
pyRevit Documentation

Release 4.6.24

eirannejad

Jun 30, 2019

1	How To Use This Documents	3
2	Create Your First Command	5
3	Anatomy of a pyRevit Script	7
3.1	Script Metadata Variables	7
3.2	pyrevit.script Module	9
3.3	Appendix A: Builtin Parameters Provided by pyRevit Engine	12
3.4	Appendix B: System Category Names	13
4	Effective Output/Input	19
4.1	Clickable Element Links	19
4.2	Tables	20
4.3	Code Output	21
4.4	Progress bars	21
4.5	Standard Prompts	22
4.6	Standard Dialogs	26
4.7	Base Forms	35
4.8	Graphs	37
5	Keyboard Shortcuts	45
5.1	Shift-Click: Alternate/Config Script	45
5.2	Ctrl-Click: Debug Mode	45
5.3	Alt-Click: Show Script file in Explorer	46
5.4	Ctrl-Shift-Alt-Click: Reload Engine	46
5.5	Shift-Win-Click: pyRevit Button Context Menu	46
6	Extensions and Commmands	47
6.1	Why do I need an Extension	47
6.2	Extensions	47
6.3	Command Bundles	48
6.4	Group Bundles	49
6.5	Advanced Bundles	51
6.6	Other Extension Types	53
7	pyRevit Configuration	55

8	Usage Logger	57
9	pyRevit CLI Utility	59
10	Load Sequence, Step 1: Revit Addon	61
10.1	The Complex Relationship of a C# Addin and a Python Script	61
10.2	pyRevit loader script	61
10.3	pyRevitLoader Addin Source	62
11	Load Sequence, Step 2: IronPython Module	63
12	pyrevit	65
13	pyrevit.api	71
14	pyrevit.compat	73
15	pyrevit.coreutils	75
15.1	pyrevit.coreutils	75
15.2	pyrevit.coreutils.appdata	88
15.3	pyrevit.coreutils.charts	90
15.4	pyrevit.coreutils.colors	91
15.5	pyrevit.coreutils.configparser	102
15.6	pyrevit.coreutils.dotnetcompiler	103
15.7	pyrevit.coreutils.envvars	103
15.8	pyrevit.coreutils.git	104
15.9	pyrevit.coreutils.loadertypes	106
15.10	pyrevit.coreutils.logger	106
15.11	pyrevit.coreutils.mathnet	108
15.12	pyrevit.coreutils.moduleutils	108
15.13	pyrevit.coreutils.pyutils	108
15.14	pyrevit.coreutils.ribbon	110
16	pyrevit.forms	115
16.1	pyrevit.forms.toaster	134
16.2	pyrevit.forms.utils	135
17	pyrevit.framework	137
18	pyrevit.loader	139
18.1	pyrevit.loader	139
18.2	pyrevit.loader.asmmaker	139
18.3	pyrevit.loader.sessioninfo	139
18.4	pyrevit.loader.systemdiag	141
18.5	pyrevit.loader.sessionmgr	141
18.6	pyrevit.loader.uimaker	142
19	pyrevit.output	143
19.1	pyrevit.output	143
19.2	pyrevit.output.linkmaker	149
20	pyrevit.revit	151
20.1	pyrevit.revit.files	151
20.2	pyrevit.revit.geom	151
20.3	pyrevit.revit.serverutils	152
20.4	pyrevit.revit.units	153

21	pyrevit.script	155
22	pyrevit.userconfig	161
23	pyrevit.versionmgr	165
23.1	pyrevit.versionmgr.about	166
23.2	pyrevit.versionmgr.updater	166
23.3	pyrevit.versionmgr.upgrade	167
	Python Module Index	169
	Index	171

Note: This documentation is a work-in-progress. Thanks for your patience.

Getting Started

I suggest reading this section completely as it provides 99% of what you will need to know for developing scripts in pyRevit environment. Other sections dive deeper into pyRevit inner workings.

- *How To Use This Documents*
- *Create Your First Command*
- *Anatomy of a pyRevit Script*
- *Effective Output/Input*
- *Keyboard Shortcuts*
- *Extensions and Commmands*
- *pyRevit Configuration*
- *Usage Logger*
- *pyRevit CLI Utility*

CHAPTER 1

How To Use This Documents

CHAPTER 2

Create Your First Command

Anatomy of a pyRevit Script

pyRevit provides a few basic services to python scripts that use its engine. These functionalities are accessible through a few high level modules. This article is a quick look at these services and their associated python modules.

3.1 Script Metadata Variables

pyRevit looks for certain global scope variables in each scripts that provide metadata about the script and follow the `__<name>__` format.

3.1.1 `__title__`



Button Title: When using the bundle name as the button name in Revit UI is not desired, or you want to add a newline character to the command name to better display the button name inside Revit UI Panel, you can define the `__title__` variable in your script and set it to the desired button title.

```
__title__ = 'Sample\\nCommand'
```

3.1.2 `__doc__`

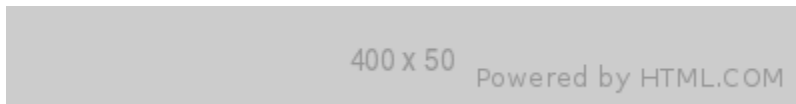


Button Tooltip: Tooltips are displayed similar to the other buttons in Revit interface. You can define the tooltip for a script using the docstring at the top of the script or by explicitly defining `__doc__` metadata variable.

```
# defining tooltip as the script docstring
"""You can place the docstring (tooltip) at the top of the script file.
This serves both as python docstring and also button tooltip in pyRevit.
You should use triple quotes for standard python docstrings."""
```

```
# defining tooltip by setting metadata variable
__doc__ = 'This is the text for the button tooltip associated with this_
↪script.'
```

3.1.3 `__author__`



Script Author: You can define the script author as shown below. This will show up on the button tooltip.

```
__author__ = 'Ehsan Iran-Nejad'
```

3.1.4 `__helpurl__`

F1 Shortcut Help Url: xx

3.1.5 `__min_revit_ver__`

Min Revit Version Required: xx

3.1.6 `__max_revit_ver__`

Max Revit Version Supported: xx

3.1.7 `__beta__`

Script In Beta: xx

3.1.8 `__context__`

Command Availability: Revit commands use standard `IExternalCommandAvailability` class to let Revit know if they are available in different contexts. For example, if a command needs to work on a set of elements, it can tell Revit to deactivate the button unless the user has selected one or more elements.

In pyRevit, command availability is set through the `__context__` variable. Currently, pyRevit support three types of command availability types.

```
# Tool activates when at least one element is selected
__context__ = 'Selection'

# Tools are active even when there are no documents available/open in Revit
__context__ = 'zerodoc'

# Tool activates when all selected elements are of the given category or categories
__context__ = '<Element Category>'
__context__ = ['<Element Category>', '<Element Category>']
```

If category name starts with **'OST_'** (e.g. 'OST_Doors') comparison will be made using `BuiltInCategory`. It allows to use your script in spite of Revit translation

List of available categories <https://www.revitapidocs.com/2020/ba1c5b30-242f-5fdc-8ea9-ec3b61e6e722.htm>

`<Element Category>` can be any of the standard Revit element categories. See *Appendix B: System Category Names* for a full list of system categories. You can use the `List` tool under `pyRevit > Spy` and list the standard categories.

Here are a few examples:

```
# Tool activates when all selected elements are of the given category

__context__ = 'Doors'
__context__ = 'Walls'
__context__ = 'Floors'
__context__ = 'OST_Doors'
__context__ = ['Space Tags', 'Spaces']
```

3.2 pyrevit.script Module

All pyRevit scripts should use the `pyrevit.script` module to access pyRevit functionality unless listed otherwise. pyRevit internals are subject to changes and accessing them directly is not suggested.

Here is a list of supported modules for pyRevit scripts. Examples of using the functionality in these modules are provided on this page.

`pyrevit.script`

This module provides access to output window (`pyrevit.output`), logging (`pyrevit.coreutils.logger`), temporary files (`pyrevit.coreutils.appdata`), and other misc features. See the module page for usage examples and full documentation of all available functions.

3.2.1 Logging

You can get the default logger for the script using `pyrevit.script.get_logger()`.

```
from pyrevit import script

logger = script.get_logger()

logger.info('Test Log Level :ok_hand_sign:')

logger.warning('Test Log Level')

logger.critical('Test Log Level')
```

Critical and warning messages are printed in color for clarity. Normally debug messages are not printed. you can hold CTRL and click on a command button to put that command in DEBUG mode and see all its debug messages

```
logger.debug('Yesss! Here is the debug message')
```

3.2.2 Controlling Output Window

Each script can control its own output window:

```
from pyrevit import script

output = script.get_output()

output.set_height(600)
output.get_title()
output.set_title('More control please!')
```

See *Effective Output/Input* for more info.

3.2.3 Script Config

Each script can save and load configuration pyRevit's user configuration file:

See *pyrevit.output* for more examples.

See *pyrevit.script.get_config()* and *pyrevit.script.save_config()* for the individual functions used here.

```
from pyrevit import script

config = script.get_config()

# set a new config parameter: firstparam
config.firstparam = True

# saving configurations
script.save_config()

# read the config parameter value
if config.firstparam:
    do_task_A()
```


3.2.4 Logging Results

pyRevit has a usage logging system that can record all tool usages to either a json file or to a web server. Scripts can return custom data to this logging system.

In example below, the script reports the amount of time it saved to the logging system:

```
from pyrevit import script

results = script.get_results()
results.timesaved = 10
```

3.2.5 Using Temporary Files

Scripts can create 3 different types of data files:

- **Universal files**

These files are not marked by host Revit version and could be shared between all Revit versions and instances. These data files are saved in pyRevit's appdata directory and are NOT cleaned up at Revit restarts.

See `pyrevit.script.get_universal_data_file()`

Note: Script should take care of cleaning up these data files.

```
# provide a unique file id and file extension
# Method will return full path of the data file
from pyrevit import script
script.get_universal_data_file(file_id, file_ext)
```

- **Data files**

These files are marked by host Revit version and could be shared between instances of host Revit version Data files are saved in pyRevit's appdata directory and are NOT cleaned up when Revit restarts.

See `pyrevit.script.get_data_file()`

Note: Script should take care of cleaning up these data files.

```
# provide a unique file id and file extension
# Method will return full path of the data file
from pyrevit import script
script.get_data_file(file_id, file_ext)
```

- **Instance Data files**

These files are marked by host Revit version and process Id and are only available to current Revit instance. This avoids any conflicts between similar scripts running under two or more Revit instances. Data files are saved in pyRevit's appdata directory (with extension `.tmp`) and ARE cleaned up when Revit restarts.

See `pyrevit.script.get_instance_data_file()`

```
# provide a unique file id and file extension
# Method will return full path of the data file
from pyrevit import script
script.get_instance_data_file(file_id)
```

- **Document Data files**

(Shared only between instances of host Revit version): These files are marked by host Revit version and name of Active Project and could be shared between instances of host Revit version. Data files are saved in pyRevit's appdata directory and are NOT cleaned up when Revit restarts.

See `pyrevit.script.get_document_data_file()`

Note: Script should take care of cleaning up these data files.

```
# provide a unique file id and file extension
# Method will return full path of the data file
from pyrevit import script
script.get_document_data_file(file_id, file_ext)

# You can also pass a document object to get a data file for that
# document (use document name in file naming)
script.get_document_data_file(file_id, file_ext, doc)
```

3.3 Appendix A: Builtin Parameters Provided by pyRevit Engine

Variables listed below are provided for every script in pyRevit.

