Pywikibot Documentation

Release 2.0rc1

Pywikibot team

June 26, 2015

Contents

Warning: **This documentation is incomplete**, and needs quite some work.
If you are not familiar with pywikibot, please start at the old documentation: Manual:Pywikibot on mediawiki.org

Pywikibot is a full-stack framework for editing MediaWiki wiki's. (more intro)

Pywikibot supports Python 2.6.5, 2.7 and 3.2+.

The documentation consists of four major parts:

1. Installation
2. `scripts/index`
3. Using pywikibot as library
4. API reference

CHAPTER 1

For bot users:

1.1 Installation

Note: Please see the old documentation at Manual:Pywikibot/Installation

1.2 Getting help

Note: Please see Manual:Pywikibot/Communication

CHAPTER 2

For bot developers:

2.1 Installation

Note: Please see the old documentation at Manual:Pywikibot/Installation

2.2 Using pywikibot as library

Note: Please see the old documentation at Manual:Pywikibot/Create your own script

2.3 Getting help

Note: Please see Manual:Pywikibot/Communication

2.4 API reference

2.4.1 High-level request structure

User code mainly interacts with `pywikibot.Page` objects, which represent pages on a specific wiki. These objects get their properties by calling functions on their associated `pywikibot.Site` object, which represents a specific wiki.

The `pywikibot.Site` object then calls the MediaWiki API using the functions provided by *`pywikibot.data.api`*. This layer then uses *`pywikibot.comms.http.request()`* to do the actual HTTP request.

2.4.2 Contents

pywikibot Package

pywikibot Package

The initialization file for the Pywikibot framework.

class pywikibot.__init__.**UnicodeMixin**
Bases: object

Mixin class to add __str__ method in Python 2 or 3.

pywikibot.__init__.**translate**(*code*, *xdict*, *parameters=None*, *fallback=False*)
Return the most appropriate translation from a translation dict.

Given a language code and a dictionary, returns the dictionary's value for key 'code' if this key exists; otherwise tries to return a value for an alternative language that is most applicable to use on the wiki in language 'code' except fallback is False.

The language itself is always checked first, then languages that have been defined to be alternatives, and finally English. If none of the options gives result, we just take the one language from xdict which may not be always the same. When fallback is iterable it'll return None if no code applies (instead of returning one).

For PLURAL support have a look at the twntranslate method

Parameters

- **code** (*string or Site object*) – The language code
- **xdict** (*dict, string, unicode*) – dictionary with language codes as keys or extended dictionary with family names as keys containing language dictionaries or a single (unicode) string. May contain PLURAL tags as described in twntranslate
- **parameters** (*dict, string, unicode, int*) – For passing (plural) parameters
- **fallback** (*boolean or iterable*) – Try an alternate language code. If it's iterable it'll also try those entries and choose the first match.

class pywikibot.__init__.**Page**(*source*, *title=''*, *ns=0*)
Bases: pywikibot.page.BasePage

Page: A MediaWiki page.

set_redirect_target(*target_page*, *create=False*, *force=False*, *keep_section=False*, *save=True*, ***kwargs*)
Change the page's text to point to the redirect page.

Parameters

- **target_page** (*pywikibot.Page or string*) – target of the redirect, this argument is required.
- **create** (*bool*) – if true, it creates the redirect even if the page doesn't exist.
- **force** (*bool*) – if true, it set the redirect target even the page doesn't exist or it's not redirect.
- **keep_section** (*bool*) – if the old redirect links to a section and the new one doesn't it uses the old redirect's section.
- **save** (*bool*) – if true, it saves the page immediately.

- **kwargs** – Arguments which are used for saving the page directly afterwards, like 'summary' for edit summary.

templatesWithParams()
Iterate templates used on this Page.

Returns a generator that yields a tuple for each use of a template

in the page, with the template Page as the first entry and a list of parameters as the second entry.

class pywikibot.__init__.**FilePage**(*source*, *title=''*)
Bases: pywikibot.page.Page

A subclass of Page representing a file description page.

Supports the same interface as Page, with some added methods.

fileIsOnCommons()
DEPRECATED. Check if the image is stored on Wikimedia Commons.

Returns bool

fileIsShared()
Check if the file is stored on any known shared repository.

Returns bool

fileUrl()
Return the URL for the file described on this page.

getFileMd5Sum()
Return image file's MD5 checksum.

getFileSHA1Sum()
Return the file's SHA1 checksum.

getFileVersionHistory()
Return the file's version history.

Returns

A list of dictionaries with the following keys:

```
[comment, sha1, url, timestamp, metadata,
```

height, width, mime, user, descriptionurl, size]

getFileVersionHistoryTable()
Return the version history in the form of a wiki table.

getFirstUploader()
Return a list with first uploader of the FilePage and timestamp.

For compatibility with compat only.

getImagePageHtml()
Download the file page, and return the HTML, as a unicode string.

Caches the HTML code, so that if you run this method twice on the same FilePage object, the page will only be downloaded once.

getLatestUploader()
Return a list with latest uploader of the FilePage and timestamp.

For compatibility with compat only.

get_file_history()
Return the file's version history.

Returns dictionary with: key: timestamp of the entry value: instance of FileInfo()

latest_file_info
Retrieve and store information of latest Image rev. of FilePage.

At the same time, the whole history of Image is fetched and cached in self._file_revisions

Returns instance of FileInfo()

oldest_file_info
Retrieve and store information of oldest Image rev. of FilePage.

At the same time, the whole history of Image is fetched and cached in self._file_revisions

Returns instance of FileInfo()

usingPages(*step=None*, *total=None*, *content=False*)
Yield Pages on which the file is displayed.

Parameters

- **step** – limit each API call to this number of pages
- **total** – iterate no more than this number of pages in total
- **content** – if True, load the current content of each iterated page (default False)

class pywikibot.__init__.**Category**(*source*, *title=''*, *sortKey=None*)
Bases: pywikibot.page.Page

A page in the Category: namespace.

articles(*recurse=False*, *step=None*, *total=None*, *content=False*, *namespaces=None*, *sortby=None*, *starttime=None*, *endtime=None*, *startsort=None*, *endsort=None*)
Yield all articles in the current category.

By default, yields all *pages* in the category that are not subcategories!

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate articles in subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate articles in first-level subcats, but no deeper.)
- **step** – limit each API call to this number of pages
- **total** – iterate no more than this number of pages in total (at all levels)
- **namespaces** (*int or list of ints*) – only yield pages in the specified namespaces
- **content** – if True, retrieve the content of the current version of each page (default False)
- **sortby** (*str*) – determines the order in which results are generated, valid values are "sortkey" (default, results ordered by category sort key) or "timestamp" (results ordered by time page was added to the category). This applies recursively.
- **starttime** (*pywikibot.Timestamp*) – if provided, only generate pages added after this time; not valid unless sortby="timestamp"
- **endtime** (*pywikibot.Timestamp*) – if provided, only generate pages added before this time; not valid unless sortby="timestamp"
- **startsort** (*str*) – if provided, only generate pages >= this title lexically; not valid if sortby="timestamp"

- **endsort** (*str*) – if provided, only generate pages <= this title lexically; not valid if sortby="timestamp"

articlesList(*recurse=False*)
DEPRECATED: equivalent to list(self.articles(...)).

aslink(*sortKey=None*)
Return a link to place a page in this Category.

Use this only to generate a "true" category link, not for interwikis or text links to category pages.

Parameters **sortKey** (*(optional) unicode*) – The sort key for the article to be placed in this Category; if omitted, default sort key is used.

categoryinfo
Return a dict containing information about the category.

The dict contains values for:

Numbers of pages, subcategories, files, and total contents.

Returns dict

copyAndKeep(*catname*, *cfdTemplates*, *message*)
Copy partial category page text (not contents) to a new title.

Like copyTo above, except this removes a list of templates (like deletion templates) that appear in the old category text. It also removes all text between the two HTML comments BEGIN CFD TEMPLATE and END CFD TEMPLATE. (This is to deal with CFD templates that are substituted.)

Returns true if copying was successful, false if target page already existed.

Parameters

- **catname** – New category title (without namespace)
- **cfdTemplates** – A list (or iterator) of templates to be removed from the page text

Returns True if copying was successful, False if target page already existed.

copyTo(*cat*, *message*)
Copy text of category page to a new page. Does not move contents.

Parameters

- **cat** (*unicode or Category*) – New category title (without namespace) or Category object
- **message** (*unicode*) – message to use for category creation message If two %s are provided in message, will be replaced by (self.title, authorsList)

Returns True if copying was successful, False if target page already existed.

isEmptyCategory()
Return True if category has no members (including subcategories).

isHiddenCategory()
Return True if the category is hidden.

members(*recurse=False*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Yield all category contents (subcats, pages, and files).

newest_pages(*total=None*)
Return pages in a category ordered by the creation date.

If two or more pages are created at the same time, the pages are returned in the order they were added to the category. The most recently added page is returned first.

It only allows to return the pages ordered from newest to oldest, as it is impossible to determine the oldest page in a category without checking all pages. But it is possible to check the category in order with the newly added first and it yields all pages which were created after the currently checked page was added (and thus there is no page created after any of the cached but added before the currently checked).

Parameters **total** (*int*) – The total number of pages queried.

Returns A page generator of all pages in a category ordered by the creation date. From newest to oldest. Note: It currently only returns Page instances and not a subclass of it if possible. This might change so don't expect to only get Page instances.

Return type generator

subcategories(*recurse=False*, *step=None*, *total=None*, *content=False*)
Iterate all subcategories of the current category.

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate subcategories of subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate direct subcats and first-level sub-sub-cats, but no deeper.)
- **step** – limit each API call to this number of categories
- **total** – iterate no more than this number of subcategories in total (at all levels)
- **content** – if True, retrieve the content of the current version of each category description page (default False)

subcategoriesList(*recurse=False*)
DEPRECATED: Equivalent to list(self.subcategories(...)).

supercategories()
DEPRECATED: equivalent to self.categories().

supercategoriesList()
DEPRECATED: equivalent to list(self.categories(...)).

class pywikibot.__init__.**Link**(*text*, *source=None*, *defaultNamespace=0*)
Bases: *pywikibot.tools.ComparableMixin*

A MediaWiki link (local or interwiki).

Has the following attributes:

```
- site:  The Site object for the wiki linked to
- namespace: The namespace of the page linked to (int)
- title: The title of the page linked to (unicode); does not include
  namespace or section
- section: The section of the page linked to (unicode or None); this
  contains any text following a '#' character in the title
- anchor: The anchor text (unicode or None); this contains any text
  following a '|' character inside the link
```

anchor
Return the anchor of the link.

Returns unicode

astext(*onsite=None*)
Return a text representation of the link.

Parameters **onsite** – if specified, present as a (possibly interwiki) link from the given site; otherwise, present as an internal link on the source site.

canonical_title()
Return full page title, including localized namespace.

classmethod **fromPage**(*page*, *source=None*)
Create a Link to a Page.

Parameters

- **page** (Page) – target Page
- **source** – Link from site source
- **source** – Site

Returns Link

illegal_titles_pattern = re.compile('[\\x00-\\x1f\\x23\\x3c\\x3e\\x5b\\x5d\\x7b\\x7c\\x7d\\x7f]|%[0-9A-Fa-f]{2}|&[

classmethod **langlinkUnsafe**(*lang*, *title*, *source*)
Create a "lang:title" Link linked from source.

Assumes that the lang & title come clean, no checks are made.

Parameters

- **lang** (*str*) – target site code (language)
- **title** (*unicode*) – target Page
- **source** – Link from site source
- **source** – Site

Returns Link

namespace
Return the namespace of the link.

Returns unicode

ns_title(*onsite=None*)
Return full page title, including namespace.

Parameters **onsite** – site object if specified, present title using onsite local namespace, otherwise use self canonical namespace.

if no corresponding namespace is found in onsite, pywikibot.Error is raised.

parse()
Parse wikitext of the link.

Called internally when accessing attributes.

parse_site()
Parse only enough text to determine which site the link points to.

This method does not parse anything after the first ":"; links with multiple interwiki prefixes (such as "wikt:fr:Parlais") need to be re-parsed on the first linked wiki to get the actual site.

Returns The family name and site code for the linked site. If the site is not supported by the configured families it returns None instead of a str.

Return type str or None, str or None

section
Return the section of the link.

Returns unicode

site
Return the site of the link.

Returns unicode

title
Return the title of the link.

Returns unicode

class pywikibot.__init__.**User**(*source*, *title=''*)
Bases: pywikibot.page.Page

A class that represents a Wiki user.

This class also represents the Wiki page User:<username>

block(*expiry*, *reason*, *anononly=True*, *nocreate=True*, *autoblock=True*, *noemail=False*, *reblock=False*)
Block user.

Parameters

- **expiry** (*pywikibot.Timestamp|str*) – When the block should expire
- **reason** (*basestring*) – Block reason
- **anononly** (*bool*) – Whether block should only affect anonymous users
- **nocreate** (*bool*) – Whether to block account creation
- **autoblock** (*bool*) – Whether to enable autoblock
- **noemail** (*bool*) – Whether to disable email access
- **reblock** (*bool*) – Whether to reblock if a block already is set

Returns None

contributions(*total=500*, *namespaces=[]*)
Yield tuples describing this user edits.

Each tuple is composed of a pywikibot.Page object, the revision id (int), the edit timestamp (as a pywikibot.Timestamp object), and the comment (unicode). Pages returned are not guaranteed to be unique.

Parameters

- **total** (*int*) – limit result to this number of pages
- **namespaces** (*list*) – only iterate links in these namespaces

editCount(*force=False*)
Return edit count for a registered user.

Always returns 0 for 'anonymous' users.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long

editedPages(*total=500*)
DEPRECATED. Use contributions().

Yields pywikibot.Page objects that this user has edited, with an upper bound of 'total'. Pages returned are not guaranteed to be unique.

Parameters **total** (*int.*) – limit result to this number of pages.

getUserPage(*subpage=''*)
Return a Page object relative to this user's main page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main page title (optional)

getUserTalkPage(*subpage=''*)
Return a Page object relative to this user's main talk page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main talk page title (optional)

getprops(*force=False*)
Return a properties about the user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns dict

groups(*force=False*)
Return a list of groups to which this user belongs.

The list of groups may be empty.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns list

isAnonymous()
Determine if the user is editing as an IP address.

Returns bool

isBlocked(*force=False*)
Determine whether the user is currently blocked.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isEmailable(*force=False*)
Determine whether emails may be send to this user through MediaWiki.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isRegistered(*force=False*)
Determine if the user is registered on the site.

It is possible to have a page named User:xyz and not have a corresponding user with username xyz.

The page does not need to exist for this method to return True.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

name()
The username.

Returns unicode

registration(*force=False*)
Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns pywikibot.Timestamp or None

registrationTime (*force=False*)
DEPRECATED. Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long (MediaWiki's internal timestamp format) or 0

sendMail (*subject*, *text*, *ccme=False*)
Send an email to this user via MediaWiki's email interface.

Outputs 'Email sent' if the email was sent.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body
- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises **_EmailUserError** logged in user does not have 'sendemail' right or the target has disabled receiving emails

Returns operation successful indicator

Return type bool

send_email (*subject*, *text*, *ccme=False*)
Send an email to this user via MediaWiki's email interface.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body
- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises

- **NotEmailableError** – the user of this User is not emailable
- **UserRightsError** – logged in user does not have 'sendemail' right

Returns operation successful indicator

Return type bool

uploadedImages (*total=10*)
Yield tuples describing files uploaded by this user.

Each tuple is composed of a pywikibot.Page, the timestamp (str in ISO8601 format), comment (unicode) and a bool for pageid > 0. Pages returned are not guaranteed to be unique.

Parameters **total** (*int*) – limit result to this number of pages

username
The username.

Convenience method that returns the title of the page with namespace prefix omitted, which is the username.

Returns unicode

class pywikibot.__init__.**ItemPage** (*site*, *title=None*, *ns=None*)
Bases: pywikibot.page.WikibasePage

Wikibase entity of type 'item'.

A Wikibase item may be defined by either a 'Q' id (qid), or by a site & title.

If an item is defined by site & title, once an item's qid has been looked up, the item is then defined by the qid.

addClaim(*claim*, *bot=True*, ***kwargs*)
Add a claim to the item.

Parameters

- **claim** (Claim) – The claim to add
- **bot** (*bool*) – Whether to flag as bot (if possible)

classmethod fromPage(*page*, *lazy_load=False*)
Get the ItemPage for a Page that links to it.

Parameters

- **page** (*pywikibot.Page*) – Page to look for corresponding data item
- **lazy_load** (*bool*) – Do not raise NoPage if either page or corresponding ItemPage does not exist.

Returns ItemPage

Raises NoPage There is no corresponding ItemPage for the page

get(*force=False*, **args*, ***kwargs*)
Fetch all item data, and cache it.

Parameters

- **force** (*bool*) – override caching
- **args** – values of props

getRedirectTarget()
Return the redirect target for this page.

getSitelink(*site*, *force=False*)
Return the title for the specific site.

If the item doesn't have that language, raise NoPage.

Parameters

- **site** (*pywikibot.Site or database name*) – Site to find the linked page of.
- **force** – override caching

Returns unicode

iterlinks(*family=None*)
Iterate through all the sitelinks.

Parameters **family** (*str|pywikibot.family.Family*) – string/Family object which represents what family of links to iterate

Returns iterator of pywikibot.Page objects

mergeInto(*item*, ***kwargs*)
Merge the item into another item.

Parameters **item** (*pywikibot.ItemPage*) – The item to merge into

removeClaims(*claims*, ***kwargs*)
Remove the claims from the item.

removeSitelink(*site*, ***kwargs*)
Remove a sitelink.

A site can either be a Site object, or it can be a dbName.

removeSitelinks(*sites*, ***kwargs*)
Remove sitelinks.

Sites should be a list, with values either being Site objects, or dbNames.

setSitelink(*sitelink*, ***kwargs*)
Set sitelinks. Calls setSitelinks().

A sitelink can either be a Page object, or a {'site':dbname,'title':title} dictionary.

setSitelinks(*sitelinks*, ***kwargs*)
Set sitelinks.

Sitelinks should be a list. Each item in the list can either be a Page object, or a dict with a value for 'site' and 'title'.

set_redirect_target(*target_page*, *create=False*, *force=False*, *keep_section=False*, *save=True*, ***kwargs*)
Make the item redirect to another item.

You need to define an extra argument to make this work, like save=True :param target_page: target of the redirect, this argument is required. :type target_page: pywikibot.Item or string :param force: if true, it sets the redirect target even the page

is not redirect.

title(***kwargs*)
Return ID as title of the ItemPage.

If the ItemPage was lazy-loaded via ItemPage.fromPage, this method will fetch the wikibase item ID for the page, potentially raising NoPage with the page on the linked wiki if it does not exist, or does not have a corresponding wikibase item ID.

This method also refreshes the title if the id property was set. i.e. item.id = 'Q60'

All optional keyword parameters are passed to the superclass.

toJSON(*diffto=None*)
Create JSON suitable for Wikibase API.

When diffto is provided, JSON representing differences to the provided data is created.

Parameters **diffto** (*dict*) – JSON containing claim data

Returns dict

class pywikibot.__init__.**PropertyPage**(*source*, *title=''*)
Bases: pywikibot.page.WikibasePage, pywikibot.page.Property

A Wikibase entity in the property namespace.

Should be created as::

```
PropertyPage(DataSite, 'P21')
```

get (*force=False*, **args*)
Fetch the property entity, and cache it.

Parameters

- **force** – override caching
- **args** – values of props

newClaim (**args*, ***kwargs*)
Helper function to create a new claim object for this property.

Returns Claim

class pywikibot.__init__.**Claim** (*site*, *pid*, *snak=None*, *hash=None*, *isReference=False*, *isQualifier=False*, ***kwargs*)
Bases: pywikibot.page.Property

A Claim on a Wikibase entity.

Claims are standard claims as well as references.

TARGET_CONVERTER = {'time': <function Claim.<lambda> at 0x7f941ec7f268>, 'globe-coordinate': <bound method typ

addQualifier (*qualifier*, ***kwargs*)
Add the given qualifier.

Parameters **qualifier** (Claim) – the qualifier to add

addSource (*claim*, ***kwargs*)
Add the claim as a source.

Parameters **claim** (*pywikibot.Claim*) – the claim to add

addSources (*claims*, ***kwargs*)
Add the claims as one source.

Parameters **claims** (*list of pywikibot.Claim*) – the claims to add

changeRank (*rank*)
Change the rank of the Claim and save.

changeSnakType (*value=None*, ***kwargs*)
Save the new snak value.

TODO: Is this function really needed?

changeTarget (*value=None*, *snaktype='value'*, ***kwargs*)
Set the target value in the data repository.

Parameters

- **value** (*object*) – The new target value.
- **snaktype** (*str ('value', 'somevalue', or 'novalue')*) – The new snak type.

classmethod fromJSON (*site*, *data*)
Create a claim object from JSON returned in the API call.

Parameters **data** (*dict*) – JSON containing claim data

Returns Claim

getRank ()
Return the rank of the Claim.

getSnakType ()
Return the type of snak.

Returns str ('value', 'somevalue' or 'novalue')

getSources()
Return a list of sources, each being a list of Claims.

Returns list

getTarget()
Return the target value of this Claim.

None is returned if no target is set

Returns object

has_qualifier(*qualifier_id*, *target*)
Check whether Claim contains specified qualifier.

Parameters

- **qualifier_id** (*str*) – id of the qualifier
- **target** – qualifier target to check presence of

Returns true if the qualifier was found, false otherwise

Return type bool

classmethod qualifierFromJSON(*site*, *data*)
Create a Claim for a qualifier from JSON.

Qualifier objects are represented a bit differently like references, but I'm not sure if this even requires it's own function.

Returns Claim

classmethod referenceFromJSON(*site*, *data*)
Create a dict of claims from reference JSON returned in the API call.

Reference objects are represented a bit differently, and require some more handling.

Returns dict

removeSource(*source*, ***kwargs*)
Remove the source. Calls removeSources().

Parameters **source** (*pywikibot.Claim*) – the source to remove

removeSources(*sources*, ***kwargs*)
Remove the sources.

Parameters **sources** (*list of pywikibot.Claim*) – the sources to remove

setRank(*rank*)
Set the rank of the Claim.

setSnakType(*value*)
Set the type of snak.

Parameters **value** (*str ('value', 'somevalue', or 'novalue')*) – Type of snak

setTarget(*value*)
Set the target value in the local object.

Parameters **value** (*object*) – The new target value.

Raises ValueError if value is not of the type required for the Claim type.

target_equals(*value*)

Check whether the Claim's target is equal to specified value.

The function checks for::

- ItemPage ID equality
- WbTime year equality
- Coordinate equality, regarding precision
- direct equality

Parameters **value** – the value to compare with

Returns true if the Claim's target is equal to the value provided, false otherwise

Return type bool

toJSON()

pywikibot.__init__.**html2unicode**(*text*, *ignore=None*)

Replace HTML entities with equivalent unicode.

Parameters

- **ignore** – HTML entities to ignore
- **ignore** – list of int

Returns unicode

pywikibot.__init__.**url2unicode**(*title*, *encodings='utf-8'*)

Convert URL-encoded text to unicode using several encoding.

Uses the first encoding that doesn't cause an error.

Parameters

- **title** (*str*) – URL-encoded character data to convert
- **encodings** (*str, list or Site*) – Encodings to attempt to use during conversion.

Returns unicode

Raises UnicodeError Could not convert using any encoding.

pywikibot.__init__.**unicode2html**(*x*, *encoding*)

Convert unicode string to requested HTML encoding.

Attempt to encode the string into the desired format; if that doesn't work, encode the unicode into HTML &#; entities. If it does work, return it unchanged.

Parameters

- **x** (*unicode*) – String to update
- **encoding** (*str*) – Encoding to use

Returns str

pywikibot.__init__.**stdout**(*text*, *decoder=None*, *newline=True*, ***kwargs*)

Output script results to the user via the userinterface.

pywikibot.__init__.**output**(*text*, *decoder=None*, *newline=True*, *toStdout=False*, ***kwargs*)

Output a message to the user via the userinterface.

Works like print, but uses the encoding used by the user's console (console_encoding in the configuration file) instead of ASCII.

If decoder is None, text should be a unicode string. Otherwise it should be encoded in the given encoding.

If newline is True, a line feed will be added after printing the text.

If toStdout is True, the text will be sent to standard output, so that it can be piped to another process. All other text will be sent to stderr. See: https://en.wikipedia.org/wiki/Pipeline_%28Unix%29

text can contain special sequences to create colored output. These consist of the escape character 03 and the color name in curly braces, e. g. 03{lightpurple}. 03{default} resets the color.

Other keyword arguments are passed unchanged to the logger; so far, the only argument that is useful is "exc_info=True", which causes the log message to include an exception traceback.

pywikibot.__init__.**warning**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a warning message to the user via the userinterface.

pywikibot.__init__.**error**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output an error message to the user via the userinterface.

pywikibot.__init__.**critical**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a critical record to the log file.

pywikibot.__init__.**debug**(*text*, *layer*, *decoder=None*, *newline=True*, ***kwargs*)
Output a debug record to the log file.

Parameters **layer** – The name of the logger that text will be sent to.

pywikibot.__init__.**exception**(*msg=None*, *decoder=None*, *newline=True*, *tb=False*, ***kwargs*)
Output an error traceback to the user via the userinterface.

Use directly after an 'except' statement::

```
...
except:
    pywikibot.exception()
...
```

or alternatively::

```
...
except Exception as e:
    pywikibot.exception(e)
...
```

Parameters **tb** – Set to True in order to output traceback also.

pywikibot.__init__.**input_choice**(*question*, *answers*, *default=None*, *return_shortcut=True*, *automatic_quit=True*, *force=False*)
Ask the user the question and return one of the valid answers.

Parameters

- **question** (*basestring*) – The question asked without trailing spaces.
- **answers** (*Iterable containing an iterable of length two*) – The valid answers each containing a full length answer and a shortcut. Each value must be unique.
- **default** (*basestring*) – The result if no answer was entered. It must not be in the valid answers and can be disabled by setting it to None. If it should be linked with the valid answers it must be its shortcut.
- **return_shortcut** (*bool*) – Whether the shortcut or the index of the answer is returned.

- **automatic_quit** (*bool*) – Adds the option 'Quit' ('q') and throw a *QuitKeyboardInterrupt* if selected.
- **force** (*bool*) – Automatically use the default

Returns The selected answer shortcut or index. Is -1 if the default is selected, it does not return the shortcut and the default is not a valid shortcut.

Return type int (if not return shortcut), basestring (otherwise)

pywikibot.__init__.**input**(*question*, *password=False*, *default=''*, *force=False*)
Ask the user a question, return the user's answer.

Parameters

- **question** (*unicode*) – a string that will be shown to the user. Don't add a space after the question mark/colon, this method will do this for you.
- **password** (*bool*) – if True, hides the user's input (for password entry).
- **default** (*basestring*) – The default answer if none was entered. None to require an answer.
- **force** (*bool*) – Automatically use the default

Return type unicode

pywikibot.__init__.**input_yn**(*question*, *default=None*, *automatic_quit=True*, *force=False*)
Ask the user a yes/no question and returns the answer as a bool.

Parameters

- **question** (*basestring*) – The question asked without trailing spaces.
- **default** (*basestring or bool*) – The result if no answer was entered. It must be a bool or 'y' or 'n' and can be disabled by setting it to None.
- **automatic_quit** (*bool*) – Adds the option 'Quit' ('q') and throw a *QuitKeyboardInterrupt* if selected.
- **force** (*bool*) – Automatically use the default

Returns Return True if the user selected yes and False if the user selected no. If the default is not None it'll return True if default is True or 'y' and False if default is False or 'n'.

Return type bool

pywikibot.__init__.**inputChoice**(*question*, *answers*, *hotkeys*, *default=None*)
Ask the user a question with several options, return the user's choice.

DEPRECATED: Use *input_choice* instead!

The user's input will be case-insensitive, so the hotkeys should be distinctive case-insensitively.

Parameters

- **question** (*basestring*) – a string that will be shown to the user. Don't add a space after the question mark/colon, this method will do this for you.
- **answers** (*list of basestring*) – a list of strings that represent the options.
- **hotkeys** – a list of one-letter strings, one for each answer.
- **default** – an element of hotkeys, or None. The default choice that will be returned when the user just presses Enter.

Returns a one-letter string in lowercase.

Return type str

pywikibot.__init__.**handle_args**(*args=None*, *do_help=True*)
Handle standard command line arguments, and return the rest as a list.

Takes the command line arguments as Unicode strings, processes all global parameters such as -lang or -log, initialises the logging layer, which emits startup information into log at level 'verbose'.

This makes sure that global arguments are applied first, regardless of the order in which the arguments were given.

args may be passed as an argument, thereby overriding sys.argv

Parameters

- **args** (*list of unicode*) – Command line arguments
- **do_help** (*bool*) – Handle parameter '-help' to show help and invoke sys.exit

Returns list of arguments not recognised globally

Return type list of unicode

pywikibot.__init__.**handleArgs**(**args*)
DEPRECATED. Use handle_args().

pywikibot.__init__.**showHelp**(*module_name=None*)
Show help for the Bot.

pywikibot.__init__.**log**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a record to the log file.

pywikibot.__init__.**calledModuleName**()
Return the name of the module calling this function.

This is required because the -help option loads the module's docstring and because the module name will be used for the filename of the log.

Return type unicode

class pywikibot.__init__.**Bot**(***kwargs*)
Bases: object

Generic Bot to be subclassed.

This class provides a run() method for basic processing of a generator one page at a time.

If the subclass places a page generator in self.generator, Bot will process each page in the generator, invoking the method treat() which must then be implemented by subclasses.

If the subclass does not set a generator, or does not override treat() or run(), NotImplementedError is raised.

availableOptions = {'always': False}

current_page
Return the current working page as a property.

getOption(*option*)
Get the current value of an option.

Parameters **option** – key defined in Bot.availableOptions

quit()
Cleanup and quit processing.

run()
Process all pages in generator.

setOptions(***kwargs*)
Set the instance options.

Parameters **kwargs** (*dict*) – options

site
Site that the bot is using.

treat(*page*)
Process one page (Abstract method).

userPut(*page*, *oldtext*, *newtext*, ***kwargs*)
Save a new revision of a page, with user confirmation as required.

Print differences, ask user for confirmation, and puts the page if needed.

Option used:: * 'always'

Keyword args used:: * 'async' - passed to page.save * 'summary' - passed to page.save * 'show_diff' - show changes between oldtext and newtext (enabled) * 'ignore_save_related_errors' - report and ignore (disabled) * 'ignore_server_errors' - report and ignore (disabled)

user_confirm(*question*)
Obtain user response if bot option 'always' not enabled.

class pywikibot.__init__.**CurrentPageBot**(***kwargs*)
Bases: *pywikibot.bot.Bot*

A bot which automatically sets 'current_page' on each treat().

ignore_save_related_errors = True

ignore_server_errors = False

put_current(*new_text*, *ignore_save_related_errors=None*, *ignore_server_errors=None*, ***kwargs*)
Call *Bot.userPut* but use the current page.

It compares the new_text to the current page text.

Parameters

- **new_text** (*basestring*) – The new text
- **ignore_save_related_errors** (*bool or None*) – Ignore save related errors and automatically print a message. If None uses this instances default.
- **ignore_server_errors** (*bool or None*) – Ignore server errors and automatically print a message. If None uses this instances default.
- **kwargs** (*dict*) – Additional parameters directly given to *Bot.userPut*.

treat(*page*)
Set page to current page and treat that page.

treat_page()
Process one page (Abstract method).

class pywikibot.__init__.**WikidataBot**(***kwargs*)
Bases: *pywikibot.bot.Bot*

Generic Wikidata Bot to be subclassed.

Source claims (P143) can be created for specific sites.

cacheSources()
Fetch the sources from the list on Wikidata.

It is stored internally and reused by getSource()

getSource(*site*)
Create a Claim usable as a source for Wikibase statements.

Parameters **site** (*Site*) – site that is the source of assertions.

Returns Claim

get_property_by_name(*property_name*)
Find given property and return its ID.

Method first uses site.search() and if the property isn't found, then asks user to provide the property ID.

Parameters **property_name** (*str*) – property to find

run()
Process all pages in generator.

user_edit_entity(*item*, *data=None*, ***kwargs*)
Edit entity with data provided, with user confirmation as required.

Parameters

- **item** (ItemPage) – page to be edited
- **data** – data to be saved, or None if the diff should be created automatically
- **summary** – revision comment, passed to ItemPage.editEntity

@kwtype summary: str :kwarg show_diff: show changes between oldtext and newtext (default:

True)

@kwtype show_diff: bool :kwarg ignore_server_errors: if True, server errors will be reported

and ignored (default: False)

@kwtype ignore_server_errors: bool :kwarg ignore_save_related_errors: if True, errors related to page save will be reported and ignored (default: False) @kwtype ignore_save_related_errors: bool

exception pywikibot.__init__.**Error**(*arg*)
Bases: *pywikibot.tools.UnicodeMixin*, Exception

Pywikibot error

exception pywikibot.__init__.**InvalidTitle**(*arg*)
Bases: pywikibot.exceptions.Error

Invalid page title

exception pywikibot.__init__.**BadTitle**(*arg*)
Bases: pywikibot.exceptions.Error

Server responded with BadTitle.

exception pywikibot.__init__.**NoPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageRelatedError

Page does not exist

message = "Page %s doesn't exist."

exception pywikibot.__init__.**SectionError**(*arg*)
Bases: pywikibot.exceptions.Error

The section specified by # does not exist

exception pywikibot.__init__.**SiteDefinitionError**(*arg*)
Bases: pywikibot.exceptions.Error

Site does not exist

pywikibot.__init__.**NoSuchSite**
alias of *SiteDefinitionError*

exception pywikibot.__init__.**UnknownSite**(*arg*)
Bases: pywikibot.exceptions.SiteDefinitionError

Site does not exist in Family

exception pywikibot.__init__.**UnknownFamily**(*arg*)
Bases: pywikibot.exceptions.SiteDefinitionError

Family is not registered

exception pywikibot.__init__.**UnknownExtension**(*arg*)
Bases: pywikibot.exceptions.Error, NotImplementedError

Extension is not defined.

exception pywikibot.__init__.**NoUsername**(*arg*)
Bases: pywikibot.exceptions.Error

Username is not in user-config.py.

exception pywikibot.__init__.**UserBlocked**(*arg*)
Bases: pywikibot.exceptions.Error

Your username or IP has been blocked

pywikibot.__init__.**UserActionRefuse**()
Email related error.

exception pywikibot.__init__.**PageRelatedError**(*page*, *message=None*)
Bases: pywikibot.exceptions.Error

Abstract Exception, used when the exception concerns a particular Page.

This class should be used when the Exception concerns a particular Page, and when a generic message can be written once for all.

getPage()
Return the page related to the exception.

message = None

exception pywikibot.__init__.**IsRedirectPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageRelatedError

Page is a redirect page

message = 'Page %s is a redirect page.'

exception pywikibot.__init__.**IsNotRedirectPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageRelatedError

Page is not a redirect page

message = 'Page %s is not a redirect page.'

exception pywikibot.__init__.**PageSaveRelatedError**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageRelatedError

Saving the page has failed

args
Expose args.

message = 'Page %s was not saved.'

pywikibot.__init__.**PageNotSaved**
alias of *PageSaveRelatedError*

exception pywikibot.__init__.**OtherPageSaveError**(*page*, *reason*)
Bases: pywikibot.exceptions.PageSaveRelatedError

Saving the page has failed due to uncatchable error.

args
Expose args.

message = 'Edit to page %(title)s failed:\n%(reason)s'

exception pywikibot.__init__.**LockedPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageSaveRelatedError

Page is locked

message = 'Page %s is locked.'

exception pywikibot.__init__.**CascadeLockedPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.LockedPage

Page is locked due to cascading protection

message = 'Page %s is locked due to cascading protection.'

exception pywikibot.__init__.**LockedNoPage**(*page*, *message=None*)
Bases: pywikibot.exceptions.LockedPage

Title is locked against creation

message = 'Page %s does not exist and is locked preventing creation.'

exception pywikibot.__init__.**NoCreateError**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageSaveRelatedError

Parameter nocreate doesn't allow page creation.

message = 'Page %s could not be created due to parameter nocreate'

exception pywikibot.__init__.**EditConflict**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageSaveRelatedError

There has been an edit conflict while uploading the page

message = 'Page %s could not be saved due to an edit conflict'

exception pywikibot.__init__.**PageDeletedConflict**(*page*, *message=None*)
Bases: pywikibot.exceptions.EditConflict

Page was deleted since being retrieved

message = 'Page %s has been deleted since last retrieved.'

exception pywikibot.__init__.**PageCreatedConflict**(*page*, *message=None*)
Bases: pywikibot.exceptions.EditConflict

Page was created by another user

message = 'Page %s has been created since last retrieved.'

exception pywikibot.__init__.**UploadWarning**(*code*, *message*)
Bases: *pywikibot.data.api.APIError*

Upload failed with a warning message (passed as the argument).

message
Return warning message.

exception pywikibot.__init__.**ServerError**(*arg*)
Bases: pywikibot.exceptions.Error

Got unexpected server response

exception pywikibot.__init__.**FatalServerError**(*arg*)
Bases: pywikibot.exceptions.ServerError

A fatal server error will not be corrected by resending the request.

exception pywikibot.__init__.**Server504Error**(*arg*)
Bases: pywikibot.exceptions.Error

Server timed out with HTTP 504 code

exception pywikibot.__init__.**CaptchaError**(*arg*)
Bases: pywikibot.exceptions.Error

Captcha is asked and config.solve_captcha == False.

exception pywikibot.__init__.**SpamfilterError**(*page*, *url*)
Bases: pywikibot.exceptions.PageSaveRelatedError

Page save failed because MediaWiki detected a blacklisted spam URL.

message = 'Edit to page %(title)s rejected by spam filter due to content:\n%(url)s'

exception pywikibot.__init__.**CircularRedirect**(*page*, *message=None*)
Bases: pywikibot.exceptions.PageRelatedError

Page is a circular redirect.