Note: It's strongly advised not to read or write values from these variables unless necessary. The `pyrevit` module provides wrappers around these variables that are safe to use.

```
# Revit UIApplication is accessible through:
__revit__

# Command data provided to this command by Revit is accessible through:
__commandData__

# selection of elements provided to this command by Revit
__elements__

# pyRevit engine manager that is managing this engine
__ipyenginemanager__

# This variable is True if command is being run in a cached engine
__cachedengine__

# pyRevit external command object wrapping the command being run
__externalcommand__

# information about the pyrevit command being run
__commandpath__          # main script path
__alternatecommandpath__ # alternate script path
```

(continues on next page)

(continued from previous page)

```

__commandname__          # command name
__commandbundle__       # command bundle name
__commandextension__    # command extension name
__commanduniqueid__     # command unique id

# This variable is True if user CTRL-Clicks the button
__forceddebugmode__

# This variable is True if user SHIFT-Clicks the button
__shiftclick__

# results dictionary
__result__

```

3.4 Appendix B: System Category Names

```

Adaptive Points
Air Terminal Tags
Air Terminals
Analysis Display Style
Analysis Results
Analytical Beam Tags
Analytical Beams
Analytical Brace Tags
Analytical Braces
Analytical Column Tags
Analytical Columns
Analytical Floor Tags
Analytical Floors
Analytical Foundation Slabs
Analytical Isolated Foundation Tags
Analytical Isolated Foundations
Analytical Link Tags
Analytical Links
Analytical Node Tags
Analytical Nodes
Analytical Slab Foundation Tags
Analytical Spaces
Analytical Surfaces
Analytical Wall Foundation Tags
Analytical Wall Foundations
Analytical Wall Tags
Analytical Walls
Annotation Crop Boundary
Area Load Tags
Area Tags
Areas
Assemblies
Assembly Tags
Boundary Conditions
Brace in Plan View Symbols
Cable Tray Fitting Tags
Cable Tray Fittings
Cable Tray Runs

```

(continues on next page)

(continued from previous page)

Cable Tray Tags
Cable Trays
Callout Boundary
Callout Heads
Callouts
Cameras
Casework
Casework Tags
Ceiling Tags
Ceilings
Color Fill Legends
Columns
Communication Device Tags
Communication Devices
Conduit Fitting Tags
Conduit Fittings
Conduit Runs
Conduit Tags
Conduits
Connection Symbols
Contour Labels
Crop Boundaries
Curtain Grids
Curtain Panel Tags
Curtain Panels
Curtain System Tags
Curtain Systems
Curtain Wall Mullions
Data Device Tags
Data Devices
Detail Item Tags
Detail Items
Dimensions
Displacement Path
Door Tags
Doors
Duct Accessories
Duct Accessory Tags
Duct Color Fill
Duct Color Fill Legends
Duct Fitting Tags
Duct Fittings
Duct Insulation Tags
Duct Insulations
Duct Lining Tags
Duct Linings
Duct Placeholders
Duct Systems
Duct Tags
Ducts
Electrical Circuits
Electrical Equipment
Electrical Equipment Tags
Electrical Fixture Tags
Electrical Fixtures
Electrical Spare/Space Circuits
Elevation Marks

(continues on next page)

(continued from previous page)

Elevations
Entourage
Filled region
Fire Alarm Device Tags
Fire Alarm Devices
Flex Duct Tags
Flex Ducts
Flex Pipe Tags
Flex Pipes
Floor Tags
Floors
Foundation Span Direction Symbol
Furniture
Furniture System Tags
Furniture Systems
Furniture Tags
Generic Annotations
Generic Model Tags
Generic Models
Grid Heads
Grids
Guide Grid
HVAC Zones
Imports in Families
Internal Area Load Tags
Internal Line Load Tags
Internal Point Load Tags
Keynote Tags
Level Heads
Levels
Lighting Device Tags
Lighting Devices
Lighting Fixture Tags
Lighting Fixtures
Line Load Tags
Lines
Masking Region
Mass
Mass Floor Tags
Mass Tags
Matchline
Material Tags
Materials
Mechanical Equipment
Mechanical Equipment Tags
MEP Fabrication Containment
MEP Fabrication Containment Tags
MEP Fabrication Ductwork
MEP Fabrication Ductwork Tags
MEP Fabrication Hanger Tags
MEP Fabrication Hangers
MEP Fabrication Pipework
MEP Fabrication Pipework Tags
Multi-Category Tags
Multi-Rebar Annotations
Nurse Call Device Tags
Nurse Call Devices

(continues on next page)

(continued from previous page)

Panel Schedule Graphics
Parking
Parking Tags
Part Tags
Parts
Pipe Accessories
Pipe Accessory Tags
Pipe Color Fill
Pipe Color Fill Legends
Pipe Fitting Tags
Pipe Fittings
Pipe Insulation Tags
Pipe Insulations
Pipe Placeholders
Pipe Segments
Pipe Tags
Pipes
Piping Systems
Plan Region
Planting
Planting Tags
Plumbing Fixture Tags
Plumbing Fixtures
Point Clouds
Point Load Tags
Project Information
Property Line Segment Tags
Property Tags
Railing Tags
Railings
Ramps
Raster Images
Rebar Cover References
Rebar Set Toggle
Rebar Shape
Reference Lines
Reference Planes
Reference Points
Render Regions
Revision Cloud Tags
Revision Clouds
Roads
Roof Tags
Roofs
Room Tags
Rooms
Routing Preferences
Schedule Graphics
Scope Boxes
Section Boxes
Section Line
Section Marks
Sections
Security Device Tags
Security Devices
Shaft Openings
Sheets

(continues on next page)

(continued from previous page)

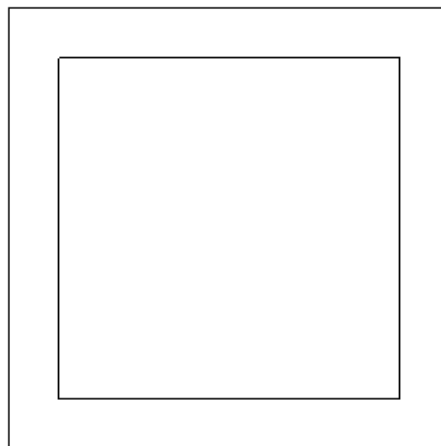
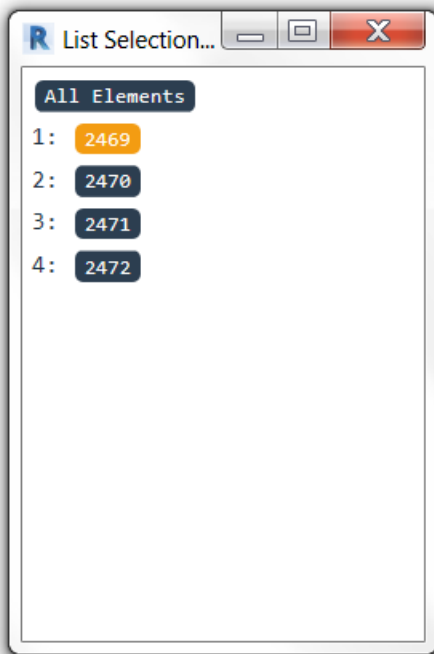
Site
Site Tags
Space Tags
Spaces
Span Direction Symbol
Specialty Equipment
Specialty Equipment Tags
Spot Coordinates
Spot Elevation Symbols
Spot Elevations
Spot Slopes
Sprinkler Tags
Sprinklers
Stair Landing Tags
Stair Paths
Stair Run Tags
Stair Support Tags
Stair Tags
Stair Tread/Riser Numbers
Stairs
Structural Annotations
Structural Area Reinforcement
Structural Area Reinforcement Symbols
Structural Area Reinforcement Tags
Structural Beam System Tags
Structural Beam Systems
Structural Column Tags
Structural Columns
Structural Connection Tags
Structural Connections
Structural Fabric Areas
Structural Fabric Reinforcement
Structural Fabric Reinforcement Symbols
Structural Fabric Reinforcement Tags
Structural Foundation Tags
Structural Foundations
Structural Framing
Structural Framing Tags
Structural Internal Loads
Structural Load Cases
Structural Loads
Structural Path Reinforcement
Structural Path Reinforcement Symbols
Structural Path Reinforcement Tags
Structural Rebar
Structural Rebar Coupler Tags
Structural Rebar Couplers
Structural Rebar Tags
Structural Stiffener Tags
Structural Stiffeners
Structural Truss Tags
Structural Trusses
Switch System
Telephone Device Tags
Telephone Devices
Text Notes
Title Blocks

(continues on next page)

(continued from previous page)

Topography
View Reference
View Titles
Viewports
Views
Wall Tags
Walls
Window Tags
Windows
Wire Tags
Wires
Zone Tags

4.1 Clickable Element Links



work in progress

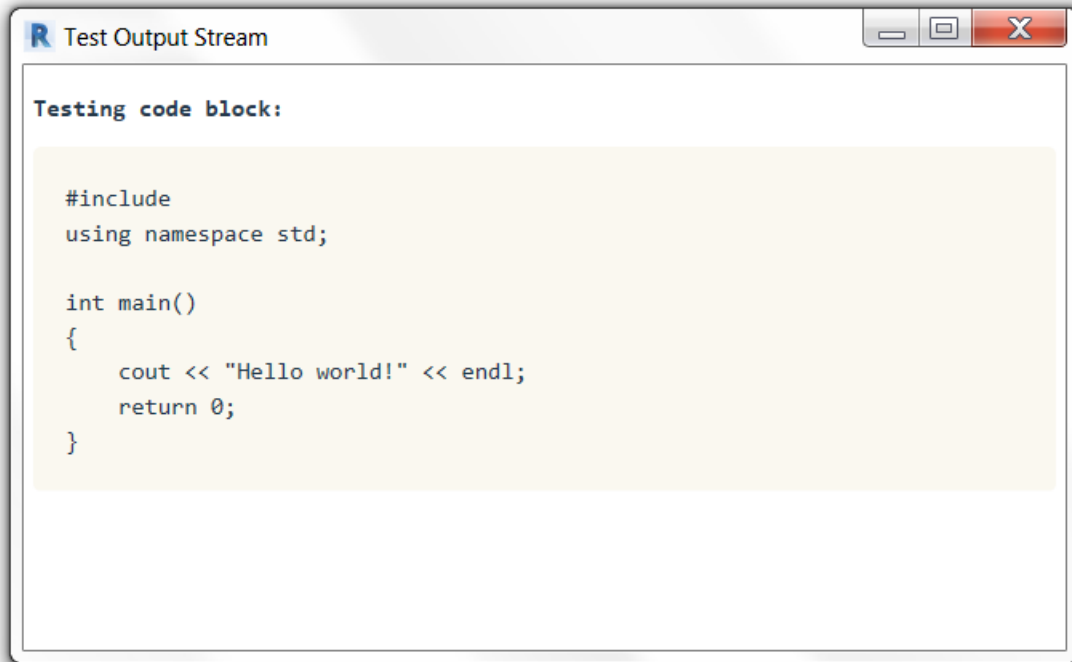
4.2 Tables

The screenshot shows a window titled "Test Output Stream" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, the text "Test Table" is displayed above a table. The table has three columns labeled "Col 1", "Col 2", and "Col 3". The first row is highlighted in dark blue. The data in the table is as follows:

Col 1	Col 2	Col 3
1	2	3%
1	2	3%
1	2	3%

work in progress

4.3 Code Output

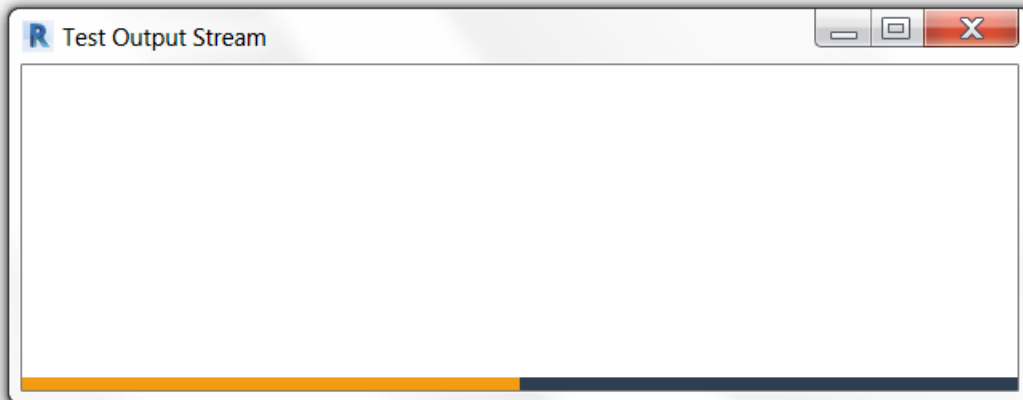


A screenshot of a window titled "Test Output Stream" with a blue 'R' icon in the title bar. The window contains a code block with the following C++ code:

```
Testing code block:  
  
#include  
using namespace std;  
  
int main()  
{  
    cout << "Hello world!" << endl;  
    return 0;  
}
```

work in progress

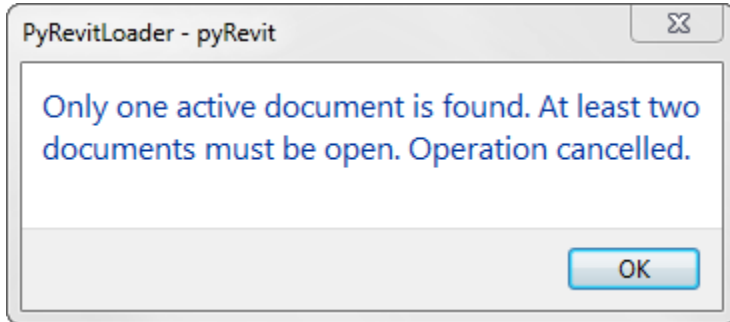
4.4 Progress bars



work in progress

4.5 Standard Prompts

4.5.1 Alerts



`pyrevit.forms.alert` (*msg*, *title=None*, *sub_msg=None*, *expanded=None*, *footer=""*, *ok=True*, *cancel=False*, *yes=False*, *no=False*, *retry=False*, *warn_icon=True*, *options=None*, *exitscript=False*)

Show a task dialog with given message.