Exception argument is the redirect target; this may be the same title as this page or a different title (in which case the target page directly or indirectly redirects back to this one)

message = 'Page %s is a circular redirect.'

exception pywikibot.__init__.**InterwikiRedirectPage**(*page*, *target_page*)
Bases: pywikibot.exceptions.PageRelatedError

Page is a redirect to another site.

This is considered invalid in Pywikibot. See Bug 73184.

message = 'Page redirects to a page on another Site.\nPage: %(page)s\nTarget page: %(target_page)s on %(target_site)

exception pywikibot.__init__.**WikiBaseError**(*arg*)
Bases: pywikibot.exceptions.Error

Wikibase related error.

exception pywikibot.__init__.**CoordinateGlobeUnknownException**(*arg*)
Bases: pywikibot.exceptions.WikiBaseError, NotImplementedError

This globe is not implemented yet in either WikiBase or pywikibot.

exception pywikibot.__init__.**QuitKeyboardInterrupt**
Bases: KeyboardInterrupt

The user has cancelled processing at a prompt.

pywikibot.__init__.**unescape**(*a, **kw)

pywikibot.__init__.**replaceExcept**(*a, **kw)

pywikibot.__init__.**removeDisabledParts**(*a, **kw)

pywikibot.__init__.**removeHTMLParts**(*a, **kw)

pywikibot.__init__.**isDisabled**(*a, **kw)

pywikibot.__init__.**interwikiFormat**(*a, **kw)

pywikibot.__init__.**interwikiSort**(*a, **kw)

pywikibot.__init__.**getLanguageLinks**(*a, **kw)

pywikibot.__init__.**replaceLanguageLinks**(*a, **kw)

pywikibot.__init__.**removeLanguageLinks**(*a, **kw)

pywikibot.__init__.**removeLanguageLinksAndSeparator**(*a, **kw)

pywikibot.__init__.**getCategoryLinks**(*a, **kw)

pywikibot.__init__.**categoryFormat**(*a, **kw)

pywikibot.__init__.**replaceCategoryLinks**(*a, **kw)

pywikibot.__init__.**removeCategoryLinks**(*a, **kw)

pywikibot.__init__.**removeCategoryLinksAndSeparator**(*a, **kw)

pywikibot.__init__.**replaceCategoryInPlace**(*a, **kw)

pywikibot.__init__.**compileLinkR**(*a, **kw)

pywikibot.__init__.**extract_templates_and_params**(*a, **kw)

bot Module

User-interface related functions for building bots.

class pywikibot.bot.**Bot**(**kwargs)

Bases: object

Generic Bot to be subclassed.

This class provides a run() method for basic processing of a generator one page at a time.

If the subclass places a page generator in self.generator, Bot will process each page in the generator, invoking the method treat() which must then be implemented by subclasses.

If the subclass does not set a generator, or does not override treat() or run(), NotImplementedError is raised.

availableOptions = {'always': False}

current_page

Return the current working page as a property.

getOption(*option*)

Get the current value of an option.

Parameters **option** – key defined in Bot.availableOptions

quit()
Cleanup and quit processing.

run()
Process all pages in generator.

setOptions(***kwargs*)
Set the instance options.

Parameters **kwargs** (*dict*) – options

site
Site that the bot is using.

treat(*page*)
Process one page (Abstract method).

userPut(*page*, *oldtext*, *newtext*, ***kwargs*)
Save a new revision of a page, with user confirmation as required.

Print differences, ask user for confirmation, and puts the page if needed.

Option used:: * 'always'

Keyword args used:: * 'async' - passed to page.save * 'summary' - passed to page.save * 'show_diff' - show changes between oldtext and newtext (enabled) * 'ignore_save_related_errors' - report and ignore (disabled) * 'ignore_server_errors' - report and ignore (disabled)

user_confirm(*question*)
Obtain user response if bot option 'always' not enabled.

class pywikibot.bot.**CreatingPageBot**(***kwargs*)
Bases: *pywikibot.bot.CurrentPageBot*

A CurrentPageBot class which only treats not exisiting pages.

treat(*page*)
Treat page if doesn't exist.

class pywikibot.bot.**CurrentPageBot**(***kwargs*)
Bases: *pywikibot.bot.Bot*

A bot which automatically sets 'current_page' on each treat().

ignore_save_related_errors = True

ignore_server_errors = False

put_current(*new_text*, *ignore_save_related_errors=None*, *ignore_server_errors=None*, ***kwargs*)
Call *Bot.userPut* but use the current page.

It compares the new_text to the current page text.

Parameters

- **new_text** (*basestring*) – The new text
- **ignore_save_related_errors** (*bool or None*) – Ignore save related errors and automatically print a message. If None uses this instances default.
- **ignore_server_errors** (*bool or None*) – Ignore server errors and automatically print a message. If None uses this instances default.
- **kwargs** (*dict*) – Additional parameters directly given to *Bot.userPut*.

treat(*page*)
Set page to current page and treat that page.

treat_page()
Process one page (Abstract method).

class pywikibot.bot.**ExistingPageBot**(***kwargs*)
Bases: *pywikibot.bot.CurrentPageBot*

A CurrentPageBot class which only treats existing pages.

treat(*page*)
Treat page if it exists and handle NoPage from it.

class pywikibot.bot.**FollowRedirectPageBot**(***kwargs*)
Bases: *pywikibot.bot.CurrentPageBot*

A CurrentPageBot class which follows the redirect.

treat(*page*)
Treat target if page is redirect and the page otherwise.

class pywikibot.bot.**LoggingFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
Bases: logging.Formatter

Format LogRecords for output to file.

This formatter *ignores* the 'newline' key of the LogRecord, because every record written to a file must end with a newline, regardless of whether the output to the user's console does.

formatException(*ei*)
Convert exception trace to unicode if necessary.

Make sure that the exception trace is converted to unicode.

exceptions.Error traces are encoded in our console encoding, which is needed for plainly printing them. However, when logging them using logging.exception, the Python logging module will try to use these traces, and it will fail if they are console encoded strings.

Formatter.formatException also strips the trailing n, which we need.

class pywikibot.bot.**NoRedirectPageBot**(***kwargs*)
Bases: *pywikibot.bot.CurrentPageBot*

A NoRedirectPageBot class which only treats non-redirects.

treat(*page*)
Treat only non-redirect pages and handle IsRedirectPage from it.

exception pywikibot.bot.**QuitKeyboardInterrupt**
Bases: KeyboardInterrupt

The user has cancelled processing at a prompt.

class pywikibot.bot.**RedirectPageBot**(***kwargs*)
Bases: *pywikibot.bot.CurrentPageBot*

A RedirectPageBot class which only treats redirects.

treat(*page*)
Treat only redirect pages and handle IsNotRedirectPage from it.

class pywikibot.bot.**RotatingFileHandler**(*filename*, *mode='a'*, *maxBytes=0*, *backupCount=0*, *encoding=None*, *delay=False*)
Bases: logging.handlers.RotatingFileHandler

Modified RotatingFileHandler supporting unlimited amount of backups.

doRollover()
Modified naming system for logging files.

Overwrites the default Rollover renaming by inserting the count number between file name root and extension. If backupCount is >= 1, the system will successively create new files with the same pathname as the base file, but with inserting ".1", ".2" etc. in front of the filename suffix. For example, with a backupCount of 5 and a base file name of "app.log", you would get "app.log", "app.1.log", "app.2.log", ... through to "app.5.log". The file being written to is always "app.log" - when it gets filled up, it is closed and renamed to "app.1.log", and if files "app.1.log", "app.2.log" etc. already exist, then they are renamed to "app.2.log", "app.3.log" etc. respectively. If backupCount is == -1 do not rotate but create new numbered filenames. The newest file has the highest number except some older numbered files where deleted and the bot was restarted. In this case the ordering starts from the lowest available (unused) number.

format(*record*)
Strip trailing newlines before outputting text to file.

class pywikibot.bot.**WikidataBot**(***kwargs*)
Bases: *pywikibot.bot.Bot*

Generic Wikidata Bot to be subclassed.

Source claims (P143) can be created for specific sites.

cacheSources()
Fetch the sources from the list on Wikidata.

It is stored internally and reused by getSource()

getSource(*site*)
Create a Claim usable as a source for Wikibase statements.

Parameters **site** (*Site*) – site that is the source of assertions.

Returns Claim

get_property_by_name(*property_name*)
Find given property and return its ID.

Method first uses site.search() and if the property isn't found, then asks user to provide the property ID.

Parameters **property_name** (*str*) – property to find

run()
Process all pages in generator.

user_edit_entity(*item*, *data=None*, ***kwargs*)
Edit entity with data provided, with user confirmation as required.

Parameters

- **item** (ItemPage) – page to be edited
- **data** – data to be saved, or None if the diff should be created automatically
- **summary** – revision comment, passed to ItemPage.editEntity

@kwtype summary: str :kwarg show_diff: show changes between oldtext and newtext (default:

True)

@kwtype show_diff: bool :kwarg ignore_server_errors: if True, server errors will be reported

and ignored (default: False)

@kwtype ignore_server_errors: bool :kwarg ignore_save_related_errors: if True, errors related to page save will be reported and ignored (default: False) @kwtype ignore_save_related_errors: bool

pywikibot.bot.**calledModuleName**()
Return the name of the module calling this function.

This is required because the -help option loads the module's docstring and because the module name will be used for the filename of the log.

Return type unicode

pywikibot.bot.**critical**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a critical record to the log file.

pywikibot.bot.**debug**(*text*, *layer*, *decoder=None*, *newline=True*, ***kwargs*)
Output a debug record to the log file.

Parameters **layer** – The name of the logger that text will be sent to.

pywikibot.bot.**error**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output an error message to the user via the userinterface.

pywikibot.bot.**exception**(*msg=None*, *decoder=None*, *newline=True*, *tb=False*, ***kwargs*)
Output an error traceback to the user via the userinterface.

Use directly after an 'except' statement::

```
...
except::
    pywikibot.exception()
...
```

or alternatively::

```
...
except Exception as e::
    pywikibot.exception(e)
...
```

Parameters **tb** – Set to True in order to output traceback also.

pywikibot.bot.**handleArgs**(**args*)
DEPRECATED. Use handle_args().

pywikibot.bot.**handle_args**(*args=None*, *do_help=True*)
Handle standard command line arguments, and return the rest as a list.

Takes the command line arguments as Unicode strings, processes all global parameters such as -lang or -log, initialises the logging layer, which emits startup information into log at level 'verbose'.

This makes sure that global arguments are applied first, regardless of the order in which the arguments were given.

args may be passed as an argument, thereby overriding sys.argv

Parameters

- **args** (*list of unicode*) – Command line arguments
- **do_help** (*bool*) – Handle parameter '-help' to show help and invoke sys.exit

Returns list of arguments not recognised globally

Return type list of unicode

pywikibot.bot.**init_handlers**(*strm=None*)
Initialize logging system for terminal-based bots.

This function must be called before using pywikibot.output(); and must be called again if the destination stream is changed.

Note: this function is called by handleArgs(), so it should normally not need to be called explicitly

All user output is routed through the logging module. Each type of output is handled by an appropriate handler object. This structure is used to permit eventual development of other user interfaces (GUIs) without modifying the core bot code.

The following output levels are defined::

- DEBUG: only for file logging; debugging messages.
- **STDOUT: output that must be sent to sys.stdout (for bots that may** have their output redirected to a file or other destination).
- VERBOSE: optional progress information for display to user.
- INFO: normal (non-optional) progress information for display to user.
- INPUT: prompts requiring user response.
- WARN: user warning messages.
- ERROR: user error messages.
- CRITICAL: fatal error messages.

Accordingly, do ''not" use print statements in bot code; instead, use pywikibot.output function.

Parameters **strm** – Output stream. If None, re-uses the last stream if one was defined, otherwise uses sys.stderr

pywikibot.bot.**input**(*question*, *password=False*, *default=''*, *force=False*)
Ask the user a question, return the user's answer.

Parameters

- **question** (*unicode*) – a string that will be shown to the user. Don't add a space after the question mark/colon, this method will do this for you.
- **password** (*bool*) – if True, hides the user's input (for password entry).
- **default** (*basestring*) – The default answer if none was entered. None to require an answer.
- **force** (*bool*) – Automatically use the default

Return type unicode

pywikibot.bot.**inputChoice**(*question*, *answers*, *hotkeys*, *default=None*)
Ask the user a question with several options, return the user's choice.

DEPRECATED: Use *input_choice* instead!

The user's input will be case-insensitive, so the hotkeys should be distinctive case-insensitively.

Parameters

- **question** (*basestring*) – a string that will be shown to the user. Don't add a space after the question mark/colon, this method will do this for you.
- **answers** (*list of basestring*) – a list of strings that represent the options.
- **hotkeys** – a list of one-letter strings, one for each answer.

- **default** – an element of hotkeys, or None. The default choice that will be returned when the user just presses Enter.

Returns a one-letter string in lowercase.

Return type str

pywikibot.bot.**input_choice**(*question*, *answers*, *default=None*, *return_shortcut=True*, *automatic_quit=True*, *force=False*)

Ask the user the question and return one of the valid answers.

Parameters

- **question** (*basestring*) – The question asked without trailing spaces.
- **answers** (*Iterable containing an iterable of length two*) – The valid answers each containing a full length answer and a shortcut. Each value must be unique.
- **default** (*basestring*) – The result if no answer was entered. It must not be in the valid answers and can be disabled by setting it to None. If it should be linked with the valid answers it must be its shortcut.
- **return_shortcut** (*bool*) – Whether the shortcut or the index of the answer is returned.
- **automatic_quit** (*bool*) – Adds the option 'Quit' ('q') and throw a *QuitKeyboardInterrupt* if selected.
- **force** (*bool*) – Automatically use the default

Returns The selected answer shortcut or index. Is -1 if the default is selected, it does not return the shortcut and the default is not a valid shortcut.

Return type int (if not return shortcut), basestring (otherwise)

pywikibot.bot.**input_list_choice**(*question*, *answers*, *default=None*, *automatic_quit=True*, *force=False*)

Ask the user the question and return one of the valid answers.

Parameters

- **question** (*basestring*) – The question asked without trailing spaces.
- **answers** (*Iterable of basestring*) – The valid answers each containing a full length answer.
- **default** (*basestring*) – The result if no answer was entered. It must not be in the valid answers and can be disabled by setting it to None.
- **force** (*bool*) – Automatically use the default

Returns The selected answer.

Return type basestring

pywikibot.bot.**input_yn**(*question*, *default=None*, *automatic_quit=True*, *force=False*)

Ask the user a yes/no question and returns the answer as a bool.

Parameters

- **question** (*basestring*) – The question asked without trailing spaces.
- **default** (*basestring or bool*) – The result if no answer was entered. It must be a bool or 'y' or 'n' and can be disabled by setting it to None.
- **automatic_quit** (*bool*) – Adds the option 'Quit' ('q') and throw a *QuitKeyboardInterrupt* if selected.
- **force** (*bool*) – Automatically use the default

Returns Return True if the user selected yes and False if the user selected no. If the default is not None it'll return True if default is True or 'y' and False if default is False or 'n'.

Return type bool

pywikibot.bot.**log**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a record to the log file.

pywikibot.bot.**logoutput**(*text*, *decoder=None*, *newline=True*, *_level=20*, *_logger=''*, ***kwargs*)
Format output and send to the logging module.

Helper function used by all the user-output convenience functions.

pywikibot.bot.**open_webbrowser**(*page*)
Open the web browser displaying the page and wait for input.

pywikibot.bot.**output**(*text*, *decoder=None*, *newline=True*, *toStdout=False*, ***kwargs*)
Output a message to the user via the userinterface.

Works like print, but uses the encoding used by the user's console (console_encoding in the configuration file) instead of ASCII.

If decoder is None, text should be a unicode string. Otherwise it should be encoded in the given encoding.

If newline is True, a line feed will be added after printing the text.

If toStdout is True, the text will be sent to standard output, so that it can be piped to another process. All other text will be sent to stderr. See: https://en.wikipedia.org/wiki/Pipeline_%28Unix%29

text can contain special sequences to create colored output. These consist of the escape character 03 and the color name in curly braces, e. g. 03{lightpurple}. 03{default} resets the color.

Other keyword arguments are passed unchanged to the logger; so far, the only argument that is useful is "exc_info=True", which causes the log message to include an exception traceback.

pywikibot.bot.**showHelp**(*module_name=None*)
Show help for the Bot.

pywikibot.bot.**stdout**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output script results to the user via the userinterface.

pywikibot.bot.**warning**(*text*, *decoder=None*, *newline=True*, ***kwargs*)
Output a warning message to the user via the userinterface.

pywikibot.bot.**writeToCommandLogFile**()
Save name of the called module along with all parameters to logs/commands.log.

This can be used by user later to track errors or report bugs.

pywikibot.bot.**writelogheader**()
Save additional version, system and status info to the log file in use.

This may help the user to track errors or report bugs.

botirc Module

User-interface related functions for building bots.

Note: the script requires the Python IRC library http://python-irclib.sourceforge.net/

class pywikibot.botirc.**IRCBot**(*site*, *channel*, *nickname*, *server*, *port=6667*, ***kwargs*)
Bases: *pywikibot.bot.Bot*, *pywikibot.botirc.SingleServerIRCBot*

A generic IRC Bot to be subclassed.

A Bot that displays the ordinal number of the new articles being created visible on the Recent Changes list. The Bot doesn't make any edits, no account needed.

availableOptions = {}

do_command(*e*, *cmd*)

on_dccchat(*c*, *e*)

on_dccmsg(*c*, *e*)

on_nicknameinuse(*c*, *e*)

on_privmsg(*c*, *e*)

on_pubmsg(*c*, *e*)

on_quit(*e*, *cmd*)

on_welcome(*c*, *e*)

class pywikibot.botirc.**SingleServerIRCBot**(**args*, ***kwargs*)
Bases: object

Fake SingleServerIRCBot.

config2 Module

Module to define and load pywikibot configuration default and user preferences.

User preferences are loaded from a python file called user-config.py, which may be located in directory specified by the environment variable PYWIKIBOT2_DIR, or the same directory as pwb.py, or in a directory within the users home. See get_base_dir for more information.

If user-config.py can not be found in any of those locations, this module will fail to load unless the environment variable PYWIKIBOT2_NO_USER_CONFIG is set to a value other than '0'. i.e. PYWIKIBOT2_NO_USER_CONFIG=1 will allow config to load without a user-config.py. However, warnings will be shown if user-config.py was not loaded. To prevent these warnings, set PYWIKIBOT2_NO_USER_CONFIG=2.

Provides two functions to register family classes which can be used in the user-config:

```
- register_family_file
- register_families_folder
```

Other functions made available to user-config:

```
- user_home_path
```

Sets module global base_dir and provides utility methods to build paths relative to base_dir:

```
- makepath
- datafilepath
- shortpath
```

pywikibot.config2.**datafilepath**(**filename*)
Return an absolute path to a data file in a standard location.

Argument(s) are zero or more directory names, optionally followed by a data file name. The return path is offset to config.base_dir. Any directories in the path that do not already exist are created.

pywikibot.config2.**get_base_dir**(*test_directory=None*)
Return the directory in which user-specific information is stored.

This is determined in the following order::

1. If the script was called with a -dir: argument, use the directory provided in this argument.
2. If the user has a PYWIKIBOT2_DIR environment variable, use the value of it.
3. If user-config is present in current directory, use the current directory.
4. If user-config is present in pwb.py directory, use that directory
5. Use (and if necessary create) a 'pywikibot' folder under 'Application Data' or 'AppDataRoaming' (Windows) or '.pywikibot' directory (Unix and similar) under the user's home directory.

Set PYWIKIBOT2_NO_USER_CONFIG=1 to disable loading user-config.py

Parameters **test_directory** (*str or None*) – Assume that a user config file exists in this directory. Used to test whether placing a user config file in this directory will cause it to be selected as the base directory.

Return type unicode

pywikibot.config2.**makepath**(*path*)
Return a normalized absolute version of the path argument.

- if the given path already exists in the filesystem the filesystem is not modified.
- otherwise makepath creates directories along the given path using the dirname() of the path. You may append a '/' to the path if you want it to be a directory path.

from holger@trillke.net 2002/03/18

pywikibot.config2.**register_families_folder**(*folder_path*)
Register all family class files contained in a directory.

pywikibot.config2.**register_family_file**(*family_name*, *file_path*)
Register a single family class file.

pywikibot.config2.**shortpath**(*path*)
Return a file path relative to config.base_dir.

pywikibot.config2.**user_home_path**(*path*)
Return a file path to a file in the user home.

date Module

Date data and manipulation module.

class pywikibot.date.**FormatDate**(*site*)
Bases: object

Format a date.

pywikibot.date.**MakeParameter**(*decoder*, *param*)

pywikibot.date.**addFmt1**(*lang*, *isMnthOfYear*, *patterns*)
Add 12 month formats for a specific type ('January','Feb..), for a given language.

The function must accept one parameter for the ->int or ->string conversions, just like everywhere else in the formats map. The patterns parameter is a list of 12 elements to be used for each month.

pywikibot.date.**addFmt2**(*lang*, *isMnthOfYear*, *pattern*, *makeUpperCase=None*)

pywikibot.date.**alwaysTrue**(*x*)
Return True, always.

It is used for multiple value selection function to accept all other values.

Parameters **x** – not used

Returns True

Return type bool

pywikibot.date.**apply_month_delta**(*date*, *month_delta=1*, *add_overlap=False*)
Add or subtract months from the date.

By default if the new month has less days then the day of the date it chooses the last day in the new month. For example a date in the March 31st added by one month will result in April 30th.

When the overlap is enabled, and there is overlap, then the new_date will be one month off and get_month_delta will report a number one higher.

It does only work on calendars with 12 months per year, and where the months are numbered consecutively beginning by 1.

Parameters

- **date** (date) – The starting date
- **month_delta** (*int*) – The amount of months added or subtracted.
- **add_overlap** (*bool*) – Add any missing days to the date, increasing the month once more.

Returns The end date

Return type type of date

pywikibot.date.**decSinglVal**(*v*)

pywikibot.date.**dh**(*value*, *pattern*, *encf*, *decf*, *filter=None*)
Function to help with year parsing.

Usually it will be used as a lambda call in a map::

```
lambda v: dh(v, u'pattern string', encf, decf)
```

Parameters

- **encf** – Converts from an integer parameter to another integer or a tuple of integers. Depending on the pattern, each integer will be converted to a proper string representation, and will be passed as a format argument to the pattern::

```
pattern % encf(value)
```

  This function is a complement of decf.

- **decf** – Converts a tuple/list of non-negative integers found in the original value string into a normalized value. The normalized value can be passed right back into dh() to produce the original string. This function is a complement of encf. dh() interprets %d as a decimal and %s as a roman numeral number.

pywikibot.date.**dh_centuryAD**(*value*, *pattern*)

pywikibot.date.**dh_centuryBC**(*value*, *pattern*)

pywikibot.date.**dh_constVal**(*value*, *ind*, *match*)
Helper function to match a single value to a constant.

formats['CurrEvents']['en'](ind) => u'Current Events' formats['CurrEvents']['en'](u'Current Events') => ind

pywikibot.date.**dh_dayOfMnth**(*value*, *pattern*)
Helper for decoding a single integer value.

The single integer should be <=31, no conversion, no rounding (used in days of month).

pywikibot.date.**dh_decAD**(*value*, *pattern*)
Helper for decoding a single integer value.

It should be no conversion, round to decimals (used in decades)

pywikibot.date.**dh_decBC**(*value*, *pattern*)
Helper for decoding a single integer value.

It should be no conversion, round to decimals (used in decades)

pywikibot.date.**dh_millenniumAD**(*value*, *pattern*)

pywikibot.date.**dh_millenniumBC**(*value*, *pattern*)

pywikibot.date.**dh_mnthOfYear**(*value*, *pattern*)
Helper for decoding a single integer value.

The value should be >=1000, no conversion, no rounding (used in month of the year)

pywikibot.date.**dh_noConv**(*value*, *pattern*, *limit*)
Helper for decoding a single integer value, no conversion, no rounding.

pywikibot.date.**dh_number**(*value*, *pattern*)

pywikibot.date.**dh_simpleYearAD**(*value*)
Helper for decoding a single integer value.

This value should be representing a year with no extra symbols.

pywikibot.date.**dh_singVal**(*value*, *match*)

pywikibot.date.**dh_yearAD**(*value*, *pattern*)
Helper for decoding a year value.

The value should have no conversion, no rounding, limits to 3000.

pywikibot.date.**dh_yearBC**(*value*, *pattern*)
Helper for decoding a year value.

The value should have no conversion, no rounding, limits to 3000.

pywikibot.date.**encDec0**(*i*)

pywikibot.date.**encDec1**(*i*)

pywikibot.date.**encNoConv**(*i*)

pywikibot.date.**escapePattern2**(*pattern*)
Convert a string pattern into a regex expression and cache.

Allows matching of any _digitDecoders inside the string. Returns a compiled regex object and a list of digit decoders.

pywikibot.date.**formatYear**(*lang*, *year*)

pywikibot.date.**getAutoFormat**(*lang*, *title*, *ignoreFirstLetterCase=True*)
Return first matching formatted date value.

Parameters

- **lang** – language code
- **title** – value to format

Returns dictName ('YearBC', 'December', ...) and value (a year, date, ...)

Return type tuple

pywikibot.date.**getNumberOfDaysInMonth**(*month*)
Return the number of days in a given month, 1 being January, etc.

pywikibot.date.**get_month_delta**(*date1*, *date2*)
Return the difference between to dates in months.

It does only work on calendars with 12 months per year, and where the months are consecutive and non-negative numbers.

pywikibot.date.**intToLocalDigitsStr**(*value*, *digitsToLocalDict*)

pywikibot.date.**intToRomanNum**(*i*)

pywikibot.date.**localDigitsStrToInt**(*value*, *digitsToLocalDict*, *localToDigitsDict*)

pywikibot.date.**makeMonthList**(*pattern*)

pywikibot.date.**makeMonthNamedList**(*lang*, *pattern*, *makeUpperCase=None*)
Create a list of 12 elements based on the name of the month.

The language-dependent month name is used as a formating argument to the pattern. The pattern must be have one %s that will be replaced by the localized month name. Use %%d for any other parameters that should be preserved.

pywikibot.date.**monthName**(*lang*, *ind*)

pywikibot.date.**multi**(*value*, *tuplst*)
Run multiple pattern checks for the same entry.

For example: 1st century, 2nd century, etc.

The tuplst is a list of tupples. Each tupple must contain two functions:: first to encode/decode a single value (e.g. simpleInt), second is a predicate function with an integer parameter that returns true or false. When the 2nd function evaluates to true, the 1st function is used.

pywikibot.date.**romanNumToInt**(*v*)

pywikibot.date.**slh**(*value*, *lst*)
Helper function for simple list value matching.

!!!!! The index starts at 1, so 1st element has index 1, not 0 !!!!!

Usually it will be used as a lambda call in a map::

```
lambda v: slh(v, [u'January',u'February',...])
```

Usage scenarios::

```
formats['MonthName']['en'](1) => u'January'
formats['MonthName']['en'](u'January') => 1
formats['MonthName']['en'](u'anything else') => raise ValueError
```

diff Module

Diff module.

class pywikibot.diff.**Hunk**(*a*, *b*, *grouped_opcode*)
Bases: object

One change hunk between a and b.

Note: parts of this code are taken from by difflib.get_grouped_opcodes().

APPR = 1

NOT_APPR = -1

PENDING = 0

apply()
Turn a into b for this hunk.

color_line(*line*, *line_ref=None*)
Color line characters.

If line_ref is None, the whole line is colored. If line_ref[i] is not blank, line[i] is colored. Color depends if line starts with +/-.

line: string line_ref: string.

create_diff()
Generator of diff text for this hunk, without formatting.

format_diff()
Color diff lines.

get_header()
Provide header of unified diff.

static get_header_text(*a_rng*, *b_rng*, *affix='@@'*)
Provide header for any ranges.

class pywikibot.diff.**PatchManager**(*text_a*, *text_b*, *context=0*, *by_letter=False*, *replace_invisible=False*)
Bases: object

Apply patches to text_a to obtain a new text.

If all hunks are approved, text_b will be obtained.

apply()
Apply changes. If there are undecided changes, ask to review.

get_blocks()
Return list with blocks of indexes which compose a and, where applicable, b.

Format of each block::

```
[-1, (i1, i2), (-1, -1)] -> block a[i1:i2] does not change from a to b
    then is there is no corresponding hunk.
[hunk index, (i1, i2), (j1, j2)] -> block a[i1:i2] becomes b[j1:j2]
```

print_hunks()
Print the headers and diff texts of all hunks to the output.

review_hunks()
Review hunks.

pywikibot.diff.**cherry_pick**(*oldtext*, *newtext*, *n=0*, *by_letter=False*)
Propose a list of changes for approval.

Text with approved changes will be returned. n: int, line of context as defined in difflib.get_grouped_opcodes(). by_letter: if text_a and text_b are single lines, comparison can be done

pywikibot.diff.**html_comparator**(*compare_string*)
List of added and deleted contexts from 'action=compare' html string.

This function is useful when combineds with site.py's "compare" method. Site.compare() returns HTML that is useful for displaying on a page. Here we use BeautifulSoup to get the un-HTML-ify the context of changes. Finally we present the added and deleted contexts. :param compare_string: HTML string from mediawiki API :type compare_string: str :return: deleted and added list of contexts :rtype: dict

echo Module

Classes and functions for working with the Echo extension.

class pywikibot.echo.**Notification**(*site*)
Bases: object

A notification issued by the Echo extension.

classmethod fromJSON(*site*, *data*)
Construct a Notification object from JSON data returned by the API.

Return type *Notification*

mark_as_read()
Mark the notification as read.

editor Module

Text editor class for your favourite editor.

class pywikibot.editor.**TextEditor**
Bases: object

Text editor.

command(*tempFilename*, *text*, *jumpIndex=None*)
Return editor selected in user-config.py.

edit(*text*, *jumpIndex=None*, *highlight=None*)
Call the editor and thus allows the user to change the text.

Halts the thread's operation until the editor is closed.

Parameters

- **text** (*unicode*) – the text to be edited
- **jumpIndex** (*int*) – position at which to put the caret
- **highlight** (*unicode*) – each occurrence of this substring will be highlighted

Returns the modified text, or None if the user didn't save the text file in his text editor

Return type unicode or None

exceptions Module

Exception and warning classes used throughout the framework.

Error: Base class, all exceptions should the subclass of this class.

- NoUsername: Username is not in user-config.py, or it is invalid.
- UserBlocked: Username or IP has been blocked

- AutoblockUser: requested action on a virtual autoblock user not valid
- UserRightsError: insufficient rights for requested action
- BadTitle: Server responded with BadTitle
- InvalidTitle: Invalid page title
- CaptchaError: Captcha is asked and config.solve_captcha == False
- Server504Error: Server timed out with HTTP 504 code
- PageNotFound: Page not found (deprecated)
- i18n.TranslationError: i18n/l10n message not available
- UnknownExtension: Extension is not defined for this site

SiteDefinitionError: Site loading problem

- UnknownSite: Site does not exist in Family
- UnknownFamily: Family is not registered

PageRelatedError: any exception which is caused by an operation on a Page.

- NoPage: Page does not exist
- IsRedirectPage: Page is a redirect page
- IsNotRedirectPage: Page is not a redirect page
- CircularRedirect: Page is a circular redirect
- InterwikiRedirectPage: Page is a redirect to another site
- SectionError: The section specified by # does not exist
- NotEmailableError: The target user has disabled email

PageSaveRelatedError: page exceptions within the save operation on a Page (alias: PageNotSaved).

- SpamfilterError: MediaWiki spam filter detected a blacklisted URL
- OtherPageSaveError: misc. other save related exception.
- **LockedPage: Page is locked**
  - LockedNoPage: Title is locked against creation
  - CascadeLockedPage: Page is locked due to cascading protection
- **EditConflict: Edit conflict while uploading the page**
  - PageDeletedConflict: Page was deleted since being retrieved
  - PageCreatedConflict: Page was created by another user
  - ArticleExistsConflict: Page article already exists
- NoCreateError: parameter nocreate not allow page creation

ServerError: a problem with the server.

- FatalServerError: A fatal/non-recoverable server error

WikiBaseError: any issue specific to Wikibase.

- CoordinateGlobeUnknownException: globe is not implemented yet.
- EntityTypeUnknownException: entity type is not available on the site.

DeprecationWarning: old functionality replaced by new functionality

PendingDeprecationWarning: problematic code which has not yet been fully deprecated, possibly because a replacement is not available

RuntimeWarning: problems developers should have fixed, and users need to

be aware of its status.

- tools._NotImplementedWarning: do not use
- NotImplementedWarning: functionality not implemented

UserWarning: warnings targetted at users

- config2._ConfigurationDeprecationWarning: user configuration file problems
- login._PasswordFileWarning: password file problems
- ArgumentDeprecationWarning: command line argument problems
- FamilyMaintenanceWarning: missing information in family definition

`family` Module

Objects representing MediaWiki families.

class pywikibot.family.**AutoFamily**(*name*, *url*, *site=None*)
Bases: *pywikibot.family.Family*

Family that automatically loads the site configuration.

protocol(*code*)
Return the protocol of the URL.

scriptpath(*code*)
Extract the script path from the URL.

class pywikibot.family.**Family**
Bases: object

Parent class for all wiki families.

apipath(*code*)

base_url(*code*, *uri*)

category_redirects(*code*, *fallback='_default'*)

code2encoding(*code*)
Return the encoding for a specific language wiki.

code2encodings(*code*)
Return list of historical encodings for a specific language Wiki.

dbName(*code*)

disambig(*code*, *fallback='_default'*)

encoding(*code*)
Return the encoding for a specific language Wiki.

encodings(*code*)
Return list of historical encodings for a specific language Wiki.

force_version(*code*)
Return a manual version number.

The site is usually using the version number from the servers' siteinfo, but if there is a problem with that it's possible to return a non-empty string here representing another version number.

For example, *pywikibot.tools.MediaWikiVersion* treats version numbers ending with 'alpha', 'beta' or 'rc' as newer than any version ending with 'wmf<number>'. But if that causes breakage it's possible to override it here to a version number which doesn't cause breakage.

Returns A version number which can be parsed using *pywikibot.tools.MediaWikiVersion*. If empty/None it uses the version returned via siteinfo.

Return type str

from_url(*url*)
Return whether this family matches the given url.

It must match URLs generated via `self.langs` and *Family.nice_get_address* or *Family.path*. If the protocol doesn't match but is present in the interwikimap it'll log this.

It ignores $1 in the url, and anything that follows it.

Returns The language code of the url. None if that url is not from this family.

Return type str or None

Raises RuntimeError Mismatch between Family langs dictionary and URL regex.

get_address(*code*, *title*)

get_cr_templates(*code*, *fallback*)

get_known_families(*site*)

hostname(*code*)
The hostname to use for standard http connections.

ignore_certificate_error(*code*)
Return whether a HTTPS certificate error should be ignored.

Parameters **code** (*string*) – language code

Returns flag to allow access if certificate has an error.

Return type bool

isPublic(*code*)
Check the wiki require logging in before viewing it.

iwkeys

linktrail(*code*, *fallback='_default'*)
Return regex for trailing chars displayed as part of a link.

Returns a string, not a compiled regular expression object.

This reads from the family file, and ''not" from [[MediaWiki:Linktrail]], because the MW software currently uses a built-in linktrail from its message files and ignores the wiki value.

static **load**(*fam=None*)
Import the named family.

Parameters **fam** (*str*) – family name (if omitted, uses the configured default)

Returns a Family instance configured for the named family.

Raises UnknownFamily family not known

maximum_GET_length(*code*)

nice_get_address(*code*, *title*)

nicepath(*code*)

obsolete
Old codes that are not part of the family.

Interwiki replacements override removals for the same code.

Returns mapping of old codes to new codes (or None)

Return type dict

path(*code*)

post_get_convert(*site*, *getText*)
Do a conversion on the retrieved text from the Wiki.

For example a X-conversion in Esperanto https://en.wikipedia.org/wiki/Esperanto_orthography#X-system.

pre_put_convert(*site*, *putText*)
Do a conversion on the text to insert on the Wiki.

For example a X-conversion in Esperanto https://en.wikipedia.org/wiki/Esperanto_orthography#X-system.

protocol(*code*)
The protocol to use to connect to the site.

May be overridden to return 'https'. Other protocols are not supported.

Parameters **code** (*string*) – language code

Returns protocol that this family uses

Return type string

querypath(*code*)

rcstream_host(*code*)

scriptpath(*code*)
The prefix used to locate scripts on this wiki.

This is the value displayed when you enter {{SCRIPTPATH}} on a wiki page (often displayed at [[Help:Variables]] if the wiki has copied the master help page correctly).

The default value is the one used on Wikimedia Foundation wikis, but needs to be overridden in the family file for any wiki that uses a different value.

server_time(*code*)
DEPRECATED, use Site.getcurrenttime() instead.

Return a datetime object representing server time.

shared_data_repository(*code*, *transcluded=False*)
Return the shared Wikibase repository, if any.

shared_image_repository(*code*)
Return the shared image repository, if any.

ssl_hostname(*code*)
The hostname to use for SSL connections.

ssl_pathprefix(*code*)
The path prefix for secure HTTP access.

version(*code*)
Return MediaWiki version number as a string.

Use *pywikibot.tools.MediaWikiVersion* to compare version strings.

versionnumber(*code*)
DEPRECATED, use version() instead.

Use *pywikibot.tools.MediaWikiVersion* to compare version strings. Return an int identifying MediaWiki version.

Currently this is implemented as returning the minor version number; i.e., 'X' in version '1.X.Y'

class pywikibot.family.**WikimediaFamily**
Bases: *pywikibot.family.Family*

Class for all wikimedia families.

closed_wikis = []

code_aliases = {'zh-tw': 'zh', 'dk': 'da', 'jp': 'ja', 'nan': 'zh-min-nan', 'nb': 'no', 'zh-cn': 'zh', 'nl_nds': 'nl-nds', 'n

interwiki_removals

interwiki_replacement_overrides = {'mo': 'ro'}

interwiki_replacements

protocol(*code*)
Return 'https' as the protocol.

rcstream_host(*code*)

removed_wikis = []

shared_image_repository(*code*)

fixes Module

File containing all standard fixes. Currently available predefined fixes are:

```
* HTML         - Convert HTML tags to wiki syntax, and
```

fix XHTML.