Parameters

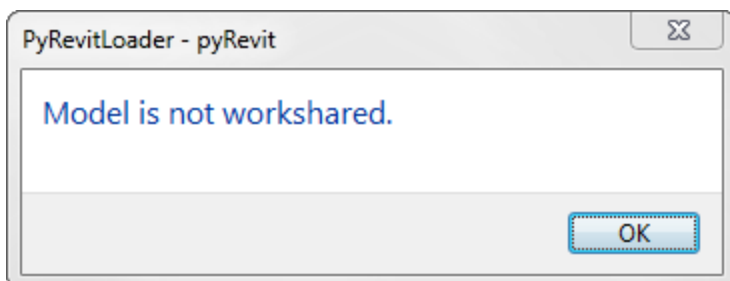
- **msg** (*str*) – message to be displayed
- **title** (*str*, *optional*) – task dialog title
- **ok** (*bool*, *optional*) – show OK button, defaults to True
- **cancel** (*bool*, *optional*) – show Cancel button, defaults to False
- **yes** (*bool*, *optional*) – show Yes button, defaults to False
- **no** (*bool*, *optional*) – show NO button, defaults to False
- **retry** (*bool*, *optional*) – show Retry button, defaults to False
- **options** (*list[str]*, *optional*) – list of command link titles in order
- **exitscript** (*bool*, *optional*) – exit if cancel or no, defaults to False

Returns True if okay, yes, or retry, otherwise False

Return type bool

Example

```
>>> from pyrevit import forms
>>> forms.alert('Are you sure?',
...           ok=False, yes=True, no=True, exitscript=True)
```



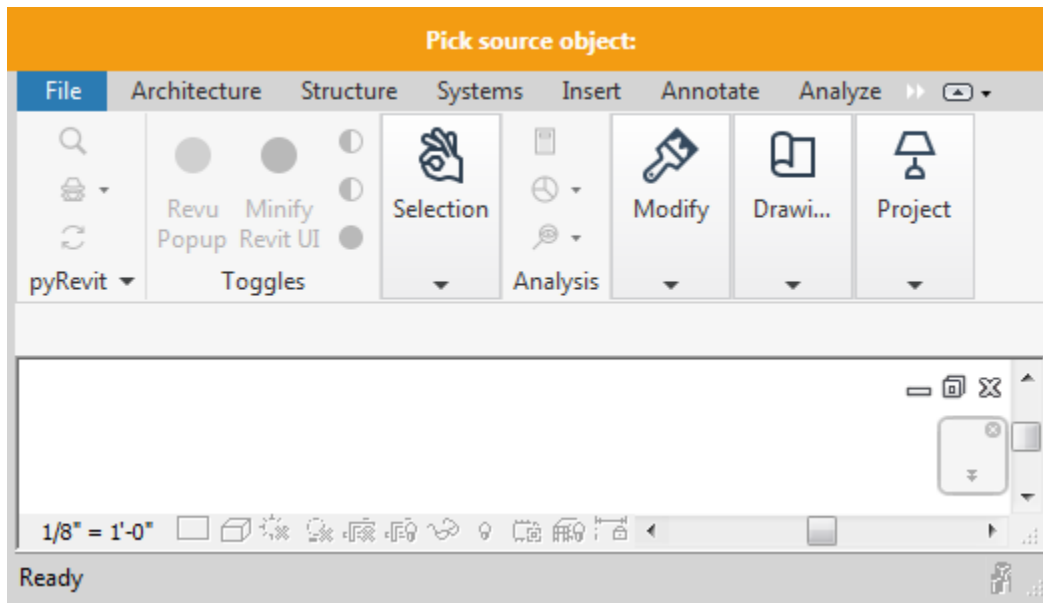
`pyrevit.forms.check_workshared` (*doc=None, message='Model is not workshared.'*)
Verify if model is workshared and notify user if not.

Parameters

- **doc** (*DB.Document*) – target document, current if not provided
- **message** (*str*) – prompt message if returning False

Returns True if doc is workshared

Return type bool



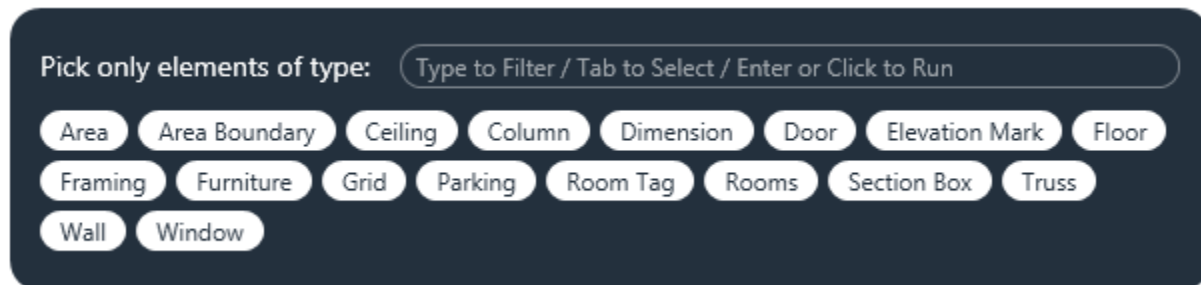
class `pyrevit.forms.WarningBar` (*height=32, **kwargs*)
Show warning bar at the top of Revit window.

Parameters **title** (*string*) – warning bar text

Example

```
>>> with WarningBar(title='my warning'):  
...     # do stuff
```

4.5.2 Command Options



class `pyrevit.forms.CommandSwitchWindow` (*context, title, width, height, **kwargs*)
Standard form to select from a list of command options.

Parameters

- **context** (*list[str]*) – list of command options to choose from
- **switches** (*list[str]*) – list of on/off switches
- **message** (*str*) – window title message
- **config** (*dict*) – dictionary of config dicts for options or switches
- **recognize_access_key** (*bool*) – recognize ‘_’ as mark of access key

Returns name of selected option

Return type str

Returns if `switches` option is used, returns a tuple of selection option name and dict of switches

Return type tuple(str, dict)

Example

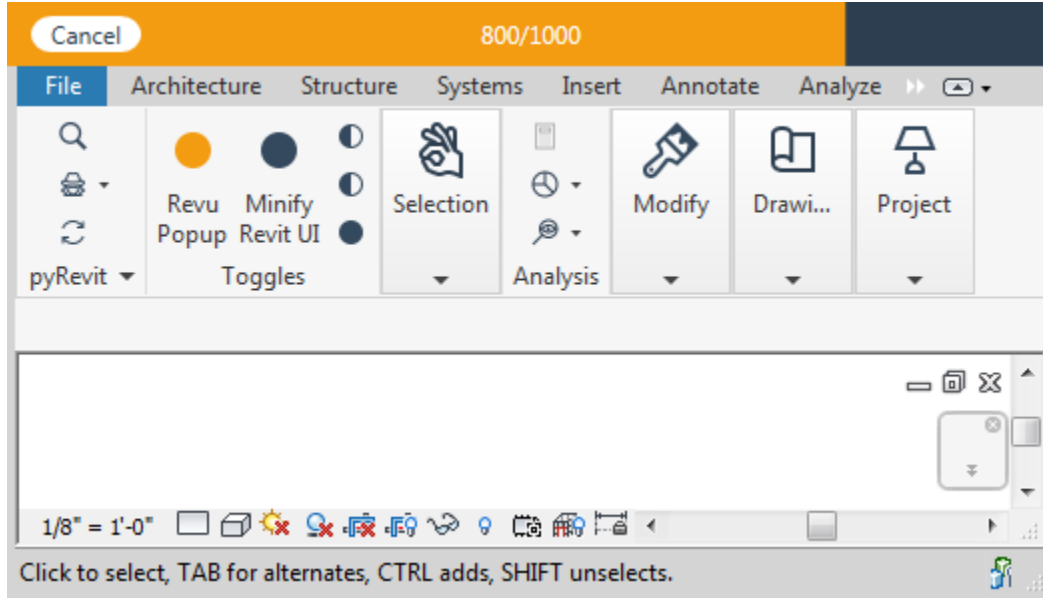
This is an example with series of command options:

```
>>> from pyrevit import forms
>>> ops = ['option1', 'option2', 'option3', 'option4']
>>> forms.CommandSwitchWindow.show(ops, message='Select Option')
'option2'
```

A more advanced example of combining command options, on/off switches, and option or switch configuration options:

```
>>> from pyrevit import forms
>>> ops = ['option1', 'option2', 'option3', 'option4']
>>> switches = ['switch1', 'switch2']
>>> cfgs = {'option1': {'background': '0xFF55FF'}}
>>> rops, rswitches = forms.CommandSwitchWindow.show(
...     ops,
...     switches=switches
...     message='Select Option',
...     config=cfgs,
...     recognize_access_key=False
... )
>>> rops
'option2'
>>> rswitches
{'switch1': False, 'switch2': True}
```

4.5.3 Showing Progress



class `pyrevit.forms.ProgressBar` (*height=32, **kwargs*)
 Show progress bar at the top of Revit window.

Parameters

- **title** (*string*) – progress bar text, defaults to 0/100 progress format
- **indeterminate** (*bool*) – create indeterminate progress bar
- **cancellable** (*bool*) – add cancel button to progress bar
- **step** (*int*) – update progress intervals

Example

```
>>> from pyrevit import forms
>>> count = 1
>>> with forms.ProgressBar(title='my command progress message') as pb:
...     # do stuff
...     pb.update_progress(count, 100)
...     count += 1
```

Progress bar title could also be customized to show the current and total progress values. In example below, the progress bar message will be in format “0 of 100”

```
>>> with forms.ProgressBar(title='{value} of {max_value}') as pb:
```

By default progress bar updates the progress every time the `.update_progress` method is called. For operations with a large number of max steps, the gui update process time will have a significant effect on the overall execution time of the command. In these cases, set the value of `step` argument to something larger than 1. In example below, the progress bar updates once per every 10 units of progress.

```
>>> with forms.ProgressBar(title='message', steps=10):
```

Progress bar could also be set to indeterminate for operations of unknown length. In this case, the progress bar will show an infinitely running ribbon:

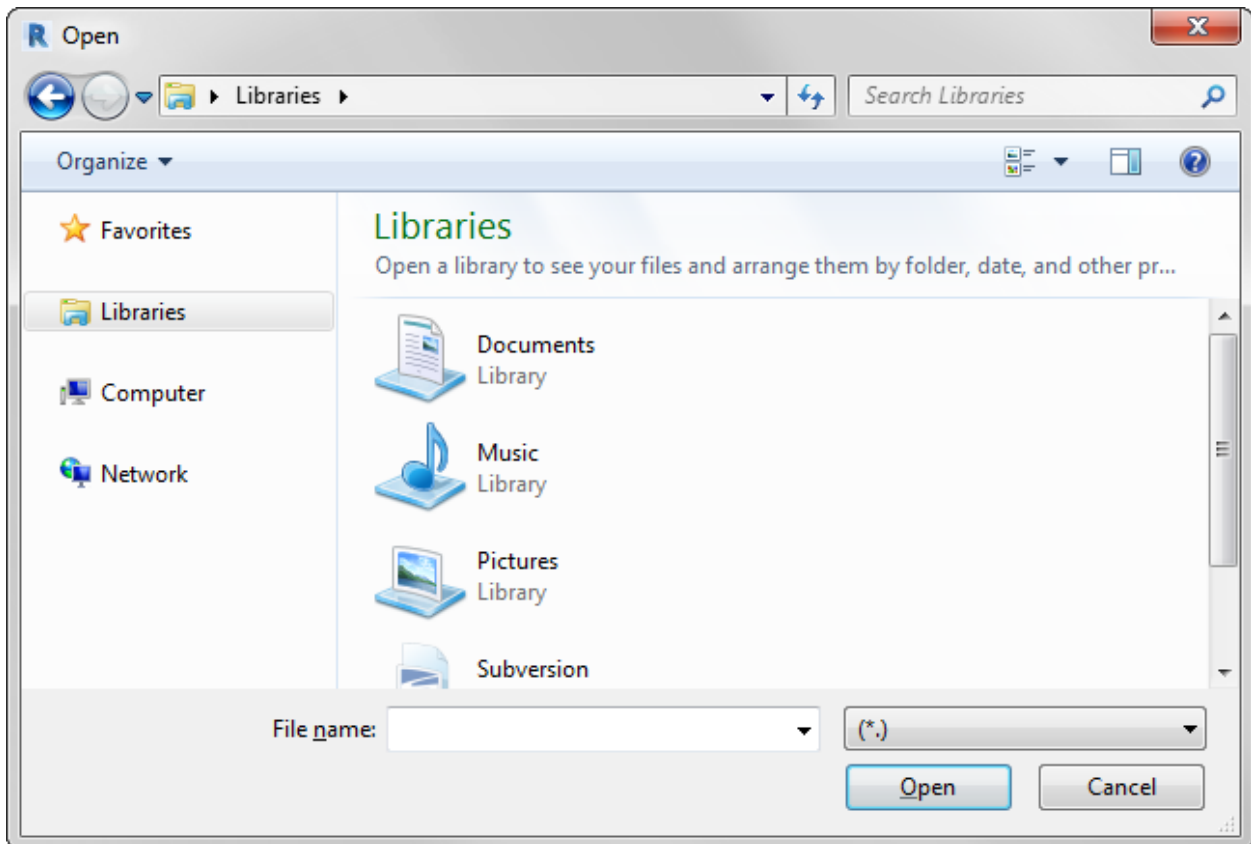
```
>>> with forms.ProgressBar(title='message', indeterminate=True):
```

if cancellable is set on the object, a cancel button will show on the progress bar and .cancelled attribute will be set on the ProgressBar instance if users clicks on cancel button:

```
>>> with forms.ProgressBar(title='message',
...                       cancellable=True) as pb:
...     # do stuff
...     if pb.cancelled:
...         # wrap up and cancel operation
```

4.6 Standard Dialogs

4.6.1 Pick File



```
pyrevit.forms.pick_file(file_ext="", files_filter="", init_dir="", restore_dir=True, multi_file=False,
                        unc_paths=False)
```

Pick file dialog to select a destination file.