- isbn - Fix badly formatted ISBNs.
- **syntax - Try to fix bad wiki markup. Do not run** this in automatic mode, as the bot may make mistakes.
- **syntax-safe - Like syntax, but less risky, so you can** run this in automatic mode.
- case-de - fix upper/lower case errors in German
- grammar-de - fix grammar and typography in German
- **vonbis - Ersetze Binde-/Gedankenstrich durch "bis"** in German
- music - Links auf Begriffsklärungen in German
- datum - specific date formats in German

- **correct-ar - Corrections for Arabic Wikipedia and any** Arabic wiki.
- **yu-tld - the yu top-level domain will soon be** disabled, see
- **fckeditor - Try to convert FCKeditor HTML tags to wiki** syntax. https://lists.wikimedia.org/pipermail/wikibots-l/2009-February/000290.html

i18n Module

Various i18n functions.

Helper functions for both the internal translation system and for TranslateWiki-based translations.

By default messages are assumed to reside in a package called 'scripts.i18n'. In pywikibot 2.0, that package is not packaged with pywikibot, and pywikibot 2.0 does not have a hard dependency on any i18n messages. However, there are three user input questions in pagegenerators which will use i18 messages if they can be loaded.

The default message location may be changed by calling `set_message_package` with a package name. The package must contain an __init__.py, and a message bundle called 'pywikibot' containing messages. See *twntranslate* for more information on the messages.

exception pywikibot.i18n.**TranslationError**(*arg*)
Bases: pywikibot.exceptions.Error, ImportError

Raised when no correct translation could be found.

pywikibot.i18n.**input**(*twtitle*, *parameters=None*, *password=False*, *fallback_prompt=None*)
Ask the user a question, return the user's answer.

The prompt message is retrieved via *twtranslate* and either uses the config variable 'userinterface_lang' or the default locale as the language code.

Parameters

- **twtitle** – The TranslateWiki string title, in <package>-<key> format
- **parameters** – The values which will be applied to the translated text
- **password** – Hides the user's input (for password entry)
- **fallback_prompt** – The English prompt if i18n is not available.

Return type unicode string

pywikibot.i18n.**messages_available**()
Return False if there are no i18n messages available.

To determine if messages are available, it looks for the package name set using *set_messages_package* for a message bundle called 'pywikibot' containing messages.

Return type bool

pywikibot.i18n.**set_messages_package**(*package_name*)
Set the package name where i18n messages are located.

pywikibot.i18n.**translate**(*code*, *xdict*, *parameters=None*, *fallback=False*)
Return the most appropriate translation from a translation dict.

Given a language code and a dictionary, returns the dictionary's value for key 'code' if this key exists; otherwise tries to return a value for an alternative language that is most applicable to use on the wiki in language 'code' except fallback is False.

The language itself is always checked first, then languages that have been defined to be alternatives, and finally English. If none of the options gives result, we just take the one language from xdict which may not be always the same. When fallback is iterable it'll return None if no code applies (instead of returning one).

For PLURAL support have a look at the twntranslate method

Parameters

- **code** (*string or Site object*) – The language code
- **xdict** (*dict, string, unicode*) – dictionary with language codes as keys or extended dictionary with family names as keys containing language dictionaries or a single (unicode) string. May contain PLURAL tags as described in twntranslate
- **parameters** (*dict, string, unicode, int*) – For passing (plural) parameters
- **fallback** (*boolean or iterable*) – Try an alternate language code. If it's iterable it'll also try those entries and choose the first match.

pywikibot.i18n.**twget_keys**(*twtitle*)

Return all language codes for a special message.

Parameters **twtitle** – The TranslateWiki string title, in <package>-<key> format

pywikibot.i18n.**twhas_key**(*code*, *twtitle*)

Check if a message has a translation in the specified language code.

The translations are retrieved from i18n.<package>, based on the callers import table.

No code fallback is made.

Parameters

- **code** – The language code
- **twtitle** – The TranslateWiki string title, in <package>-<key> format

pywikibot.i18n.**twntranslate**(*code*, *twtitle*, *parameters=None*)

Translate a message with plural support.

Support is implemented like in MediaWiki extension. If the TranslateWiki message contains a plural tag inside which looks like::

```
{{PLURAL:<number>|<variant1>|<variant2>[|<variantn>]}}
```

it takes that variant calculated by the plural_rules depending on the number value. Multiple plurals are allowed.

As an examples, if we had several json dictionaries in test folder like:

en.json:

```
{
    "test-plural": "Bot: Changing %(num)s {{PLURAL:%(num)d|page|pages}}.",
}
```

fr.json:

```
{
    "test-plural": "Robot: Changer %(descr)s {{PLURAL:num|une page|quelques pages}}.",
}
```

and so on.

```
>>> from pywikibot import i18n
>>> i18n.set_messages_package('tests.i18n')
>>> # use a number
>>> str(i18n.twntranslate('en', 'test-plural', 0) % {'num': 'no'})
'Bot: Changing no pages.'
>>> # use a string
>>> str(i18n.twntranslate('en', 'test-plural', '1') % {'num': 'one'})
'Bot: Changing one page.'
>>> # use a dictionary
>>> str(i18n.twntranslate('en', 'test-plural', {'num':2}))
'Bot: Changing 2 pages.'
>>> # use additional format strings
>>> str(i18n.twntranslate('fr', 'test-plural', {'num': 1, 'descr': 'seulement'}))
'Robot: Changer seulement une page.'
>>> # use format strings also outside
>>> str(i18n.twntranslate('fr', 'test-plural', 10) % {'descr': 'seulement'})
'Robot: Changer seulement quelques pages.'
```

The translations are retrieved from i18n.<package>, based on the callers import table.

Parameters

- **code** – The language code
- **twtitle** – The TranslateWiki string title, in <package>-<key> format
- **parameters** – For passing (plural) parameters.

pywikibot.i18n.**twtranslate**(*code*, *twtitle*, *parameters=None*, *fallback=True*)

Translate a message.

The translations are retrieved from json files in messages_package_name.

fallback parameter must be True for i18n and False for L10N or testing purposes.

Parameters

- **code** – The language code
- **twtitle** – The TranslateWiki string title, in <package>-<key> format
- **parameters** – For passing parameters.
- **fallback** (*boolean*) – Try an alternate language code

interwiki_graph Module

Module with the Graphviz drawing calls.

class pywikibot.interwiki_graph.**GraphDrawer**(*subject*)

Bases: object

Graphviz (dot) code creator.

addDirectedEdge(*page*, *refPage*)

Add a directed edge from refPage to page.

addNode(*page*)

Add a node for page.

createGraph()

Create graph of the interwiki links.

For more info see http://meta.wikimedia.org/wiki/Interwiki_graphs

getLabel(*page*)
Get label for page.

saveGraphFile()
Write graphs to the data directory.

exception pywikibot.interwiki_graph.**GraphImpossible**
Bases: Exception

Drawing a graph is not possible on your system.

class pywikibot.interwiki_graph.**GraphSavingThread**(*graph*, *originPage*)
Bases: threading.Thread

Threaded graph renderer.

Rendering a graph can take extremely long. We use multithreading because of that.

TODO: Find out if several threads running in parallel can slow down the system too much. Consider adding a mechanism to kill a thread if it takes too long.

run()
Write graphs to the data directory.

class pywikibot.interwiki_graph.**Subject**(*origin=None*)
Bases: object

Data about a page with translations on multiple wikis.

foundIn
Mapping of pages to others pages interwiki linked to it.

DEPRECATED. Use found_in.

origin
Page on the origin wiki.

originPage
Deprecated property for the origin page.

DEPRECATED. Use origin.

pywikibot.interwiki_graph.**getFilename**(*page*, *extension=None*)
Create a filename that is unique for the page.

Parameters

- **page** (Page) – page used to create the new filename
- **extension** (*str*) – file extension

Returns filename of <family>-<lang>-<page>.<ext>

Return type str

logentries Module

Objects representing Mediawiki log entries.

class pywikibot.logentries.**BlockEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Block log entry.

duration()
Return a datetime.timedelta representing the block duration.

Returns datetime.timedelta, or None if block is indefinite.

expiry()
Return a Timestamp representing the block expiry date.

Return type pywikibot.Timestamp or None

flags()
Return a list of (str) flags associated with the block entry.

It raises an Error if the entry is an unblocking log entry.

Return type list of flag strings

title()
Return the blocked account or IP.

Returns the Page object of username or IP if this block action targets a username or IP, or the blockid if this log reflects the removal of an autoblock

Return type Page or int

class pywikibot.logentries.**DeleteEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Deletion log entry.

class pywikibot.logentries.**ImportEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Import log entry.

class pywikibot.logentries.**LogDict**
Bases: dict

Simple custom dict that raises a custom KeyError when a key is missing.

It also logs debugging information when a key is missing.

class pywikibot.logentries.**LogEntry**(*apidata*, *site*)
Bases: object

Generic log entry.

action()

comment()

logid()

ns()

pageid()

timestamp()
Timestamp object corresponding to event timestamp.

title()
Page on which action was performed.

Note: title may be missing in data dict e.g. by oversight action to hide the title. In that case a KeyError exception will raise

type()

user()

class pywikibot.logentries.**LogEntryFactory**(*site*, *logtype=None*)
Bases: object

LogEntry Factory.

Only available method is create()

create(*logdata*)
Instantiate the LogEntry object representing logdata.

Parameters **logdata** (*dict*) – <item> returned by the api

Returns LogEntry object representing logdata

class pywikibot.logentries.**MoveEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Move log entry.

new_ns()
Return namespace id of target page.

new_title()
Return page object of the new title.

suppressedredirect()
Return True if no redirect was created during the move.

Return type bool

target_ns
Return namespace object of target page.

target_page
Return target page object.

target_title
Return the target title.

class pywikibot.logentries.**NewUsersEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

New user log entry.

class pywikibot.logentries.**PatrolEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Patrol log entry.

auto
Return auto patrolled.

current_id
Return the current id.

previous_id
Return the previous id.

class pywikibot.logentries.**ProtectEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Protection log entry.

class pywikibot.logentries.**RightsEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Rights log entry.

newgroups
Return new rights groups.

oldgroups
Return old rights groups.

class pywikibot.logentries.**UploadEntry**(*apidata*, *site*)
Bases: *pywikibot.logentries.LogEntry*

Upload log entry.

login Module

Library to log the bot in to a wiki account.

class pywikibot.login.**LoginManager**(*password=None*, *sysop=False*, *site=None*, *user=None*)
Bases: object

Site login manager.

botAllowed()
Check whether the bot is listed on a specific page.

This allows bots to comply with the policy on the respective wiki.

getCookie(*remember=True*, *captcha=None*)
Login to the site.

remember Remember login (default: True) captchaId A dictionary containing the captcha id and answer, if any

Returns cookie data if successful, None otherwise.

login(*retry=False*)
Attempt to log into the server.

Parameters **retry** (*bool*) – infinitely retry if the API returns an unknown error

Raises **NoUsername** Username is not recognised by the site.

readPassword()
Read passwords from a file.

DO NOT FORGET TO REMOVE READ ACCESS FOR OTHER USERS!!! Use chmod 600 password-file.

All lines below should be valid Python tuples in the form (code, family, username, password), (family, username, password) or (username, password) to set a default password for an username. The last matching entry will be used, so default usernames should occur above specific usernames.

The file must be either encoded in ASCII or UTF-8.

Example:

```
(u"my_username", u"my_default_password")
(u"my_sysop_user", u"my_sysop_password")
(u"wikipedia", u"my_wikipedia_user", u"my_wikipedia_pass")
(u"en", u"wikipedia", u"my_en_wikipedia_user", u"my_en_wikipedia_pass")
```

showCaptchaWindow(*url*)
Open a window to show the captcha for the given URL.

storecookiedata(*data*)
Store cookie data.

The argument data is the raw data, as returned by getCookie().

Returns nothing.

page Module

Objects representing various types of MediaWiki, including Wikibase, pages.

This module also includes objects:: * Link: an internal or interwiki link in wikitext. * Revision: a single change to a wiki page. * Property: a type of semantic data. * Claim: an instance of a semantic assertion.

pagegenerators Module

This module offers a wide variety of page generators.

A page generator is an object that is iterable (see http://legacy.python.org/dev/peps/pep-0255/) and that yields page objects on which other scripts can then work.

Pagegenerators.py cannot be run as script. For testing purposes listpages.py can be used instead, to print page titles to standard output.

These parameters are supported to specify which pages titles to print:

```
-cat              Work on all pages which are in a specific category.
                  Argument can also be given as "-cat:categoryname" or
                  as "-cat:categoryname|fromtitle" (using # instead of |
                  is also allowed in this one and the following)

-catr             Like -cat, but also recursively includes pages in
                  subcategories, sub-subcategories etc. of the
                  given category.
                  Argument can also be given as "-catr:categoryname" or
                  as "-catr:categoryname|fromtitle".

-subcats          Work on all subcategories of a specific category.
                  Argument can also be given as "-subcats:categoryname" or
                  as "-subcats:categoryname|fromtitle".

-subcatsr         Like -subcats, but also includes sub-subcategories etc. of
                  the given category.
                  Argument can also be given as "-subcatsr:categoryname" or
                  as "-subcatsr:categoryname|fromtitle".

-uncat            Work on all pages which are not categorised.

-uncatcat         Work on all categories which are not categorised.

-uncatfiles       Work on all files which are not categorised.

-file             Read a list of pages to treat from the named text file.
                  Page titles in the file may be either enclosed with
                  [[brackets]], or be separated by new lines.
```

```
                  Argument can also be given as "-file:filename".

-filelinks        Work on all pages that use a certain image/media file.
                  Argument can also be given as "-filelinks:filename".

-search           Work on all pages that are found in a MediaWiki search
                  across all namespaces.

-logevents        Work on articles that were on a specified Special:Log.
                 The value may be a comma separated list of three values::
                      logevent,username,total
                  To use the default value, use an empty string.
                  You have options for every type of logs given by the
                 log event parameter which could be one of the following::
                      block, protect, rights, delete, upload, move, import,
                      patrol, merge, suppress, review, stable, gblblock,
                      renameuser, globalauth, gblrights, abusefilter, newusers
                  It uses the default number of pages 10.
                 Examples::
                  -logevents:move gives pages from move log (usually redirects)
                  -logevents:delete,,20 gives 20 pages from deletion log
                  -logevents:protect,Usr gives pages from protect by user Usr
                  -logevents:patrol,Usr,20 gives 20 patroled pages by user Usr
                  In some cases it must be written as -logevents:"patrol,Usr,20"

-namespaces       Filter the page generator to only yield pages in the
-namespace        specified namespaces. Separate multiple namespace
-ns               numbers or names with commas.
                 Examples::
                  -ns:0,2,4
                  -ns:Help,MediaWiki
                  If used with -newpages, -namepace/ns must be provided
                  before -newpages.
                  If used with -recentchanges, efficiency is improved if
                  -namepace/ns is provided before -recentchanges.
                  If used with -titleregex, -namepace/ns must be provided
                  before -titleregex and shall contain only one value.

-interwiki        Work on the given page and all equivalent pages in other
                  languages. This can, for example, be used to fight
                  multi-site spamming.
                  Attention: this will cause the bot to modify
                  pages on several wiki sites, this is not well tested,
                  so check your edits!

-limit:n          When used with any other argument that specifies a set
                  of pages, work on no more than n pages in total.

-links            Work on all pages that are linked from a certain page.
                  Argument can also be given as "-links:linkingpagetitle".

-liverecentchanges Work on pages from the live recent changes feed. If used as
                  -liverecentchanges:x, work on x recent changes.

-imagesused       Work on all images that contained on a certain page.
                  Argument can also be given as "-imagesused:linkingpagetitle".

-newimages        Work on the 100 newest images. If given as -newimages:x,
```

```
                  will work on the x newest images.

-newpages         Work on the most recent new pages. If given as -newpages:x,
                  will work on the x newest pages.

-recentchanges    Work on the pages with the most recent changes. If
                  given as -recentchanges:x, will work on the x most recently
                  changed pages.

-ref              Work on all pages that link to a certain page.
                  Argument can also be given as "-ref:referredpagetitle".

-start            Specifies that the robot should go alphabetically through
                  all pages on the home wiki, starting at the named page.
                  Argument can also be given as "-start:pagetitle".

                  You can also include a namespace. For example,
                  "-start:Template:!" will make the bot work on all pages
                  in the template namespace.

-prefixindex      Work on pages commencing with a common prefix.

-step:n           When used with any other argument that specifies a set
                  of pages, only retrieve n pages at a time from the wiki
                  server.

-titleregex       Work on titles that match the given regular expression.

-transcludes      Work on all pages that use a certain template.
                  Argument can also be given as "-transcludes:Title".

-unusedfiles      Work on all description pages of images/media files that are
                  not used anywhere.
                  Argument can be given as "-unusedfiles:n" where
                  n is the maximum number of articles to work on.

-lonelypages      Work on all articles that are not linked from any other
                  article.
                  Argument can be given as "-lonelypages:n" where
                  n is the maximum number of articles to work on.

-unwatched        Work on all articles that are not watched by anyone.
                  Argument can be given as "-unwatched:n" where
                  n is the maximum number of articles to work on.

-usercontribs     Work on all articles that were edited by a certain user.
                  (Example : -usercontribs:DumZiBoT)

-weblink          Work on all articles that contain an external link to
                  a given URL; may be given as "-weblink:url"

-withoutinterwiki Work on all pages that don't have interlanguage links.
                  Argument can be given as "-withoutinterwiki:n" where
                  n is the total to fetch.

-mysqlquery       Takes a Mysql query string like
                  "SELECT page_namespace, page_title, FROM page
                  WHERE page_namespace = 0" and works on the resulting pages.
```

```
-wikidataquery    Takes a WikidataQuery query string like claim[31:12280]
                  and works on the resulting pages.

-searchitem       Takes a search string and works on Wikibase pages that
                  contain it.
                  Argument can be given as "-searchitem:text", where text
                  is the string to look for, or "-searchitem:lang:text", where
                  lang is the langauge to search items in.

-random           Work on random pages returned by [[Special:Random]].
                  Can also be given as "-random:n" where n is the number
                  of pages to be returned, otherwise the default is 10 pages.

-randomredirect   Work on random redirect pages returned by
                  [[Special:RandomRedirect]]. Can also be given as
                  "-randomredirect:n" where n is the number of pages to be
                  returned, else 10 pages are returned.

-untagged         Work on image pages that don't have any license template on a
                  site given in the format "<language>.<project>.org, e.g.
                  "ja.wikipedia.org" or "commons.wikimedia.org".
                  Using an external Toolserver tool.

-google           Work on all pages that are found in a Google search.
                  You need a Google Web API license key. Note that Google
                  doesn't give out license keys anymore. See google_key in
                  config.py for instructions.
                  Argument can also be given as "-google:searchstring".

-yahoo            Work on all pages that are found in a Yahoo search.
                  Depends on python module pYsearch.  See yahoo_appid in
                  config.py for instructions.

-page             Work on a single page. Argument can also be given as
                  "-page:pagetitle", and supplied multiple times for
                  multiple pages.

-grep             A regular expression that needs to match the article
                  otherwise the page won't be returned.
                  Multiple -grep:regexpr can be provided and the page will
                  be returned if content is matched by any of the regexpr
                  provided.
                  Case insensitive regular expressions will be used and
                  dot matches any character, including a newline.

-onlyif           A claim the page needs to contain, otherwise the item won't
                  be returned.
                  The format is property=value,qualifier=value. Multiple (or
                  none) qualifiers can be passed, separated by commas.
                  Examples: P1=Q2 (property P1 must contain value Q2),
                  P3=Q4,P5=Q6,P6=Q7 (property P3 with value Q4 and
                  qualifiers: P5 with value Q6 and P6 with value Q7).
                 Value can be page ID, coordinate in format::
                  latitude,longitude[,precision] (all values are in decimal
                  degrees), year, or plain string.
                  The argument can be provided multiple times and the item
                  page will be returned only if all of the claims are present.
                  Argument can be also given as "-onlyif:expression".
```

```
-onlyifnot        A claim the page must not contain, otherwise the item won't
                  be returned.
                  For usage and examples, see -onlyif above.

-intersect        Work on the intersection of all the provided generators.
```

pywikibot.pagegenerators.**AllpagesPageGenerator**(*start='!'*, *namespace=0*, *includeredirects=True*, *site=None*, *step=None*, *total=None*, *content=False*)

Iterate Page objects for all titles in a single namespace.

If includeredirects is False, redirects are not included. If includeredirects equals the string 'only', only redirects are added.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **content** – If True, load current version of each page (default False)
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**AncientPagesPageGenerator**(*total=100*, *site=None*)

Ancient page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**CategorizedPageGenerator**(*category*, *recurse=False*, *start=None*, *step=None*, *total=None*, *content=False*, *namespaces=None*)

Yield all pages in a specific category.

If recurse is True, pages in subcategories are included as well; if recurse is an int, only subcategories to that depth will be included (e.g., recurse=2 will get pages in subcats and sub-subcats, but will not go any further).

If start is a string value, only pages whose sortkey comes after start alphabetically are included.

If content is True (default is False), the current page text of each retrieved page will be downloaded.

pywikibot.pagegenerators.**CategoryGenerator**(*generator*)

Yield pages from another generator as Category objects.

Makes sense only if it is ascertained that only categories are being retrieved.

pywikibot.pagegenerators.**CombinedPageGenerator**(*generators*)

Yield from each iterable until exhausted, then proceed with the next.

pywikibot.pagegenerators.**DayPageGenerator**(*startMonth=1*, *endMonth=12*, *site=None*)

Day page generator.

Parameters **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**DeadendPagesPageGenerator**(*total=100*, *site=None*)

Dead-end page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total

- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**DequePreloadingGenerator**(*generator*, *step=50*)
Preload generator of type DequeGenerator.

pywikibot.pagegenerators.**DuplicateFilterPageGenerator**(*generator*)
Yield all unique pages from another generator, omitting duplicates.

pywikibot.pagegenerators.**EdittimeFilterPageGenerator**(*generator*, *last_edit_start=None*, *last_edit_end=None*, *first_edit_start=None*, *first_edit_end=None*, *show_filtered=False*)
Wrap a generator to filter pages outside last or first edit range.

Parameters

- **generator** – A generator object
- **last_edit_start** (*datetime*) – Only yield pages last edited after this time
- **last_edit_end** (*datetime*) – Only yield pages last edited before this time
- **first_edit_start** (*datetime*) – Only yield pages first edited after this time
- **first_edit_end** (*datetime*) – Only yield pages first edited before this time
- **show_filtered** (*bool*) – Output a message for each page not yielded

pywikibot.pagegenerators.**FileGenerator**(*generator*)
Yield pages from another generator as FilePage objects.

Makes sense only if it is ascertained that only images are being retrieved.

pywikibot.pagegenerators.**FileLinksGenerator**(*referredFilePage*, *step=None*, *total=None*, *content=False*)
Yield Pages on which the file referredFilePage is displayed.

class pywikibot.pagegenerators.**GeneratorFactory**(*site=None*)
Bases: object

Process command line arguments and return appropriate page generator.

This factory is responsible for processing command line arguments that are used by many scripts and that determine which pages to work on.

getCategoryGen(*arg*, *recurse=False*, *content=False*, *gen_func=None*)
Return generator based on Category defined by arg and gen_func.

getCombinedGenerator(*gen=None*)
Return the combination of all accumulated generators.

Only call this after all arguments have been parsed.

handleArg(*arg*)
Parse one argument at a time.

If it is recognized as an argument that specifies a generator, a generator is created and added to the accumulation list, and the function returns true. Otherwise, it returns false, so that caller can try parsing the argument. Call getCombinedGenerator() after all arguments have been parsed to get the final output generator.

namespaces
List of Namespace parameters.

Converts int or string namespaces to Namespace objects and change the storage to immutable once it has been accessed.

The resolving and validation of namespace command line arguments is performed in this method, as it depends on the site property which is lazy loaded to avoid being cached before the global arguments are handled.

Returns namespaces selected using arguments

Return type list of Namespace

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

site

Generator site.

The generator site should not be accessed until after the global arguments have been handled, otherwise the default Site may be changed by global arguments, which will cause this cached value to be stale.

Returns Site given to constructor, otherwise the default Site at the time this property is first accessed.

Return type *pywikibot.site.BaseSite*

class pywikibot.pagegenerators.**GoogleSearchPageGenerator**(*query=None*, *site=None*)

Bases: object

Page generator using Google search results.

To use this generator, you need to install the package 'google'. https://pypi.python.org/pypi/google

This package has been available since 2010, hosted on github since 2012, and provided by pypi since 2013.

As there are concerns about Google's Terms of Service, this generator prints a warning for each query.

queryGoogle(*query*)

Perform a query using python package 'google'.

The terms of service as at June 2014 give two conditions that may apply to use of search:

```
1. Dont access [Google Services] using a method other than
   the interface and the instructions that [they] provide.
2. Don't remove, obscure, or alter any legal notices
   displayed in or along with [Google] Services.
```

Both of those issues should be managed by the package 'google', however Pywikibot will at least ensure the user sees the TOS in order to comply with the second condition.

pywikibot.pagegenerators.**ImageGenerator**(*generator*)

Yield pages from another generator as FilePage objects.

Makes sense only if it is ascertained that only images are being retrieved.

pywikibot.pagegenerators.**ImagesPageGenerator**(*pageWithImages*, *step=None*, *total=None*, *content=False*)

Yield FilePages displayed on pageWithImages.

pywikibot.pagegenerators.**InterwikiPageGenerator**(*page*)

Iterate over all interwiki (non-language) links on a page.

class pywikibot.pagegenerators.**ItemClaimFilter**
Bases: object

Item claim filter.

classmethod filter(*generator*, *prop*, *claim*, *qualifiers=None*, *negate=False*)
Yield all ItemPages which does contain certain claim in a property.

Parameters

- **prop** (*str*) – property id to check
- **claim** – value of the property to check. Can be exact value (for instance, ItemPage instance) or ItemPage ID string (e.g. 'Q37470').
- **qualifiers** (*dict or None*) – dict of qualifiers that must be present, or None if qualifiers are irrelevant
- **negate** (*bool*) – true if pages that does *not* contain specified claim should be yielded, false otherwise

pywikibot.pagegenerators.**LanguageLinksPageGenerator**(*page*, *step=None*, *total=None*)
Iterate over all interwiki language links on a page.

pywikibot.pagegenerators.**LinkedPageGenerator**(*linkingPage*, *step=None*, *total=None*, *content=False*)
Yield all pages linked from a specific page.

pywikibot.pagegenerators.**LinksearchPageGenerator**(*link*, *namespaces=None*, *step=None*, *total=None*, *site=None*)
Yield all pages that include a specified link.

Obtains data from [[Special:Linksearch]].

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**LiveRCPageGenerator**(*site=None*, *total=None*)
Yield pages from a socket.io RC stream.

Generates pages based on the socket.io recent changes stream. The Page objects will have an extra property ._rcinfo containing the literal rc data. This can be used to e.g. filter only new pages. See *pywikibot.comms.rcstream.rc_listener* for details on the .rcinfo format.

Parameters

- **site** (*pywikibot.BaseSite*) – site to return recent changes for
- **total** (*int*) – the maximum number of changes to return

pywikibot.pagegenerators.**LogeventsPageGenerator**(*logtype=None*, *user=None*, *site=None*, *namespace=0*, *total=None*)
Generate Pages for specified modes of logevents.

Parameters

- **logtype** (*basestring*) – Mode of logs to retrieve
- **user** (*basestring*) – User of logs retrieved
- **site** (*pywikibot.site.BaseSite*) – Site for generator results
- **namespace** (*int*) – Namespace to retrieve logs from

- **total** (*int*) – Maximum number of pages to retrieve in total

pywikibot.pagegenerators.**LogpagesPageGenerator**(*total=500*, *logtype=''*, *user=None*, *site=None*, *namespace=[]*)

Generate Pages for specified modes of logevents.

This is the backwards compatible one. See LogeventsPageGenerator

Parameters

- **mode** (*basestring*) – Mode of logs to retrieve
- **user** (*basestring*) – User of logs retrieved
- **site** (*pywikibot.site.BaseSite*) – Site for generator results
- **namespace** (*int*) – Namespace to retrieve logs from
- **total** (*int*) – Maximum number of pages to retrieve in total

pywikibot.pagegenerators.**LonelyPagesPageGenerator**(*total=100*, *site=None*)

Lonely page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**LongPagesPageGenerator**(*total=100*, *site=None*)

Long page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**MySQLPageGenerator**(*query*, *site=None*)

Yield a list of pages based on a MySQL query.

Each query should provide the page namespace and page title. An example query that yields all ns0 pages might look like::

```
SELECT
 page_namespace,
 page_title,
FROM page
WHERE page_namespace = 0;
```

Requires oursql <https://pythonhosted.org/oursql/> or MySQLdb <https://sourceforge.net/projects/mysql-python/>

Parameters

- **query** – MySQL query to execute
- **site** (*pywikibot.site.BaseSite*) – Site object

Returns iterator of pywikibot.Page

pywikibot.pagegenerators.**NamespaceFilterPageGenerator**(*generator*, *namespaces*, *site=None*)

A generator yielding pages from another generator in given namespaces.

If a site is provided, the namespaces are validated using the namespaces of that site, otherwise the namespaces are validated using the default site.

NOTE: API-based generators that have a "namespaces" parameter perform namespace filtering more efficiently than this generator.

Parameters

- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types.*) – list of namespace identifiers to limit results
- **site** (*pywikibot.site.BaseSite*) – Site for generator results; mandatory if namespaces contains namespace names. Defaults to the default site.

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool, or more than one namespace if the API module does not support multiple namespaces

pywikibot.pagegenerators.**NewimagesPageGenerator**(*step=None*, *total=None*, *site=None*)

New file generator.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**NewpagesPageGenerator**(*get_redirect=False*, *site=None*, *namespaces=[0]*, *step=None*, *total=None*)

Iterate Page objects for all new titles in a single namespace.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**PageTitleFilterPageGenerator**(*generator*, *ignore_list*)

Yield only those pages are not listed in the ignore list.

Parameters **ignore_list** (*dict*) – family names are mapped to dictionaries in which language codes are mapped to lists of page titles. Each title must be a valid regex as they are compared using `re.search`.

pywikibot.pagegenerators.**PageWithTalkPageGenerator**(*generator*, *return_talk_only=False*)

Yield pages and associated talk pages from another generator.

Only yields talk pages if the original generator yields a non-talk page, and does not check if the talk page in fact exists.

pywikibot.pagegenerators.**PagesFromTitlesGenerator**(*iterable*, *site=None*)

Generate pages from the titles (unicode strings) yielded by iterable.

Parameters **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**PrefixingPageGenerator**(*prefix*, *namespace=None*, *includeredirects=True*, *site=None*, *step=None*, *total=None*, *content=False*)

Prefixed Page generator.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **content** – If True, load current version of each page (default False)
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**PreloadingGenerator**(*generator*, *step=50*)
Yield preloaded pages taken from another generator.

Parameters

- **generator** – pages to iterate over
- **step** (*int*) – how many pages to preload at once

pywikibot.pagegenerators.**PreloadingItemGenerator**(*generator*, *step=50*)
Yield preloaded pages taken from another generator.

Function basically is copied from above, but for ItemPage's

Parameters

- **generator** – pages to iterate over
- **step** (*int*) – how many pages to preload at once

pywikibot.pagegenerators.**RandomPageGenerator**(*total=10*, *site=None*)
Random page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**RandomRedirectPageGenerator**(*total=10*, *site=None*)
Random redirect generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**RecentChangesPageGenerator**(*start=None*, *end=None*, *reverse=False*, *namespaces=None*, *pagelist=None*, *changetype=None*, *showMinor=None*, *showBot=None*, *showAnon=None*, *showRedirects=None*, *showPatrolled=None*, *topOnly=False*, *step=None*, *total=None*, *user=None*, *excludeuser=None*, *site=None*)
Generate pages that are in the recent changes list.

Parameters

- **start** (*pywikibot.Timestamp*) – Timestamp to start listing from
- **end** (*pywikibot.Timestamp*) – Timestamp to end listing at
- **reverse** (*bool*) – if True, start with oldest changes (default: newest)

- **pagelist** – iterate changes to pages in this list only
- **pagelist** – list of Pages
- **changetype** (*basestring*) – only iterate changes of this type ("edit" for edits to existing pages, "new" for new pages, "log" for log entries)
- **showMinor** (*bool or None*) – if True, only list minor edits; if False, only list non-minor edits; if None, list all
- **showBot** (*bool or None*) – if True, only list bot edits; if False, only list non-bot edits; if None, list all
- **showAnon** (*bool or None*) – if True, only list anon edits; if False, only list non-anon edits; if None, list all
- **showRedirects** (*bool or None*) – if True, only list edits to redirect pages; if False, only list edits to non-redirect pages; if None, list all
- **showPatrolled** (*bool or None*) – if True, only list patrolled edits; if False, only list non-patrolled edits; if None, list all
- **topOnly** (*bool*) – if True, only list changes that are the latest revision (default False)
- **user** (*basestring|list*) – if not None, only list edits by this user or users
- **excludeuser** (*basestring|list*) – if not None, exclude edits by this user or users
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**RedirectFilterPageGenerator**(*generator*, *no_redirects=True*, *show_filtered=False*)

Yield pages from another generator that are redirects or not.

Parameters

- **no_redirects** – Exclude redirects if True, else only include redirects.
- **no_redirects** – bool
- **show_filtered** (*bool*) – Output a message for each page not yielded

pywikibot.pagegenerators.**ReferringPageGenerator**(*referredPage*, *followRedirects=False*, *withTemplateInclusion=True*, *onlyTemplateInclusion=False*, *step=None*, *total=None*, *content=False*)

Yield all pages referring to a specific page.

class pywikibot.pagegenerators.**RegexFilter**

Bases: object

Regex filter.

classmethod contentfilter(*generator*, *regex*, *quantifier='any'*)

Yield pages from another generator whose body matches regex.

Uses regex option re.IGNORECASE depending on the quantifier parameter.

For parameters see titlefilter above.

classmethod titlefilter(*generator*, *regex*, *quantifier='any'*, *ignore_namespace=True*)

Yield pages from another generator whose title matches regex.

Uses regex option re.IGNORECASE depending on the quantifier parameter.

If ignore_namespace is False, the whole page title is compared. NOTE: if you want to check for a match at the beginning of the title, you have to start the regex with "^"

Parameters

- **generator** (*any generator or iterator*) – another generator
- **regex** (*a single regex string or a list of regex strings or a compiled regex or a list of compiled regexes*) – a regex which should match the page title
- **quantifier** (*string of ('all', 'any', 'none')*) – must be one of the following values: 'all' - yields page if title is matched by all regexes 'any' - yields page if title is matched by any regexes 'none' - yields page if title is NOT matched by any regexes
- **ignore_namespace** (*bool*) – ignore the namespace when matching the title

Returns return a page depending on the matching parameters

pywikibot.pagegenerators.**RepeatingGenerator**(*generator*, *key_func=<function <lambda>>*, *sleep_duration=60*, *total=None*, ***kwargs*)

Yield items in live time.

The provided generator must support parameter 'start', 'end', 'reverse', and 'total' such as site.recentchanges(), site.logevents().

To fetch revisions in recentchanges in live time::

```
gen = RepeatingGenerator(site.recentchanges, lambda x: x['revid'])
```

To fetch new pages in live time::

```
gen = RepeatingGenerator(site.newpages, lambda x: x[0])
```

Note that other parameters not listed below will be passed to the generator function. Parameter 'reverse', 'start', 'end' will always be discarded to prevent the generator yielding items in wrong order.

Parameters

- **generator** – a function returning a generator that will be queried
- **key_func** – a function returning key that will be used to detect duplicate entry
- **sleep_duration** – duration between each query
- **total** (*int or None*) – if it is a positive number, iterate no more than this number of items in total. Otherwise, iterate forever

Returns a generator yielding items in ascending order by time

pywikibot.pagegenerators.**SearchPageGenerator**(*query*, *step=None*, *total=None*, *namespaces=None*, *site=None*)

Yield pages from the MediaWiki internal search engine.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**ShortPagesPageGenerator**(*total=100*, *site=None*)

Short page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total

- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**SubCategoriesPageGenerator**(*category*, *recurse=False*, *start=None*, *step=None*, *total=None*, *content=False*)

Yield all subcategories in a specific category.

If recurse is True, pages in subcategories are included as well; if recurse is an int, only subcategories to that depth will be included (e.g., recurse=2 will get pages in subcats and sub-subcats, but will not go any further).

If start is a string value, only categories whose sortkey comes after start alphabetically are included.

If content is True (default is False), the current page text of each category description page will be downloaded.

pywikibot.pagegenerators.**TextfilePageGenerator**(*filename=None*, *site=None*)

Iterate pages from a list in a text file.

The file must contain page links between double-square-brackets or, in alternative, separated by newlines. The generator will yield each corresponding Page object.

Parameters

- **filename** (*unicode*) – the name of the file that should be read. If no name is given, the generator prompts the user.
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnCategorizedCategoryGenerator**(*total=100*, *site=None*)

Uncategorized category generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnCategorizedImageGenerator**(*total=100*, *site=None*)

Uncategorized file generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnCategorizedPageGenerator**(*total=100*, *site=None*)

Uncategorized page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnCategorizedTemplateGenerator**(*total=100*, *site=None*)

Uncategorized template generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UntaggedPageGenerator**(*untaggedProject*, *limit=500*, *site=None*)

Yield pages from defunct toolserver UntaggedImages.php.

It was using this tool:: https://toolserver.org/~daniel/WikiSense/UntaggedImages.php

Parameters **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnusedFilesGenerator**(*total=100*, *site=None*, *extension=None*)
Unused files generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UnwatchedPagesPageGenerator**(*total=100*, *site=None*)
Unwatched page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**UserContributionsGenerator**(*username*, *namespaces=None*, *site=None*, *step=None*, *total=None*)
Yield unique pages edited by user:username.

Parameters

- **step** (*int*) – Maximum number of pages to retrieve per API query
- **total** (*int*) – Maxmum number of pages to retrieve in total
- **namespaces** (*list of int*) – list of namespace numbers to fetch contribs from
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**WantedPagesPageGenerator**(*total=100*, *site=None*)
Wanted page generator.

Parameters

- **total** (*int*) – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**WikibaseItemFilterPageGenerator**(*generator*, *has_item=True*, *show_filtered=False*)
A wrapper generator used to exclude if page has a wikibase item or not.

Parameters

- **gen** (*generator*) – Generator to wrap.
- **has_item** (*bool*) – Exclude pages without an item if True, or only include pages without an item if False
- **show_filtered** (*bool*) – Output a message for each page not yielded

Returns Wrapped generator

Return type generator

pywikibot.pagegenerators.**WikibaseItemGenerator**(*gen*)
A wrapper generator used to yield Wikibase items of another generator.

Parameters **gen** (*generator*) – Generator to wrap.

Returns Wrapped generator

Return type generator

pywikibot.pagegenerators.**WikibaseSearchItemPageGenerator**(*text*, *language=None*, *total=None*, *site=None*)

Generate pages that contain the provided text.

Parameters

- **text** (*str*) – Text to look for.
- **language** (*str*) – Code of the language to search in. If not specified, value from pywikibot.config.data_lang is used.
- **total** (*int or None*) – Maximum number of pages to retrieve in total, or None in case of no limit.
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**WikidataItemGenerator**(*gen*)

A wrapper generator used to yield Wikibase items of another generator.

Parameters **gen** (*generator*) – Generator to wrap.

Returns Wrapped generator

Return type generator

pywikibot.pagegenerators.**WikidataQueryPageGenerator**(*query*, *site=None*)

Generate pages that result from the given WikidataQuery.

Parameters

- **query** – the WikidataQuery query string.
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

pywikibot.pagegenerators.**WithoutInterwikiPageGenerator**(*total=100*, *site=None*)

Page lacking interwikis generator.

Parameters

- **total** – Maxmum number of pages to retrieve in total
- **site** (*pywikibot.site.BaseSite*) – Site for generator results.

class pywikibot.pagegenerators.**YahooSearchPageGenerator**(*query=None*, *count=100*, *site=None*)

Bases: object

Page generator using Yahoo! search results.

To use this generator, you need to install the package 'pYsearch'. https://pypi.python.org/pypi/pYsearch

To use this generator, install pYsearch

queryYahoo(*query*)

Perform a query using python package 'pYsearch'.

pywikibot.pagegenerators.**YearPageGenerator**(*start=1*, *end=2050*, *site=None*)

Year page generator.

Parameters **site** (*pywikibot.site.BaseSite*) – Site for generator results.

plural Module

Module containing plural rules of various languages.

site Module

Objects representing MediaWiki sites (wikis).

This module also includes functions to load families, which are groups of wikis on the same topic in different languages.

class pywikibot.site.**APISite**(*code*, *fam=None*, *user=None*, *sysop=None*)
Bases: *pywikibot.site.BaseSite*

API interface to MediaWiki site.

Do not use directly; use pywikibot.Site function.

class **OnErrorExc**(*exception*, *on_new_page*)
Bases: tuple

_asdict()
Return a new OrderedDict which maps field names to their values.

_fields = ('exception', 'on_new_page')

classmethod **_make**(*iterable*, *new=<built-in method __new__ of type object at 0x9ca6e0>*, *len=<built-in function len>*)
Make a new OnErrorExc object from a sequence or iterable

_replace(*_self*, ***kwds*)
Return a new OnErrorExc object replacing specified fields with new values

_source = "from builtins import property as _property, tuple as _tuple\nfrom operator import itemgetter as _itemg

exception
Alias for field number 0

on_new_page
Alias for field number 1

APISite.**_build_namespaces**()

APISite.**_cache_proofreadinfo**(*expiry=False*)
Retrieve proofreadinfo from site and cache response.

Applicable only to sites with ProofreadPage extension installed.

The following info is returned by the query and cached::

- self._proofread_index_ns: Index Namespace
- self._proofread_page_ns: Page Namespace

•self._proofread_levels: a dictionary with::

keys: int in the range [0, 1, ..., 4] values: category name corresponding to the 'key' quality level

e.g. on en.wikisource:: {0: u'Without text', 1: u'Not proofread', 2: u'Problematic',

3: u'Proofread', 4: u'Validated'}

Parameters **expiry** (int (days), datetime.timedelta, False (config)) – either a number of days or a datetime.timedelta object

Returns A tuple containing _proofread_index_ns, self._proofread_page_ns and self._proofread_levels.

Return type Namespace, Namespace, dict

APISite.**_dl_errors = {'permissiondenied': 'User %(user)s not authorized to (un)delete pages on %(site)s wiki.', 'no**

APISite.**_ep_errors = {'cantcreate-anon': 'Bot is not logged in, and anon users are not authorized to create new pag**

APISite.**_generator**(*gen_class*, *type_arg=None*, *namespaces=None*, *step=None*, *total=None*, ***args*)

Convenience method that returns an API generator.

All keyword args not listed below are passed to the generator's constructor unchanged.

Parameters

- **gen_class** – the type of generator to construct (must be a subclass of pywikibot.data.api.QueryGenerator)
- **type_arg** (*str*) – query type argument to be passed to generator's constructor unchanged (not all types require this)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – if not None, limit the query to namespaces in this list
- **step** (*int*) – if not None, limit each API call to this many items
- **total** (*int*) – if not None, limit the generator to yielding this many items in total

Returns iterable with parameters set

Return type *QueryGenerator*

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**_get_titles_with_hash**(*hash_found=None*)

Helper for the deprecated method get(Files|Images)FromAnHash.

APISite.**_mv_errors = {'protectedpage': OnErrorExc(exception=<class 'pywikibot.exceptions.LockedPage'>, on_ne**

APISite.**_patrol_errors = {'noautopatrol': "User %(user)s has no permission to patrol its own changes, 'autopatr**

APISite.**_protect_errors = {'permissiondenied': 'User %(user)s not authorized to protect pages on %(site)s wiki.'**

APISite.**_rb_errors = {'noapiwrite': 'API editing not enabled on %(site)s wiki', 'writeapidenied': 'User %(user)s n**

APISite.**_update_page**(*page*, *query*, *method_name*)

APISite.**allcategories**(*start='!'*, *prefix=''*, *step=None*, *total=None*, *reverse=False*, *content=False*)

Iterate categories used (which need not have a Category page).

Iterator yields Category objects. Note that, in practice, links that were found on pages that have been deleted may not have been removed from the database table, so this method can return false positives.

Parameters

- **start** – Start at this category title (category need not exist).
- **prefix** – Only yield categories starting with this string.
- **reverse** – if True, iterate in reverse Unicode lexigraphic order (default: iterate in forward order)

- **content** – if True, load the current content of each iterated page (default False); note that this means the contents of the category description page, not the pages that are members of the category

APISite.**allimages**(*start='!'*, *prefix=''*, *minsize=None*, *maxsize=None*, *reverse=False*, *sha1=None*, *sha1base36=None*, *step=None*, *total=None*, *content=False*)

Iterate all images, ordered by image title.

Yields FilePages, but these pages need not exist on the wiki.

Parameters

- **start** – start at this title (name need not exist)
- **prefix** – only iterate titles starting with this substring
- **minsize** – only iterate images of at least this many bytes
- **maxsize** – only iterate images of no more than this many bytes
- **reverse** – if True, iterate in reverse lexigraphic order
- **sha1** – only iterate image (it is theoretically possible there could be more than one) with this sha1 hash
- **sha1base36** – same as sha1 but in base 36
- **content** – if True, load the current content of each iterated page (default False); note that this means the content of the image description page, not the image itself

APISite.**alllinks**(*start='!'*, *prefix=''*, *namespace=0*, *unique=False*, *fromids=False*, *step=None*, *total=None*)

Iterate all links to pages (which need not exist) in one namespace.

Note that, in practice, links that were found on pages that have been deleted may not have been removed from the links table, so this method can return false positives.

Parameters

- **start** – Start at this title (page need not exist).
- **prefix** – Only yield pages starting with this string.
- **namespace** (*int or Namespace*) – Iterate pages from this (single) namespace
- **unique** – If True, only iterate each link title once (default: iterate once for each linking page)
- **fromids** – if True, include the pageid of the page containing each link (default: False) as the '_fromid' attribute of the Page; cannot be combined with unique

Raises

- **KeyError** – the namespace identifier was not resolved
- **TypeError** – the namespace identifier has an inappropriate type such as bool, or an iterable with more than one namespace

APISite.**allpages**(*start='!'*, *prefix=''*, *namespace=0*, *filterredir=None*, *filterlanglinks=None*, *minsize=None*, *maxsize=None*, *protect_type=None*, *protect_level=None*, *reverse=False*, *includeredirects=None*, *step=None*, *total=None*, *content=False*)

Iterate pages in a single namespace.

Note: parameters includeRedirects and throttle are deprecated and included only for backwards compatibility.

Parameters

- **start** – Start at this title (page need not exist).
- **prefix** – Only yield pages starting with this string.
- **namespace** (*int or Namespace.*) – Iterate pages from this (single) namespace
- **filterredir** – if True, only yield redirects; if False (and not None), only yield non-redirects (default: yield both)
- **filterlanglinks** – if True, only yield pages with language links; if False (and not None), only yield pages without language links (default: yield both)
- **minsize** – if present, only yield pages at least this many bytes in size
- **maxsize** – if present, only yield pages at most this many bytes in size
- **protect_type** (*str*) – only yield pages that have a protection of the specified type
- **protect_level** – only yield pages that have protection at this level; can only be used if protect_type is specified
- **reverse** – if True, iterate in reverse Unicode lexigraphic order (default: iterate in forward order)
- **includeredirects** – DEPRECATED, use filterredir instead
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – the namespace identifier was not resolved
- **TypeError** – the namespace identifier has an inappropriate type such as bool, or an iterable with more than one namespace

APISite.**allusers**(*start='!'*, *prefix=''*, *group=None*, *step=None*, *total=None*)

Iterate registered users, ordered by username.

Iterated values are dicts containing 'name', 'editcount', 'registration', and (sometimes) 'groups' keys. 'groups' will be present only if the user is a member of at least 1 group, and will be a list of unicodes; all the other values are unicodes and should always be present.

Parameters

- **start** – start at this username (name need not exist)
- **prefix** – only iterate usernames starting with this substring
- **group** (*str*) – only iterate users that are members of this group

APISite.**ancientpages**(*step=None*, *total=None*)

Yield Pages, datestamps from Special:Ancientpages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**assert_valid_iter_params**(*msg_prefix*, *start*, *end*, *reverse*)

Validate iterating API parameters.

APISite.**blocks**(*starttime=None*, *endtime=None*, *reverse=False*, *blockids=None*, *users=None*, *step=None*, *total=None*)

Iterate all current blocks, in order of creation.

Note that logevents only logs user blocks, while this method iterates all blocks including IP ranges. The iterator yields dicts containing keys corresponding to the block properties (see https://www.mediawiki.org/wiki/API:Query_-_Lists for documentation).

Parameters

- **starttime** – start iterating at this Timestamp
- **endtime** – stop iterating at this Timestamp
- **reverse** – if True, iterate oldest blocks first (default: newest)
- **blockids** – only iterate blocks with these id numbers
- **users** – only iterate blocks affecting these usernames or IPs

APISite.**blockuser**(*user*, *expiry*, *reason*, *anononly=True*, *nocreate=True*, *autoblock=True*, *noemail=False*, *reblock=False*)

Block a user for certain amount of time and for a certain reason.

Parameters

- **user** (pywikibot.User) – The username/IP to be blocked without a namespace.
- **expiry** (*Timestamp/datetime (absolute), basestring (relative/absolute) or False ('never')*) – The length or date/time when the block expires. If 'never', 'infinite', 'indefinite' it never does. If the value is given as a basestring it's parsed by php's strtotime function:

```
http://php.net/manual/en/function.strtotime.php
```

The relative format is described there:: http://php.net/manual/en/datetime.formats.relative.php

It is recommended to not use a basestring if possible to be independent of the API.

- **reason** (*basestring*) – The reason for the block.
- **anononly** (*boolean*) – Disable anonymous edits for this IP.
- **nocreate** (*boolean*) – Prevent account creation.
- **autoblock** (*boolean*) – Automatically block the last used IP address and all subsequent IP addresses from which this account logs in.
- **noemail** (*boolean*) – Prevent user from sending email through the wiki.
- **reblock** (*boolean*) – If the user is already blocked, overwrite the existing block.

Returns The data retrieved from the API request.

Return type dict

APISite.**botusers**(*step=None*, *total=None*)

Iterate bot users.

Iterated values are dicts containing 'name', 'userid', 'editcount', 'registration', and 'groups' keys. 'groups' will be present only if the user is a member of at least 1 group, and will be a list of unicodes; all the other values are unicodes and should always be present.

APISite.**broken_redirects**(*step=None*, *total=None*)

Yield Pages without language links from Special:BrokenRedirects.

Parameters

- **step** – request batch size

- **total** – number of pages to return

APISite.**case**()
Return this site's capitalization rule.

APISite.**categories**(*number=10*, *repeat=False*)
DEPRECATED.

APISite.**categoryinfo**(*category*)

APISite.**categorymembers**(*category*, *namespaces=None*, *sortby=None*, *reverse=False*, *starttime=None*, *endtime=None*, *startsort=None*, *endsort=None*, *step=None*, *total=None*, *content=False*, *member_type=None*)
Iterate members of specified category.

Parameters

- **category** – The Category to iterate.
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – If present, only return category members from these namespaces. To yield subcategories or files, use parameter member_type instead.
- **sortby** (*str*) – determines the order in which results are generated, valid values are "sortkey" (default, results ordered by category sort key) or "timestamp" (results ordered by time page was added to the category)
- **reverse** – if True, generate results in reverse order (default False)
- **starttime** (*pywikibot.Timestamp*) – if provided, only generate pages added after this time; not valid unless sortby="timestamp"
- **endtime** (*pywikibot.Timestamp*) – if provided, only generate pages added before this time; not valid unless sortby="timestamp"
- **startsort** (*str*) – if provided, only generate pages >= this title lexically; not valid if sortby="timestamp"
- **endsort** (*str*) – if provided, only generate pages <= this title lexically; not valid if sortby="timestamp"
- **content** (*bool*) – if True, load the current content of each iterated page (default False)
- **member_type** (*str or iterable of str; values: page, subcat, file*) – member type; if member_type includes 'page' and is used in conjunction with sortby="timestamp", the API may limit results to only pages in the first 50 namespaces.

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**checkBlocks**(*sysop=False*)
Raise an exception when the user is blocked. DEPRECATED.

Parameters **sysop** (*bool*) – If true, log in to sysop account (if available)

Raises UserBlocked The logged in user/sysop account is blocked.

APISite.**compare**(*old*, *diff*)
Corresponding method to the 'action=compare' API action.

See: https://en.wikipedia.org/w/api.php?action=help&modules=compare Use pywikibot.diff's html_comparator() method to parse result. :param old: starting revision ID, title, Page, or Revision

:type old: int, str, pywikibot.Page, or pywikibot.Page.Revision :param diff: ending revision ID, title, Page, or Revision :type diff: int, str, pywikibot.Page, or pywikibot.Page.Revision :return: Returns an HTML string of a diff between two revisions. :rtype: str

APISite.**data_repository**()
Return Site object for data repository e.g. Wikidata.

APISite.**dbName**()
Return this site's internal id.

APISite.**deadendpages**(*step=None*, *total=None*)
Yield Page objects retrieved from Special:Deadendpages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**deletedrevs**(*page*, *start=None*, *end=None*, *reverse=None*, *get_text=False*, *step=None*, *total=None*)
Iterate deleted revisions.

Each value returned by the iterator will be a dict containing the 'title' and 'ns' keys for a particular Page and a 'revisions' key whose value is a list of revisions in the same format as recentchanges (plus a 'content' element if requested). If get_text is true, the toplevel dict will contain a 'token' key as well.

Parameters

- **page** – The page to check for deleted revisions
- **start** – Iterate revisions starting at this Timestamp
- **end** – Iterate revisions ending at this Timestamp
- **reverse** – Iterate oldest revisions first (default: newest)
- **get_text** – If True, retrieve the content of each revision and an undelete token

APISite.**deletepage**(*page*, *reason*)
Delete page from the wiki. Requires appropriate privilege level.

Parameters

- **page** (Page) – Page to be deleted.
- **reason** (*basestring*) – Deletion reason.

APISite.**double_redirects**(*step=None*, *total=None*)
Yield Pages without language links from Special:BrokenRedirects.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**editpage**(*page*, *summary*, *minor=True*, *notminor=False*, *bot=True*, *recreate=True*, *createonly=False*, *nocreate=False*, *watch=None*)
Submit an edited Page object to be saved to the wiki.

Parameters

- **page** – The Page to be saved; its .text property will be used as the new text to be saved to the wiki
- **summary** – the edit summary (required!)

- **minor** – if True (default), mark edit as minor
- **notminor** – if True, override account preferences to mark edit as non-minor
- **recreate** – if True (default), create new page even if this title has previously been deleted
- **createonly** – if True, raise an error if this title already exists on the wiki
- **nocreate** – if True, raise an error if the page does not exist
- **watch** – Specify how the watchlist is affected by this edit, set to one of "watch", "unwatch", "preferences", "nochange":: * watch: add the page to the watchlist * unwatch: remove the page from the watchlist The following settings are supported by mw >= 1.16 only * preferences: use the preference settings (default) * nochange: don't change the watchlist
- **bot** – if True, mark edit with bot flag

Returns True if edit succeeded, False if it failed

APISite.**expand_text**(*text*, *title=None*, *includecomments=None*)
Parse the given text for preprocessing and rendering.

e.g expand templates and strip comments if includecomments parameter is not True. Keeps text inside <nowiki></nowiki> tags unchanges etc. Can be used to parse magic parser words like {{CURRENTTIMESTAMP}}.

Parameters

- **text** (*unicode*) – text to be expanded
- **title** (*unicode*) – page title without section
- **includecomments** (*bool*) – if True do not strip comments

Returns unicode

APISite.**exturlusage**(*url=None*, *protocol='http'*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Iterate Pages that contain links to the given URL.

Parameters

- **url** – The URL to search for (without the protocol prefix); this many include a '*' as a wildcard, only at the start of the hostname
- **protocol** – The protocol prefix (default: "http")

APISite.**forceLogin**(**a*, ***kw*)

classmethod APISite.**fromDBName**(*dbname*)

APISite.**getFilesFromAnHash**(*hash_found=None*)
Return all files that have the same hash.

DEPRECATED: Use *APISite.allimages* instead using 'sha1'.

APISite.**getImagesFromAnHash**(*hash_found=None*)
Return all images that have the same hash.

DEPRECATED: Use *APISite.allimages* instead using 'sha1'.

APISite.**getPatrolToken**(*sysop=False*)
DEPRECATED: Get patrol token.

APISite.**getToken**(*getalways=True*, *getagain=False*, *sysop=False*)
DEPRECATED: Get edit token.

APISite.**get_searched_namespaces**(*force=False*)
Retrieve the default searched namespaces for the user.

If no user is logged in, it returns the namespaces used by default. Otherwise it returns the user preferences. It caches the last result and returns it, if the username or login status hasn't changed.

Parameters **force** – Whether the cache should be discarded.

Returns The namespaces which are searched by default.

Return type set of *Namespace*

APISite.**get_tokens**(*types*, *all=False*)
Preload one or multiple tokens.

For all MediaWiki versions prior to 1.20, only one token can be retrieved at once. For MediaWiki versions since 1.24wmfXXX a new token system was introduced which reduced the amount of tokens available. Most of them were merged into the 'csrf' token. If the token type in the parameter is not known it will default to the 'csrf' token.

The other token types available are::

- deleteglobalaccount
- patrol (*)
- rollback
- setglobalaccountstatus
- userrights
- watch

(*) Patrol was added in v1.14. Until v1.16, the patrol token is same as the edit token. For v1.17-19, the patrol token must be obtained from the query list recentchanges.

Parameters

- **types** (*iterable*) – the types of token (e.g., "edit", "move", "delete"); see API documentation for full list of types
- **all** (*bool*) – load all available tokens, if None only if it can be done in one request.

return: a dict with retrieved valid tokens.

APISite.**getcategoryinfo**(*category*)
Retrieve data on contents of category.

APISite.**getcurrenttime**()
Return a Timestamp object representing the current server time.

For wikis with a version newer than 1.16 it uses the 'time' property of the siteinfo 'general'. It'll force a reload before returning the time. It requests to expand the text '{{CURRENTTIMESTAMP}}' for older wikis.

Returns the current server time

Return type Timestamp

APISite.**getcurrenttimestamp**()
Return the server time as a MediaWiki timestamp string.

It calls *getcurrenttime* first so it queries the server to get the current server time.

Returns the server time

Return type str (as 'yyyymmddhhmmss')

APISite.**getglobaluserinfo**()
Retrieve globaluserinfo from site and cache it.

self._globaluserinfo will be a dict with the following keys and values:

```
- id: user id (numeric str)
- home: dbname of home wiki
- registration: registration date as Timestamp
- groups: list of groups (could be empty)
- rights: list of rights (could be empty)
- editcount: global editcount
```

APISite.**getmagicwords**(*word*)
Return list of localized "word" magic words for the site.

APISite.**getredirtarget**(*page*)
Return page object for the redirect target of page.

Parameters **page** (*BasePage*) – page to search redirects for

Returns redirect target of page

Return type BasePage

@raise IsNotRedirectPage: page is not a redirect @raise RuntimeError: no redirects found @raise CircularRedirect: page is a circular redirect @raise InterwikiRedirectPage: the redirect target is

on another site

APISite.**getuserinfo**()
Retrieve userinfo from site and store in _userinfo attribute.

self._userinfo will be a dict with the following keys and values:

```
- id: user id (numeric str)
- name: username (if user is logged in)
- anon: present if user is not logged in
- groups: list of groups (could be empty)
- rights: list of rights (could be empty)
- message: present if user has a new message on talk page
- blockinfo: present if user is blocked (dict)
```

APISite.**globaluserinfo**
Retrieve userinfo from site and store in _userinfo attribute.

self._userinfo will be a dict with the following keys and values:

```
- id: user id (numeric str)
- name: username (if user is logged in)
- anon: present if user is not logged in
- groups: list of groups (could be empty)
- rights: list of rights (could be empty)
- message: present if user has a new message on talk page
- blockinfo: present if user is blocked (dict)
```

APISite.**hasExtension**(*name*, *unknown=None*)
Determine whether extension *name* is loaded.

Use *has_extension* instead!

Parameters

- **name** (*str*) – The extension to check for, case insensitive
- **unknown** – Old parameter which shouldn't be used anymore.

Returns If the extension is loaded

Return type bool

APISite.**has_all_mediawiki_messages**(*keys*)
Confirm that the site defines a set of MediaWiki messages.

Parameters **keys** (*set of str*) – names of MediaWiki messages

Returns bool

APISite.**has_data_repository**
Return True if site has a shared data repository like Wikidata.

APISite.**has_extension**(*name*)
Determine whether extension *name* is loaded.

Parameters **name** (*str*) – The extension to check for, case sensitive

Returns If the extension is loaded

Return type bool

APISite.**has_group**(*group*, *sysop=False*)
Return true if and only if the user is a member of specified group.

Possible values of 'group' may vary depending on wiki settings, but will usually include bot.

APISite.**has_image_repository**
Return True if site has a shared image repository like Commons.

APISite.**has_mediawiki_message**(*key*)
Determine if the site defines a MediaWiki message.

Parameters **key** (*str*) – name of MediaWiki message

Returns bool

APISite.**has_right**(*right*, *sysop=False*)
Return true if and only if the user has a specific right.

Possible values of 'right' may vary depending on wiki settings, but will usually include:

```
    * Actions: edit, move, delete, protect, upload
    * User levels: autoconfirmed, sysop, bot
```

APISite.**has_transcluded_data**
Return True if site has a shared data repository like Wikidata.

APISite.**image_repository**()
Return Site object for image repository e.g. commons.

APISite.**imageusage**(*image*, *namespaces=None*, *filterredir=None*, *step=None*, *total=None*, *content=False*)
Iterate Pages that contain links to the given FilePage.

Parameters

- **image** (FilePage) – the image to search for (FilePage need not exist on the wiki)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – If present, only iterate pages in these namespaces
- **filterredir** – if True, only yield redirects; if False (and not None), only yield non-redirects (default: yield both)
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**isAllowed**(**a*, ***kw*)

APISite.**isBlocked**(**a*, ***kw*)

APISite.**isBot**(*username*)
Return True is username is a bot user.

APISite.**is_blocked**(*sysop=False*)
Return True when logged in user is blocked.

To check whether a user can perform an action, the method has_right should be used.

Parameters **sysop** (*bool*) – If true, log in to sysop account (if available)

Return type bool

APISite.**is_data_repository**()
Return True if Site object is the data repository.

APISite.**is_image_repository**()
Return True if Site object is the image repository.

APISite.**is_uploaddisabled**()
Return True if upload is disabled on site.

If not called directly, it is cached by the first attempted upload action.

APISite.**lang**
Return the code for the language of this Site.

APISite.**language**()
Return the code for the language of this Site.

APISite.**linksearch**(*siteurl*, *limit=None*)
Backwards-compatible interface to exturlusage().

APISite.**list_to_text**(*args*)
Convert a list of strings into human-readable text.

The MediaWiki message 'and' is used as separator between the last two arguments. If present, other arguments are joined using a comma.

Parameters **args** (*iterable*) – text to be expanded

Returns unicode

APISite.**live_version**(*force=False*)
Return the 'real' version number found on [[Special:Version]].

By default the version number is cached for one day.

Parameters **force** (*bool*) – If the version should be read always from the server and never from the cache.

Returns A tuple containing the major, minor version number and any text after that. If an error occured (0, 0, 0) is returned.

Return type int, int, str

APISite.**loadcoordinfo**(*page*)
Load [[mw:Extension:GeoData]] info.

APISite.**loadflowinfo**(*page*)
Load Flow-related information about a given page.

FIXME: Assumes that the Flow extension is installed.

APISite.**loadimageinfo**(*page*, *history=False*)
Load image info from api and save in page attributes.

Parameters **history** – if true, return the image's version history

APISite.**loadpageinfo**(*page*, *preload=False*)
Load page info from api and store in page attributes.

APISite.**loadpageprops**(*page*)
Load page props for the given page.

APISite.**loadrevisions**(*page*, *getText=False*, *revids=None*, *startid=None*, *endid=None*, *starttime=None*, *endtime=None*, *rvdir=None*, *user=None*, *excludeuser=None*, *section=None*, *sysop=False*, *step=None*, *total=None*, *rollback=False*)
Retrieve and store revision information.

By default, retrieves the last (current) revision of the page, unless any of the optional parameters revids, startid, endid, starttime, endtime, rvdir, user, excludeuser, or limit are specified. Unless noted below, all parameters not specified default to False.

If rvdir is False or not specified, startid must be greater than endid if both are specified; likewise, starttime must be greater than endtime. If rvdir is True, these relationships are reversed.

Parameters

- **page** – retrieve revisions of this Page (required unless ids is specified)
- **getText** – if True, retrieve the wiki-text of each revision; otherwise, only retrieve the revision metadata (default)
- **section** (*int*) – if specified, retrieve only this section of the text (getText must be True); section must be given by number (top of the article is section 0), not name
- **revids** (*an int, a str or a list of ints or strings*) – retrieve only the specified revision ids (raise Exception if any of revids does not correspond to page
- **startid** – retrieve revisions starting with this revid
- **endid** – stop upon retrieving this revid
- **starttime** – retrieve revisions starting at this Timestamp
- **endtime** – stop upon reaching this Timestamp

- **rvdir** – if false, retrieve newest revisions first (default); if true, retrieve earliest first
- **user** – retrieve only revisions authored by this user
- **excludeuser** – retrieve all revisions not authored by this user
- **sysop** – if True, switch to sysop account (if available) to retrieve this page

APISite.**logevents**(*logtype=None*, *user=None*, *page=None*, *namespace=None*, *start=None*, *end=None*, *reverse=False*, *step=None*, *total=None*)

Iterate all log entries.

Parameters

- **logtype** – only iterate entries of this type (see wiki documentation for available types, which will include "block", "protect", "rights", "delete", "upload", "move", "import", "patrol", "merge")
- **user** – only iterate entries that match this user name
- **page** – only iterate entries affecting this page
- **namespace** (*int or Namespace*) – namespace to retrieve logevents from
- **start** – only iterate entries from and after this Timestamp
- **end** – only iterate entries up to and through this Timestamp
- **reverse** – if True, iterate oldest entries first (default: newest)

Raises

- **KeyError** – the namespace identifier was not resolved
- **TypeError** – the namespace identifier has an inappropriate type such as bool, or an iterable with more than one namespace

APISite.**loggedInAs**(*sysop=False*)

Return the current username if logged in, otherwise return None.

DEPRECATED (use .user() method instead)

Parameters **sysop** (*bool*) – if True, test if user is logged in as the sysop user instead of the normal user.

Returns bool

APISite.**logged_in**(*sysop=False*)

Verify the bot is logged into the site as the expected user.

The expected usernames are those provided as either the user or sysop parameter at instantiation.

Parameters **sysop** (*bool*) – if True, test if user is logged in as the sysop user instead of the normal user.

Returns bool

APISite.**login**(*sysop=False*)

Log the user in if not already logged in.

APISite.**logout**()

Logout of the site and load details for the logged out user.

Also logs out of the global account if linked to the user.

APISite.**lonelypages**(*step=None*, *total=None*)

Yield Pages retrieved from Special:Lonelypages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**longpages**(*step=None*, *total=None*)
Yield Pages and lengths from Special:Longpages.

Yields a tuple of Page object, length(int).

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**mediawiki_message**(*key*)
Fetch the text for a MediaWiki message.

Parameters **key** (*str*) – name of MediaWiki message

Returns unicode

APISite.**mediawiki_messages**(*keys*)
Fetch the text of a set of MediaWiki messages.

If keys is '*' or ['*'], all messages will be fetched. The returned dict uses each key to store the associated message.

Parameters **keys** (*set of str, '*' or ['*']*) – MediaWiki messages to fetch

Returns dict

APISite.**messages**(*sysop=False*)
Return true if the user has new messages, and false otherwise.

APISite.**months_names**
Obtain month names from the site messages.

The list is zero-indexed, ordered by month in calendar, and should be in the original site language.

Returns list of tuples (month name, abbreviation)

APISite.**movepage**(*page*, *newtitle*, *summary*, *movetalk=True*, *noredirect=False*)
Move a Page to a new title.

Parameters

- **page** – the Page to be moved (must exist)
- **newtitle** (*unicode*) – the new title for the Page
- **summary** – edit summary (required!)
- **movetalk** – if True (default), also move the talk page if possible
- **noredirect** – if True, suppress creation of a redirect from the old title to the new one

Returns Page object with the new title

APISite.**namespace**(*num*, *all=False*)
Return string containing local name of namespace 'num'.

If optional argument 'all' is true, return a list of all recognized values for this namespace.

APISite.**newfiles**(*user=None*, *start=None*, *end=None*, *reverse=False*, *step=None*, *total=None*)
Yield information about newly uploaded files.

Yields a tuple of FilePage, Timestamp, user(unicode), comment(unicode).

N.B. the API does not provide direct access to Special:Newimages, so this is derived from the "upload" log events instead.

APISite.**newimages**(**args*, ***kwargs*)
Yield information about newly uploaded files.

DEPRECATED: Use newfiles() instead.

APISite.**newpages**(*user=None*, *returndict=False*, *start=None*, *end=None*, *reverse=False*, *showBot=False*, *showRedirects=False*, *excludeuser=None*, *showPatrolled=None*, *namespaces=None*, *step=None*, *total=None*)
Yield new articles (as Page objects) from recent changes.

Starts with the newest article and fetches the number of articles specified in the first argument.

The objects yielded are dependent on parameter returndict. When true, it yields a tuple composed of a Page object and a dict of attributes. When false, it yields a tuple composed of the Page object, timestamp (unicode), length (int), an empty unicode string, username or IP address (str), comment (unicode).

Parameters **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – only iterate pages in these namespaces

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**nice_get_address**(*title*)
Return shorter URL path to retrieve page titled 'title'.

APISite.**notifications**(***kwargs*)
Yield Notification objects from the Echo extension.

APISite.**notifications_mark_read**(***kwargs*)
Mark selected notifications as read.

Returns whether the action was successful

Return type bool

APISite.**page_can_be_edited**(*page*)
Determine if the page can be edited.

Return True if and only if::

- page is unprotected, and bot has an account for this site, or
- page is protected, and bot has a sysop account for this site.

Returns bool

APISite.**page_embeddedin**(*page*, *filterRedirects=None*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Iterate all pages that embedded the given page as a template.

Parameters

- **page** – The Page to get inclusions for.

- **filterRedirects** – If True, only return redirects that embed the given page. If False, only return non-redirect links. If None, return both (no filtering).
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – If present, only return links from the namespaces in this list.
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**page_exists**(*page*)
Return True if and only if page is an existing page on site.

APISite.**page_extlinks**(*page*, *step=None*, *total=None*)
Iterate all external links on page, yielding URL strings.

APISite.**page_isredirect**(*page*)
Return True if and only if page is a redirect.

APISite.**page_restrictions**(*page*)
Return a dictionary reflecting page protections.

APISite.**pagebacklinks**(*page*, *followRedirects=False*, *filterRedirects=None*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Iterate all pages that link to the given page.

Parameters

- **page** – The Page to get links to.
- **followRedirects** – Also return links to redirects pointing to the given page.
- **filterRedirects** – If True, only return redirects to the given page. If False, only return non-redirect links. If None, return both (no filtering).
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – If present, only return links from the namespaces in this list.
- **step** – Limit on number of pages to retrieve per API query.
- **total** – Maximum number of pages to retrieve in total.
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**pagecategories**(*page*, *step=None*, *total=None*, *content=False*)
Iterate categories to which page belongs.

Parameters **content** – if True, load the current content of each iterated page (default False); note that this means the contents of the category description page, not the pages contained in the category

APISite.**pageimages**(*page*, *step=None*, *total=None*, *content=False*)
Iterate images used (not just linked) on the page.

Parameters **content** – if True, load the current content of each iterated page (default False); note that this means the content of the image description page, not the image itself

APISite.**pageinterwiki**(*page*)

APISite.**pagelanglinks**(*page*, *step=None*, *total=None*, *include_obsolete=False*)
Iterate all interlanguage links on page, yielding Link objects.

Parameters **include_obsolete** – if true, yield even Link objects whose site is obsolete

APISite.**pagelinks**(*page*, *namespaces=None*, *follow_redirects=False*, *step=None*, *total=None*, *content=False*)
Iterate internal wikilinks contained (or transcluded) on page.

Parameters

- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – Only iterate pages in these namespaces (default: all)
- **follow_redirects** – if True, yields the target of any redirects, rather than the redirect page
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**pagename2codes**()
Return list of localized PAGENAMEE tags for the site.

APISite.**pagenamecodes**()
Return list of localized PAGENAME tags for the site.

APISite.**pagereferences**(*page*, *followRedirects=False*, *filterRedirects=None*, *withTemplateInclusion=True*, *onlyTemplateInclusion=False*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Convenience method combining pagebacklinks and page_embeddedin.

Parameters **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – If present, only return links from the namespaces in this list.

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**pagetemplates**(*page*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Iterate templates transcluded (not just linked) on the page.

Parameters

- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – Only iterate pages in these namespaces
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved

- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**patrol**(*rcid=None*, *revid=None*, *revision=None*)
Return a generator of patrolled pages.

Pages to be patrolled are identified by rcid, revid or revision. At least one of the parameters is mandatory. See https://www.mediawiki.org/wiki/API:Patrol.

Parameters

- **rcid** (*iterable/iterator which returns a number or string which contains only digits; it also supports a string (as above) or int*) – an int/string/iterable/iterator providing rcid of pages to be patrolled.
- **revid** (*iterable/iterator which returns a number or string which contains only digits; it also supports a string (as above) or int.*) – an int/string/iterable/iterator providing revid of pages to be patrolled.
- **revision** (*iterable/iterator which returns a Revision object; it also supports a single Revision.*) – an Revision/iterable/iterator providing Revision object of pages to be patrolled.

Return type iterator of dict with 'rcid', 'ns' and 'title' of the patrolled page.

APISite.**prefixindex**(*prefix*, *namespace=0*, *includeredirects=True*)
Yield all pages with a given prefix. Deprecated.

Use allpages() with the prefix= parameter instead of this method.

APISite.**preloadpages**(*pagelist*, *groupsize=50*, *templates=False*, *langlinks=False*)
Return a generator to a list of preloaded pages.

Note that [at least in current implementation] pages may be iterated in a different order than in the underlying pagelist.

Parameters

- **pagelist** – an iterable that returns Page objects
- **groupsize** (*int*) – how many Pages to query at a time
- **templates** – preload list of templates in the pages
- **langlinks** – preload list of language links found in the pages

APISite.**proofread_index_ns**
Return Index namespace for the ProofreadPage extension.

APISite.**proofread_levels**
Return Quality Levels for the ProofreadPage extension.

APISite.**proofread_page_ns**
Return Page namespace for the ProofreadPage extension.

APISite.**protect**(*page*, *protections*, *reason*, *expiry=None*, ***kwargs*)
(Un)protect a wiki page. Requires administrator status.

Parameters

- **protections** (*dict*) – A dict mapping type of protection to protection level of that type. Valid types of protection are 'edit', 'move', 'create', and 'upload'. Valid protection levels (in MediaWiki 1.12) are '' (equivalent to 'none'), 'autoconfirmed', and 'sysop'. If None is given, however, that protection will be skipped.
- **reason** (*basestring*) – Reason for the action

- **expiry** (*pywikibot.Timestamp, string in GNU timestamp format (including ISO 8601).*) – When the block should expire. This expiry will be applied to all protections. If None, 'infinite', 'indefinite', 'never', or '' is given, there is no expiry.

APISite.**protectedpages**(*namespace=0*, *type='edit'*, *level=False*, *total=None*)
Return protected pages depending on protection level and type.