Parameters

- **file_ext** (*str*) – file extension
- **files_filter** (*str*) – file filter

- **init_dir** (*str*) – initial directory
- **restore_dir** (*bool*) – restore last directory
- **multi_file** (*bool*) – allow select multiple files
- **unc_paths** (*bool*) – return unc paths

Returns file path or list of file paths if multi_file=True

Return type str or list[str]

Example

```
>>> from pyrevit import forms
>>> forms.pick_file(file_ext='csv')
... r'C:\output\somefile.csv'
```

```
>>> forms.pick_file(file_ext='csv', multi_file=True)
... [r'C:\output\somefile1.csv', r'C:\output\somefile2.csv']
```

```
>>> forms.pick_file(files_filter='All Files (*.*)|*.*|'
                    'Excel Workbook (*.xlsx)|*.xlsx|'
                    'Excel 97-2003 Workbook|*.xls',
                    multi_file=True)
... [r'C:\output\somefile1.xlsx', r'C:\output\somefile2.xls']
```

`pyrevit.forms.save_file(file_ext="", files_filter="", init_dir="", default_name="", restore_dir=True, unc_paths=False)`

Save file dialog to select a destination file for data.

Parameters

- **file_ext** (*str*) – file extension
- **files_filter** (*str*) – file filter
- **init_dir** (*str*) – initial directory
- **default_name** (*str*) – default file name
- **restore_dir** (*bool*) – restore last directory
- **unc_paths** (*bool*) – return unc paths

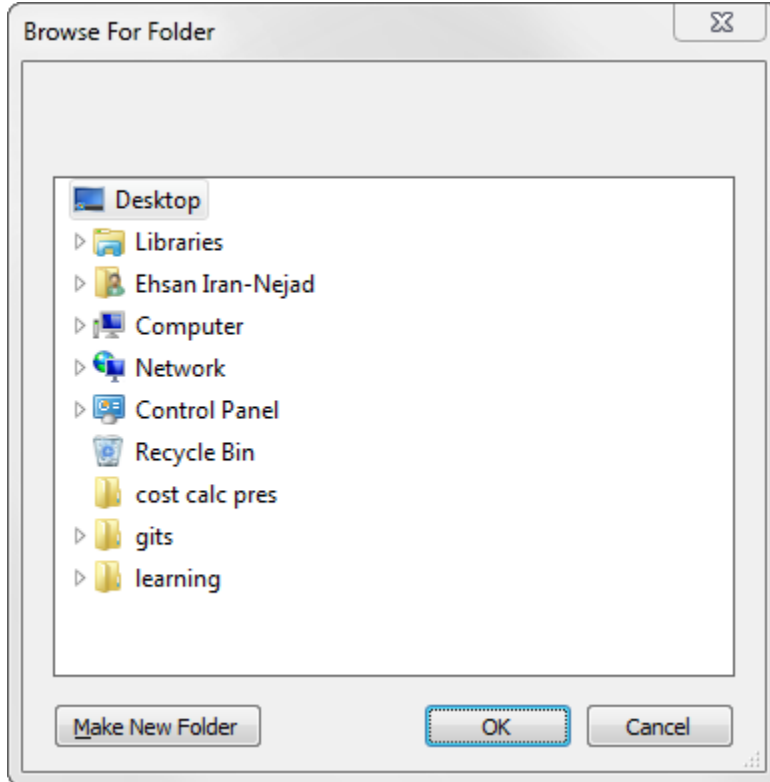
Returns file path

Return type str

Example

```
>>> from pyrevit import forms
>>> forms.save_file(file_ext='csv')
... r'C:\output\somefile.csv'
```

4.6.2 Pick Folder



`pyrevit.forms.pick_folder(title=None)`

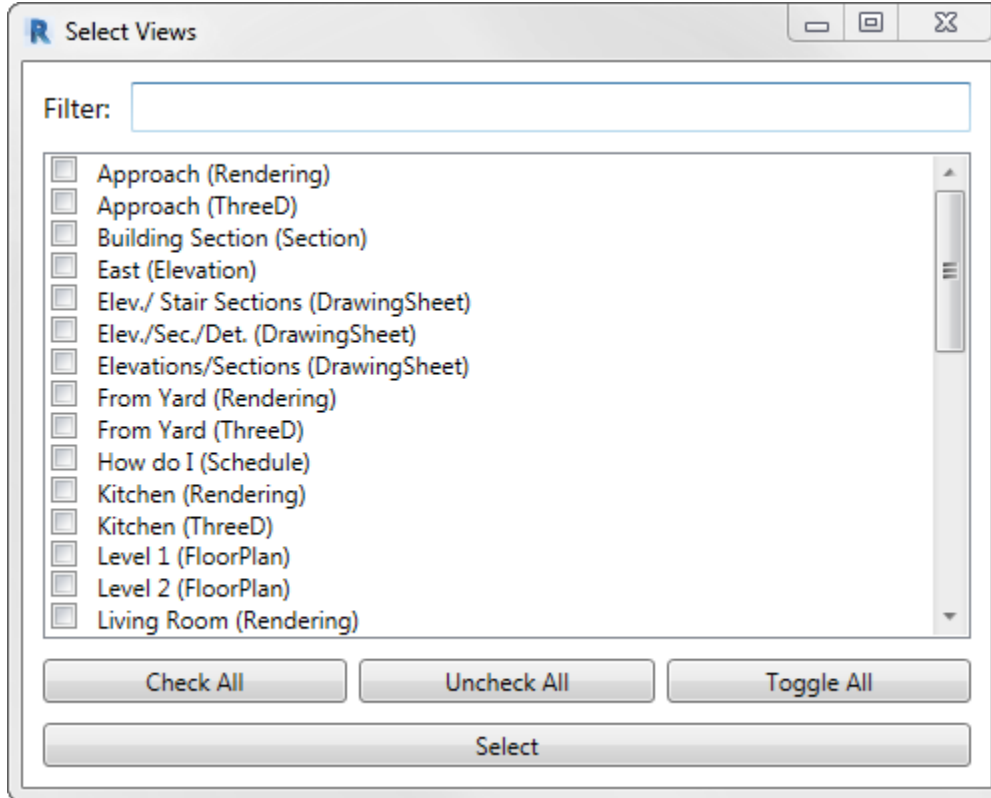
Show standard windows pick folder dialog.

Parameters `title` (*str, optional*) – title for the window

Returns folder path

Return type `str`

4.6.3 Select Views



```
pyrevit.forms.select_views(title='Select Views', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None)
```

Standard form for selecting views.

Parameters

- **title** (*str*, *optional*) – list window title
- **button_name** (*str*, *optional*) – list window button caption
- **width** (*int*, *optional*) – width of list window
- **multiselect** (*bool*, *optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document*, *optional*) – source document for views; defaults to active document

Returns list of selected views

Return type list[DB.View]

Example

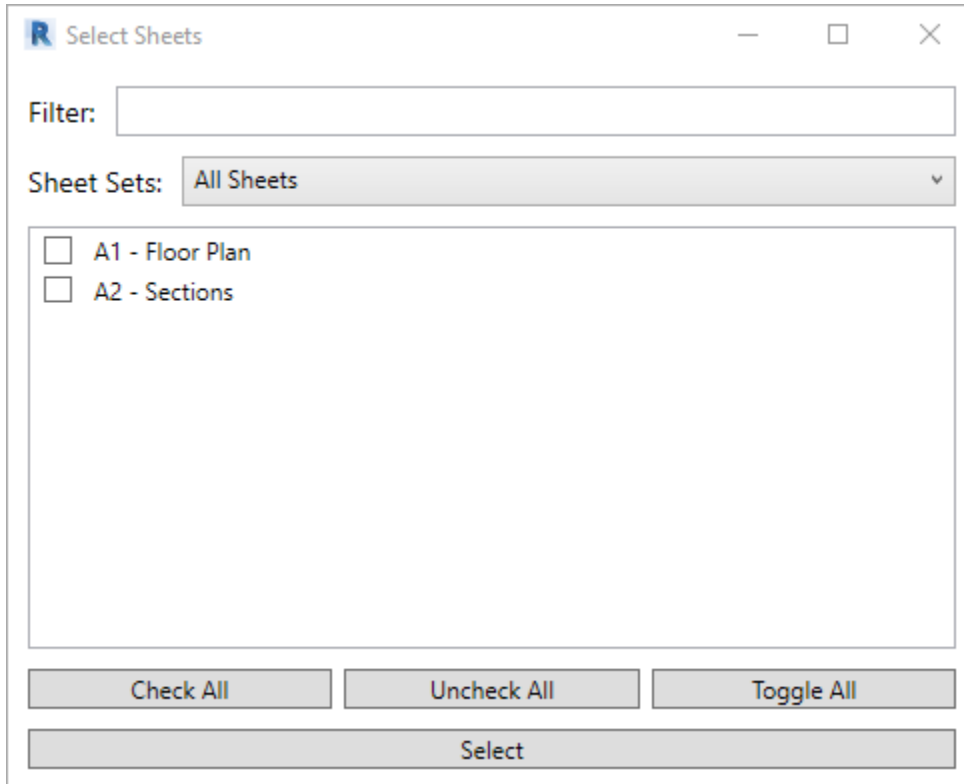
```
>>> from pyrevit import forms
>>> forms.select_views()
```

(continues on next page)

(continued from previous page)

```
... [<Autodesk.Revit.DB.View object>,
... <Autodesk.Revit.DB.View object>]
```

4.6.4 Select Sheets



`pyrevit.forms.select_sheets` (*title='Select Sheets', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None*)

Standard form for selecting sheets.

Sheets are grouped into sheet sets and sheet set can be selected from a drop down box at the top of window.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for sheets; defaults to active document

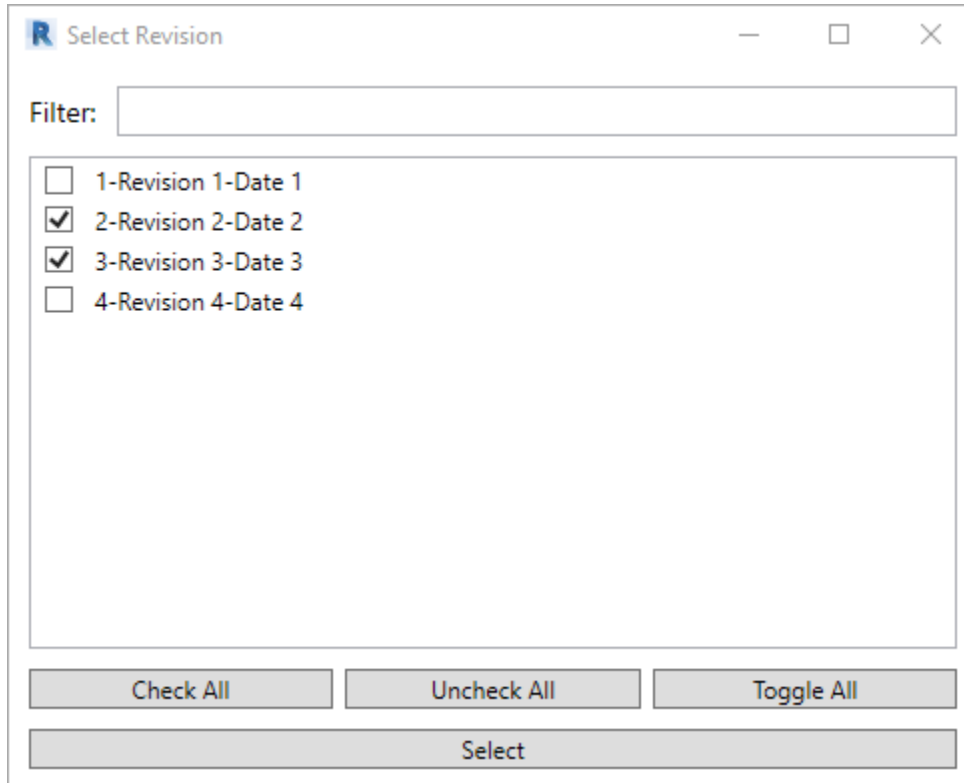
Returns list of selected sheets

Return type list[DB.ViewSheet]