For protection types which aren't 'create' it uses *APISite.allpages*, while it uses for 'create' the 'query+protectedtitles' module.

Parameters

- **namespaces** (*int or Namespace or str*) – The searched namespace.
- **type** (*str*) – The protection type to search for (default 'edit').
- **level** (*str or False*) – The protection level (like 'autoconfirmed'). If False it shows all protection levels.

Returns The pages which are protected.

Return type generator of Page

APISite.**protection_levels**()
Return the protection levels available on this site.

Returns protection types available

Return type set of unicode instances

See *Siteinfo._get_default()*

APISite.**protection_types**()
Return the protection types available on this site.

Returns protection types available

Return type set of unicode instances

See *Siteinfo._get_default()*

APISite.**purgepages**(*pages*, ***kwargs*)
Purge the server's cache for one or multiple pages.

Parameters **pages** – list of Page objects

Returns True if API returned expected response; False otherwise

APISite.**randompage**(*redirect=False*)
DEPRECATED.

Parameters **redirect** – Return a random redirect page

Returns pywikibot.Page

APISite.**randompages**(*step=None*, *total=10*, *namespaces=None*, *redirects=False*, *content=False*)
Iterate a number of random pages.

Pages are listed in a fixed sequence, only the starting point is random.

Parameters

- **total** – the maximum number of pages to iterate (default: 1)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a* '|' *separated list of namespace identifiers.*) – only iterate pages in these namespaces.

- **redirects** – if True, include only redirect pages in results (default: include only non-redirects)
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**randomredirectpage**()
DEPRECATED: Use Site.randompages() instead.

Returns Return a random redirect page

APISite.**recentchanges**(*start=None*, *end=None*, *reverse=False*, *namespaces=None*, *pagelist=None*, *changetype=None*, *showMinor=None*, *showBot=None*, *showAnon=None*, *showRedirects=None*, *showPatrolled=None*, *topOnly=False*, *step=None*, *total=None*, *user=None*, *excludeuser=None*)
Iterate recent changes.

Parameters

- **start** (*pywikibot.Timestamp*) – Timestamp to start listing from
- **end** (*pywikibot.Timestamp*) – Timestamp to end listing at
- **reverse** (*bool*) – if True, start with oldest changes (default: newest)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – only iterate pages in these namespaces
- **pagelist** – iterate changes to pages in this list only
- **pagelist** – list of Pages
- **changetype** (*basestring*) – only iterate changes of this type ("edit" for edits to existing pages, "new" for new pages, "log" for log entries)
- **showMinor** (*bool or None*) – if True, only list minor edits; if False, only list non-minor edits; if None, list all
- **showBot** (*bool or None*) – if True, only list bot edits; if False, only list non-bot edits; if None, list all
- **showAnon** (*bool or None*) – if True, only list anon edits; if False, only list non-anon edits; if None, list all
- **showRedirects** (*bool or None*) – if True, only list edits to redirect pages; if False, only list edits to non-redirect pages; if None, list all
- **showPatrolled** (*bool or None*) – if True, only list patrolled edits; if False, only list non-patrolled edits; if None, list all
- **topOnly** (*bool*) – if True, only list changes that are the latest revision (default False)
- **user** (*basestring|list*) – if not None, only list edits by this user or users
- **excludeuser** (*basestring|list*) – if not None, exclude edits by this user or users

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**redirect**()
Return the localized #REDIRECT keyword.

APISite.**redirectRegex**()
Return a compiled regular expression matching on redirect pages.

Group 1 in the regex match object will be the target title.

APISite.**redirectpages**(*step=None*, *total=None*)
Yield redirect pages from Special:ListRedirects.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**rollbackpage**(*page*, ***kwargs*)
Roll back page to version before last user's edits.

The keyword arguments are those supported by the rollback API.

As a precaution against errors, this method will fail unless the page history contains at least two revisions, and at least one that is not by the same user who made the last edit.

Parameters **page** – the Page to be rolled back (must exist)

APISite.**search**(*searchstring*, *namespaces=None*, *where='text'*, *getredirects=False*, *step=None*, *total=None*, *content=False*)
Iterate Pages that contain the searchstring.

Note that this may include non-existing Pages if the wiki's database table contains outdated entries.

Parameters

- **searchstring** (*unicode*) – the text to search for
- **where** – Where to search; value must be "text" or "titles" (many wikis do not support title search)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers.*) – search only in these namespaces (defaults to all)
- **getredirects** – if True, include redirects in results. Since version MediaWiki 1.23 it will always return redirects.
- **content** – if True, load the current content of each iterated page (default False)

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**shortpages**(*step=None*, *total=None*)
Yield Pages and lengths from Special:Shortpages.

Yields a tuple of Page object, length(int).

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**siteinfo**
Site information dict.

APISite.**token**(*page*, *tokentype*)
Return token retrieved from wiki to allow changing page content.

Parameters

- **page** – the Page for which a token should be retrieved
- **tokentype** – the type of token (e.g., "edit", "move", "delete"); see API documentation for full list of types

APISite.**unblockuser**(*user*, *reason*)
Remove the block for the user.

Parameters

- **user** (pywikibot.User) – The username/IP without a namespace.
- **reason** (*basestring*) – Reason for the unblock.

APISite.**uncategorizedcategories**(*number=None*, *repeat=True*, *step=None*, *total=None*)
Yield Categories from Special:Uncategorizedcategories.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**uncategorizedfiles**(*number=None*, *repeat=True*, *step=None*, *total=None*)
Yield FilePages from Special:Uncategorizedimages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**uncategorizedimages**(*number=None*, *repeat=True*, *step=None*, *total=None*)
Yield FilePages from Special:Uncategorizedimages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**uncategorizedpages**(*number=None*, *repeat=True*, *step=None*, *total=None*)
Yield Pages from Special:Uncategorizedpages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**uncategorizedtemplates**(*number=None*, *repeat=True*, *step=None*, *total=None*)
Yield Pages from Special:Uncategorizedtemplates.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**undelete_page**(*page*, *reason*, *revisions=None*)
Undelete page from the wiki. Requires appropriate privilege level.

Parameters

- **page** (Page) – Page to be deleted.
- **revisions** (*list*) – List of timestamps to restore. If None, restores all revisions.
- **reason** (*basestring*) – Undeletion reason.

APISite.**unusedcategories**(*step=None*, *total=None*)
Yield Category objects from Special:Unusedcategories.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**unusedfiles**(*step=None*, *total=None*)
Yield FilePage objects from Special:Unusedimages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**unusedimages**(**args*, ***kwargs*)
Yield FilePage objects from Special:Unusedimages.

DEPRECATED: Use *APISite.unusedfiles* instead.

APISite.**unwatchedpages**(*step=None*, *total=None*)
Yield Pages from Special:Unwatchedpages (requires Admin privileges).

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**upload**(*filepage*, *source_filename=None*, *source_url=None*, *comment=None*, *text=None*, *watch=False*, *ignore_warnings=False*, *chunk_size=0*)
Upload a file to the wiki.

Either source_filename or source_url, but not both, must be provided.

Parameters

- **filepage** – a FilePage object from which the wiki-name of the file will be obtained.
- **source_filename** – path to the file to be uploaded
- **source_url** – URL of the file to be uploaded
- **comment** – Edit summary; if this is not provided, then filepage.text will be used. An empty summary is not permitted. This may also serve as the initial page text (see below).
- **text** – Initial page text; if this is not set, then filepage.text will be used, or comment.
- **watch** – If true, add filepage to the bot user's watchlist
- **ignore_warnings** – if true, ignore API warnings and force upload (for example, to overwrite an existing file); default False
- **chunk_size** (*int*) – The chunk size in bytesfor chunked uploading (see https://www.mediawiki.org/wiki/API:Upload#Chunked_uploading). It will only upload in chunks, if the version number is 1.20 or higher and the chunk size is positive but lower than the file size.

APISite.**usercontribs**(*user=None*, *userprefix=None*, *start=None*, *end=None*, *reverse=False*, *namespaces=None*, *showMinor=None*, *step=None*, *total=None*, *top_only=False*)

Iterate contributions by a particular user.

Iterated values are in the same format as recentchanges.

Parameters

- **user** – Iterate contributions by this user (name or IP)
- **userprefix** – Iterate contributions by all users whose names or IPs start with this substring
- **start** – Iterate contributions starting at this Timestamp
- **end** – Iterate contributions ending at this Timestamp
- **reverse** – Iterate oldest contributions first (default: newest)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a* '|' *separated list of namespace identifiers.*) – only iterate pages in these namespaces
- **showMinor** – if True, iterate only minor edits; if False and not None, iterate only non-minor edits (default: iterate both)
- **top_only** – if True, iterate only edits which are the latest revision

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**userinfo**

Retrieve userinfo from site and store in _userinfo attribute.

self._userinfo will be a dict with the following keys and values:

```
- id: user id (numeric str)
- name: username (if user is logged in)
- anon: present if user is not logged in
- groups: list of groups (could be empty)
- rights: list of rights (could be empty)
- message: present if user has a new message on talk page
- blockinfo: present if user is blocked (dict)
```

APISite.**users**(*usernames*)

Iterate info about a list of users by name or IP.

Parameters **usernames** (*list, or other iterable, of unicodes*) – a list of user names

APISite.**validate_tokens**(*types*)

Validate if requested tokens are acceptable.

Valid tokens depend on mw version.

APISite.**version**()

Return live project version number as a string.

This overwrites the corresponding family method for APISite class. Use *pywikibot.tools.MediaWikiVersion* to compare MediaWiki versions.

APISite.**wantedcategories**(*step=None*, *total=None*)

Yield Pages from Special:Wantedcategories.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**wantedpages**(*step=None*, *total=None*)
Yield Pages from Special:Wantedpages.

Parameters

- **step** – request batch size
- **total** – number of pages to return

APISite.**watchlist_revs**(*start=None*, *end=None*, *reverse=False*, *namespaces=None*, *showMinor=None*, *showBot=None*, *showAnon=None*, *step=None*, *total=None*)
Iterate revisions to pages on the bot user's watchlist.

Iterated values will be in same format as recentchanges.

Parameters

- **start** – Iterate revisions starting at this Timestamp
- **end** – Iterate revisions ending at this Timestamp
- **reverse** – Iterate oldest revisions first (default: newest)
- **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a* '|' *separated list of namespace identifiers.*) – only iterate pages in these namespaces
- **showMinor** – if True, only list minor edits; if False (and not None), only list non-minor edits
- **showBot** – if True, only list bot edits; if False (and not None), only list non-bot edits
- **showAnon** – if True, only list anon edits; if False (and not None), only list non-anon edits

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

APISite.**watchpage**(*page*, *unwatch=False*)
Add or remove page from watchlist.

Parameters **unwatch** – If True, remove page from watchlist; if False (default), add it.

Returns True if API returned expected response; False otherwise

APISite.**withoutinterwiki**(*step=None*, *total=None*)
Yield Pages without language links from Special:Withoutinterwiki.

Parameters

- **step** – request batch size
- **total** – number of pages to return

class pywikibot.site.**BaseSite**(*code*, *fam=None*, *user=None*, *sysop=None*)
Bases: *pywikibot.tools.ComparableMixin*

Site methods that are independent of the communication interface.

_build_namespaces()
Create default namespaces.

_cache_interwikimap(*force=False*)
Cache the interwikimap with usable site instances.

_cmpkey()
Perform equality and inequality tests on Site objects.

category_namespace()
Return local name for the Category namespace.

category_namespaces()
Return names for the Category namespace.

category_on_one_line()
Return True if this site wants all category links on one line.

code
The identifying code for this Site.

By convention, this is usually an ISO language code, but it does not have to be.

disambcategory()
Return Category in which disambig pages are listed.

doc_subpage
Return the documentation subpage for this Site.

Returns tuple

fam()
Return Family object for this Site.

family
The Family object for this Site's wiki family.

getNamespaceIndex(**a*, ***kw*)

getSite(*code*)
Return Site object for language 'code' in this Family.

getUrl(*path*, *retry=True*, *sysop=False*, *data=None*, *compress=True*, *no_hostname=False*, *cookie_only=False*)
DEPRECATED.

Retained for compatibility only. All arguments except path and data are ignored.

image_namespace()
Return local name for the File namespace.

interwiki(*prefix*)
Return the site for a corresponding interwiki prefix.

@**raise SiteDefinitionError: if the url given in the interwiki table** doesn't match any of the existing families.

@raise KeyError: if the prefix is not an interwiki prefix.

interwiki_prefix(*site*)
Return the interwiki prefixes going to that site.

The interwiki prefixes are ordered first by length (shortest first) and then alphabetically.

Parameters **site** (*BaseSite*) – The targeted site, which might be it's own.

Returns The interwiki prefixes

Return type list (guaranteed to be not empty)

@raise KeyError: if there is no interwiki prefix for that site.

interwiki_putfirst()
Return list of language codes for ordering of interwiki links.

interwiki_putfirst_doubled(*list_of_links*)

isInterwikiLink(*text*)
Return True if text is in the form of an interwiki link.

If a link object constructed using "text" as the link text parses as belonging to a different site, this method returns True.

lang
The ISO language code for this Site.

Presumed to be equal to the wiki prefix, but this can be overridden.

languages()
Return list of all valid language codes for this site's Family.

linkto(*title*, *othersite=None*)
DEPRECATED. Return a wikilink to a page.

Parameters

- **title** (*unicode*) – Title of the page to link to
- **othersite** (*Site (optional)*) – Generate a interwiki link for use on this site.

Returns unicode

local_interwiki(*prefix*)
Return whether the interwiki prefix is local.

A local interwiki prefix is handled by the target site like a normal link. So if that link also contains an interwiki link it does follow it as long as it's a local link.

@**raise SiteDefinitionError: if the url given in the interwiki table** doesn't match any of the existing families.

@raise KeyError: if the prefix is not an interwiki prefix.

lock_page(*page*, *block=True*)
Lock page for writing. Must be called before writing any page.

We don't want different threads trying to write to the same page at the same time, even to different sections.

Parameters

- **page** (*pywikibot.Page*) – the page to be locked
- **block** – if true, wait until the page is available to be locked; otherwise, raise an exception if page can't be locked

mediawiki_namespace()
Return local name for the MediaWiki namespace.

namespaces
Return dict of valid namespaces on this wiki.

nocapitalize

normalizeNamespace(**a*, ***kw*)

ns_index(*namespace*)
Return the Namespace for a given namespace name.

Parameters **namespace** (*unicode*) – name

Returns The matching Namespace object on this Site

Return type Namespace, or None if invalid

ns_normalize(*value*)
Return canonical local form of namespace name.

Parameters **value** (*unicode*) – A namespace name

pagename2codes()
Return list of localized PAGENAMEE tags for the site.

pagenamecodes()
Return list of localized PAGENAME tags for the site.

postData(*address*, *data*, *contentType=None*, *sysop=False*, *compress=True*, *cookies=None*)
DEPRECATED.

postForm(*address*, *predata*, *sysop=False*, *cookies=None*)
DEPRECATED.

redirect()
Return list of localized redirect tags for the site.

redirectRegex(*pattern=None*)
Return a compiled regular expression matching on redirect pages.

Group 1 in the regex match object will be the target title.

sametitle(*title1*, *title2*)
Return True if title1 and title2 identify the same wiki page.

title1 and title2 may be unequal but still identify the same page, if they use different aliases for the same namespace.

sitename
String representing this Site's name and code.

special_namespace()
Return local name for the Special: namespace.

template_namespace()
Return local name for the Template namespace.

throttle
Return this Site's throttle. Initialize a new one if needed.

unlock_page(*page*)
Unlock page. Call as soon as a write operation has completed.

Parameters **page** (*pywikibot.Page*) – the page to be locked

urlEncode(*query*)
DEPRECATED.

user()
Return the currently-logged in bot user, or None.

username(*sysop=False*)
Return the username/sysopname used for the site.

validLanguageLinks()
Return list of language codes that can be used in interwiki links.

class pywikibot.site.**DataSite**(**args*, ***kwargs*)
Bases: *pywikibot.site.APISite*

Wikibase data capable site.

_cache_entity_namespaces()
Find namespaces for each known wikibase entity type.

_get_propertyitem(*props*, *source*, ***params*)
Generic method to get the data for multiple Wikibase items.

addClaim(*item*, *claim*, *bot=True*, ***kwargs*)

categories(**args*, ***kwargs*)

changeClaimTarget(*claim*, *snaktype='value'*, *bot=True*, ***kwargs*)
Set the claim target to the value of the provided claim target.

Parameters

- **claim** (Claim) – The source of the claim target value
- **snaktype** (*str ('value', 'novalue' or 'somevalue')*) – An optional snaktype. Default: 'value'

checkBlocks(**args*, ***kwargs*)

createNewItemFromPage(*page*, *bot=True*, ***kwargs*)
Create a new Wikibase item for a provided page.

Parameters

- **page** (*pywikibot.Page*) – page to fetch links from
- **bot** – whether to mark the edit as bot

Returns pywikibot.ItemPage of newly created item

editEntity(*identification*, *data*, *bot=True*, ***kwargs*)

editQualifier(*claim*, *qualifier*, *new=False*, *bot=True*, ***kwargs*)
Create/Edit a qualifier.

Parameters

- **claim** (Claim) – A Claim object to add the qualifier to
- **qualifier** (Claim) – A Claim object to be used as a qualifier

editSource(*claim*, *source*, *new=False*, *bot=True*, ***kwargs*)
Create/Edit a source.

Parameters

- **claim** (Claim) – A Claim object to add the source to
- **source** (Claim) – A Claim object to be used as a source
- **new** (*bool*) – Whether to create a new one if the "source" already exists

fam()

getPropertyType(*prop*)
Obtain the type of a property.

This is used specifically because we can cache the value for a much longer time (near infinite).

getUrl(**args*, ***kwargs*)

get_item(*source*, ***params*)
Get the data for multiple Wikibase items.

isAllowed(**args*, ***kwargs*)

isBlocked(**args*, ***kwargs*)

item_namespace
Return namespace for items.

Returns item namespace

Return type *Namespace*

linkTitles(*page1*, *page2*, *bot=True*)
Link two pages together.

Parameters

- **page1** (*pywikibot.Page*) – First page to link
- **page2** (*pywikibot.Page*) – Second page to link
- **bot** – whether to mark edit as bot

Returns dict API output

linksearch(**args*, ***kwargs*)

linkto(**args*, ***kwargs*)

loadcontent(*identification*, **props*)
Fetch the current content of a Wikibase item.

This is called loadcontent since wbgetentities does not support fetching old revisions. Eventually this will get replaced by an actual loadrevisions.

Parameters

- **identification** (*dict*) – Parameters used to identify the page(s)
- **props** – the optional properties to fetch.

loggedInAs(**args*, ***kwargs*)

mergeItems(*fromItem*, *toItem*, ***kwargs*)
Merge two items together.

Parameters

- **fromItem** (*pywikibot.ItemPage*) – Item to merge from
- **toItem** (*pywikibot.ItemPage*) – Item to merge into

Returns dict API output

newimages(**args*, ***kwargs*)

postData(**args*, ***kwargs*)

postForm(**args*, ***kwargs*)

prefixindex(**args*, ***kwargs*)

preloaditempages(*pagelist*, *groupsize=50*)
Yield ItemPages with content prefilled.

Note that pages will be iterated in a different order than in the underlying pagelist.

Parameters

- **pagelist** – an iterable that yields either WikibasePage objects, or Page objects linked to an ItemPage.
- **groupsize** (*int*) – how many pages to query at a time

property_namespace
Return namespace for properties.

Returns property namespace

Return type *Namespace*

removeClaims(*claims*, *bot=True*, ***kwargs*)

removeSources(*claim*, *sources*, *bot=True*, ***kwargs*)
Remove sources.

Parameters

- **claim** (Claim) – A Claim object to remove the sources from
- **sources** (Claim) – A list of Claim objects that are sources

save_claim(*claim*, ***kwargs*)
Save the whole claim to the wikibase site.

Parameters **claim** (Claim) – The claim to save

search_entities(*search*, *language*, *limit=None*, ***kwargs*)
Search for pages or properties that contain the given text.

Parameters

- **search** (*str*) – Text to find.
- **language** (*str*) – Language to search in.
- **limit** (*int or None*) – Maximum number of pages to retrieve in total, or None in case of no limit.

Returns 'search' list from API output.

set_redirect_target(*from_item*, *to_item*)
Make a redirect to another item.

Parameters

- **to_item** (*pywikibot.ItemPage*) – title of target item.
- **from_item** (*pywikibot.ItemPage*) – Title of the item to be redirected.

urlEncode(**args*, ***kwargs*)

class pywikibot.site.**LoginStatus**(*state*)
Bases: object

Enum for Login statuses.

```
>>> LoginStatus.NOT_ATTEMPTED
 -3
>>> LoginStatus.AS_USER
0
>>> LoginStatus.name(-3)
'NOT_ATTEMPTED'
>>> LoginStatus.name(0)
'AS_USER'
```

AS_SYSOP = 1

AS_USER = 0

IN_PROGRESS = -2

NOT_ATTEMPTED = -3

NOT_LOGGED_IN = -1

classmethod name (*search_value*)
Return the name of a LoginStatus by it's value.

class pywikibot.site.**Namespace**(*id*, *canonical_name=None*, *custom_name=None*, *aliases=None*, *use_image_name=False*, ***kwargs*)
Bases: collections.abc.Iterable, *pywikibot.tools.ComparableMixin*, *pywikibot.tools.UnicodeMixin*

Namespace site data object.

This is backwards compatible with the structure of entries in site._namespaces which were a list of::

```
[customised namespace,
 canonical namespace name?,
 namespace alias*]
```

If the canonical_name is not provided for a namespace between -2 and 15, the MediaWiki 1.14+ built-in names are used. Enable use_image_name to use built-in names from MediaWiki 1.13 and earlier as the details.

Image and File are aliases of each other by default.

If only one of canonical_name and custom_name are available, both properties will have the same value.

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 28

_abc_registry = <_weakrefset.WeakSet object>

_cmpkey()
Return the ID as a comparison key.

static _colons(*id*, *name*)
Return the name with required colons, depending on the ID.

_contains_lowercase_name(*name*)
Determine a lowercase normalised name is a name of this namespace.

Return type bool

_distinct()

classmethod builtin_namespaces(*use_image_name=False*, *case='first-letter'*)
Return a dict of the builtin namespaces.

canonical_namespaces = {0: '', 1: 'Talk', 2: 'User', 3: 'User talk', 4: 'Project', 5: 'Project talk', 6: 'File', 7: 'File tal

canonical_prefix()
Return the canonical name with required colons.

custom_prefix()
Return the custom name with required colons.

static **default_case**(*id*, *default_case=None*)
Return the default fixed case value for the namespace ID.

classmethod **lookup_name**(*name*, *namespaces=None*)
Find the Namespace for a name.

Parameters

- **name** (*basestring*) – Name of the namespace.
- **namespaces** (*dict of Namespace*) – namespaces to search default: builtins only

Returns Namespace or None

static **normalize_name**(*name*)
Remove an optional colon before and after name.

TODO: reject illegal characters.

static **resolve**(*identifiers*, *namespaces=None*)
Resolve namespace identifiers to obtain Namespace objects.

Identifiers may be any value for which int() produces a valid namespace id, except bool, or any string which Namespace.lookup_name successfully finds. A numerical string is resolved as an integer.

Parameters

- **identifiers** (*iterable of basestring or Namespace key, or a single instance of those types*) – namespace identifiers
- **namespaces** (*dict of Namespace*) – namespaces to search (default: builtins only)

Returns list of Namespace objects in the same order as the identifiers

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool

class pywikibot.site.**NonMWAPISite**(*url*)
Bases: *pywikibot.site.BaseSite*

API interface to non MediaWiki sites.

exception pywikibot.site.**PageInUse**(*arg*)
Bases: pywikibot.exceptions.Error

Page cannot be reserved for writing due to existing lock.

class pywikibot.site.**Siteinfo**(*site*)
Bases: collections.abc.Container

A 'dictionary' like container for siteinfo.

This class queries the server to get the requested siteinfo property. Optionally it can cache this directly in the instance so that later requests don't need to query the server.

All values of the siteinfo property 'general' are directly available.

WARNING_REGEX = re.compile("^Unrecognized values? for parameter 'siprop': ([^,]+(?:, [^,]+)*)$")

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 28

_abc_registry = <_weakrefset.WeakSet object>

_get_cached(*key*)
Return the cached value or a KeyError exception if not cached.

static _get_default(*key*)
Return the default value for different properties.

If the property is 'restrictions' it returns a dictionary with::

- 'cascadinglevels': 'sysop'
- 'semiprotectedlevels': 'autoconfirmed'
- 'levels': '' (everybody), 'autoconfirmed', 'sysop'
- 'types': 'create', 'edit', 'move', 'upload'

Otherwise it returns `pywikibot.tools.EMPTY_DEFAULT`.

Parameters **key** (*str*) – The property name

Returns The default value

Return type dict or *pywikibot.tools.EmptyDefault*

_get_general(*key*, *expiry*)
Return a siteinfo property which is loaded by default.

The property 'general' will be queried if it wasn't yet or it's forced. Additionally all uncached default properties are queried. This way multiple default properties are queried with one request. It'll cache always all results.

Parameters

- **key** (*str*) – The key to search for.
- **expiry** (int (days), `datetime.timedelta`, False (never)) – If the cache is older than the expiry it ignores the cache and queries the server to get the newest value.

Returns If that property was retrived via this method. Returns None if the key was not in the retreived values.

Return type various (the value), bool (if the default value is used)

_get_siteinfo(*prop*, *expiry*)
Retrieve a siteinfo property.

All properties which the site doesn't support contain the default value. Because pre-1.12 no data was returned when a property doesn't exists, it queries each property independetly if a property is invalid.

Parameters

- **prop** (*str or iterable*) – The property names of the siteinfo.
- **expiry** (int (days), `datetime.timedelta`, False (config)) – The expiry date of the cached request.

Returns A dictionary with the properties of the site. Each entry in the dictionary is a tuple of the value and a boolean to save if it is the default value.

Return type dict (the values)

See https://www.mediawiki.org/wiki/API:Meta#siteinfo_.2F_si

static _is_expired(*cache_date*, *expire*)
Return true if the cache date is expired.

get(*key*, *get_default=True*, *cache=True*, *expiry=False*)
Return a siteinfo property.

It will never throw an APIError if it only stated, that the siteinfo property doesn't exist. Instead it will use the default value.

Parameters

- **key** (*str*) – The name of the siteinfo property.
- **get_default** (*bool*) – Whether to throw an KeyError if the key is invalid.
- **cache** (*bool*) – Caches the result interally so that future accesses via this method won't query the server.
- **expiry** (int/float (days), `datetime.timedelta`, False (never)) – If the cache is older than the expiry it ignores the cache and queries the server to get the newest value.

Returns The gathered property

Return type various

@raise KeyError: If the key is not a valid siteinfo property and the get_default option is set to False.

See *_get_siteinfo*

get_requested_time(*key*)
Return when 'key' was successfully requested from the server.

If the property is actually in the siprop 'general' it returns the last request from the 'general' siprop.

Parameters **key** (*basestring*) – The siprop value or a property of 'general'.

Returns The last time the siprop of 'key' was requested.

Return type None (never), False (default), `datetime.datetime` (cached)

is_recognised(*key*)
Return if 'key' is a valid property name. 'None' if not cached.

class `pywikibot.site.`**TokenWallet**(*site*)
Bases: `object`

Container for tokens.

load_tokens(*types*, *all=False*)
Preload one or multiple tokens.

Parameters

- **types** (*iterable*) – the types of token.
- **all** (*bool*) – load all available tokens, if None only if it can be done in one request.

class `pywikibot.site.`**_NamespacesDict**
Bases: *pywikibot.tools.SelfCallDict*

A wrapper to add a deprecation message when called.

_own_desc = 'the namespaces property'

pywikibot.site.**must_be**(*group=None*, *right=None*)
Decorator to require a certain user status when method is called.

Parameters

- **group** (*str ('user' or 'sysop')*) – The group the logged in user should belong to this parameter can be overridden by keyword argument 'as_group'.
- **right** – The rights the logged in user should have. Not supported yet and thus ignored.

Returns method decorator

pywikibot.site.**need_version**(*version*)
Decorator to require a certain MediaWiki version number.

Parameters **version** (*str*) – the mw version number required

Returns a decorator to make sure the requirement is satisfied when the decorated function is called.

textlib Module

Functions for manipulating wiki-text.

Unless otherwise noted, all functions take a unicode string as the argument and return a unicode string.

class pywikibot.textlib.**TimeStripper**(*site=None*)
Bases: object

Find timestamp in page and return it as timezone aware datetime object.

findmarker(*text*, *base='@@'*, *delta='@'*)
Find a string which is not part of text.

fix_digits(*line*)
Make non-latin digits like Persian to latin to parse.

last_match_and_replace(*txt*, *pat*)
Take the rightmost match and replace with marker.

It does so to prevent spurious earlier matches.

timestripper(*line*)
Find timestamp in line and convert it to time zone aware datetime.

All the following items must be matched, otherwise None is returned:: -. year, month, hour, time, day, minute, tzinfo

pywikibot.textlib.**categoryFormat**(*categories*, *insite=None*)
Return a string containing links to all categories in a list.

'categories' should be a list of Category or Page objects or strings which can be either the raw name, [[Category:..]] or [[cat_localised_ns:...]].

The string is formatted for inclusion in insite. Category namespace is converted to localised namespace.

pywikibot.textlib.**compileLinkR**(*withoutBracketed=False*, *onlyBracketed=False*)
Return a regex that matches external links.

pywikibot.textlib.**does_text_contain_section**(*pagetext*, *section*)
Determine whether the page text contains the given section title.

It does not care whether a section string may contain spaces or underlines. Both will match.

If a section parameter contains a internal link, it will match the section with or without a preceding colon which is required for a text link e.g. for categories and files.

Parameters

- **pagetext** (*unicode or string*) – The wikitext of a page
- **section** (*unicode or string*) – a section of a page including wikitext markups

pywikibot.textlib.**expandmarker**(*text*, *marker=''*, *separator=''*)

pywikibot.textlib.**extract_templates_and_params**(*text*)

Return a list of templates found in text.

Return value is a list of tuples. There is one tuple for each use of a template in the page, with the template title as the first entry and a dict of parameters as the second entry. Parameters are indexed by strings; as in MediaWiki, an unnamed parameter is given a parameter name with an integer value corresponding to its position among the unnamed parameters, and if this results multiple parameters with the same name only the last value provided will be returned.

This uses the package `mwparserfromhell` (mwpfh) if it is installed and enabled by config.mwparserfromhell. Otherwise it falls back on a regex based implementation.

There are minor differences between the two implementations.

The two implementations return nested templates in a different order. i.e. for {{a|b={{c}}}}, mwpfh returns [a, c], whereas regex returns [c, a].

mwpfh preserves whitespace in parameter names and values. regex excludes anything between <!– –> before parsing the text.

Parameters **text** (*unicode or string*) – The wikitext from which templates are extracted

Returns list of template name and params

Return type list of tuple

pywikibot.textlib.**extract_templates_and_params_mwpfh**(*text*)

Extract templates with params using mwparserfromhell.

This function should not be called directly.

Use extract_templates_and_params, which will select this mwparserfromhell implementation if based on whether the mwparserfromhell package is installed and enabled by config.mwparserfromhell.

Parameters **text** (*unicode or string*) – The wikitext from which templates are extracted

Returns list of template name and params

Return type list of tuple

pywikibot.textlib.**extract_templates_and_params_regex**(*text*)

Extract templates with params using a regex.

This function should not be called directly.

Use extract_templates_and_params, which will fallback to using this regex based implementation when the mwparserfromhell implementation is not used.

Parameters **text** (*unicode or string*) – The wikitext from which templates are extracted

Returns list of template name and params

Return type list of tuple

pywikibot.textlib.**findmarker**(*text*, *startwith='@@'*, *append=None*)

Find a string which is not part of text.

pywikibot.textlib.**getCategoryLinks**(*text*, *site=None*, *include=[]*)
Return a list of category links found in text.

Parameters **include** (*list*) – list of tags which should not be removed by removeDisabledParts() and where CategoryLinks can be searched.

Returns all category links found

Return type list of Category objects

pywikibot.textlib.**getLanguageLinks**(*text*, *insite=None*, *pageLink='[[]]'*, *template_subpage=False*)
Return a dict of inter-language links found in text.

The returned dict uses language codes as keys and Page objects as values.

Do not call this routine directly, use Page.interwiki() method instead.

pywikibot.textlib.**glue_template_and_params**(*template_and_params*)
Return wiki text of template glued from params.

You can use items from extract_templates_and_params here to get an equivalent template wiki text (it may happen that the order of the params changes).

pywikibot.textlib.**interwikiFormat**(*links*, *insite=None*)
Convert interwiki link dict into a wikitext string.

Parameters

- **links** (*dict with the Site objects as keys, and Page or Link objects as values.*) – interwiki links to be formatted
- **insite** (BaseSite) – site the interwiki links will be formatted for (defaulting to the current site).

Returns string including wiki links formatted for inclusion in insite

Return type unicode

pywikibot.textlib.**interwikiSort**(*sites*, *insite=None*)
Sort sites according to local interwiki sort logic.

pywikibot.textlib.**isDisabled**(*text, index, tags=['*']*)
Return True if text[index] is disabled, e.g. by a comment or by nowiki tags.

For the tags parameter, see *removeDisabledParts*.

pywikibot.textlib.**reformat_ISBNs**(*text*, *match_func*)
Reformat ISBNs.

Parameters

- **text** (*str*) – text containing ISBNs
- **match_func** (*callable*) – function to reformat matched ISBNs

Returns reformatted text

Return type str

pywikibot.textlib.**removeCategoryLinks**(*text*, *site=None*, *marker=''*)
Return text with all category links removed.

Put the string marker after the last replacement (at the end of the text if there is no replacement).

pywikibot.textlib.**removeCategoryLinksAndSeparator**(*text*, *site=None*, *marker=''*, *separator=''*)
Return text with all category links and preceding separators removed.

Put the string marker after the last replacement (at the end of the text if there is no replacement).

pywikibot.textlib.**removeDisabledParts**(*text, tags=['*'], include=[]*)
Return text without portions where wiki markup is disabled.

Parts that can/will be removed are – * HTML comments * nowiki tags * pre tags * includeonly tags

The exact set of parts which should be removed can be passed as the 'tags' parameter, which defaults to all. Or, in alternative, default parts that shall not be removed can be specified in the 'include' param.

pywikibot.textlib.**removeHTMLParts**(*text, keeptags=['tt', 'nowiki', 'small', 'sup']*)
Return text without portions where HTML markup is disabled.

Parts that can/will be removed are – * HTML and all wiki tags

The exact set of parts which should NOT be removed can be passed as the 'keeptags' parameter, which defaults to ['tt', 'nowiki', 'small', 'sup'].

pywikibot.textlib.**removeLanguageLinks**(*text*, *site=None*, *marker=''*)
Return text with all inter-language links removed.

If a link to an unknown language is encountered, a warning is printed. If a marker is defined, that string is placed at the location of the last occurrence of an interwiki link (at the end if there are no interwiki links).

pywikibot.textlib.**removeLanguageLinksAndSeparator**(*text*, *site=None*, *marker=''*, *separator=''*)
Return text with inter-language links and preceding separators removed.

If a link to an unknown language is encountered, a warning is printed. If a marker is defined, that string is placed at the location of the last occurrence of an interwiki link (at the end if there are no interwiki links).

pywikibot.textlib.**replaceCategoryInPlace**(*oldtext*, *oldcat*, *newcat*, *site=None*)
Replace old category with new one and return the modified text.

Parameters

- **oldtext** – Content of the old category
- **oldcat** – pywikibot.Category object of the old category
- **newcat** – pywikibot.Category object of the new category

Returns the modified text

Return type unicode

pywikibot.textlib.**replaceCategoryLinks**(*oldtext*, *new*, *site=None*, *addOnly=False*)
Replace all existing category links with new category links.

Parameters

- **oldtext** – The text that needs to be replaced.
- **new** – Should be a list of Category objects or strings which can be either the raw name or [[Category:..]].
- **addOnly** – If addOnly is True, the old category won't be deleted and the category(s) given will be added (and so they won't replace anything).

pywikibot.textlib.**replaceExcept**(*text*, *old*, *new*, *exceptions*, *caseInsensitive=False*, *allowoverlap=False*, *marker=''*, *site=None*)
Return text with 'old' replaced by 'new', ignoring specified types of text.

Skips occurrences of 'old' within exceptions; e.g., within nowiki tags or HTML comments. If caseInsensitive is true, then use case insensitive regex matching. If allowoverlap is true, overlapping occurrences are all replaced (watch out when using this, it might lead to infinite loops!).

Parameters

- **old** – a compiled or uncompiled regular expression
- **new** – a unicode string (which can contain regular expression references), or a function which takes a match object as parameter. See parameter repl of re.sub().
- **exceptions** – a list of strings which signal what to leave out, e.g. ['math', 'table', 'template']
- **marker** – a string that will be added to the last replacement; if nothing is changed, it is added at the end

pywikibot.textlib.**replaceLanguageLinks**(*oldtext*, *new*, *site=None*, *addOnly=False*, *template=False*, *template_subpage=False*)

Replace inter-language links in the text with a new set of links.

'new' should be a dict with the Site objects as keys, and Page or Link objects as values (i.e., just like the dict returned by getLanguageLinks function).

pywikibot.textlib.**to_local_digits**(*phrase*, *lang*)

Change Latin digits based on language to localized version.

Be aware that this function only returns for several language And doesn't touch the input if other languages are asked. :param phrase: The phrase to convert to localized numerical :param lang: language code :return: The localized version :rtype: unicode

class pywikibot.textlib.**tzoneFixedOffset**(*offset*, *name*)

Bases: datetime.tzinfo

Class building tzinfo objects for fixed-offset time zones.

Parameters

- **offset** – a number indicating fixed offset in minutes east from UTC
- **name** – a string with name of the timezone

dst(*dt*)

Return no daylight savings time.

tzname(*dt*)

Return the name of the timezone.

utcoffset(*dt*)

Return the offset to UTC.

pywikibot.textlib.**unescape**(*s*)

Replace escaped HTML-special characters by their originals.

throttle Module

Mechanics to slow down wiki read and/or write rate.

class pywikibot.throttle.**Throttle**(*site*, *mindelay=None*, *maxdelay=None*, *writedelay=None*, *multiplydelay=True*)

Bases: object

Control rate of access to wiki server.