Example

```
>>> from pyrevit import forms
>>> forms.select_sheets()
... [<Autodesk.Revit.DB.ViewSheet object>,
...  <Autodesk.Revit.DB.ViewSheet object>]
```

4.6.5 Select Revisions



`pyrevit.forms.select_revisions` (*title='Select Revision', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None*)

Standard form for selecting revisions.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for revisions; defaults to active document

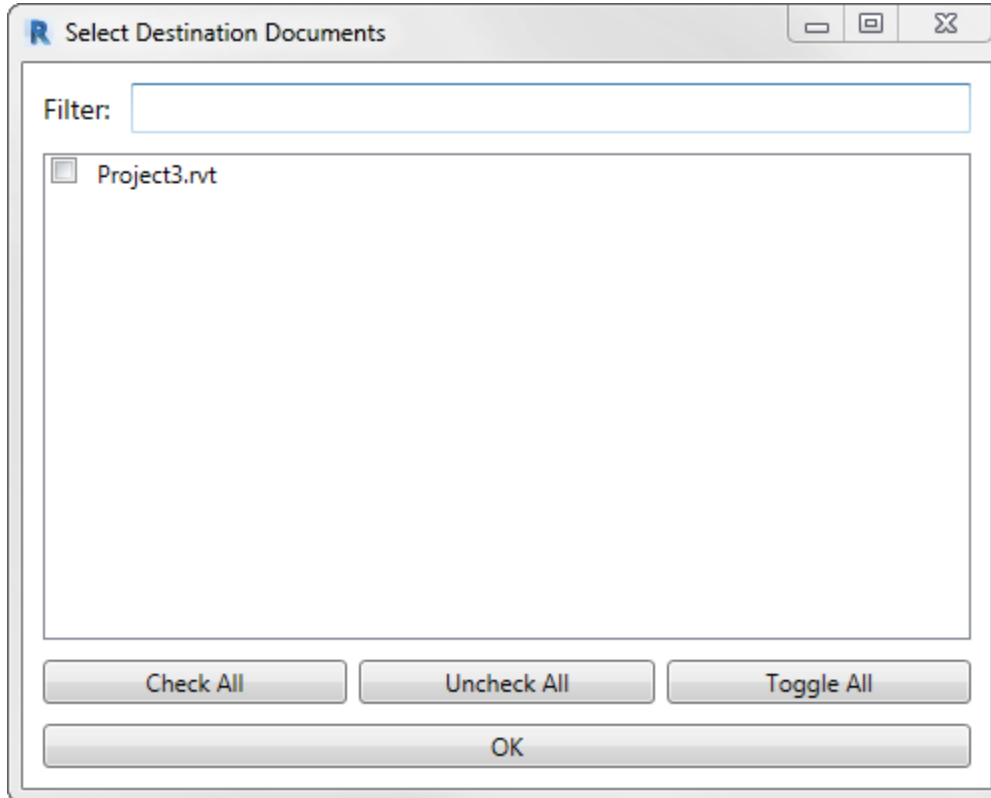
Returns list of selected revisions

Return type list[DB.Revision]

Example

```
>>> from pyrevit import forms
>>> forms.select_revisions()
... [<Autodesk.Revit.DB.Revision object>,
...  <Autodesk.Revit.DB.Revision object>]
```

4.6.6 Select From Open Documents



```
pyrevit.forms.select_open_docs (title='Select Open Documents', button_name='OK', width=500,
                                multiple=True, filterfunc=None)
```

Standard form for selecting open documents.

Parameters

- **title** (*str*, *optional*) – list window title
- **button_name** (*str*, *optional*) – list window button caption
- **width** (*int*, *optional*) – width of list window
- **multiselect** (*bool*, *optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.

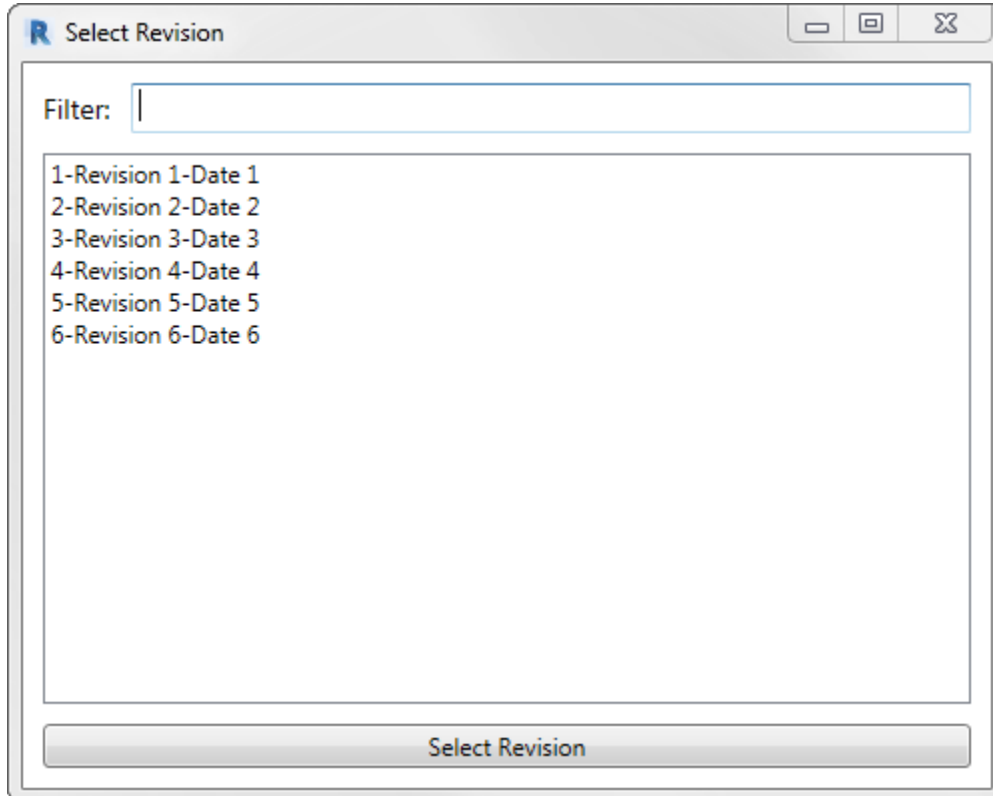
Returns list of selected documents

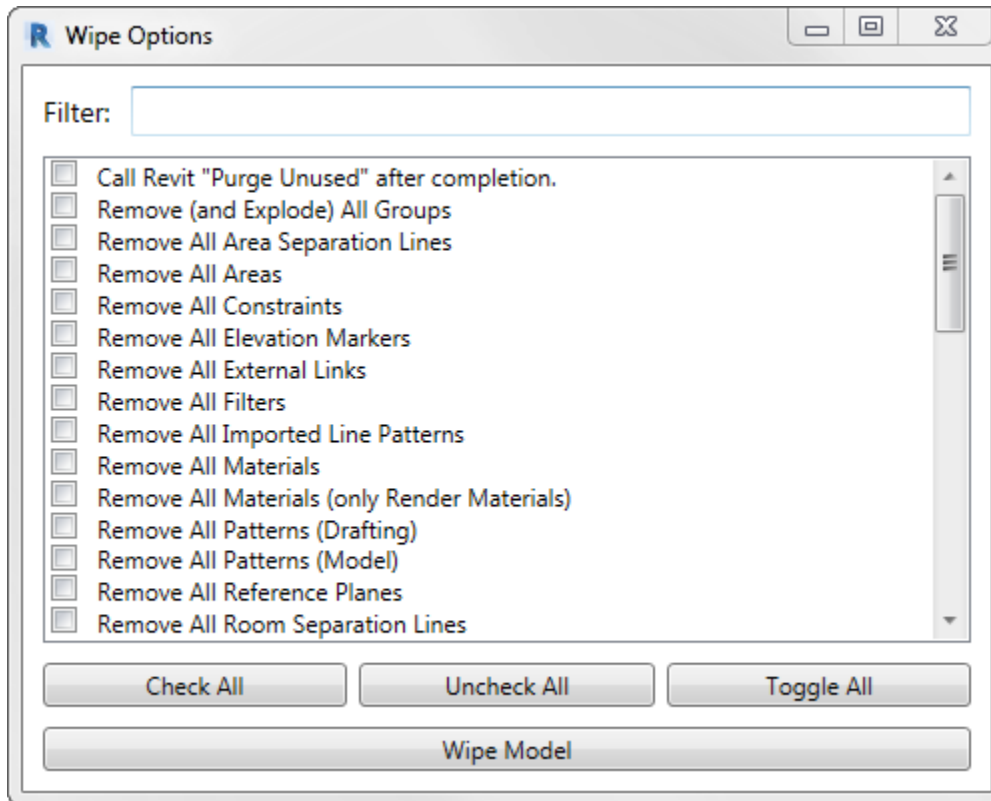
Return type list[DB.Document]

Example

```
>>> from pyrevit import forms
>>> forms.select_open_docs()
... [<Autodesk.Revit.DB.Document object>,
...  <Autodesk.Revit.DB.Document object>]
```

4.6.7 Select From List (Single, Multiple)





class `pyrevit.forms.SelectFromList` (*context, title, width, height, **kwargs*)
 Standard form to select from a list of items.

Any object can be passed in a list to the `context` argument. This class wraps the objects passed to context, in `TemplateListItem`. This class provides the necessary mechanism to make this form work both for selecting items from a list, and from a list of checkboxes. See the list of arguments below for additional options and features.

Parameters

- **context** (*list[str]* or *dict[list[str]]*) – list of items to be selected from OR dict of list of items to be selected from. use dict when input items need to be grouped e.g. List of sheets grouped by sheet set.
- **title** (*str, optional*) – window title. see super class for defaults.
- **width** (*int, optional*) – window width. see super class for defaults.
- **height** (*int, optional*) – window height. see super class for defaults.
- **button_name** (*str, optional*) – name of select button. defaults to ‘Select’
- **name_attr** (*str, optional*) – object attribute that should be read as item name.
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to False
- **return_all** (*bool, optional*) – return all items. This is handy when some input items have states and the script needs to check the state changes on all items. This options works in multiselect mode only. defaults to False
- **filterfunc** (*function*) – filter function to be applied to context items.
- **group_selector_title** (*str*) – title for list group selector. defaults to ‘List Group’

- `default_group` (*str*) – name of default group to be selected

Example

```
>>> from pyrevit import forms
>>> items = ['item1', 'item2', 'item3']
>>> forms.SelectFromList.show(items, button_name='Select Item')
>>> ['item1']
```

```
>>> from pyrevit import forms
>>> ops = [viewsheet1, viewsheet2, viewsheet3]
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=False,
...                               name_attr='Name',
...                               button_name='Select Sheet')
```

```
>>> from pyrevit import forms
>>> ops = {'Sheet Set A': [viewsheet1, viewsheet2, viewsheet3],
...       'Sheet Set B': [viewsheet4, viewsheet5, viewsheet6]}
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=True,
...                               name_attr='Name',
...                               group_selector_title='Sheet Sets',
...                               button_name='Select Sheets')
```

This module also provides a wrapper base class `TemplateListItem` for when the checkbox option is wrapping another element, e.g. a Revit ViewSheet. Derive from this base class and define the name property to customize how the checkbox is named on the dialog.

```
>>> from pyrevit import forms
>>> class MyOption(forms.TemplateListItem)
...     @property
...     def name(self):
...         return '{} - {}'.format(self.item.SheetNumber,
...                                 self.item.SheetNumber)
>>> ops = [MyOption('op1'), MyOption('op2', True), MyOption('op3')]
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=True,
...                               button_name='Select Item')
>>> [bool(x) for x in res] # or [x.state for x in res]
[True, False, True]
```

`class pyrevit.forms.TemplateListItem` (*orig_item*, *checkable=True*, *name_attr=None*)
Base class for checkbox option wrapping another object.

4.7 Base Forms

4.7.1 Input Dialog for pyRevit Search



`pyrevit.forms.SearchPrompt` (*search_db*, *width*, *height*, ***kwargs*)
Standard prompt for pyRevit search.

Parameters

- **search_db** (*list*) – list of possible search targets
- **search_tip** (*str*) – text to show in grayscale when search box is empty
- **switches** (*str*) – list of switches
- **width** (*int*) – width of search prompt window
- **height** (*int*) – height of search prompt window

Returns matched strings, and dict of switches if provided str: matched string if switches are not provided.

Return type str, dict

Example

```
>>> from pyrevit import forms
>>> # assume search input of '/switch1 target1'
>>> matched_str, args, switches = forms.SearchPrompt.show(
...     search_db=['target1', 'target2', 'target3', 'target4'],
...     switches=['/switch1', '/switch2'],
...     search_tip='pyRevit Search'
... )
... matched_str
'target1'
... args
['--help', '--branch', 'branchname']
... switches
{'/switch1': True, '/switch2': False}
```

4.7.2 Generic Forms

class `pyrevit.forms.TemplatePromptBar` (*height=32*, ***kwargs*)
Template context-manager class for creating prompt bars.

Prompt bars are show at the top of the active Revit window and are designed for better prompt visibility.

Parameters

- **height** (*int*) – window height
- ****kwargs** – other arguments to be passed to `_setup()`

class `pyrevit.forms.TemplateUserInputWindow` (*context*, *title*, *width*, *height*, ***kwargs*)
Base class for pyRevit user input standard forms.

Parameters

- **context** (*any*) – window context element(s)
- **title** (*str*) – window title
- **width** (*int*) – window width
- **height** (*int*) – window height

- ****kwargs** – other arguments to be passed to `_setup()`

class `pyrevit.forms.WPFWindow`(*xaml_source*, *literal_string=False*, *handle_esc=True*)
WPF Window base class for all pyRevit forms.

Parameters

- **xaml_source** (*str*) – xaml source filepath or xaml content
- **literal_string** (*bool*) – xaml_source contains xaml content, not filepath
- **handle_esc** (*bool*) – handle Escape button and close the window

Example

```
>>> from pyrevit import forms
>>> layout = '<Window ' \
>>>         'xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" ' \
↵\
>>>         'xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" ' \
>>>         'ShowInTaskbar="False" ResizeMode="NoResize" ' \
>>>         'WindowStartupLocation="CenterScreen" ' \
>>>         'HorizontalContentAlignment="Center">' \
>>>         '</Window>'
>>> w = forms.WPFWindow(layout, literal_string=True)
>>> w.show()
```

4.8 Graphs

Step 1: Create a chart object for the chart type that you want. We'll add data to this later...

```
from pyrevit import script

output = script.get_output()

# Line chart
chart = output.make_line_chart()
# Bar chart
chart = output.make_bar_chart()
# Bubble chart
chart = output.make_bubble_chart()
# Radar chart
chart = output.make_radar_chart()
# Polar chart
chart = output.make_polar_chart()
# Pie chart
chart = output.make_pie_chart()
# Doughnut chart
chart = output.make_doughnut_chart()
```

Step 1-a: Optional: Setup the chart title, and other options. the full list of options for every chart is available on [Charts.js Documentation](#) page. Some of the properties have their own sub-properties, for example the `title` option for the charts has multiple sub-properties as shown below. The value for these type of properties should be a dictionary of the sub-properties you'd like to set. All this is explained clearly in the [Charts.js Documentation](#)

```
chart.set_style('height:150px')

chart.options.title = {'display': True,
                       'text': 'Chart Title',
                       'fontSize': 18,
                       'fontColor': '#000',
                       'fontStyle': 'bold'}
```

Step 2: Now let's add data to the chart. Every chart object has a data property `chart.data` that we can interact with to add datasets to the chart. Different types of charts need different types of data sets in terms of how data is organized, so the chart can present multiple data sets correctly. I'm providing two examples here, one for a simple line chart (showing 3 different data sets) and another for a radial chart (also showing 3 different data sets within the same chart). They're all very similar to each other though.

```
# setting the charts x line data labels
chart.data.labels = ['Monday', 'Tuesday',
                    'Wednesday', 'Thursday',
                    'Friday', 'Saturday', 'Sunday']

# Let's add the first dataset to the chart object
# we'll give it a name: set_a
set_a = chart.data.new_dataset('set_a')
# And let's add data to it.
# These are the data for the Y axis of the graph
# The data length should match the length of data for the X axis
set_a.data = [12, 19, 3, 17, 6, 3, 7]
# Set the color for this graph
set_a.set_color(0xFF, 0x8C, 0x8D, 0.8)
```

Step 3: The last step is to ask the chart object to draw itself.

```
# Before drawing the chart you can randomize the colors
# if you have not added any color to the datasets.
chart.randomize_colors()

# Finally let's draw the chart
chart.draw()
```

4.8.1 Line charts

See the comments in the script for more info

```
# get line chart object
chart = output.make_line_chart()

# this is a list of labels for the X axis of the line graph
chart.data.labels = ['Monday', 'Tuesday',
                    'Wednesday', 'Thursday',
                    'Friday', 'Saturday', 'Sunday']

# Let's add the first dataset to the chart object
# we'll give it a name: set_a
set_a = chart.data.new_dataset('set_a')
# And let's add data to it.
# These are the data for the Y axis of the graph
```

(continues on next page)

(continued from previous page)

```

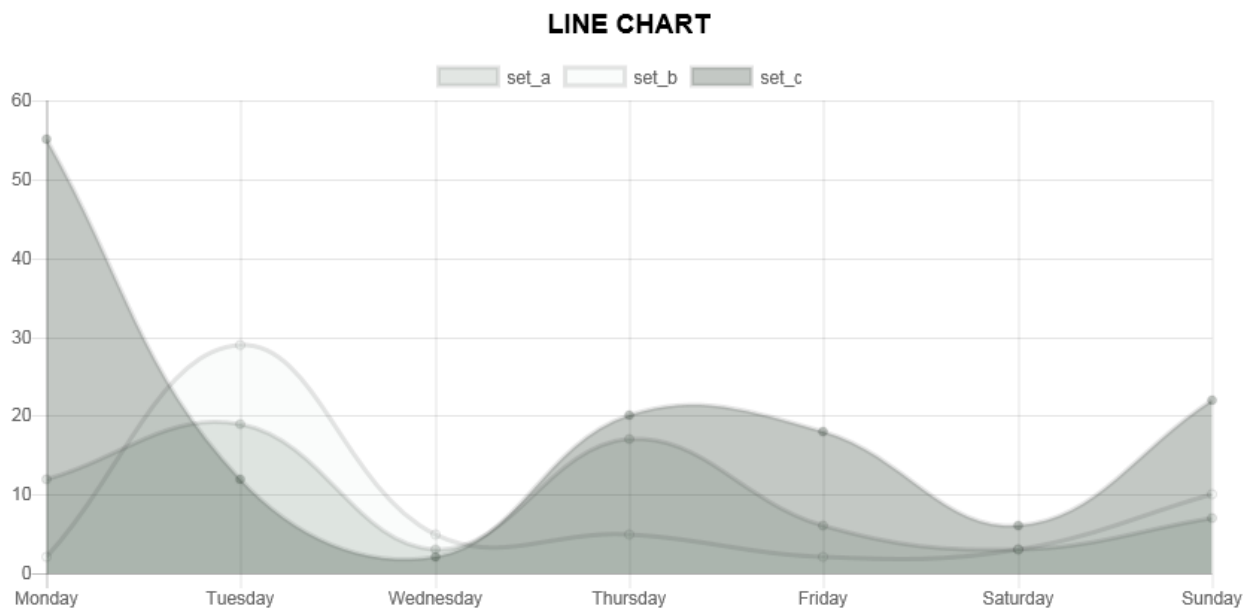
# The data length should match the length of data for the X axis
set_a.data = [12, 19, 3, 17, 6, 3, 7]
# Set the color for this graph
set_a.set_color(0xFF, 0x8C, 0x8D, 0.8)
# You can also set custom options for this graph
# See the Charts.js documentation for all the options
set_b.fill = False

# Same as above for a new data set: set_b
set_b = chart.data.new_dataset('set_b')
# Obviously a different set of data and a different color
set_b.data = [2, 29, 5, 5, 2, 3, 10]
set_b.set_color(0xFF, 0xCE, 0x56, 0.8)

# Same as above for a new data set: set_c
set_c = chart.data.new_dataset('set_c')
# Obviously a different set of data and a different color
set_c.data = [55, 12, 2, 20, 18, 6, 22]
set_c.set_color(0x36, 0xA2, 0xEB, 0.8)

```

And here is the result:



4.8.2 Pie charts

See the comments in the script for more info

```

# get pie chart object
chart = output.make_pie_chart()

# Set the labels for the circumference axis
chart.data.labels = ['A', 'B', 'C']

# Create new data sets

```

(continues on next page)

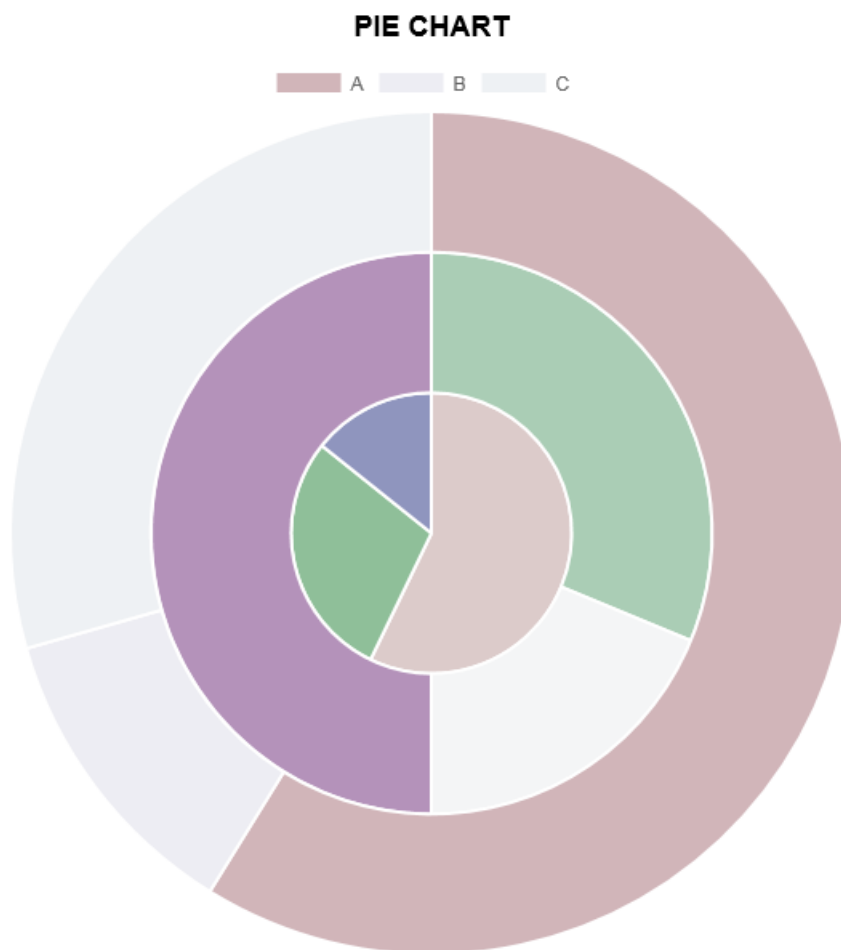
(continued from previous page)

```
set_a = chart.data.new_dataset('set_a')
set_a.data = [100, 20, 50]
# You can set a different color for each pie of the chart
set_a.backgroundColor = ["#560764", "#1F6CB0", "#F98B60"]

set_b = chart.data.new_dataset('set_b')
set_b.data = [50, 30, 80]
set_b.backgroundColor = ["#913175", "#70A3C4", "#FFC057"]

set_c = chart.data.new_dataset('set_c')
set_c.data = [40, 20, 10]
set_c.backgroundColor = ["#DD5B82", "#E7E8F5", "#FFE084"]
```

And here is the result:

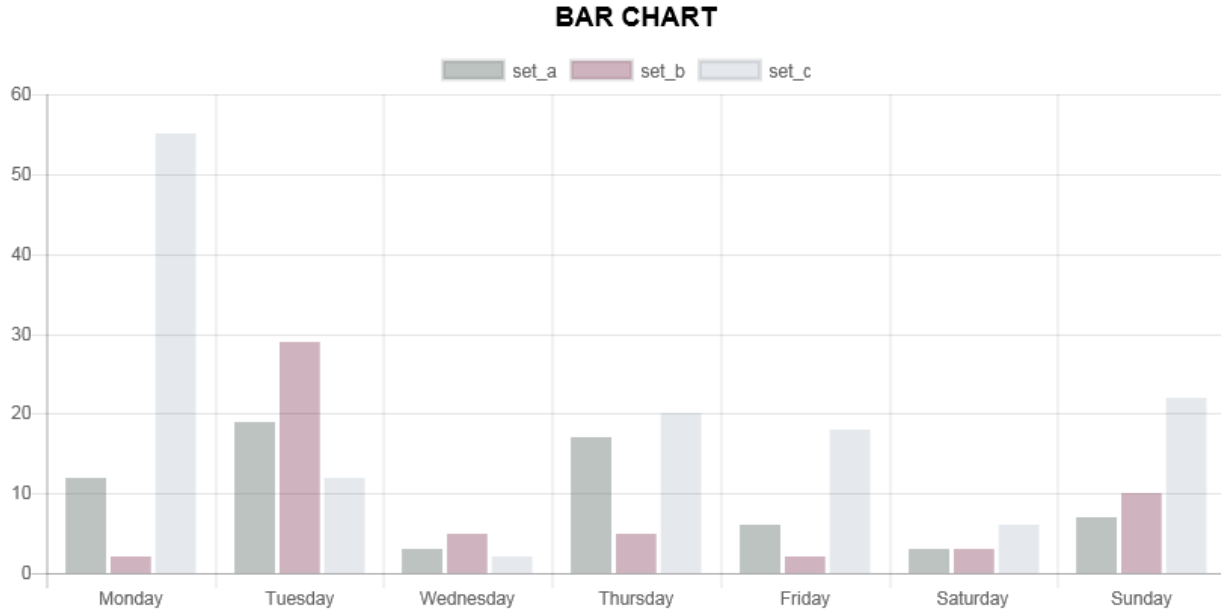


4.8.3 Bar charts

See the comments in the script for more info

```
# get bar chart object
chart = output.make_bar_chart()
```