Calling this object blocks the calling thread until at least 'delay' seconds have passed since the previous call.

Each Site initiates one Throttle object (site.throttle) to control the rate of access.

checkMultiplicity()
Count running processes for site and set process_multiplicity.

drop()
Remove me from the list of running bot processes.

getDelay(*write=False*)
Return the actual delay, accounting for multiple processes.

This value is the maximum wait between reads/writes, not taking account of how much time has elapsed since the last access.

lag(*lagtime*)
Seize the throttle lock due to server lag.

This will prevent any thread from accessing this site.

setDelays(*delay=None*, *writedelay=None*, *absolute=False*)
Set the nominal delays in seconds. Defaults to config values.

wait(*seconds*)
Wait for seconds seconds.

Announce the delay if it exceeds a preset limit.

waittime(*write=False*)
Return waiting time in seconds if a query would be made right now.

titletranslate Module

Title translate module.

pywikibot.titletranslate.**appendFormatedDates**(*result*, *dictName*, *value*)

pywikibot.titletranslate.**getPoisonedLinks**(*pl*)
Return a list of known corrupted links that should be removed if seen.

pywikibot.titletranslate.**translate**(*page*, *hints=None*, *auto=True*, *removebrackets=False*, *site=None*, *family=None*)
Return a list of links to pages on other sites based on hints.

Entries for single page titles list those pages. Page titles for entries such as "all:" or "xyz:" or "20:" are first built from the page title of 'page' and then listed. When 'removebrackets' is True, a trailing pair of brackets and the text between them is removed from the page title. If 'auto' is true, known year and date page titles are autotranslated to all known target languages and inserted into the list.

tools Module

Miscellaneous helper functions (not wiki-dependent).

exception pywikibot.tools.**CombinedError**
Bases: KeyError, IndexError

An error that gets caught by both KeyError and IndexError.

class pywikibot.tools.**ComparableMixin**
Bases: object

Mixin class to allow comparing to other objects which are comparable.

class pywikibot.tools.**ContextManagerWrapper**(*wrapped*)
Bases: object

Wraps an object in a context manager.

It is redirecting all access to the wrapped object and executes 'close' when used as a context manager in with-statements. In such statements the value set via 'as' is directly the wrapped object. For example:

```
wrapped = ContextManagerWrapper(an_object)
with wrapped as another_object::
    assert(another_object is an_object)
```

It does not subclass the object though, so isinstance checks will fail outside a with-statement.

class pywikibot.tools.**DequeGenerator**
Bases: collections.deque

A generator that allows items to be added during generating.

next()
Python 3 iterator method.

class pywikibot.tools.**DotReadableDict**
Bases: *pywikibot.tools.UnicodeMixin*

Parent class of Revision() and FileInfo().

Provide::

- __getitem__(), __unicode__() and __repr__().

class pywikibot.tools.**EmptyDefault**
Bases: str, collections.abc.Mapping

A default for a not existing siteinfo property.

It should be chosen if there is no better default known. It acts like an empty collections, so it can be iterated through it savely if treated as a list, tuple, set or dictionary. It is also basically an empty string.

Accessing a value via __getitem__ will result in an combined KeyError and IndexError.

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 28

_abc_registry = <_weakrefset.WeakSet object>

_empty_iter()
An iterator which does nothing and drops the argument.

iteritems()
An iterator which does nothing and drops the argument.

iterkeys()
An iterator which does nothing and drops the argument.

itervalues()
An iterator which does nothing and drops the argument.

class pywikibot.tools.**FrozenDict**(*data=None*, *error=None*)
Bases: dict

Frozen dict, preventing write after initialisation.

Raises TypeError if write attempted.

update(**args*, ***kwargs*)
Prevent updates.

class pywikibot.tools.**LazyRegex**
Bases: object

Regex object that compiles the regex on usage.

flags
Get flags property.

raw
Get raw property.

class pywikibot.tools.**MediaWikiVersion**(*vstring=None*)
Bases: distutils.version.Version

Version object to allow comparing 'wmf' versions with normal ones.

The version mainly consist of digits separated by periods. After that is a suffix which may only be 'wmf<number>', 'alpha', 'beta<number>' or '-rc.<number>' (the - and . are optional). They are considered from old to new in that order with a version number without suffix is considered the newest. This secondary difference is stored in an internal _dev_version attribute.

Two versions are equal if their normal version and dev version are equal. A version is greater if the normal version or dev version is greater. For example:

```
1.24 < 1.24.1 < 1.25wmf1 < 1.25alpha < 1.25beta1 < 1.25beta2
< 1.25-rc-1 < 1.25-rc.2 < 1.25
```

Any other suffixes are considered invalid.

MEDIAWIKI_VERSION = re.compile('^(\\d+(?:\\.\\d+)+)(wmf(\\d+)|alpha|beta(\\d+)|-?rc\\.?(\\d+))?$')

_cmp(*other*)

parse(*vstring*)
Parse version string.

class pywikibot.tools.**ModuleDeprecationWrapper**(*module*)
Bases: module

A wrapper for a module to deprecate classes or variables of it.

_add_deprecated_attr(*name*, *replacement=None*, *replacement_name=None*, *warning_message=None*)
Add the name to the local deprecated names dict.

Parameters

- **name** (*str*) – The name of the deprecated class or variable. It may not be already deprecated.
- **replacement** (*any*) – The replacement value which should be returned instead. If the name is already an attribute of that module this must be None. If None it'll return the attribute of the module.

- **replacement_name** (*str*) – The name of the new replaced value. Required if `replacement` is not None and it has no __name__ attribute.
- **warning_message** (*basestring*) – The warning to display, with positional variables: {0} = module, {1} = attribute name, {2} = replacement.

class pywikibot.tools.**NotImplementedClass**(**args*, ***kwargs*)
Bases: object

No implementation is available.

class pywikibot.tools.**SelfCallDict**
Bases: *pywikibot.tools.SelfCallMixin*, dict

Dict with SelfCallMixin.

class pywikibot.tools.**SelfCallMixin**
Bases: object

Return self when called.

When '_own_desc' is defined it'll also issue a deprecation warning using issue_deprecation_warning('Calling ' + _own_desc, 'it directly').

class pywikibot.tools.**SelfCallString**
Bases: *pywikibot.tools.SelfCallMixin*, str

Unicode string with SelfCallMixin.

class pywikibot.tools.**ThreadList**(*limit=128*, **args*)
Bases: list

A simple threadpool class to limit the number of simultaneous threads.

Any threading.Thread object can be added to the pool using the append() method. If the maximum number of simultaneous threads has not been reached, the Thread object will be started immediately; if not, the append() call will block until the thread is able to start.

```
>>> pool = ThreadList(limit=10)
>>> def work()::
...     time.sleep(1)
...
>>> for x in range(20)::
...     pool.append(threading.Thread(target=work))
...
```

_logger = 'threadlist'

active_count()
Return the number of alive threads, and delete all non-alive ones.

append(*thd*)
Add a thread to the pool and start it.

stop_all()
Stop all threads the pool.

class pywikibot.tools.**ThreadedGenerator**(*group=None*, *target=None*, *name='GeneratorThread'*, *args=()*, *kwargs=None*, *qsize=65536*)
Bases: threading.Thread

Look-ahead generator class.

Runs a generator in a separate thread and queues the results; can be called like a regular generator.

Subclasses should override self.generator, I{not} self.run

Important: the generator thread will stop itself if the generator's internal queue is exhausted; but, if the calling program does not use all the generated values, it must call the generator's stop() method to stop the background thread. Example usage:

```
>>> gen = ThreadedGenerator(target=range, args=(20,))
>>> try::
```

... data = list(gen) ... finally:: ... gen.stop() >>> data [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

run()
Run the generator and store the results on the queue.

stop()
Stop the background thread.

class pywikibot.tools.**UnicodeMixin**
Bases: object

Mixin class to add __str__ method in Python 2 or 3.

exception pywikibot.tools.**_NotImplementedWarning**
Bases: RuntimeWarning

Feature that is no longer implemented.

pywikibot.tools.**add_decorated_full_name**(*obj*, *stacklevel=1*)
Extract full object name, including class, and store in __full_name__.

This must be done on all decorators that are chained together, otherwise the second decorator will have the wrong full name.

Parameters

- **obj** (*object*) – A object being decorated
- **stacklevel** (*int*) – level to use

pywikibot.tools.**add_full_name**(*obj*)
A decorator to add __full_name__ to the function being decorated.

This should be done for all decorators used in pywikibot, as any decorator that does not add __full_name__ will prevent other decorators in the same chain from being able to obtain it.

This can be used to monkey-patch decorators in other modules. e.g. <xyz>.foo = add_full_name(<xyz>.foo)

Parameters **obj** (*callable*) – The function to decorate

Returns decorating function

Return type function

pywikibot.tools.**concat_options**(*message*, *line_length*, *options*)
Concatenate options.

pywikibot.tools.**deprecate_arg**(*old_arg*, *new_arg*)
Decorator to declare old_arg deprecated and replace it with new_arg.

pywikibot.tools.**deprecated**(**outer_args*, ***outer_kwargs*)
Outer wrapper.

The outer wrapper may be the replacement function if the decorated decorator was called without arguments, or the replacement decorator if the decorated decorator was called without arguments.

Parameters

- **outer_args** (*list*) – args
- **outer_kwargs** – kwargs

Type outer_kwargs: dict

pywikibot.tools.**deprecated_args**(***arg_pairs*)
Decorator to declare multiple args deprecated.

Parameters **arg_pairs** – Each entry points to the new argument name. With True or None it drops the value and prints a warning. If False it just drops the value.

pywikibot.tools.**empty_iterator**()
An iterator which does nothing.

pywikibot.tools.**first_lower**(*string*)
Return a string with the first character uncapitalized.

Empty strings are supported. The original string is not changed.

pywikibot.tools.**first_upper**(*string*)
Return a string with the first character capitalized.

Empty strings are supported. The original string is not changed.

pywikibot.tools.**get_wrapper_depth**(*wrapper*)
Return depth of wrapper function.

pywikibot.tools.**intersect_generators**(*genlist*)
Intersect generators listed in genlist.

Yield items only if they are yielded by all generators in genlist. Threads (via ThreadedGenerator) are used in order to run generators in parallel, so that items can be yielded before generators are exhausted.

Threads are stopped when they are either exhausted or Ctrl-C is pressed. Quitting before all generators are finished is attempted if there is no more chance of finding an item in all queues.

Parameters **genlist** (*list*) – list of page generators

pywikibot.tools.**issue_deprecation_warning**(*name*, *instead*, *depth*)
Issue a deprecation warning.

pywikibot.tools.**itergroup**(*iterable*, *size*)
Make an iterator that returns lists of (up to) size items from iterable.

Example:

```
>>> i = itergroup(range(25), 10)
>>> print(next(i))
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] >>> print(next(i)) [10, 11, 12, 13, 14, 15, 16, 17, 18, 19] >>> print(next(i)) [20, 21, 22, 23, 24] >>> print(next(i)) Traceback (most recent call last):

...

StopIteration

pywikibot.tools.**manage_wrapping**(*wrapper*, *obj*)
Add attributes to wrapper and wrapped functions.

pywikibot.tools.**normalize_username**(*username*)
Normalize the username.

pywikibot.tools.**open_compressed**(*filename*, *use_extension=False*)
Open a file and uncompress it if needed.

This function supports bzip2, gzip and 7zip as compression containers. It uses the packages available in the standard library for bzip2 and gzip so they are always available. 7zip is only available when a 7za program is available.

The compression is selected via the file ending.

Parameters **filename** (*str*) – The filename.

Raises

- **ValueError** – When 7za is not available.
- **OSError** – When it's not a 7z archive but the file extension is 7z. It is also raised by bz2 when its content is invalid. gzip does not immediately raise that error but only on reading it.

Returns A file like object returning the uncompressed data in binary mode. Before Python 2.7 it's wrapping the object returned by BZ2File and gzip in a ContextManagerWrapper so it's advantages/disadvantages apply there.

Return type file like object

pywikibot.tools.**print_debug**(*msg*, **args*, ***kwargs*)
Simple debug routine.

pywikibot.tools.**redirect_func**(*target*, *source_module=None*, *target_module=None*, *old_name=None*, *class_name=None*)
Return a function which can be used to redirect to 'target'.

It also acts like marking that function deprecated and copies all parameters.

Parameters

- **target** (*callable*) – The targeted function which is to be executed.
- **source_module** (*basestring*) – The module of the old function. If '.' defaults to target_module. If 'None' (default) it tries to guess it from the executing function.
- **target_module** (*basestring*) – The module of the target function. If 'None' (default) it tries to get it from the target. Might not work with nested classes.
- **old_name** (*basestring*) – The old function name. If None it uses the name of the new function.
- **class_name** (*basestring*) – The name of the class. It's added to the target and source module (separated by a '.').

Returns A new function which adds a warning prior to each execution.

Return type callable

pywikibot.tools.**remove_last_args**(*arg_names*)
Decorator to declare all args additionally provided deprecated.

All positional arguments appearing after the normal arguments are marked deprecated. It marks also all keyword arguments present in arg_names as deprecated. Any arguments (positional or keyword) which are not present in arg_names are forwarded. For example a call with 3 parameters and the original function requests one and arg_names contain one name will result in an error, because the function got called with 2 parameters.

The decorated function may not use *args or **kwargs.

Parameters **arg_names** (*iterable; for the most explanatory message it should retain the given order (so not a set for example).*) – The names of all arguments.

pywikibot.tools.**signature**(*obj*)
Safely return function Signature object (PEP 362).

inspect.signature was introduced in 3.3, however backports are available. In Python 3.3, it does not support all types of callables, and should not be relied upon. Python 3.4 works correctly.

Any exception calling inspect.signature is ignored and None is returned.

Parameters **obj** – Function to inspect

Rtype obj callable

Return type inpect.Signature or None

version Module

Module to determine the pywikibot version (tag, revision and date).

exception pywikibot.version.**ParseError**
Bases: Exception

Parsing went wrong.

pywikibot.version.**get_module_filename**(*module*)
Retrieve filename from an imported pywikibot module.

It uses the __file__ attribute of the module. If it's file extension ends with py and another character the last character is discarded when the py file exist.

Parameters **module** (*module*) – The module instance.

Returns The filename if it's a pywikibot module otherwise None.

Return type str or None

pywikibot.version.**get_module_mtime**(*module*)
Retrieve the modification time from an imported module.

Parameters **module** (*module*) – The module instance.

Returns The modification time if it's a pywikibot module otherwise None.

Return type datetime or None

pywikibot.version.**get_module_version**(*module*)
Retrieve __version__ variable from an imported module.

Parameters **module** (*module*) – The module instance.

Returns The version hash without the surrounding text. If not present None.

Return type str or None

pywikibot.version.**getfileversion**(*filename*)
Retrieve revision number of file.

Extracts __version__ variable containing Id tag, without importing it. (thus can be done for any file)

The version variable containing the Id tag is read and returned. Because it doesn't import it, the version can be retrieved from any file. :param filename: Name of the file to get version :type filename: string

pywikibot.version.**getversion**(*online=True*)
Return a pywikibot version string.

Parameters **online** – (optional) Include information obtained online

pywikibot.version.**getversion_git**(*path=None*)
Get version info for a Git clone.

Parameters **path** – directory of the Git checkout

Returns

- tag (name for the repository),
- rev (current revision identifier),
- date (date of current revision),
- hash (git hash for the current revision)

Return type tuple of three str and a time.struct_time

pywikibot.version.**getversion_nightly**(*path=None*)
Get version info for a nightly release.

Parameters **path** – directory of the uncompressed nightly.

Returns

- tag (name for the repository),
- rev (current revision identifier),
- date (date of current revision),
- hash (git hash for the current revision)

Return type tuple of three str and a time.struct_time

pywikibot.version.**getversion_onlinerepo**(*repo=None*)
Retrieve current framework revision number from online repository.

Parameters **repo** (*URL or string*) – (optional) Online repository location

pywikibot.version.**getversion_package**(*path=None*)
Get version info for an installed package.

Parameters **path** – Unused argument

Returns

- tag: 'pywikibot/__init__.py'
- rev: '-1 (unknown)'
- date (date the package was installed locally),
- hash (git hash for the current revision of 'pywikibot/__init__.py')

Return type tuple of four str

pywikibot.version.**getversion_svn**(*path=None*)
Get version info for a Subversion checkout.

Parameters **path** – directory of the Subversion checkout

Returns

- tag (name for the repository),
- rev (current Subversion revision identifier),
- date (date of current revision),
- hash (git hash for the Subversion revision)

Return type tuple of three str and a time.struct_time

pywikibot.version.**getversion_svn_setuptools**(*path=None*)
Get version info for a Subversion checkout using setuptools.

Parameters **path** – directory of the Subversion checkout

Returns

- tag (name for the repository),
- rev (current Subversion revision identifier),
- date (date of current revision),
- hash (git hash for the Subversion revision)

Return type tuple of three str and a time.struct_time

pywikibot.version.**getversiondict**()
Get version info for the package.

Returns

- tag (name for the repository),
- rev (current revision identifier),
- date (date of current revision),
- hash (git hash for the current revision)

Return type dict of four str

pywikibot.version.**github_svn_rev2hash**(*tag*, *rev*)
Convert a Subversion revision to a Git hash using Github.

Parameters

- **tag** – name of the Subversion repo on Github
- **rev** – Subversion revision identifier

Returns the git hash

Return type str

pywikibot.version.**package_versions**(*modules=None*, *builtins=False*, *standard_lib=None*)
Retrieve package version information.

When builtins or standard_lib are None, they will be included only if a version was found in the package.

Parameters

- **modules** (*list of strings*) – Modules to inspect
- **builtins** (*Boolean, or None for automatic selection*) – Include builtins
- **standard_lib** (*Boolean, or None for automatic selection*) – Include standard library packages

pywikibot.version.**svn_rev_info**(*path*)
Fetch information about the current revision of an Subversion checkout.

Parameters **path** – directory of the Subversion checkout

Returns

- tag (name for the repository),

- rev (current Subversion revision identifier),
- date (date of current revision),

Return type `tuple` of two `str` and a `time.struct_time`

`weblib` Module

Functions for manipulating external links or querying third-party sites.

`pywikibot.weblib.`**`getInternetArchiveURL`**(*url*, *timestamp=None*)
Return archived URL by Internet Archive.

See [[:mw:Archived Pages]] and https://archive.org/help/wayback_api.php for more details.

Parameters

- **`url`** – url to search an archived version for
- **`timestamp`** – requested archive date. The version closest to that moment is returned. Format: YYYYMMDDhhmmss or part thereof.

`pywikibot.weblib.`**`getWebCitationURL`**(*url*, *timestamp=None*)
Return archived URL by Web Citation.

See http://www.webcitation.org/doc/WebCiteBestPracticesGuide.pdf for more details

Parameters

- **`url`** – url to search an archived version for
- **`timestamp`** – requested archive date. The version closest to that moment is returned. Format: YYYYMMDDhhmmss or part thereof.

`xmlreader` Module

XML reading module.

Each XmlEntry object represents a page, as read from an XML source

The XmlDump class reads a pages_current XML dump (like the ones offered on https://dumps.wikimedia.org/backup-index.html) and offers a generator over XmlEntry objects which can be used by other bots.

class `pywikibot.xmlreader.`**`XmlDump`**(*filename*, *allrevisions=False*)
Bases: `object`

Represents an XML dump file.

Reads the local file at initialization, parses it, and offers access to the resulting XmlEntries via a generator.

Parameters **`allrevisions`** – boolean If True, parse all revisions instead of only the latest one. Default: False.

`parse`()
Generator using cElementTree iterparse function.

class `pywikibot.xmlreader.`**`XmlEntry`**(*title*, *ns*, *id*, *text*, *username*, *ipedit*, *timestamp*, *editRestriction*, *moveRestriction*, *revisionid*, *comment*, *redirect*)
Bases: `object`

Represent a page.

class pywikibot.xmlreader.**XmlParserThread**(*filename*, *handler*)
Bases: threading.Thread

XML parser that will run as a single thread.

This allows the XmlDump generator to yield pages before the parser has finished reading the entire dump.

There surely are more elegant ways to do this.

run()
Parse the file in a single thread.

pywikibot.xmlreader.**parseRestrictions**(*restrictions*)
Parse the characters within a restrictions tag.

Returns strings representing user groups allowed to edit and to move a page, where None means there are no restrictions.

Subpackages

comms Package

comms Package Communication layer.

http Module Basic HTTP access interface.

This module handles communication between the bot and the HTTP threads.

This module is responsible for

- Setting up a connection pool
- Providing a (blocking) interface for HTTP requests
- Translate site objects with query strings into URLs
- URL-encoding all data
- Basic HTTP error handling

pywikibot.comms.http.**error_handling_callback**(*request*)
Raise exceptions and log alerts.

Parameters **request** – Request that has completed

Rtype request threadedhttp.HttpRequest

pywikibot.comms.http.**fetch**(*uri*, *method='GET'*, *body=None*, *headers=None*, *default_error_handling=True*, ***kwargs*)
Blocking HTTP request.

Note: The callback runs in the HTTP thread, where exceptions are logged but are not able to be caught.

See httplib2.Http.request for parameters.

Parameters **default_error_handling** (*bool*) – Use default error handling

Return type threadedhttp.HttpRequest

pywikibot.comms.http.**request**(*site=None*, *uri=None*, *method='GET'*, *body=None*, *headers=None*, ***kwargs*)
Request to Site with default error handling and response decoding.

See httplib2.Http.request for additional parameters.

If the site argument is provided, the uri is a relative uri from and including the document root '/'.

If the site argument is None, the uri must be absolute.

Parameters

- **site** (*pywikibot.site.BaseSite*) – The Site to connect to
- **uri** (*str*) – the URI to retrieve
- **charset** (*CodecInfo, str, None*) – Either a valid charset (usable for str.decode()) or None to automatically chose the charset from the returned header (defaults to latin-1)

Returns The received data

Return type a unicode string

pywikibot.comms.http.**user_agent**(*site=None*, *format_string=None*)
Generate the user agent string for a given site and format.

Parameters

- **site** (BaseSite) – The site for which this user agent is intended. May be None.
- **format_string** (*basestring*) – The string to which the values will be added using str.format. Is using config.user_agent_format when it is None.

Returns The formatted user agent

Return type unicode

pywikibot.comms.http.**user_agent_username**(*username=None*)
Reduce username to a representation permitted in HTTP headers.

To achieve that, this function:: 1) replaces spaces (' ') with '_' 2) encodes the username as 'utf-8' and if the username is not ASCII 3) URL encodes the username if it is not ASCII, or contains '%'

threadedhttp Module Httplib2 threaded cookie layer.

This class extends httplib2, adding support for::

- Cookies, guarded for cross-site redirects
- Thread safe ConnectionPool class
- HttpProcessor thread class
- HttpRequest object

class pywikibot.comms.threadedhttp.**ConnectionPool**(*maxnum=5*)
Bases: object

A thread-safe connection pool.

pop_connection(*identifier*)
Get a connection from identifier's connection pool.

Parameters **identifier** – The pool identifier

Returns A connection object if found, None otherwise

push_connection(*identifier*, *connection*)
Add a connection to identifier's connection pool.

Parameters

- **identifier** – The pool identifier

- **connection** – The connection to add to the pool

class pywikibot.comms.threadedhttp.**DummyMessage**(*response*)
Bases: object

Simulated mimetools.Message object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

get_all(*k*, *failobj=None*)

getheaders(*k*)

class pywikibot.comms.threadedhttp.**DummyRequest**(*url*, *headers=None*)
Bases: object

Simulated urllib2.Request object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

add_unredirected_header(*key*, *val*)

get_full_url()

get_header(*key*, *default=None*)

get_host()

get_origin_req_host()

get_type()

has_header(*key*)

is_unverifiable()

unverifiable

class pywikibot.comms.threadedhttp.**DummyResponse**(*response*)
Bases: object

Simulated urllib2.Request object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

info()

class pywikibot.comms.threadedhttp.**Http**(**args*, ***kwargs*)
Bases: httplib2.Http

Subclass of httplib2.Http that stores cookies.

Overrides httplib2's internal redirect support to prevent cookies being eaten by the wrong sites.

request(*uri*, *method='GET'*, *body=None*, *headers=None*, *max_redirects=None*, *connection_type=None*)
Start an HTTP request.

Parameters

- **uri** – The uri to retrieve
- **method** – (optional) The HTTP method to use. Default is 'GET'
- **body** – (optional) The request body. Default is no body.
- **headers** – (optional) Additional headers to send. Defaults include `connection: keep-alive`, `user-agent` and `content-type`.

- **max_redirects** – (optional) The maximum number of redirects to use for this request. The class instance's max_redirects is default
- **connection_type** – (optional) see `httplib2.Http.request`

Returns (response, content) tuple

class `pywikibot.comms.threadedhttp.`**HttpProcessor**(*queue*, *cookiejar*, *connection_pool*)
Bases: `threading.Thread`

Thread object to spawn multiple HTTP connection threads.

run()

class `pywikibot.comms.threadedhttp.`**HttpRequest**(*uri*, *method='GET'*, *body=None*, *headers=None*, *callbacks=None*, *charset=None*, ***kwargs*)
Bases: *pywikibot.tools.UnicodeMixin*

Object wrapper for HTTP requests that need to block origin thread.

Usage:

```
>>> from .http import Queue
>>> queue = Queue.Queue()
>>> cookiejar = cookielib.CookieJar()
>>> connection_pool = ConnectionPool()
>>> proc = HttpProcessor(queue, cookiejar, connection_pool)
>>> proc.setDaemon(True)
>>> proc.start()
>>> request = HttpRequest('https://hostname.invalid/')
>>> queue.put(request)
>>> request.lock.acquire()
```

True >>> print(type(request.data)) <class 'httplib2.ServerNotFoundError'> >>> print(request.data) Unable to find the server at hostname.invalid >>> queue.put(None) # Stop the http processor thread

`request.lock.acquire()` will block until the data is available.

self.data will be either:: * a tuple of (dict, unicode) if the request was successful * an exception

content
Return the response decoded by the detected encoding.

data
Return the httplib2 response tuple.

decode(*encoding*, *errors='strict'*)
Return the decoded response.

encoding
Detect the response encoding.

exception
Get the exception raised by httplib2, if any.

header_encoding
Return charset given by the response header.

hostname
Return the host of the request.

parsed_uri
Return the parsed requested uri.

raw
Return the raw response body.

response_headers
Return the response headers.

status
HTTP response status.

Return type int

compat Package

compat Package Package to provide compatibility with compat scripts.

catlib Module WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

class pywikibot.compat.catlib.**Category**(*source*, *title=''*, *sortKey=None*)
Bases: pywikibot.page.Page

A page in the Category: namespace.

articles(*recurse=False*, *step=None*, *total=None*, *content=False*, *namespaces=None*, *sortby=None*, *starttime=None*, *endtime=None*, *startsort=None*, *endsort=None*)
Yield all articles in the current category.

By default, yields all *pages* in the category that are not subcategories!

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate articles in subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate articles in first-level subcats, but no deeper.)
- **step** – limit each API call to this number of pages
- **total** – iterate no more than this number of pages in total (at all levels)
- **namespaces** (*int or list of ints*) – only yield pages in the specified namespaces
- **content** – if True, retrieve the content of the current version of each page (default False)
- **sortby** (*str*) – determines the order in which results are generated, valid values are "sortkey" (default, results ordered by category sort key) or "timestamp" (results ordered by time page was added to the category). This applies recursively.
- **starttime** (*pywikibot.Timestamp*) – if provided, only generate pages added after this time; not valid unless sortby="timestamp"
- **endtime** (*pywikibot.Timestamp*) – if provided, only generate pages added before this time; not valid unless sortby="timestamp"
- **startsort** (*str*) – if provided, only generate pages >= this title lexically; not valid if sortby="timestamp"
- **endsort** (*str*) – if provided, only generate pages <= this title lexically; not valid if sortby="timestamp"

articlesList(*recurse=False*)
DEPRECATED: equivalent to list(self.articles(...)).

aslink(*sortKey=None*)
Return a link to place a page in this Category.

Use this only to generate a "true" category link, not for interwikis or text links to category pages.

Parameters **sortKey** (*(optional) unicode*) – The sort key for the article to be placed in this Category; if omitted, default sort key is used.

categoryinfo
Return a dict containing information about the category.

The dict contains values for:

Numbers of pages, subcategories, files, and total contents.

Returns dict

copyAndKeep(*catname*, *cfdTemplates*, *message*)
Copy partial category page text (not contents) to a new title.

Like copyTo above, except this removes a list of templates (like deletion templates) that appear in the old category text. It also removes all text between the two HTML comments BEGIN CFD TEMPLATE and END CFD TEMPLATE. (This is to deal with CFD templates that are substituted.)

Returns true if copying was successful, false if target page already existed.

Parameters

- **catname** – New category title (without namespace)
- **cfdTemplates** – A list (or iterator) of templates to be removed from the page text

Returns True if copying was successful, False if target page already existed.

copyTo(*cat*, *message*)
Copy text of category page to a new page. Does not move contents.

Parameters

- **cat** (*unicode or Category*) – New category title (without namespace) or Category object
- **message** (*unicode*) – message to use for category creation message If two %s are provided in message, will be replaced by (self.title, authorsList)

Returns True if copying was successful, False if target page already existed.

isEmptyCategory()
Return True if category has no members (including subcategories).

isHiddenCategory()
Return True if the category is hidden.

members(*recurse=False*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Yield all category contents (subcats, pages, and files).

newest_pages(*total=None*)
Return pages in a category ordered by the creation date.

If two or more pages are created at the same time, the pages are returned in the order they were added to the category. The most recently added page is returned first.

It only allows to return the pages ordered from newest to oldest, as it is impossible to determine the oldest page in a category without checking all pages. But it is possible to check the category in order with the

newly added first and it yields all pages which were created after the currently checked page was added (and thus there is no page created after any of the cached but added before the currently checked).

Parameters **total** (*int*) – The total number of pages queried.

Returns A page generator of all pages in a category ordered by the creation date. From newest to oldest. Note: It currently only returns Page instances and not a subclass of it if possible. This might change so don't expect to only get Page instances.

Return type generator

subcategories(*recurse=False*, *step=None*, *total=None*, *content=False*)
Iterate all subcategories of the current category.

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate subcategories of subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate direct subcats and first-level sub-sub-cats, but no deeper.)
- **step** – limit each API call to this number of categories
- **total** – iterate no more than this number of subcategories in total (at all levels)
- **content** – if True, retrieve the content of the current version of each category description page (default False)

subcategoriesList(*recurse=False*)
DEPRECATED: Equivalent to list(self.subcategories(...)).

supercategories()
DEPRECATED: equivalent to self.categories().

supercategoriesList()
DEPRECATED: equivalent to list(self.categories(...)).

pywikibot.compat.catlib.**change_category**(*article*, *oldCat*, *newCat*, *comment=None*, *sortKey=None*, *inPlace=True*)
Change the category of the article.

query Module WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

pywikibot.compat.query.**GetData**(*request*, *site=None*, *back_response=False*)
Query the server with the given request dict.

DEPRECATED: Use pywikibot.data.api.Request instead.

userlib Module WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

class pywikibot.compat.userlib.**User**(*source*, *title=''*)
Bases: pywikibot.page.Page

A class that represents a Wiki user.

This class also represents the Wiki page User:<username>

block(*expiry*, *reason*, *anononly=True*, *nocreate=True*, *autoblock=True*, *noemail=False*, *reblock=False*)
Block user.

Parameters

- **expiry** (*pywikibot.Timestamp|str*) – When the block should expire
- **reason** (*basestring*) – Block reason
- **anononly** (*bool*) – Whether block should only affect anonymous users
- **nocreate** (*bool*) – Whether to block account creation
- **autoblock** (*bool*) – Whether to enable autoblock
- **noemail** (*bool*) – Whether to disable email access
- **reblock** (*bool*) – Whether to reblock if a block already is set

Returns None

contributions(*total=500*, *namespaces=[]*)
Yield tuples describing this user edits.

Each tuple is composed of a pywikibot.Page object, the revision id (int), the edit timestamp (as a pywikibot.Timestamp object), and the comment (unicode). Pages returned are not guaranteed to be unique.

Parameters

- **total** (*int*) – limit result to this number of pages
- **namespaces** (*list*) – only iterate links in these namespaces

editCount(*force=False*)
Return edit count for a registered user.

Always returns 0 for 'anonymous' users.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long

editedPages(*total=500*)
DEPRECATED. Use contributions().

Yields pywikibot.Page objects that this user has edited, with an upper bound of 'total'. Pages returned are not guaranteed to be unique.

Parameters **total** (*int.*) – limit result to this number of pages.

getUserPage(*subpage=''*)
Return a Page object relative to this user's main page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main page title (optional)

getUserTalkPage(*subpage=''*)
Return a Page object relative to this user's main talk page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main talk page title (optional)

getprops(*force=False*)
Return a properties about the user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns dict

groups (*force=False*)
Return a list of groups to which this user belongs.

The list of groups may be empty.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns list

isAnonymous ()
Determine if the user is editing as an IP address.

Returns bool

isBlocked (*force=False*)
Determine whether the user is currently blocked.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isEmailable (*force=False*)
Determine whether emails may be send to this user through MediaWiki.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isRegistered (*force=False*)
Determine if the user is registered on the site.

It is possible to have a page named User:xyz and not have a corresponding user with username xyz.

The page does not need to exist for this method to return True.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

name ()
The username.

Returns unicode

registration (*force=False*)
Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns pywikibot.Timestamp or None

registrationTime (*force=False*)
DEPRECATED. Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long (MediaWiki's internal timestamp format) or 0

sendMail (*subject*, *text*, *ccme=False*)
Send an email to this user via MediaWiki's email interface.

Outputs 'Email sent' if the email was sent.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body

- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises _EmailUserError logged in user does not have 'sendemail' right or the target has disabled receiving emails

Returns operation successful indicator

Return type bool

send_email(*subject*, *text*, *ccme=False*)

Send an email to this user via MediaWiki's email interface.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body
- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises

- **NotEmailableError** – the user of this User is not emailable
- **UserRightsError** – logged in user does not have 'sendemail' right

Returns operation successful indicator

Return type bool

uploadedImages(*total=10*)

Yield tuples describing files uploaded by this user.

Each tuple is composed of a pywikibot.Page, the timestamp (str in ISO8601 format), comment (unicode) and a bool for pageid > 0. Pages returned are not guaranteed to be unique.

Parameters **total** (*int*) – limit result to this number of pages

username

The username.

Convenience method that returns the title of the page with namespace prefix omitted, which is the username.

Returns unicode

data Package

data Package

Module providing several layers of data access to the wiki.

api Module

Interface to Mediawiki's api.php.

exception pywikibot.data.api.**APIError**(*code*, *info*, ***kwargs*)

Bases: pywikibot.exceptions.Error

The wiki site returned an error message.

class pywikibot.data.api.**APIGenerator**(*action*, *continue_name='continue'*, *limit_name='limit'*, *data_name='data'*, ***kwargs*)

Bases: object

Iterator that handle API responses containing lists.

The iterator will iterate each item in the query response and use the continue request parameter to retrieve the next portion of items automatically. If the limit attribute is set, the iterator will stop after iterating that many values.

set_maximum_items(*value*)
Set the maximum number of items to be retrieved from the wiki.

If not called, most queries will continue as long as there is more data to be retrieved from the API.

Parameters **value** (*int*) – The value of maximum number of items to be retrieved in total to set.

set_query_increment(*value*)
Set the maximum number of items to be retrieved per API query.

If not called, the default is 50.

Parameters **value** (*int*) – The value of maximum number of items to be retrieved per API request to set.

exception pywikibot.data.api.**APIMWException**(*mediawiki_exception_class_name*, *info*, ***kwargs*)
Bases: *pywikibot.data.api.APIError*

The API site returned an error about a MediaWiki internal exception.

class pywikibot.data.api.**CTEBinaryBytesGenerator**(**args*, ***kwargs*)
Bases: email.generator.BytesGenerator

Workaround for bug in python 3 email handling of CTE binary.

class pywikibot.data.api.**CTEBinaryMIMEMultipart**(*_subtype='mixed'*, *boundary=None*, *_subparts=None*, ***_params*)
Bases: email.mime.multipart.MIMEMultipart

Workaround for bug in python 3 email handling of CTE binary.

as_bytes(*unixfrom=False*, *policy=None*)
Return unmodified binary payload.

class pywikibot.data.api.**CachedRequest**(*expiry*, **args*, ***kwargs*)
Bases: *pywikibot.data.api.Request*

Cached request.

submit()
Submit cached request.

pywikibot.data.api.**CategoryPageGenerator**()
Like PageGenerator, but yields Category objects instead of Pages.

class pywikibot.data.api.**EnableSSLSiteWrapper**(*site*)
Bases: object

Wrapper to change the site protocol to https.

protocol()
Return protocol.

pywikibot.data.api.**ImagePageGenerator**()
Like PageGenerator, but yields FilePage objects instead of Pages.

class pywikibot.data.api.**ListGenerator**(*listaction*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for queries of type action=query&list=foo.

See the API documentation for types of lists that can be queried. Lists include both side-wide information (such as 'allpages') and page-specific information (such as 'backlinks').

This iterator yields a dict object for each member of the list returned by the API, with the format of the dict depending on the particular list command used. For those lists that contain page information, it may be easier to use the PageGenerator class instead, as that will convert the returned information into a Page object.

class pywikibot.data.api.**LogEntryListGenerator**(*logtype=None*, ***kwargs*)
Bases: *pywikibot.data.api.ListGenerator*

Iterator for queries of list 'logevents'.

Yields LogEntry objects instead of dicts.

result(*pagedata*)
Instatiate LogEntry from data from api.

class pywikibot.data.api.**LoginManager**(*password=None*, *sysop=False*, *site=None*, *user=None*)
Bases: *pywikibot.login.LoginManager*

Supply getCookie() method to use API interface.

getCookie(*remember=True*, *captchaId=None*, *captchaAnswer=None*)
Login to the site.

Parameters are all ignored.

Returns cookie data if successful, None otherwise.

storecookiedata(*data*)
Ignore data; cookies are set by threadedhttp module.

pywikibot.data.api.**MIMEMultipart**
alias of *CTEBinaryMIMEMultipart*

class pywikibot.data.api.**OptionSet**(*site=None*, *module=None*, *param=None*, *dict=None*)
Bases: collections.abc.MutableMapping

A class to store a set of options which can be either enabled or not.

If it is instantiated with the associated site, module and parameter it will only allow valid names as options. If instantiated 'lazy loaded' it won't checks if the names are valid until the site has been set (which isn't required, but recommended). The site can only be set once if it's not None and after setting it, any site (even None) will fail.

api_iter()
Iterate over each option as they appear in the URL.

clear()
Clear all enabled and disabled options.

from_dict(*dict*)
Load options from the dict.

The options are not cleared before. If changes have been made previously, but only the dict values should be applied it needs to be cleared first.

Parameters **dict** (*dict (keys are strings, values are bool/None)*) – A dictionary containing for each entry either the value False, True or None. The names must be valid depending on whether they enable or disable the option. All names with the value None can be in either of the list.

class pywikibot.data.api.**PageGenerator**(*generator*, *g_content=False*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for response to a request of type action=query&generator=foo.

This class can be used for any of the query types that are listed in the API documentation as being able to be used as a generator. Instances of this class iterate Page objects.

result (*pagedata*)
Convert page dict entry from api to Page object.

This can be overridden in subclasses to return a different type of object.

class pywikibot.data.api.**ParamInfo**(*site*, *preloaded_modules=None*, *modules_only_mode=None*)
Bases: collections.abc.Container

API parameter information data object.

Provides cache aware fetching of parameter information.

Full support for MW 1.12+, when 'paraminfo' was introduced to the API. Partially supports MW 1.11, using data extracted from API 'help'. MW 1.10 not supported as module prefixes are not extracted from API 'help'.

TODO: Rewrite help parser to support earlier releases.

TODO: establish a data structure in the class which prefills the param information available for a site given its version, using the API information available for each API version.

TODO: module aliases: in 1.25wmf list=deletedrevs becomes list=alldeletedrevisions prop=deletedrevs becomes prop=deletedrevisions

TODO: share API parameter information between sites using similar versions of the API, especially all sites in the same family.

action_modules
Set of all action modules.

fetch(*modules*, *_init=False*)
Fetch paraminfo for multiple modules.

No exception is raised when paraminfo for a module does not exist. Use __getitem__ to cause an exception if a module does not exist.

Parameters **modules** (*set*) – API modules to load

Return type NoneType

init_modules = frozenset({'paraminfo', 'main'})

module_attribute_map(*attribute*, *modules=None*)
Mapping of modules with an attribute to the attribute value.

Parameters

- **attribute** (*basestring*) – attribute name
- **modules** (*set*) – modules to include (default: all modules)

Return type dict using modules as keys

modules
Set of all module names without path prefixes.

Only includes one 'tokens', even if it appears as both a action and a query submodule.

Returns module names

Return type set of str

normalize_modules(*modules*)
Convert the modules into module paths.

Add query+ to any query module name not also in action modules.

Returns The modules converted into a module paths

Return type set

classmethod normalize_paraminfo(*data*)
Convert both old and new API JSON into a new-ish data structure.

parameter(*module*, *param_name*)
Get details about one modules parameter.

Returns None if the parameter does not exist.

Parameters

- **module** (*str*) – API module name
- **param_name** (*str*) – parameter name in the module

Returns metadata that describes how the parameter may be used

Return type dict or None

paraminfo_keys = frozenset({'mainmodule', 'formatmodules', 'querymodules', 'pagesetmodule', 'modules'})

prefixes
Mapping of module to its prefix for all modules with a prefix.

This loads paraminfo for all modules.

query_modules
Set of all query module names without query+ path prefix.

query_modules_with_limits
Set of all query modules which have limits.

root_modules = frozenset({'pageset', 'main'})

class pywikibot.data.api.**PropertyGenerator**(*prop*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for queries of type action=query&prop=foo.

See the API documentation for types of page properties that can be queried.

This iterator yields one or more dict object(s) corresponding to each "page" item(s) from the API response; the calling module has to decide what to do with the contents of the dict. There will be one dict for each page queried via a titles= or ids= parameter (which must be supplied when instantiating this class).

props
The requested property names.

class pywikibot.data.api.**QueryGenerator**(***kwargs*)
Bases: object

Base class for iterators that handle responses to API action=query.

By default, the iterator will iterate each item in the query response, and use the (query-)continue element, if present, to continue iterating as long as the wiki returns additional values. However, if the iterator's limit attribute is set to a positive int, the iterator will stop after iterating that many values. If limit is negative, the limit parameter will not be passed to the API at all.

Most common query types are more efficiently handled by subclasses, but this class can be used directly for custom queries and miscellaneous types (such as "meta=...") that don't return the usual list of pages or links. See the API documentation for specific query options.

result(*data*)
Process result data as needed for particular subclass.

set_maximum_items(*value*)
Set the maximum number of items to be retrieved from the wiki.

If not called, most queries will continue as long as there is more data to be retrieved from the API.

If set to -1 (or any negative value), the "limit" parameter will be omitted from the request. For some request types (such as prop=revisions), this is necessary to signal that only current revision is to be returned.

set_namespace(*namespaces*)
Set a namespace filter on this query.

Parameters **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers. An empty iterator clears any namespace restriction.*) – namespace identifiers to limit query results

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool, or more than one namespace if the API module does not support multiple namespaces

set_query_increment(*value*)
Set the maximum number of items to be retrieved per API query.

If not called, the default is to ask for "max" items and let the API decide how many to send.

class pywikibot.data.api.**Request**(***kwargs*)
Bases: collections.abc.MutableMapping

A request to a Site's api.php interface.

Attributes of this object (except for the special parameters listed below) get passed as commands to api.php, and can be get or set using the dict interface. All attributes must be strings (or unicode). Use an empty string for parameters that don't require a value. For example, Request(action="query", titles="Foo bar", prop="info", redirects="") corresponds to the API request "api.php?action=query&titles=Foo%20bar&prop=info&redirects"

This is the lowest-level interface to the API, and can be used for any request that a particular site's API supports. See the API documentation (https://www.mediawiki.org/wiki/API) and site-specific settings for details on what parameters are accepted for each request type.

Uploading files is a special case: to upload, the parameter "mime" must be true, and the parameter "file" must be set equal to a valid filename on the local computer, _not_ to the content of the file.

Returns a dict containing the JSON data returned by the wiki. Normally, one of the dict keys will be equal to the value of the 'action' parameter. Errors are caught and raise an APIError exception.

Example:

```
>>> r = Request(action="query", meta="userinfo")
>>> # This is equivalent to
>>> # https://{path}/api.php?action=query&meta=userinfo&format=json
>>> # change a parameter
>>> r['meta'] = "userinfo|siteinfo"
>>> # add a new parameter
>>> r['siprop'] = "namespaces"
```

```
>>> # note that "uiprop" param gets added automatically
>>> r.action
```

u'query' >>> sorted(r._params.keys()) # doctest: +IGNORE_UNICODE [u'action', u'meta', u'siprop'] >>> r._params['action'] # doctest: +IGNORE_UNICODE [u'query'] >>> r._params['meta'] # doctest: +IGNORE_UNICODE [u'userinfo', u'siteinfo'] >>> r._params['siprop'] # doctest: +IGNORE_UNICODE [u'namespaces'] >>> data = r.submit() # doctest: +IGNORE_UNICODE >>> isinstance(data, dict) True >>> set(['query', 'batchcomplete', 'warnings']).issuperset(data.keys()) True >>> 'query' in data True >>> sorted(data[u'query'].keys()) # doctest: +IGNORE_UNICODE ['namespaces', 'userinfo']

http_params()
Return the parameters formatted for inclusion in an HTTP request.

DEPRECATED. See _encoded_items for explanation of encoding used.

items()
Return a list of tuples containg the parameters in any order.

iteritems()
Implement dict interface.

keys()
Implement dict interface.

mime
Return whether mime parameters are defined.

submit()
Submit a query and parse the response.

Returns a dict containing data retrieved from api.php

wait()
Determine how long to wait after a failed request.

exception pywikibot.data.api.**TimeoutError**(*arg*)
Bases: pywikibot.exceptions.Error

API request failed with a timeout error.

exception pywikibot.data.api.**UploadWarning**(*code*, *message*)
Bases: *pywikibot.data.api.APIError*

Upload failed with a warning message (passed as the argument).

message
Return warning message.

pywikibot.data.api.**update_page**(*page*, *pagedict*, *props=[]*)
Update attributes of Page object page, based on query data in pagedict.

Parameters

- **page** (Page) – object to be updated
- **pagedict** (*dict*) – the contents of a "page" element of a query response
- **props** (*iterable of string*) – the property names which resulted in pagedict. If a missing value in pagedict can indicate both 'false' and 'not present' the property which would make the value present must be in the props parameter.

wikidataquery Module Objects representing WikidataQuery query syntax and API.

class pywikibot.data.wikidataquery.**Around**(*prop*, *coord*, *rad*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form around[PROPERTY,LATITUDE,LONGITUDE,RADIUS].

queryType = 'around'

validate()

class pywikibot.data.wikidataquery.**Between**(*prop*, *begin=None*, *end=None*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form between[PROP, BEGIN, END].

You have to give prop and one of begin or end. Note that times have to be in UTC, timezones are not supported by the API

Parameters

- **prop** – the property
- **begin** – WbTime object representing the beginning of the period
- **end** – WbTime object representing the end of the period

queryType = 'between'

validate()

class pywikibot.data.wikidataquery.**HasClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.Query*

This is a Query of the form "claim[prop:val]".

It is subclassed by the other similar forms like noclaim and string

formatItems()

queryType = 'claim'

validate()

class pywikibot.data.wikidataquery.**Link**(*link*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form link[LINK,...], which also includes nolink.

All link elements have to be strings, or validation will throw

queryType = 'link'

validate()

class pywikibot.data.wikidataquery.**NoClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.HasClaim*

Query of the form noclaim[PROPERTY].

queryType = 'noclaim'

class pywikibot.data.wikidataquery.**NoLink**(*link*)
Bases: *pywikibot.data.wikidataquery.Link*

A query in the form nolink[..].

queryType = 'nolink'

class pywikibot.data.wikidataquery.**Query**
Bases: object

A query is a single query for the WikidataQuery API.

For example:: claim[100:60] or link[enwiki]

Construction of a Query can throw a TypeError if you feed it bad parameters. Exactly what these need to be depends on the Query

AND(*ands*)
Produce a query set ANDing this query and all the given query/sets.

OR(*ors*)
Produce a query set ORing this query and all the given query/sets.

convertWDType(*item*)
Convert Wikibase items like ItemPage or PropertyPage into integer IDs.

The resulting IDs may be used in query strings.

Parameters **item** – A single item. One of ItemPages, PropertyPages, int or anything that can be fed to int()

Returns the int ID of the item

convertWDTypes(*items*)

formatItem(*item*)
Default item formatting is string.

This will work for queries, querysets, ints and strings

formatList(*l*)
Format and comma-join a list.

static isOrContainsOnlyTypes(*items*, *types*)
Either this item is one of the given types, or it is a list of only those types.

Return type bool

validate()
Validate the query parameters.

Default validate result is a pass - subclasses need to implement this if they want to check their parameters.

Returns True

Return type bool

validateOrRaise(*msg=None*)

class pywikibot.data.wikidataquery.**QuerySet**(*q*)
Bases: object

A QuerySet represents a set of queries or other query sets.

Queries may be joined by operators (AND and OR).

A QuerySet stores this information as a list of Query(Sets) and a joiner operator to join them all together

AND(*args*)
Add the given args (Queries or QuerySets) to the Query set as a logical conjuction (AND).

OR(*args*)
Add the given args (Queries or QuerySets) to the Query set as a logical disjunction (OR).

addJoiner(*args*, *joiner*)
Add to this QuerySet using the given joiner.

If the given joiner is not the same as we used before in this QuerySet, nest the current one in parens before joining. This makes the implicit grouping of the API explicit.

Returns a new query set representing the joining of this one and the arguments

class pywikibot.data.wikidataquery.**StringClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.HasClaim*

Query of the form string[PROPERTY:"STRING",...].

formatItem(*x*)
Add quotes around string.

queryType = 'string'

validate()

class pywikibot.data.wikidataquery.**Tree**(*item*, *forward=[]*, *reverse=[]*)
Bases: *pywikibot.data.wikidataquery.Query*

Query of the form tree[ITEM,...][PROPERTY,...]<PROPERTY,...>.

queryType = 'tree'

validate()

class pywikibot.data.wikidataquery.**WikidataQuery**(*host='https://wdq.wmflabs.org'*, *cacheDir=None*, *cacheMaxAge=60*)
Bases: object

An interface to the WikidataQuery API.

Default host is https://wdq.wmflabs.org/, but you can substitute a different one.

Caching defaults to a subdir of the system temp directory with a 1 hour max cache age.

Set a zero or negative maxCacheAge to disable caching

getCacheFilename(*queryStr*)
Encode a query into a unique and universally safe format.

Return type unicode

getDataFromHost(*queryStr*)
Go and fetch a query from the host's API.

Return type dict

getQueryString(*q*, *labels=[]*, *props=[]*)
Get the query string for a given query or queryset.

Returns string including labels and props

getUrl(*queryStr*)

query(*q*, *labels=[]*, *props=[]*)
Actually run a query over the API.

Returns dict of the interpreted JSON or None on failure

readFromCache(*queryStr*)
Load the query result from the cache, if possible.

Returns None if the data is not there or if it is too old.

saveToCache(*q*, *data*)
Save data from a query to a cache file, if enabled.

Return type None

pywikibot.data.wikidataquery.**fromClaim**(*claim*)
Construct from a pywikibot.page Claim object.

Return type *Query*

pywikibot.data.wikidataquery.**listify**(*x*)
If given a non-list, encapsulate in a single-element list.

Return type list

families Package

families Package Families package.

anarchopedia_family Module Family module for Anarchopedia wiki.

class pywikibot.families.anarchopedia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Anarchopedia wiki.

apipath(*code*)
Return the path to api.php for this family.

interwiki_replacements = {'nob': 'no', 'epo': 'eo', 'lav': 'lv', 'heb': 'he', 'lit': 'lt', 'nor': 'no', 'jpn': 'ja', 'por': 'p

path(*code*)
Return the path to index.php for this family.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

battlestarwiki_family Module Family module for Battlestar Wiki.

class pywikibot.families.battlestarwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Battlestar Wiki.

langs = {'fr': 'fr.battlestarwiki.org', 'ms': 'ms.battlestarwiki.org', 'es': 'es.battlestarwiki.org', 'en': 'en.battlestarwiki.o

languages_by_size = ['en', 'de', 'fr', 'zh', 'es', 'ms', 'tr', 'simple']

name = 'battlestarwiki'

commons_family Module Family module for Wikimedia Commons.

class pywikibot.families.commons_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia Commons.

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

i18n_family Module Family module for Translate Wiki.

class pywikibot.families.i18n_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Translate Wiki.

langs = {'i18n': 'translatewiki.net'}

name = 'i18n'

protocol(*code*)
Return https as the protocol for this family.

incubator_family Module Family module for Incubator Wiki.

class pywikibot.families.incubator_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Incubator Wiki.

lyricwiki_family Module Family module for LyricWiki.

class pywikibot.families.lyricwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for LyricWiki.

langs = {'en': 'lyrics.wikia.com'}

name = 'lyricwiki'

scriptpath(*code*)
Return the script path for this family.

mediawiki_family Module Family module for MediaWiki wiki.

class pywikibot.families.mediawiki_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family module for MediaWiki wiki.

meta_family Module Family module for Meta Wiki.

class pywikibot.families.meta_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Meta Wiki.

omegawiki_family Module Family module for Omega Wiki.

class pywikibot.families.omegawiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Omega Wiki.

ignore_certificate_error(*code*)
Ignore certificate errors.

langs = {'omegawiki': 'www.omegawiki.org'}

name = 'omegawiki'

protocol(*code*)
Return https as the protocol for this family.

scriptpath(*code*)
Return the script path for this family.

osm_family Module Family module for OpenStreetMap wiki.

class pywikibot.families.osm_family.**Family**
Bases: *pywikibot.family.Family*

Family class for OpenStreetMap wiki.

langs = {'en': 'wiki.openstreetmap.org'}

name = 'osm'

protocol(*code*)
Return https as the protocol for this family.

outreach_family Module Family module for Wikimedia outreach wiki.

class pywikibot.families.outreach_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia outreach wiki.

species_family Module Family module for Wikimedia species wiki.

class pywikibot.families.species_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia species wiki.

strategy_family Module Family module for Wikimedia Strategy Wiki.

class pywikibot.families.strategy_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia Strategy Wiki.

dbName(*code*)
Return the database name for this family.

test_family Module Family module for test.wikipedia.org.

class pywikibot.families.test_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for test.wikipedia.org.

from_url(*url*)
Return None to indicate no code of this family is accepted.

langs = {'test': 'test.wikipedia.org'}

name = 'test'

vikidia_family Module Family module for Vikidia.

class pywikibot.families.vikidia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Vikidia.

ignore_certificate_error(*code*)
Ignore certificate errors.

langs = {'ca': 'ca.vikidia.org', 'fr': 'fr.vikidia.org', 'es': 'es.vikidia.org', 'en': 'en.vikidia.org', 'it': 'it.vikidia.org', 'ru':

name = 'vikidia'

protocol(*code*)
Return https as the protocol for this family.

wikia_family Module Family module for Wikia.

class pywikibot.families.wikia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Wikia.

apipath(*code*)
Return the path to api.php for this family.

hostname(*code*)
Return the hostname for every site in this family.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

wikibooks_family Module Family module for Wikibooks.

class pywikibot.families.wikibooks_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikibooks.

closed_wikis = ['aa', 'ak', 'als', 'as', 'ast', 'ay', 'ba', 'bi', 'bm', 'bo', 'ch', 'co', 'ga', 'got', 'gn', 'gu', 'kn', 'ks', 'lb', 'ln

removed_wikis = ['tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this family.

wikidata_family Module Family module for Wikidata.

class pywikibot.families.wikidata_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikidata.

calendarmodel(*code*)
Default calendar model for WbTime datatype.

globes(*code*)
Supported globes for Coordinate datatype.

shared_data_repository(*code*, *transcluded=False*)
Indicate Wikidata is both a repository and its own client.

Until 20 August 2014, Wikidata was only a data repository, and this method only returned a tuple with data if transcluded was False.

On that date, the software was enhanced so that Wikidata could store sitelinks to itself.

wikimedia_family Module Family module for Wikimedia chapter wikis.

class pywikibot.families.wikimedia_family.**Family**
Bases: *pywikibot.family.Family*

Family for Wikimedia chapters hosted on wikimedia.org.

wikinews_family Module Family module for Wikinews.

class pywikibot.families.wikinews_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikinews.

closed_wikis = ['hu', 'nl', 'sd', 'th']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikipedia_family Module Family module for Wikipedia.

class pywikibot.families.wikipedia_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family module for Wikipedia.

closed_wikis = ['aa', 'cho', 'ho', 'hz', 'ii', 'kj', 'kr', 'mh', 'mo', 'mus']

code2encodings(*code*)
Return a list of historical encodings for a specific site.

get_known_families(*site*)
Override the family interwiki prefixes for each site.

removed_wikis = ['ng', 'ru-sib', 'tlh', 'tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikiquote_family Module Family module for Wikiquote.

class pywikibot.families.wikiquote_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikiquote.

closed_wikis = ['als', 'ang', 'ast', 'bm', 'co', 'cr', 'ga', 'kk', 'kr', 'ks', 'kw', 'lb', 'na', 'nds', 'qu', 'simple', 'tk', 'tt', 'u

code2encodings(*code*)
Return a list of historical encodings for a specific language.

Parameters **code** – site code

removed_wikis = ['tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this family.

wikisource_family Module Family module for Wikisource.

class pywikibot.families.wikisource_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikisource.

closed_wikis = ['ang', 'ht']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikitech_family Module Family module for Wikitech.

class pywikibot.families.wikitech_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Wikitech.

langs = {'en': 'wikitech.wikimedia.org'}

name = 'wikitech'

protocol(*code*)
Return the protocol for this family.

wikiversity_family Module Family module for Wikiversity.

class pywikibot.families.wikiversity_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikiversity.

wikivoyage_family Module Family module for Wikivoyage.

class pywikibot.families.wikivoyage_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikivoyage.

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wiktionary_family Module Family module for Wiktionary.

class pywikibot.families.wiktionary_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wiktionary.

closed_wikis = ['aa', 'ab', 'ak', 'als', 'as', 'av', 'ba', 'bh', 'bi', 'bm', 'bo', 'ch', 'cr', 'dz', 'ik', 'mh', 'mo', 'pi', 'rm', 'r

removed_wikis = ['tokipona']

wowwiki_family Module Family module for WOW Wiki.

class pywikibot.families.wowwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for WOW Wiki.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

userinterfaces Package

userinterfaces Package User interfaces.

cgi_interface Module CGI user interface.

class pywikibot.userinterfaces.cgi_interface.**UI**
Bases: object

CGI user interface.

input(*question*, *colors=None*)

output(*text*, *colors=None*, *newline=True*, *toStdout=False*)

gui Module

terminal_interface Module Platform independent terminal interface module.

It imports the appropriate operating system specific implementation.

pywikibot.userinterfaces.terminal_interface.**UI**
alias of UnixUI

terminal_interface_base Module Base for terminal user interfaces.

class pywikibot.userinterfaces.terminal_interface_base.**MaxLevelFilter**(*level=None*)
Bases: logging.Filter

Filter that only passes records at or below a specific level.

(setting handler level only passes records at or *above* a specified level, so this provides the opposite functionality)

filter(*record*)
Return true if the level is below or equal to the set level.

class pywikibot.userinterfaces.terminal_interface_base.**TerminalFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
Bases: logging.Formatter

Terminal logging formatter.

class pywikibot.userinterfaces.terminal_interface_base.**TerminalHandler**(*UI*, *strm=None*)
Bases: logging.Handler

A handler class that writes logging records to a terminal.

This class does not close the stream, as sys.stdout or sys.stderr may be (and usually will be) used.

Slightly modified version of the StreamHandler class that ships with logging module, plus code for colorization of output.

emit(*record*)
Emit the record formatted to the output and return it.

flush()
Flush the stream.

sharedlock = <_thread.RLock owner=0 count=0>

threading = <module 'threading' from '/usr/lib/python3.4/threading.py'>

class pywikibot.userinterfaces.terminal_interface_base.**UI**
Bases: object

Base for terminal user interfaces.

argvu()
Return the decoded arguments from argv.

askForCaptcha(*url*)
Show the user a CAPTCHA image and return the answer.

editText(*text*, *jumpIndex=None*, *highlight=None*)
Return the text as edited by the user.

Uses a Tkinter edit box because we don't have a console editor

Parameters

- **text** (*unicode*) – the text to be edited
- **jumpIndex** (*int*) – position at which to put the caret
- **highlight** (*unicode*) – each occurrence of this substring will be highlighted

Returns the modified text, or None if the user didn't save the text file in his text editor

Return type unicode or None

init_handlers(*root_logger*, *default_stream='stderr'*)
Initialize the handlers for user output.

This method initializes handler(s) for output levels VERBOSE (if enabled by config.verbose_output), INFO, STDOUT, WARNING, ERROR, and CRITICAL. STDOUT writes its output to sys.stdout; all the others write theirs to sys.stderr.

input(*question*, *password=False*, *default=''*, *force=False*)
Ask the user a question and return the answer.

Works like raw_input(), but returns a unicode string instead of ASCII.

Unlike raw_input, this function automatically adds a colon and space after the question if they are not already present. Also recognises a trailing question mark.

Parameters

- **question** (*basestring*) – The question, without trailing whitespace.
- **password** (*bool*) – if True, hides the user's input (for password entry).
- **default** (*basestring*) – The default answer if none was entered. None to require an answer.
- **force** (*bool*) – Automatically use the default

Return type unicode

inputChoice(*question*, *options*, *hotkeys*, *default=None*)
Ask the user a question with a predefined list of acceptable answers.

DEPRECATED: Use *input_choice* instead!

Directly calls *input_choice* with the options and hotkeys zipped into a tuple list. It always returns the hotkeys and throws no `QuitKeyboardInterrupt` if quit was selected.

input_choice(*question*, *options*, *default=None*, *return_shortcut=True*, *automatic_quit=True*, *force=False*)
Ask the user and returns a value from the options.

Parameters

- **question** (*basestring*) – The question, without trailing whitespace.
- **options** (*iterable containing iterables of length 2*) – All available options. Each entry contains the full length answer and a shortcut of only one character. The shortcut must not appear in the answer.
- **default** (*basestring*) – The default answer if no was entered. None to require an answer.
- **return_shortcut** (*bool*) – Whether the shortcut or the index in the option should be returned.
- **automatic_quit** (*bool or int*) – Adds the option 'Quit' ('q') and throw a `QuitKeyboardInterrupt` if selected. If it's an integer it doesn't add the option but throw the exception when the option was selected.
- **force** (*bool*) – Automatically use the default

Returns If return_shortcut the shortcut of options or the value of default (if it's not None). Otherwise the index of the answer in options. If default is not a shortcut, it'll return -1.

Return type int (if not return_shortcut), lowercased basestring (otherwise)

input_list_choice(*question*, *answers*, *default=None*, *force=False*)
Ask the user to select one entry from a list of entries.

output(*text*, *toStdout=False*, *targetStream=None*)
Output text to a stream.

If a character can't be displayed in the encoding used by the user's terminal, it will be replaced with a question mark or by a transliteration.

printColorized(*text*, *targetStream*)
Write the text non colorized to the target stream.

To each line which contains a color tag a ' ***' is added at the end.

printNonColorized(*text*, *targetStream*)
Write the text non colorized to the target stream.

To each line which contains a color tag a ' ***' is added at the end.

terminal_interface_unix Module User interface for unix terminals.

class pywikibot.userinterfaces.terminal_interface_unix.**UnixUI**
Bases: *pywikibot.userinterfaces.terminal_interface_base.UI*

User interface for unix terminals.

printColorized(*text*, *targetStream*)
Print the text colorized using the Unix colors.

terminal_interface_win32 Module User interface for Win32 terminals.

class pywikibot.userinterfaces.terminal_interface_win32.**Win32BaseUI**
Bases: *pywikibot.userinterfaces.terminal_interface_base.UI*

User interface for Win32 terminals without ctypes.

class pywikibot.userinterfaces.terminal_interface_win32.**Win32CtypesUI**
Bases: *pywikibot.userinterfaces.terminal_interface_win32.Win32BaseUI*

User interface for Win32 terminals using ctypes.

printColorized(*text*, *targetStream*)

pywikibot.userinterfaces.terminal_interface_win32.**Win32UI**
alias of *Win32CtypesUI*

transliteration Module Module to transliterate text.

class pywikibot.userinterfaces.transliteration.**transliterator**(*encoding*)
Bases: object

Class to transliterating text.

transliterate(*char*, *default='?'*, *prev='-'*, *next='-'*)

win32_unicode Module Stdout, stderr and argv support for unicode.

comms Package

comms Package

Communication layer.

http Module

Basic HTTP access interface.

This module handles communication between the bot and the HTTP threads.

This module is responsible for

- Setting up a connection pool
- Providing a (blocking) interface for HTTP requests
- Translate site objects with query strings into URLs
- URL-encoding all data
- Basic HTTP error handling

pywikibot.comms.http.**error_handling_callback**(*request*)
Raise exceptions and log alerts.

Parameters **request** – Request that has completed

Rtype request threadedhttp.HttpRequest

pywikibot.comms.http.**fetch**(*uri*, *method='GET'*, *body=None*, *headers=None*, *default_error_handling=True*, ***kwargs*)
Blocking HTTP request.

Note: The callback runs in the HTTP thread, where exceptions are logged but are not able to be caught.

See httplib2.Http.request for parameters.

Parameters **default_error_handling** (*bool*) – Use default error handling

Return type threadedhttp.HttpRequest

pywikibot.comms.http.**request**(*site=None*, *uri=None*, *method='GET'*, *body=None*, *headers=None*, ***kwargs*)
Request to Site with default error handling and response decoding.

See httplib2.Http.request for additional parameters.

If the site argument is provided, the uri is a relative uri from and including the document root '/'.

If the site argument is None, the uri must be absolute.

Parameters

- **site** (*pywikibot.site.BaseSite*) – The Site to connect to
- **uri** (*str*) – the URI to retrieve
- **charset** (*CodecInfo, str, None*) – Either a valid charset (usable for str.decode()) or None to automatically chose the charset from the returned header (defaults to latin-1)

Returns The received data

Return type a unicode string

pywikibot.comms.http.**user_agent**(*site=None*, *format_string=None*)
Generate the user agent string for a given site and format.

Parameters

- **site** (BaseSite) – The site for which this user agent is intended. May be None.

- **format_string** (*basestring*) – The string to which the values will be added using str.format. Is using config.user_agent_format when it is None.

Returns The formatted user agent

Return type unicode

pywikibot.comms.http.**user_agent_username**(*username=None*)
Reduce username to a representation permitted in HTTP headers.

To achieve that, this function:: 1) replaces spaces (' ') with '_' 2) encodes the username as 'utf-8' and if the username is not ASCII 3) URL encodes the username if it is not ASCII, or contains '%'

threadedhttp Module

Httplib2 threaded cookie layer.

This class extends httplib2, adding support for::

- Cookies, guarded for cross-site redirects
- Thread safe ConnectionPool class
- HttpProcessor thread class
- HttpRequest object

class pywikibot.comms.threadedhttp.**ConnectionPool**(*maxnum=5*)
Bases: object

A thread-safe connection pool.

pop_connection(*identifier*)
Get a connection from identifier's connection pool.

Parameters **identifier** – The pool identifier

Returns A connection object if found, None otherwise

push_connection(*identifier*, *connection*)
Add a connection to identifier's connection pool.

Parameters

- **identifier** – The pool identifier
- **connection** – The connection to add to the pool

class pywikibot.comms.threadedhttp.**DummyMessage**(*response*)
Bases: object

Simulated mimetools.Message object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

get_all(*k*, *failobj=None*)

getheaders(*k*)

class pywikibot.comms.threadedhttp.**DummyRequest**(*url*, *headers=None*)
Bases: object

Simulated urllib2.Request object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

add_unredirected_header(*key*, *val*)

get_full_url()

get_header(*key*, *default=None*)

get_host()

get_origin_req_host()

get_type()

has_header(*key*)

is_unverifiable()

unverifiable

class pywikibot.comms.threadedhttp.**DummyResponse**(*response*)
Bases: object

Simulated urllib2.Request object for httplib2.

Implements only what's necessary for cookielib.CookieJar to work.

info()

class pywikibot.comms.threadedhttp.**Http**(**args*, ***kwargs*)
Bases: httplib2.Http

Subclass of httplib2.Http that stores cookies.

Overrides httplib2's internal redirect support to prevent cookies being eaten by the wrong sites.

request(*uri*, *method='GET'*, *body=None*, *headers=None*, *max_redirects=None*, *connection_type=None*)
Start an HTTP request.

Parameters

- **uri** – The uri to retrieve
- **method** – (optional) The HTTP method to use. Default is 'GET'
- **body** – (optional) The request body. Default is no body.
- **headers** – (optional) Additional headers to send. Defaults include connection: keep-alive, user-agent and content-type.
- **max_redirects** – (optional) The maximum number of redirects to use for this request. The class instance's max_redirects is default
- **connection_type** – (optional) see httplib2.Http.request

Returns (response, content) tuple

class pywikibot.comms.threadedhttp.**HttpProcessor**(*queue*, *cookiejar*, *connection_pool*)
Bases: threading.Thread

Thread object to spawn multiple HTTP connection threads.

run()

class pywikibot.comms.threadedhttp.**HttpRequest**(*uri*, *method='GET'*, *body=None*, *headers=None*, *callbacks=None*, *charset=None*, ***kwargs*)
Bases: *pywikibot.tools.UnicodeMixin*

Object wrapper for HTTP requests that need to block origin thread.

Usage:

```
>>> from .http import Queue
>>> queue = Queue.Queue()
>>> cookiejar = cookielib.CookieJar()
>>> connection_pool = ConnectionPool()
>>> proc = HttpProcessor(queue, cookiejar, connection_pool)
>>> proc.setDaemon(True)
>>> proc.start()
>>> request = HttpRequest('https://hostname.invalid/')
>>> queue.put(request)
>>> request.lock.acquire()
```

True >>> print(type(request.data)) <class 'httplib2.ServerNotFoundError'> >>> print(request.data) Unable to find the server at hostname.invalid >>> queue.put(None) # Stop the http processor thread

`request.lock.acquire()` will block until the data is available.

self.data will be either:: * a tuple of (dict, unicode) if the request was successful * an exception

content
Return the response decoded by the detected encoding.

data
Return the httplib2 response tuple.

decode (*encoding*, *errors='strict'*)
Return the decoded response.

encoding
Detect the response encoding.

exception
Get the exception raised by httplib2, if any.

header_encoding
Return charset given by the response header.

hostname
Return the host of the request.

parsed_uri
Return the parsed requested uri.

raw
Return the raw response body.

response_headers
Return the response headers.

status
HTTP response status.

Return type int

compat Package

compat Package

Package to provide compatibility with compat scripts.

catlib Module

WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

class pywikibot.compat.catlib.**Category**(*source*, *title=''*, *sortKey=None*)
Bases: pywikibot.page.Page

A page in the Category: namespace.

articles(*recurse=False*, *step=None*, *total=None*, *content=False*, *namespaces=None*, *sortby=None*, *starttime=None*, *endtime=None*, *startsort=None*, *endsort=None*)
Yield all articles in the current category.

By default, yields all *pages* in the category that are not subcategories!

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate articles in subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate articles in first-level subcats, but no deeper.)
- **step** – limit each API call to this number of pages
- **total** – iterate no more than this number of pages in total (at all levels)
- **namespaces** (*int or list of ints*) – only yield pages in the specified namespaces
- **content** – if True, retrieve the content of the current version of each page (default False)
- **sortby** (*str*) – determines the order in which results are generated, valid values are "sortkey" (default, results ordered by category sort key) or "timestamp" (results ordered by time page was added to the category). This applies recursively.
- **starttime** (*pywikibot.Timestamp*) – if provided, only generate pages added after this time; not valid unless sortby="timestamp"
- **endtime** (*pywikibot.Timestamp*) – if provided, only generate pages added before this time; not valid unless sortby="timestamp"
- **startsort** (*str*) – if provided, only generate pages >= this title lexically; not valid if sortby="timestamp"
- **endsort** (*str*) – if provided, only generate pages <= this title lexically; not valid if sortby="timestamp"

articlesList(*recurse=False*)
DEPRECATED: equivalent to list(self.articles(...)).

aslink(*sortKey=None*)
Return a link to place a page in this Category.

Use this only to generate a "true" category link, not for interwikis or text links to category pages.

Parameters **sortKey** (*(optional) unicode*) – The sort key for the article to be placed in this Category; if omitted, default sort key is used.

categoryinfo
Return a dict containing information about the category.

The dict contains values for:

Numbers of pages, subcategories, files, and total contents.

Returns dict

copyAndKeep(*catname*, *cfdTemplates*, *message*)
Copy partial category page text (not contents) to a new title.

Like copyTo above, except this removes a list of templates (like deletion templates) that appear in the old category text. It also removes all text between the two HTML comments BEGIN CFD TEMPLATE and END CFD TEMPLATE. (This is to deal with CFD templates that are substituted.)

Returns true if copying was successful, false if target page already existed.

Parameters

- **catname** – New category title (without namespace)
- **cfdTemplates** – A list (or iterator) of templates to be removed from the page text

Returns True if copying was successful, False if target page already existed.

copyTo(*cat*, *message*)
Copy text of category page to a new page. Does not move contents.

Parameters

- **cat** (*unicode or Category*) – New category title (without namespace) or Category object
- **message** (*unicode*) – message to use for category creation message If two %s are provided in message, will be replaced by (self.title, authorsList)

Returns True if copying was successful, False if target page already existed.

isEmptyCategory()
Return True if category has no members (including subcategories).

isHiddenCategory()
Return True if the category is hidden.

members(*recurse=False*, *namespaces=None*, *step=None*, *total=None*, *content=False*)
Yield all category contents (subcats, pages, and files).

newest_pages(*total=None*)
Return pages in a category ordered by the creation date.

If two or more pages are created at the same time, the pages are returned in the order they were added to the category. The most recently added page is returned first.

It only allows to return the pages ordered from newest to oldest, as it is impossible to determine the oldest page in a category without checking all pages. But it is possible to check the category in order with the newly added first and it yields all pages which were created after the currently checked page was added (and thus there is no page created after any of the cached but added before the currently checked).

Parameters **total** (*int*) – The total number of pages queried.

Returns A page generator of all pages in a category ordered by the creation date. From newest to oldest. Note: It currently only returns Page instances and not a subclass of it if possible. This might change so don't expect to only get Page instances.

Return type generator

subcategories(*recurse=False*, *step=None*, *total=None*, *content=False*)
Iterate all subcategories of the current category.

Parameters

- **recurse** (*int or bool*) – if not False or 0, also iterate subcategories of subcategories. If an int, limit recursion to this number of levels. (Example: recurse=1 will iterate direct subcats and first-level sub-sub-cats, but no deeper.)

- **step** – limit each API call to this number of categories
- **total** – iterate no more than this number of subcategories in total (at all levels)
- **content** – if True, retrieve the content of the current version of each category description page (default False)

subcategoriesList(*recurse=False*)
DEPRECATED: Equivalent to list(self.subcategories(...)).

supercategories()
DEPRECATED: equivalent to self.categories().

supercategoriesList()
DEPRECATED: equivalent to list(self.categories(...)).

pywikibot.compat.catlib.**change_category**(*article*, *oldCat*, *newCat*, *comment=None*, *sortKey=None*, *inPlace=True*)
Change the category of the article.

query Module

WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

pywikibot.compat.query.**GetData**(*request*, *site=None*, *back_response=False*)
Query the server with the given request dict.

DEPRECATED: Use pywikibot.data.api.Request instead.

userlib Module

WARNING: THIS MODULE EXISTS SOLELY TO PROVIDE BACKWARDS-COMPATIBILITY.

Do not use in new scripts; use the source to find the appropriate function/method instead.

class pywikibot.compat.userlib.**User**(*source*, *title=''*)
Bases: pywikibot.page.Page

A class that represents a Wiki user.

This class also represents the Wiki page User:<username>

block(*expiry*, *reason*, *anononly=True*, *nocreate=True*, *autoblock=True*, *noemail=False*, *reblock=False*)
Block user.

Parameters

- **expiry** (*pywikibot.Timestamp|str*) – When the block should expire
- **reason** (*basestring*) – Block reason
- **anononly** (*bool*) – Whether block should only affect anonymous users
- **nocreate** (*bool*) – Whether to block account creation
- **autoblock** (*bool*) – Whether to enable autoblock
- **noemail** (*bool*) – Whether to disable email access
- **reblock** (*bool*) – Whether to reblock if a block already is set

Returns None

contributions (*total=500*, *namespaces=[]*)
Yield tuples describing this user edits.

Each tuple is composed of a pywikibot.Page object, the revision id (int), the edit timestamp (as a pywikibot.Timestamp object), and the comment (unicode). Pages returned are not guaranteed to be unique.

Parameters

- **total** (*int*) – limit result to this number of pages
- **namespaces** (*list*) – only iterate links in these namespaces

editCount (*force=False*)
Return edit count for a registered user.

Always returns 0 for 'anonymous' users.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long

editedPages (*total=500*)
DEPRECATED. Use contributions().

Yields pywikibot.Page objects that this user has edited, with an upper bound of 'total'. Pages returned are not guaranteed to be unique.

Parameters **total** (*int.*) – limit result to this number of pages.

getUserPage (*subpage=''*)
Return a Page object relative to this user's main page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main page title (optional)

getUserTalkPage (*subpage=''*)
Return a Page object relative to this user's main talk page.

Parameters **subpage** (*unicode*) – subpage part to be appended to the main talk page title (optional)

getprops (*force=False*)
Return a properties about the user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns dict

groups (*force=False*)
Return a list of groups to which this user belongs.

The list of groups may be empty.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns list

isAnonymous ()
Determine if the user is editing as an IP address.

Returns bool

isBlocked (*force=False*)
Determine whether the user is currently blocked.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isEmailable(*force=False*)
Determine whether emails may be send to this user through MediaWiki.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

isRegistered(*force=False*)
Determine if the user is registered on the site.

It is possible to have a page named User:xyz and not have a corresponding user with username xyz.

The page does not need to exist for this method to return True.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns bool

name()
The username.

Returns unicode

registration(*force=False*)
Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns pywikibot.Timestamp or None

registrationTime(*force=False*)
DEPRECATED. Fetch registration date for this user.

Parameters **force** (*bool*) – if True, forces reloading the data from API

Returns long (MediaWiki's internal timestamp format) or 0

sendMail(*subject*, *text*, *ccme=False*)
Send an email to this user via MediaWiki's email interface.

Outputs 'Email sent' if the email was sent.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body
- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises **_EmailUserError** logged in user does not have 'sendemail' right or the target has disabled receiving emails

Returns operation successful indicator

Return type bool

send_email(*subject*, *text*, *ccme=False*)
Send an email to this user via MediaWiki's email interface.

Parameters

- **subject** (*unicode*) – the subject header of the mail
- **text** (*unicode*) – mail body
- **ccme** (*bool*) – if True, sends a copy of this email to the bot

Raises

- **NotEmailableError** – the user of this User is not emailable
- **UserRightsError** – logged in user does not have 'sendemail' right

Returns operation successful indicator

Return type bool

uploadedImages(*total=10*)

Yield tuples describing files uploaded by this user.

Each tuple is composed of a pywikibot.Page, the timestamp (str in ISO8601 format), comment (unicode) and a bool for pageid > 0. Pages returned are not guaranteed to be unique.

Parameters **total** (*int*) – limit result to this number of pages

username

The username.

Convenience method that returns the title of the page with namespace prefix omitted, which is the username.

Returns unicode

data Package

data Package

Module providing several layers of data access to the wiki.

api Module

Interface to Mediawiki's api.php.

exception pywikibot.data.api.**APIError**(*code*, *info*, ***kwargs*)

Bases: pywikibot.exceptions.Error

The wiki site returned an error message.

class pywikibot.data.api.**APIGenerator**(*action*, *continue_name='continue'*, *limit_name='limit'*, *data_name='data'*, ***kwargs*)

Bases: object

Iterator that handle API responses containing lists.

The iterator will iterate each item in the query response and use the continue request parameter to retrieve the next portion of items automatically. If the limit attribute is set, the iterator will stop after iterating that many values.

set_maximum_items(*value*)

Set the maximum number of items to be retrieved from the wiki.

If not called, most queries will continue as long as there is more data to be retrieved from the API.

Parameters **value** (*int*) – The value of maximum number of items to be retrieved in total to set.

set_query_increment(*value*)
Set the maximum number of items to be retrieved per API query.

If not called, the default is 50.

Parameters **value** (*int*) – The value of maximum number of items to be retrieved per API request to set.

exception pywikibot.data.api.**APIMWException**(*mediawiki_exception_class_name*, *info*, ***kwargs*)
Bases: *pywikibot.data.api.APIError*

The API site returned an error about a MediaWiki internal exception.

class pywikibot.data.api.**CTEBinaryBytesGenerator**(**args*, ***kwargs*)
Bases: email.generator.BytesGenerator

Workaround for bug in python 3 email handling of CTE binary.

class pywikibot.data.api.**CTEBinaryMIMEMultipart**(*_subtype='mixed'*, *boundary=None*, *_subparts=None*, ***_params*)
Bases: email.mime.multipart.MIMEMultipart

Workaround for bug in python 3 email handling of CTE binary.

as_bytes(*unixfrom=False*, *policy=None*)
Return unmodified binary payload.

class pywikibot.data.api.**CachedRequest**(*expiry*, **args*, ***kwargs*)
Bases: *pywikibot.data.api.Request*

Cached request.

submit()
Submit cached request.

pywikibot.data.api.**CategoryPageGenerator**()
Like PageGenerator, but yields Category objects instead of Pages.

class pywikibot.data.api.**EnableSSLSiteWrapper**(*site*)
Bases: object

Wrapper to change the site protocol to https.

protocol()
Return protocol.

pywikibot.data.api.**ImagePageGenerator**()
Like PageGenerator, but yields FilePage objects instead of Pages.

class pywikibot.data.api.**ListGenerator**(*listaction*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for queries of type action=query&list=foo.

See the API documentation for types of lists that can be queried. Lists include both side-wide information (such as 'allpages') and page-specific information (such as 'backlinks').

This iterator yields a dict object for each member of the list returned by the API, with the format of the dict depending on the particular list command used. For those lists that contain page information, it may be easier to use the PageGenerator class instead, as that will convert the returned information into a Page object.

class pywikibot.data.api.**LogEntryListGenerator**(*logtype=None*, ***kwargs*)
Bases: *pywikibot.data.api.ListGenerator*

Iterator for queries of list 'logevents'.

Yields LogEntry objects instead of dicts.

result (*pagedata*)
Instatiate LogEntry from data from api.

class pywikibot.data.api.**LoginManager** (*password=None*, *sysop=False*, *site=None*, *user=None*)
Bases: *pywikibot.login.LoginManager*

Supply getCookie() method to use API interface.

getCookie (*remember=True*, *captchaId=None*, *captchaAnswer=None*)
Login to the site.

Parameters are all ignored.

Returns cookie data if successful, None otherwise.

storecookiedata (*data*)
Ignore data; cookies are set by threadedhttp module.

pywikibot.data.api.**MIMEMultipart**
alias of *CTEBinaryMIMEMultipart*

class pywikibot.data.api.**OptionSet** (*site=None*, *module=None*, *param=None*, *dict=None*)
Bases: collections.abc.MutableMapping

A class to store a set of options which can be either enabled or not.

If it is instantiated with the associated site, module and parameter it will only allow valid names as options. If instantiated 'lazy loaded' it won't checks if the names are valid until the site has been set (which isn't required, but recommended). The site can only be set once if it's not None and after setting it, any site (even None) will fail.

api_iter()
Iterate over each option as they appear in the URL.

clear()
Clear all enabled and disabled options.

from_dict (*dict*)
Load options from the dict.

The options are not cleared before. If changes have been made previously, but only the dict values should be applied it needs to be cleared first.

Parameters **dict** (*dict (keys are strings, values are bool/None)*) – A dictionary containing for each entry either the value False, True or None. The names must be valid depending on whether they enable or disable the option. All names with the value None can be in either of the list.

class pywikibot.data.api.**PageGenerator** (*generator*, *g_content=False*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for response to a request of type action=query&generator=foo.

This class can be used for any of the query types that are listed in the API documentation as being able to be used as a generator. Instances of this class iterate Page objects.

result (*pagedata*)
Convert page dict entry from api to Page object.

This can be overridden in subclasses to return a different type of object.

class pywikibot.data.api.**ParamInfo**(*site*, *preloaded_modules=None*, *modules_only_mode=None*)
Bases: collections.abc.Container

API parameter information data object.

Provides cache aware fetching of parameter information.

Full support for MW 1.12+, when 'paraminfo' was introduced to the API. Partially supports MW 1.11, using data extracted from API 'help'. MW 1.10 not supported as module prefixes are not extracted from API 'help'.

TODO: Rewrite help parser to support earlier releases.

TODO: establish a data structure in the class which prefills the param information available for a site given its version, using the API information available for each API version.

TODO: module aliases: in 1.25wmf list=deletedrevs becomes list=alldeletedrevisions prop=deletedrevs becomes prop=deletedrevisions

TODO: share API parameter information between sites using similar versions of the API, especially all sites in the same family.

action_modules
Set of all action modules.

fetch(*modules*, *_init=False*)
Fetch paraminfo for multiple modules.

No exception is raised when paraminfo for a module does not exist. Use __getitem__ to cause an exception if a module does not exist.

Parameters **modules** (*set*) – API modules to load

Return type NoneType

init_modules = frozenset({'paraminfo', 'main'})

module_attribute_map(*attribute*, *modules=None*)
Mapping of modules with an attribute to the attribute value.

Parameters

- **attribute** (*basestring*) – attribute name
- **modules** (*set*) – modules to include (default: all modules)

Return type dict using modules as keys

modules
Set of all module names without path prefixes.

Only includes one 'tokens', even if it appears as both a action and a query submodule.

Returns module names

Return type set of str

normalize_modules(*modules*)
Convert the modules into module paths.

Add query+ to any query module name not also in action modules.

Returns The modules converted into a module paths

Return type set

classmethod normalize_paraminfo(*data*)
Convert both old and new API JSON into a new-ish data structure.

parameter(*module*, *param_name*)
Get details about one modules parameter.

Returns None if the parameter does not exist.

Parameters

- **module** (*str*) – API module name
- **param_name** (*str*) – parameter name in the module

Returns metadata that describes how the parameter may be used

Return type dict or None

paraminfo_keys = frozenset({'mainmodule', 'formatmodules', 'querymodules', 'pagesetmodule', 'modules'})

prefixes
Mapping of module to its prefix for all modules with a prefix.

This loads paraminfo for all modules.

query_modules
Set of all query module names without query+ path prefix.

query_modules_with_limits
Set of all query modules which have limits.

root_modules = frozenset({'pageset', 'main'})

class pywikibot.data.api.**PropertyGenerator**(*prop*, ***kwargs*)
Bases: *pywikibot.data.api.QueryGenerator*

Iterator for queries of type action=query&prop=foo.

See the API documentation for types of page properties that can be queried.

This iterator yields one or more dict object(s) corresponding to each "page" item(s) from the API response; the calling module has to decide what to do with the contents of the dict. There will be one dict for each page queried via a titles= or ids= parameter (which must be supplied when instantiating this class).

props
The requested property names.

class pywikibot.data.api.**QueryGenerator**(***kwargs*)
Bases: object

Base class for iterators that handle responses to API action=query.

By default, the iterator will iterate each item in the query response, and use the (query-)continue element, if present, to continue iterating as long as the wiki returns additional values. However, if the iterator's limit attribute is set to a positive int, the iterator will stop after iterating that many values. If limit is negative, the limit parameter will not be passed to the API at all.

Most common query types are more efficiently handled by subclasses, but this class can be used directly for custom queries and miscellaneous types (such as "meta=...") that don't return the usual list of pages or links. See the API documentation for specific query options.

result(*data*)
Process result data as needed for particular subclass.

set_maximum_items(*value*)
Set the maximum number of items to be retrieved from the wiki.

If not called, most queries will continue as long as there is more data to be retrieved from the API.

If set to -1 (or any negative value), the "limit" parameter will be omitted from the request. For some request types (such as prop=revisions), this is necessary to signal that only current revision is to be returned.

set_namespace(*namespaces*)
Set a namespace filter on this query.

Parameters **namespaces** (*iterable of basestring or Namespace key, or a single instance of those types. May be a '|' separated list of namespace identifiers. An empty iterator clears any namespace restriction.*) – namespace identifiers to limit query results

Raises

- **KeyError** – a namespace identifier was not resolved
- **TypeError** – a namespace identifier has an inappropriate type such as NoneType or bool, or more than one namespace if the API module does not support multiple namespaces

set_query_increment(*value*)
Set the maximum number of items to be retrieved per API query.

If not called, the default is to ask for "max" items and let the API decide how many to send.

class pywikibot.data.api.**Request**(***kwargs*)
Bases: collections.abc.MutableMapping

A request to a Site's api.php interface.

Attributes of this object (except for the special parameters listed below) get passed as commands to api.php, and can be get or set using the dict interface. All attributes must be strings (or unicode). Use an empty string for parameters that don't require a value. For example, Request(action="query", titles="Foo bar", prop="info", redirects="") corresponds to the API request "api.php?action=query&titles=Foo%20bar&prop=info&redirects"

This is the lowest-level interface to the API, and can be used for any request that a particular site's API supports. See the API documentation (https://www.mediawiki.org/wiki/API) and site-specific settings for details on what parameters are accepted for each request type.

Uploading files is a special case: to upload, the parameter "mime" must be true, and the parameter "file" must be set equal to a valid filename on the local computer, _not_ to the content of the file.

Returns a dict containing the JSON data returned by the wiki. Normally, one of the dict keys will be equal to the value of the 'action' parameter. Errors are caught and raise an APIError exception.

Example:

```
>>> r = Request(action="query", meta="userinfo")
>>> # This is equivalent to
>>> # https://{path}/api.php?action=query&meta=userinfo&format=json
>>> # change a parameter
>>> r['meta'] = "userinfo|siteinfo"
>>> # add a new parameter
>>> r['siprop'] = "namespaces"
>>> # note that "uiprop" param gets added automatically
>>> r.action
```

u'query' >>> sorted(r._params.keys()) # doctest: +IGNORE_UNICODE [u'action', u'meta', u'siprop'] >>> r._params['action'] # doctest: +IGNORE_UNICODE [u'query'] >>> r._params['meta'] # doctest: +IGNORE_UNICODE [u'userinfo', u'siteinfo'] >>> r._params['siprop'] # doctest: +IGNORE_UNICODE [u'namespaces'] >>> data = r.submit() # doctest: +IGNORE_UNICODE >>> isinstance(data, dict) True >>> set(['query', 'batchcomplete', 'warnings']).issuperset(data.keys()) True >>> 'query' in data True >>> sorted(data[u'query'].keys()) # doctest: +IGNORE_UNICODE ['namespaces', 'userinfo']

http_params()
Return the parameters formatted for inclusion in an HTTP request.

DEPRECATED. See _encoded_items for explanation of encoding used.

items()
Return a list of tuples containg the parameters in any order.

iteritems()
Implement dict interface.

keys()
Implement dict interface.

mime
Return whether mime parameters are defined.

submit()
Submit a query and parse the response.

Returns a dict containing data retrieved from api.php

wait()
Determine how long to wait after a failed request.

exception pywikibot.data.api.**TimeoutError**(*arg*)
Bases: pywikibot.exceptions.Error

API request failed with a timeout error.

exception pywikibot.data.api.**UploadWarning**(*code*, *message*)
Bases: *pywikibot.data.api.APIError*

Upload failed with a warning message (passed as the argument).

message
Return warning message.

pywikibot.data.api.**update_page**(*page*, *pagedict*, *props=[]*)
Update attributes of Page object page, based on query data in pagedict.

Parameters

- **page** (Page) – object to be updated
- **pagedict** (*dict*) – the contents of a "page" element of a query response
- **props** (*iterable of string*) – the property names which resulted in pagedict. If a missing value in pagedict can indicate both 'false' and 'not present' the property which would make the value present must be in the props parameter.

wikidataquery Module

Objects representing WikidataQuery query syntax and API.

class pywikibot.data.wikidataquery.**Around**(*prop*, *coord*, *rad*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form around[PROPERTY,LATITUDE,LONGITUDE,RADIUS].

queryType = 'around'

validate()

class pywikibot.data.wikidataquery.**Between**(*prop*, *begin=None*, *end=None*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form between[PROP, BEGIN, END].

You have to give prop and one of begin or end. Note that times have to be in UTC, timezones are not supported by the API

Parameters

- **prop** – the property
- **begin** – WbTime object representing the beginning of the period
- **end** – WbTime object representing the end of the period

queryType = 'between'

validate()

class pywikibot.data.wikidataquery.**HasClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.Query*

This is a Query of the form "claim[prop:val]".

It is subclassed by the other similar forms like noclaim and string

formatItems()

queryType = 'claim'

validate()

class pywikibot.data.wikidataquery.**Link**(*link*)
Bases: *pywikibot.data.wikidataquery.Query*

A query in the form link[LINK,...], which also includes nolink.

All link elements have to be strings, or validation will throw

queryType = 'link'

validate()

class pywikibot.data.wikidataquery.**NoClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.HasClaim*

Query of the form noclaim[PROPERTY].

queryType = 'noclaim'

class pywikibot.data.wikidataquery.**NoLink**(*link*)
Bases: *pywikibot.data.wikidataquery.Link*

A query in the form nolink[..].

queryType = 'nolink'

class pywikibot.data.wikidataquery.**Query**
Bases: object

A query is a single query for the WikidataQuery API.

For example:: claim[100:60] or link[enwiki]

Construction of a Query can throw a TypeError if you feed it bad parameters. Exactly what these need to be depends on the Query

AND(*ands*)
Produce a query set ANDing this query and all the given query/sets.

OR(*ors*)
Produce a query set ORing this query and all the given query/sets.

convertWDType(*item*)
Convert Wikibase items like ItemPage or PropertyPage into integer IDs.

The resulting IDs may be used in query strings.

Parameters **item** – A single item. One of ItemPages, PropertyPages, int or anything that can be fed to int()

Returns the int ID of the item

convertWDTypes(*items*)

formatItem(*item*)
Default item formatting is string.

This will work for queries, querysets, ints and strings

formatList(*l*)
Format and comma-join a list.

static **isOrContainsOnlyTypes**(*items*, *types*)
Either this item is one of the given types, or it is a list of only those types.

Return type bool

validate()
Validate the query parameters.

Default validate result is a pass - subclasses need to implement this if they want to check their parameters.

Returns True

Return type bool

validateOrRaise(*msg=None*)

class pywikibot.data.wikidataquery.**QuerySet**(*q*)
Bases: object

A QuerySet represents a set of queries or other query sets.

Queries may be joined by operators (AND and OR).

A QuerySet stores this information as a list of Query(Sets) and a joiner operator to join them all together

AND(*args*)
Add the given args (Queries or QuerySets) to the Query set as a logical conjuction (AND).

OR(*args*)
Add the given args (Queries or QuerySets) to the Query set as a logical disjunction (OR).

addJoiner(*args*, *joiner*)
Add to this QuerySet using the given joiner.

If the given joiner is not the same as we used before in this QuerySet, nest the current one in parens before joining. This makes the implicit grouping of the API explicit.

Returns a new query set representing the joining of this one and the arguments

class pywikibot.data.wikidataquery.**StringClaim**(*prop*, *items=[]*)
Bases: *pywikibot.data.wikidataquery.HasClaim*

Query of the form string[PROPERTY:"STRING",...].

formatItem(*x*)
Add quotes around string.

queryType = 'string'

validate()

class pywikibot.data.wikidataquery.**Tree**(*item*, *forward=[]*, *reverse=[]*)
Bases: *pywikibot.data.wikidataquery.Query*

Query of the form tree[ITEM,...][PROPERTY,...]<PROPERTY,...>.

queryType = 'tree'

validate()

class pywikibot.data.wikidataquery.**WikidataQuery**(*host='https://wdq.wmflabs.org'*, *cacheDir=None*, *cacheMaxAge=60*)
Bases: object

An interface to the WikidataQuery API.

Default host is https://wdq.wmflabs.org/, but you can substitute a different one.

Caching defaults to a subdir of the system temp directory with a 1 hour max cache age.

Set a zero or negative maxCacheAge to disable caching

getCacheFilename(*queryStr*)
Encode a query into a unique and universally safe format.

Return type unicode

getDataFromHost(*queryStr*)
Go and fetch a query from the host's API.

Return type dict

getQueryString(*q*, *labels=[]*, *props=[]*)
Get the query string for a given query or queryset.

Returns string including labels and props

getUrl(*queryStr*)

query(*q*, *labels=[]*, *props=[]*)
Actually run a query over the API.

Returns dict of the interpreted JSON or None on failure

readFromCache(*queryStr*)
Load the query result from the cache, if possible.

Returns None if the data is not there or if it is too old.

saveToCache(*q*, *data*)
Save data from a query to a cache file, if enabled.

Return type None

pywikibot.data.wikidataquery.**fromClaim**(*claim*)
Construct from a pywikibot.page Claim object.

Return type *Query*

pywikibot.data.wikidataquery.**listify**(*x*)
If given a non-list, encapsulate in a single-element list.

Return type list

families Package

families Package

Families package.

anarchopedia_family Module

Family module for Anarchopedia wiki.

class pywikibot.families.anarchopedia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Anarchopedia wiki.

apipath(*code*)
Return the path to api.php for this family.

interwiki_replacements = {'nob': 'no', 'epo': 'eo', 'lav': 'lv', 'heb': 'he', 'lit': 'lt', 'nor': 'no', 'jpn': 'ja', 'por': 'p

path(*code*)
Return the path to index.php for this family.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

battlestarwiki_family Module

Family module for Battlestar Wiki.

class pywikibot.families.battlestarwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Battlestar Wiki.

langs = {'fr': 'fr.battlestarwiki.org', 'ms': 'ms.battlestarwiki.org', 'es': 'es.battlestarwiki.org', 'en': 'en.battlestarwiki.o

languages_by_size = ['en', 'de', 'fr', 'zh', 'es', 'ms', 'tr', 'simple']

name = 'battlestarwiki'

commons_family Module

Family module for Wikimedia Commons.

class pywikibot.families.commons_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia Commons.

shared_data_repository (*code*, *transcluded=False*)
Return the shared data repository for this site.

i18n_family Module

Family module for Translate Wiki.

class pywikibot.families.i18n_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Translate Wiki.

langs = {'i18n': 'translatewiki.net'}

name = 'i18n'

protocol (*code*)
Return https as the protocol for this family.

incubator_family Module

Family module for Incubator Wiki.

class pywikibot.families.incubator_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Incubator Wiki.

lyricwiki_family Module

Family module for LyricWiki.

class pywikibot.families.lyricwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for LyricWiki.

langs = {'en': 'lyrics.wikia.com'}

name = 'lyricwiki'

scriptpath (*code*)
Return the script path for this family.

mediawiki_family Module

Family module for MediaWiki wiki.

class pywikibot.families.mediawiki_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family module for MediaWiki wiki.

meta_family Module

Family module for Meta Wiki.

class pywikibot.families.meta_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Meta Wiki.

omegawiki_family Module

Family module for Omega Wiki.

class pywikibot.families.omegawiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Omega Wiki.

ignore_certificate_error(*code*)
Ignore certificate errors.

langs = {'omegawiki': 'www.omegawiki.org'}

name = 'omegawiki'

protocol(*code*)
Return https as the protocol for this family.

scriptpath(*code*)
Return the script path for this family.

osm_family Module

Family module for OpenStreetMap wiki.

class pywikibot.families.osm_family.**Family**
Bases: *pywikibot.family.Family*

Family class for OpenStreetMap wiki.

langs = {'en': 'wiki.openstreetmap.org'}

name = 'osm'

protocol(*code*)
Return https as the protocol for this family.

outreach_family Module

Family module for Wikimedia outreach wiki.

class pywikibot.families.outreach_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia outreach wiki.

species_family Module

Family module for Wikimedia species wiki.

class pywikibot.families.species_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia species wiki.

strategy_family Module

Family module for Wikimedia Strategy Wiki.

class pywikibot.families.strategy_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikimedia Strategy Wiki.

dbName(*code*)
Return the database name for this family.

test_family Module

Family module for test.wikipedia.org.

class pywikibot.families.test_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for test.wikipedia.org.

from_url(*url*)
Return None to indicate no code of this family is accepted.

langs = {'test': 'test.wikipedia.org'}

name = 'test'

vikidia_family Module

Family module for Vikidia.

class pywikibot.families.vikidia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Vikidia.

ignore_certificate_error(*code*)
Ignore certificate errors.

langs = {'ca': 'ca.vikidia.org', 'fr': 'fr.vikidia.org', 'es': 'es.vikidia.org', 'en': 'en.vikidia.org', 'it': 'it.vikidia.org', 'ru':

name = 'vikidia'

protocol(*code*)
Return https as the protocol for this family.

wikia_family Module

Family module for Wikia.

class pywikibot.families.wikia_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Wikia.

apipath(*code*)
Return the path to api.php for this family.

hostname(*code*)
Return the hostname for every site in this family.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

wikibooks_family Module

Family module for Wikibooks.

class pywikibot.families.wikibooks_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikibooks.

closed_wikis = ['aa', 'ak', 'als', 'as', 'ast', 'ay', 'ba', 'bi', 'bm', 'bo', 'ch', 'co', 'ga', 'got', 'gn', 'gu', 'kn', 'ks', 'lb', 'ln

removed_wikis = ['tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this family.

wikidata_family Module

Family module for Wikidata.

class pywikibot.families.wikidata_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikidata.

calendarmodel(*code*)
Default calendar model for WbTime datatype.

globes(*code*)
Supported globes for Coordinate datatype.

shared_data_repository(*code*, *transcluded=False*)
Indicate Wikidata is both a repository and its own client.

Until 20 August 2014, Wikidata was only a data repository, and this method only returned a tuple with data if transcluded was False.

On that date, the software was enhanced so that Wikidata could store sitelinks to itself.

wikimedia_family Module

Family module for Wikimedia chapter wikis.

class pywikibot.families.wikimedia_family.**Family**
Bases: *pywikibot.family.Family*

Family for Wikimedia chapters hosted on wikimedia.org.

wikinews_family Module

Family module for Wikinews.

class pywikibot.families.wikinews_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikinews.

closed_wikis = ['hu', 'nl', 'sd', 'th']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikipedia_family Module

Family module for Wikipedia.

class pywikibot.families.wikipedia_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family module for Wikipedia.

closed_wikis = ['aa', 'cho', 'ho', 'hz', 'ii', 'kj', 'kr', 'mh', 'mo', 'mus']

code2encodings(*code*)
Return a list of historical encodings for a specific site.

get_known_families(*site*)
Override the family interwiki prefixes for each site.

removed_wikis = ['ng', 'ru-sib', 'tlh', 'tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikiquote_family Module

Family module for Wikiquote.

class pywikibot.families.wikiquote_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikiquote.

closed_wikis = ['als', 'ang', 'ast', 'bm', 'co', 'cr', 'ga', 'kk', 'kr', 'ks', 'kw', 'lb', 'na', 'nds', 'qu', 'simple', 'tk', 'tt', 'u

code2encodings(*code*)
Return a list of historical encodings for a specific language.

Parameters **code** – site code

removed_wikis = ['tokipona']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this family.

wikisource_family Module

Family module for Wikisource.

class pywikibot.families.wikisource_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikisource.

closed_wikis = ['ang', 'ht']

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wikitech_family Module

Family module for Wikitech.

class pywikibot.families.wikitech_family.**Family**
Bases: *pywikibot.family.Family*

Family class for Wikitech.

langs = {'en': 'wikitech.wikimedia.org'}

name = 'wikitech'

protocol(*code*)
Return the protocol for this family.

wikiversity_family Module

Family module for Wikiversity.

class pywikibot.families.wikiversity_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikiversity.

wikivoyage_family Module

Family module for Wikivoyage.

class pywikibot.families.wikivoyage_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wikivoyage.

shared_data_repository(*code*, *transcluded=False*)
Return the shared data repository for this site.

wiktionary_family Module

Family module for Wiktionary.

class pywikibot.families.wiktionary_family.**Family**
Bases: *pywikibot.family.WikimediaFamily*

Family class for Wiktionary.

closed_wikis = ['aa', 'ab', 'ak', 'als', 'as', 'av', 'ba', 'bh', 'bi', 'bm', 'bo', 'ch', 'cr', 'dz', 'ik', 'mh', 'mo', 'pi', 'rm', 'r

removed_wikis = ['tokipona']

wowwiki_family Module

Family module for WOW Wiki.

class pywikibot.families.wowwiki_family.**Family**
Bases: *pywikibot.family.Family*

Family class for WOW Wiki.

scriptpath(*code*)
Return the script path for this family.

version(*code*)
Return the version for this family.

userinterfaces Package

userinterfaces Package

User interfaces.

cgi_interface Module

CGI user interface.

class pywikibot.userinterfaces.cgi_interface.**UI**
Bases: object

CGI user interface.

input(*question*, *colors=None*)

output(*text*, *colors=None*, *newline=True*, *toStdout=False*)

gui Module

terminal_interface Module

Platform independent terminal interface module.

It imports the appropriate operating system specific implementation.

pywikibot.userinterfaces.terminal_interface.**UI**
alias of UnixUI

terminal_interface_base Module

Base for terminal user interfaces.

class pywikibot.userinterfaces.terminal_interface_base.**MaxLevelFilter**(*level=None*)
Bases: logging.Filter

Filter that only passes records at or below a specific level.

(setting handler level only passes records at or *above* a specified level, so this provides the opposite functionality)

filter(*record*)
Return true if the level is below or equal to the set level.

class pywikibot.userinterfaces.terminal_interface_base.**TerminalFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
Bases: logging.Formatter

Terminal logging formatter.

class pywikibot.userinterfaces.terminal_interface_base.**TerminalHandler**(*UI*, *strm=None*)
Bases: logging.Handler

A handler class that writes logging records to a terminal.

This class does not close the stream, as sys.stdout or sys.stderr may be (and usually will be) used.

Slightly modified version of the StreamHandler class that ships with logging module, plus code for colorization of output.

emit(*record*)
Emit the record formatted to the output and return it.

flush()
Flush the stream.

sharedlock = <_thread.RLock owner=0 count=0>

threading = <module 'threading' from '/usr/lib/python3.4/threading.py'>

class pywikibot.userinterfaces.terminal_interface_base.**UI**
Bases: object

Base for terminal user interfaces.

argvu()
Return the decoded arguments from argv.

askForCaptcha(*url*)
Show the user a CAPTCHA image and return the answer.

editText(*text*, *jumpIndex=None*, *highlight=None*)
Return the text as edited by the user.

Uses a Tkinter edit box because we don't have a console editor

Parameters

- **text** (*unicode*) – the text to be edited
- **jumpIndex** (*int*) – position at which to put the caret
- **highlight** (*unicode*) – each occurrence of this substring will be highlighted

Returns the modified text, or None if the user didn't save the text file in his text editor

Return type unicode or None

init_handlers(*root_logger*, *default_stream='stderr'*)
Initialize the handlers for user output.

This method initializes handler(s) for output levels VERBOSE (if enabled by config.verbose_output), INFO, STDOUT, WARNING, ERROR, and CRITICAL. STDOUT writes its output to sys.stdout; all the others write theirs to sys.stderr.

input(*question*, *password=False*, *default=''*, *force=False*)
Ask the user a question and return the answer.

Works like raw_input(), but returns a unicode string instead of ASCII.

Unlike raw_input, this function automatically adds a colon and space after the question if they are not already present. Also recognises a trailing question mark.

Parameters

- **question** (*basestring*) – The question, without trailing whitespace.
- **password** (*bool*) – if True, hides the user's input (for password entry).
- **default** (*basestring*) – The default answer if none was entered. None to require an answer.
- **force** (*bool*) – Automatically use the default

Return type unicode

inputChoice(*question*, *options*, *hotkeys*, *default=None*)
Ask the user a question with a predefined list of acceptable answers.

DEPRECATED: Use *input_choice* instead!

Directly calls *input_choice* with the options and hotkeys zipped into a tuple list. It always returns the hotkeys and throws no `QuitKeyboardInterrupt` if quit was selected.

input_choice(*question*, *options*, *default=None*, *return_shortcut=True*, *automatic_quit=True*, *force=False*)
Ask the user and returns a value from the options.

Parameters

- **question** (*basestring*) – The question, without trailing whitespace.
- **options** (*iterable containing iterables of length 2*) – All available options. Each entry contains the full length answer and a shortcut of only one character. The shortcut must not appear in the answer.
- **default** (*basestring*) – The default answer if no was entered. None to require an answer.
- **return_shortcut** (*bool*) – Whether the shortcut or the index in the option should be returned.

- **automatic_quit** (*bool or int*) – Adds the option 'Quit' ('q') and throw a `QuitKeyboardInterrupt` if selected. If it's an integer it doesn't add the option but throw the exception when the option was selected.
- **force** (*bool*) – Automatically use the default

Returns If return_shortcut the shortcut of options or the value of default (if it's not None). Otherwise the index of the answer in options. If default is not a shortcut, it'll return -1.

Return type int (if not return_shortcut), lowercased basestring (otherwise)

input_list_choice(*question*, *answers*, *default=None*, *force=False*)
Ask the user to select one entry from a list of entries.

output(*text*, *toStdout=False*, *targetStream=None*)
Output text to a stream.

If a character can't be displayed in the encoding used by the user's terminal, it will be replaced with a question mark or by a transliteration.

printColorized(*text*, *targetStream*)
Write the text non colorized to the target stream.

To each line which contains a color tag a ' ***' is added at the end.

printNonColorized(*text*, *targetStream*)
Write the text non colorized to the target stream.

To each line which contains a color tag a ' ***' is added at the end.

terminal_interface_unix Module

User interface for unix terminals.

class pywikibot.userinterfaces.terminal_interface_unix.**UnixUI**
Bases: *pywikibot.userinterfaces.terminal_interface_base.UI*

User interface for unix terminals.

printColorized(*text*, *targetStream*)
Print the text colorized using the Unix colors.

terminal_interface_win32 Module

User interface for Win32 terminals.

class pywikibot.userinterfaces.terminal_interface_win32.**Win32BaseUI**
Bases: *pywikibot.userinterfaces.terminal_interface_base.UI*

User interface for Win32 terminals without ctypes.

class pywikibot.userinterfaces.terminal_interface_win32.**Win32CtypesUI**
Bases: *pywikibot.userinterfaces.terminal_interface_win32.Win32BaseUI*

User interface for Win32 terminals using ctypes.

printColorized(*text*, *targetStream*)

pywikibot.userinterfaces.terminal_interface_win32.**Win32UI**
alias of *Win32CtypesUI*

transliteration Module

Module to transliterate text.

class pywikibot.userinterfaces.transliteration.**transliterator**(*encoding*)
Bases: object

Class to transliterating text.

transliterate(*char*, *default='?'*, *prev='-'*, *next='-'*)

win32_unicode Module

Stdout, stderr and argv support for unicode.

2.4.3 Test documentation

1. tests/index

CHAPTER 3

Miscellaneous

3.1 Licenses

The framework itself is available under the *MIT license*; the documentation is available under the *CC-BY-SA 3.0* license.

Httplib2, which is provided as 'external' library, is also available under the *MIT license*.

3.1.1 MIT License

The framework is available under the MIT license:

Copyright (c) 2004-2013 Pywikibot team

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

3.1.2 CC-BY-SA 3.0

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

3.2 Credits

3.2.1 pywikibot

```
Pywikibot is an open-source project that cannot exist without your
contributions. We would therefore like to thank everyone who has
contributed:

Ævar Arnfjörð Bjarmason
Alex Shih-Han Lin
Amir Sarabadani
Andre Engels
André Malafaya Baptista
Antoine Musso
Anton
balasyum
bananeweizen
Bináris
borgo
Brion Vibber
Bryan Tong Minh
Christian List
Cyde Weys
Daniel Friesen
Daniel Herding
DMaggot
DrTrigon
Egon
Eranroz
Erwin
Felix Reimann
Filnik
Francesco Cosoleto
Gerard Meijssen
Gerrit Holl
Hazard-SJ
Huji
jeedo
Jeroen de Dauw
Jitse Niesen
John Vandenberg
Jon Harald Søby
Jo Simoens
Karl Eichwalder
Kim Bruning
Kunal Mehta
Ladsgroup
Leonardo Gregianin
Lewis Cawte
Linedwell
Maarten Dammers
Marcin Cieslak
Maxim Razin
Meno25
Merlijn van Deen
Misza13
Mohamed Magdy
```

```
Mpaa
Nicolas Dumazet
notconfusing
Philip Tzou
Platonides
Purodha B Blissenbach
ricordisamoa
Rik Wade
Robert Leverington
Rob W.W. Hooft
Rotem Liss
Russell Blau
Scot Wilcoxon
Shinjiman
Shi Zhao
Siebrand Mazeland
Sorawee Porncharoenwase
Steve Sanbeg
Sumana Harihareswara
Thomas R. Koll
ThomasV
Toto Azéro
Victor Vasiliev
Warddr
Wieland Hoffmann
WikiWichtel
xqt
Yuri Astrakhan
```

3.2.2 documentation

The documentation was initially maintained on mediawiki.org. The contributors of the various pages are listed below:

(...to do...)

p

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y