And here is the result:

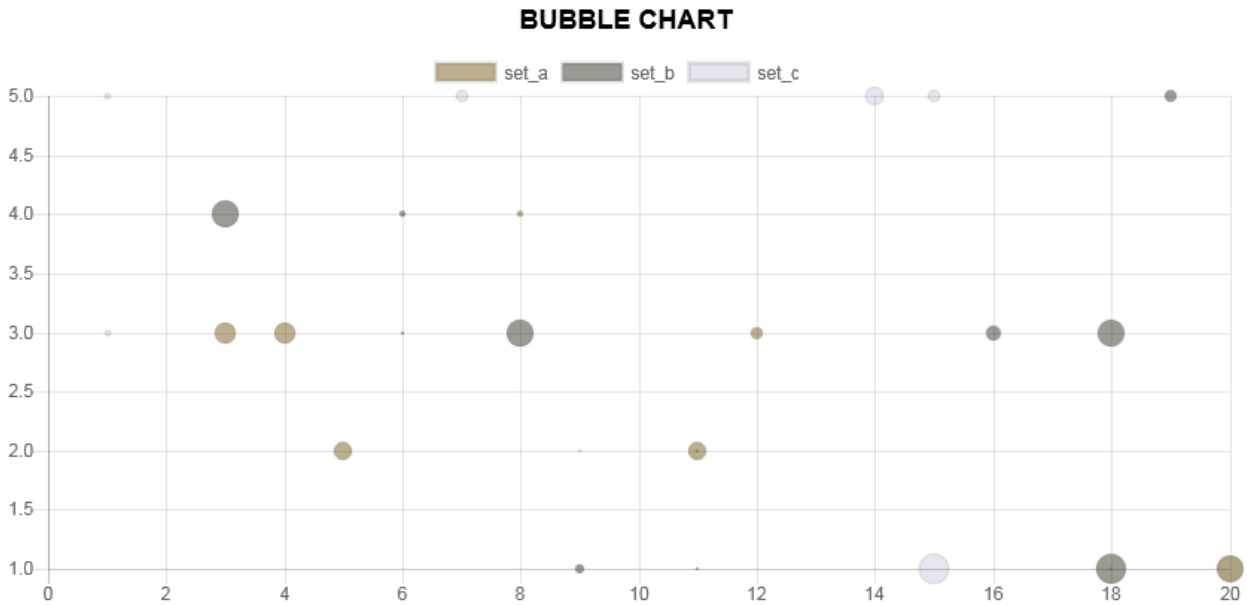


4.8.4 Bubble charts

See the comments in the script for more info

```
# get bubble chart object
chart = output.make_bubble_chart()
```

And here is the result:

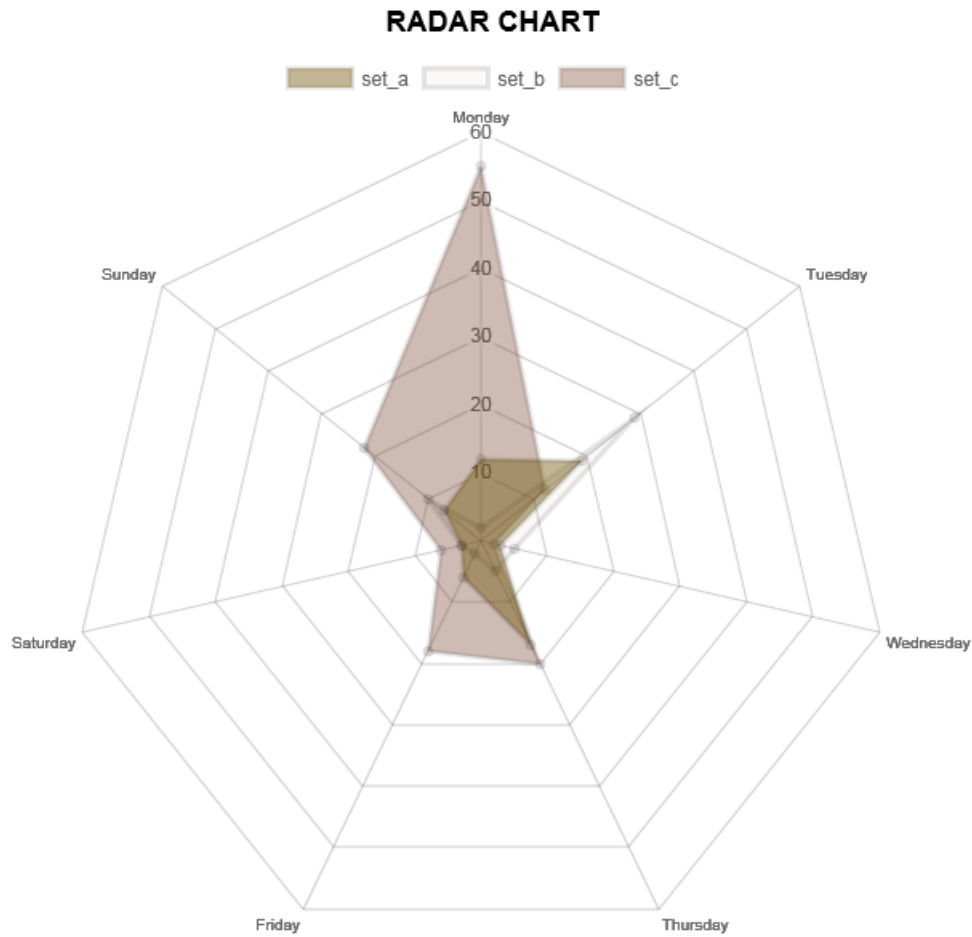


4.8.5 Radar charts

See the comments in the script for more info

```
# get radar chart object
chart = output.make_radar_chart()
```

And here is the result:

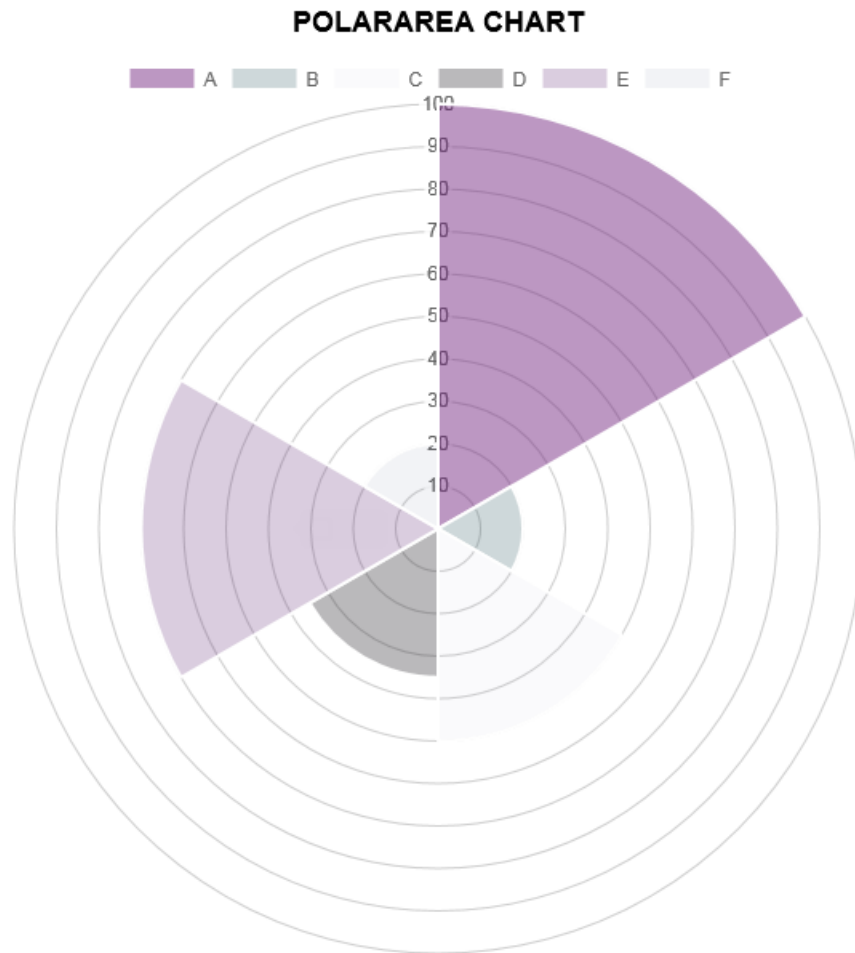


4.8.6 Polar Area charts

See the comments in the script for more info

```
# get polar chart object
chart = output.make_polar_chart()
```

And here is the result:

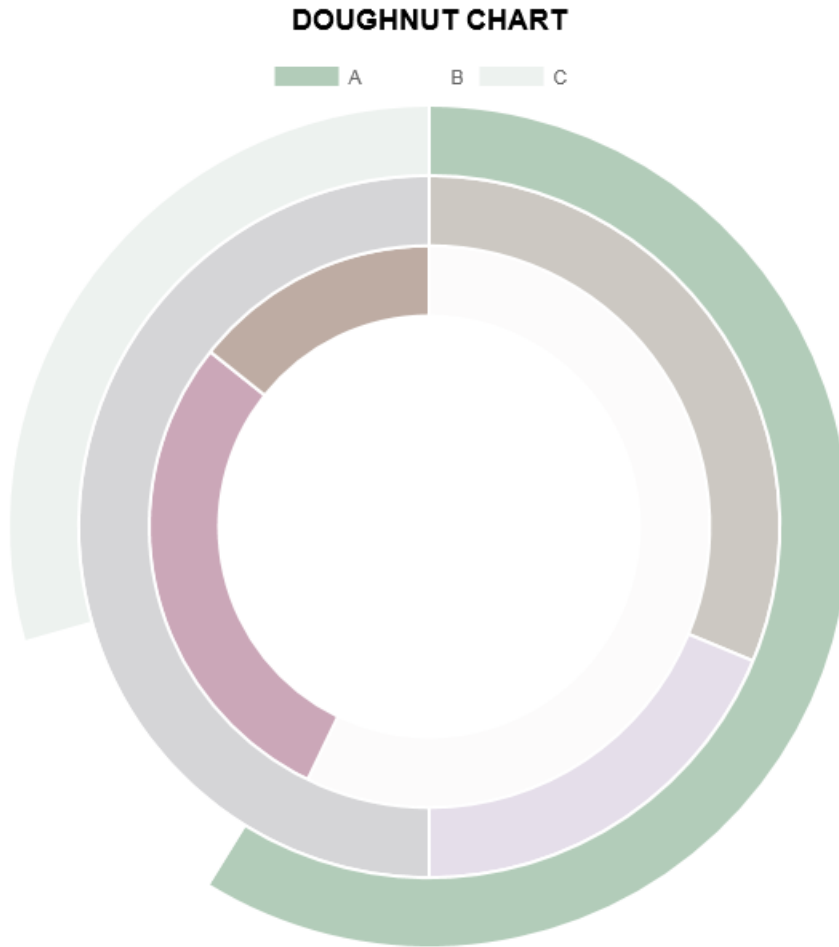


4.8.7 Doughnut charts

See the comments in the script for more info

```
# get doughnut chart object  
chart = output.make_doughnut_chart ()
```

And here is the result:



4.8.8 Charts engine

Here is a little info on how the charts engine works: the pyRevit charts module is `pyrevit.coreutils.charts`. This is the module that the output window interacts with to create the charts.

The charts module provides the chart object and handles the creation of datasets. The first thing it does when drawing the graph is to create a html `<canvas>` element and assign a unique id to it:

```
<canvas id="chart123456"></canvas>
```

Then it parses the input data and creates a JSON representation of the data. The JSON string (`json_data`) will be inserted into a template javascript. This javascript creates a Chart object from the `Chart.js` library:

```
var ctx = document.getElementById('{}').getContext('2d');
var chart = new Chart(ctx, json_data);
```

and finally, the pyRevit chart object, injects this dynamically created javascript into the `<head>` of the output window WebBrowser component:

```
output.inject_script(js_code)
```

Keyboard Shortcuts

5.1 Shift-Click: Alternate/Config Script

Each pyRevit command bundle can contain two scripts:

- *script.py is the main script.

- *config.py is the Alternate/Config script.

SHIFT-clicking on a ui button will run the alternate/config script. This alternate script is generally used to configure the main tool. Try Shift clicking on the Match tool in pyRevit > Modify panel and see the configuration window. Then try Shift clicking on the Settings tool in pyRevit panel slide-out and see what it does.

If you don't define the configuration script, you can check the value of `__shiftclick__` in your scripts to change script behaviour. This is the method that the Settings command is using to open the config file location in explorer:

```
if __shiftclick__:
    do_task_A()
else:
    do_task_B()
```

5.2 Ctrl-Click: Debug Mode

CTRL-clicking on a ui button will run the script in DEBUG mode and will allow the script to print all debug messages. You can check the value of `__forceddebugmode__` variable to see if the script is running in Debug mode to change script behaviour if necessary.

```
if __forceddebugmode__:
    do_task_A()
else:
    do_task_B()
```

5.3 Alt-Click: Show Script file in Explorer

ALT-clicking on a ui button will show the associated script file in windows explorer.

5.4 Ctrl-Shift-Alt-Click: Reload Engine

If you're using pyRevit Rocket mode, this keyboard combination will force pyRevit to discard the cached engine for this command and use a new fresh engine. If you are developing scripts for pyRevit and using external modules, you'll need to use this keyboard combination after changes to the imported module source codes. Since the modules are already imported in the cached engine, you'd need a new fresh engine to reload the modules.

5.5 Shift-Win-Click: pyRevit Button Context Menu

Shows the context menu for the pyRevit command. See image below:

Extensions and Commands

6.1 Why do I need an Extension

pyRevit's extensions system has evolved again to be more flexible and easier to work with. We'll dive right into how you can add your own extension, but first let's answer one important question:

Q: Why would I need to create a separate extension? Why Can't I just add my scripts to the current pyRevit tools?

A: Because pyRevit is a git repository and the real benefit of that is that you can keep it always updated without the need to uninstall and install the newer versions. To keep this system running without issues, I highly recommend not to mess with the pyRevit git repository folders and contents and pyRevit makes it really easy to add your own extensions. You can even add tools to the standard pyRevit tab in your own extensions. I'll show you how.

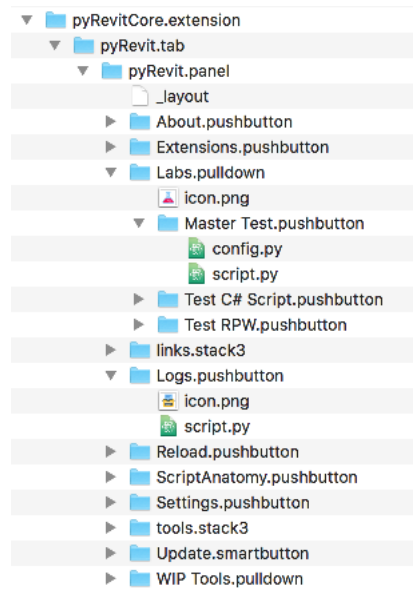
Besides, by creating a separate extension, you'll have all your precious scripts and tools in a safe place and away from the changes being made to the core pyRevit. They can even live somewhere on your company shared drives and be shared between your teams.

6.2 Extensions

Each extension is a group of tools, organized in bundles to be easily accessible through the user interface.

Extensions are organized in a folder bundle with `.extension` postfix.

Like this one:



There are two steps that you need to follow to create your own extensions:

- Setup external extension folder:

First, is to create a separate folder for all your custom extensions and tell pyRevit to load your extensions from this folder. This is done in the Settings window, under the Custom Extension folders section. This way your precious extensions will stay out of the pyRevit installation and are safe.

⤴ Custom user extension folders:

pyRevit can search in custom folders for extensions. You can setup your custom extensions in any directory and add the directory address here. pyRevit searches these folders on startup and loads the extensions. This helps to decouple the custom extensions from pyRevit's default extension folder so you can upgrade/remove pyRevit without worrying about your own extension folders.

Reload is required for changes to take effect.



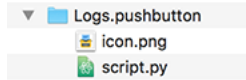
- Create custom extension bundle:

Next, create your `<your extension name>.extension` folder under your custom extensions folder. Read the sections below on how to create bundles for your commands and the user interface.

6.3 Command Bundles

A bundle is a folder named in the format `bundle_name.bundle_type`.

Like these:



The most basic bundle is a command bundle. There are more than one type of command bundles but a `.pushbutton` bundle explained here covers 90% of the use cases.

6.3.1 Pushbutton Bundle

Each command bundle needs to include a script either in python or C#:

script.py:

The first script file under the bundle that ends with `script.py` will be used as the script for this command bundle.

Examples: `BuildWall_script.py` `Analyse-script.py`

config.py:

This for python commands configuration. If this script is provided then Shift-Clicking on the button will run this command instead. Also a black dot will be added to the button name in the user interface to show that this command has a custom configuration tool. See [Shift-Click: Alternate/Config Script](#)

script.cs:

This for C# commands and works similarly to python scripts. This C# script will be compiled in runtime.

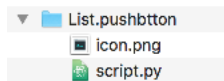
icon.png:

Command bundles can include an icon for their user interface.

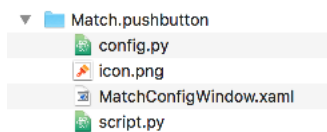
lib/:

Bundles can define a python library (a sub-folder named `lib` inside the bundle will do). This library will be accessible to the python script in this bundle. This organizes all the python modules that are necessary for this python script to work into one folder.

This is how a command bundle looks like:



And this is a more advanced command bundle with a configuration script and configuration window definition file:



6.4 Group Bundles

Now that we have explained the command bundles, we need a way to organize these commands into a user-friendly interface. Let's introduce **Group Bundles**

A group bundle is a bundle that can contain command bundles and other group bundles. They come in all different shapes and sizes but they have a few features in common:

- They can contain command bundles and other group bundles. (But I've already said that)

- **icon.png:** Bundle can include an icon for their user interface.
- **lib/:** The can define a python library (a sub-folder named `lib` inside the bundle will do). This library will be accessible to all the commands in this bundle and other child group bundles. This folder can contain all the python modules that are being shared between the child commands.
- **_layout:** This is a text file inside the bundle that defines the order in which the bundle contents should be created in the user interface. The contents of this file should be the names of the component in the order that they should be created in the user interface.

Here is `_layout` file example. This is a layout file for a Group Bundle that has a series of push buttons and other group bundles under itself:

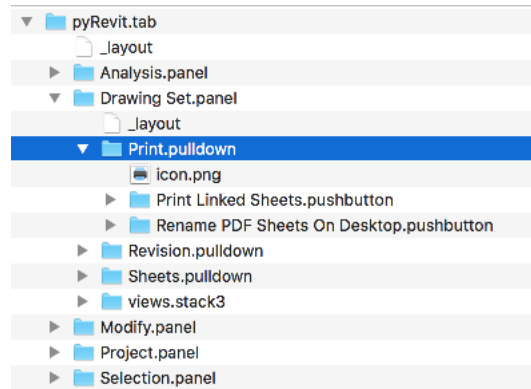
```

PushButton A
PushButton B
PullDown A
---
PullDown B
Stack3 A
>>>
PushButton C
PullDown C
    
```

Oh, and also:

- `---` This line will add a separator to the interface (You can use more than 3 - characters. For example `-----` still works as a separator)
- `>>>` Any bundle after this line will be created inside a slide-out. This works for panel bundles only. (You can use more than 3 > characters. For example `>>>>>>>>` still works as a slide-out)

And this is how a typical Group Bundle looks like:



Now let's talk about the different Group Bundles:

6.4.1 Tab Bundle

This bundle creates a Tab in the Ribbon with the bundle name.

Example	Can Contain
<code>pyRevit.tab</code>	Only <code>.panel</code> Group Bundles.

6.4.2 Panel Bundle

This bundle creates a Panel in a Ribbon Tab with the bundle name.

Example	Can Contain
<code>pyRevit.panel</code>	Any other bundle type

6.4.3 PullDown Bundle

This bundle creates a PullDown Button in a Ribbon Panel or a Stack, with the bundle name and icon.

Example	Can Contain
<code>pyRevit.pulldown</code>	Only command bundles

6.4.4 SplitButton Bundle

This bundle creates a Split Button button in a Ribbon Panel or a Stack, with the bundle name and icon.

Example	Can Contain
<code>pyRevit.splitbutton</code>	Only command bundles

6.4.5 SplitPushButton Bundle

This bundle creates a Split Push Button button (The sticky split button) in a Ribbon Panel or a Stack, with the bundle name and icon.

Example	Can Contain
<code>pyRevit.splitpushbutton</code>	Only command bundles

6.4.6 Stack Bundle: Two Buttons

This bundle creates a stack of 2 buttons in a panel.

Example	Can Contain
<code>pyRevit.stack2</code>	Can contain: <code>.pulldown</code> <code>.splitbutton</code> <code>.splitpushbutton</code>

6.4.7 Stack Bundle: Three Buttons

Just like the `.stack2` bundle but with 3 buttons instead.

6.5 Advanced Bundles

There are a few more advanced bundle types in pyRevit as well. Here is some quick intro on these bundles.

6.5.1 Smart Button Bundle

Example
<code>pyRevit.smartbutton</code>

Smart buttons are python scripts that are written like modules. They should define `__selfinit__` function as shown below. This function gets executed at startup time to give a chance to the button to initialize itself (e.g set its icon based on its state).

The `__selfinit__` must return `True` if the initialization is successful and `False` if it is not. pyRevit will not create the button if the initialization returns `False` and is unsuccessful.

```
def __selfinit__(script_cmp, ui_button_cmp, __rvt__):
    """
    Args:
        script_cmp: script component that contains info on this script
        ui_button_cmp: this is the UI button component
        __rvt__: Revit UIApplication

    Returns:
        bool: Return True if successful, False if not
    """
    run_self_initialization()
```

6.5.2 No Button Bundle

Example
<code>pyRevit.nobutton</code>

No-Button bundles are just like Pushbutton bundles except that they will never show up inside Revit UI and thus don't need any icons. The only method to run these commands is through pyRevit Search tool. These commands are meant for more advanced commands that not every user needs.

6.5.3 Panel Button Bundle

Example
<code>pyRevit.panelbutton</code>

Panel Button bundles are just like Pushbutton bundles except that they will be set as the panel configuration button (small arrow at the corner of UI Panels). These bundles do not need to have an icon as the standard Small arrow icon is used for panel configuration buttons by default. These commands work just like any pyRevit command but their primary purpose should be to configure set of related tools in a panel.

6.5.4 Link Button Bundle

Example
<code>pyRevit.linkbutton</code>

Link buttons can call a function from another Addin. To make a link button define the parameters below in the bundles `script.py`:

Note: For this button to work properly, the target addin must be already loaded when this button is being created, otherwise Revit can not tie the UI button to an assembly that is not loaded.

```
__assembly__ = 'Addin assembly name'  
__commandclass__ = 'Class name for the command'
```

For example to call the Interactive Python Shell from RevitPythonShell addin:

```
__assembly__ = 'RevitPythonShell'  
__commandclass__ = 'IronPythonConsoleCommand'
```

6.6 Other Extension Types

6.6.1 Library Extensions

Library extensions are created to share IronPython modules between all extensions. They're in essence IronPython module packages. Some users might decide to develop an IronPython library (e.g. [RevitPythonWrapper Library](#)) that other users can use in their tools and benefit from.

Library extensions are identified by `.lib` postfix. The library extension folder address will be added to the `sys.path` of all the other extensions by the loader.

CHAPTER 7

pyRevit Configuration

work in progress

CHAPTER 8

Usage Logger

work in progress

pyRevit CLI Utility

pyRevit has a command line utility tool that helps managing pRevit clones, extensions and configurations, a lot easier.

Documentation The documentation for this tool is provided in a markdown file inside the pyrevit repository. The reason is that the cli tool knows the link to its own help based on its versions, to the markdown file on the repo, and can direct the user to it upon running `pyrevit help`. It made more sense to maintain the help in a single file inside the repo, rather than a more complex documentation like this that needs sphinx to generate the actual documentation pages.

You can access the latest CLI help [HERE](#)

pyRevit Core

- *Load Sequence, Step 1: Revit Addon*
- *Load Sequence, Step 2: IronPython Module*

Load Sequence, Step 1: Revit Addon

10.1 The Complex Relationship of a C# Addin and a Python Script

Let's talk basics:

- Revit Addons are written in C# and are windows `.dll` files.
- `pyRevit` is written as an IronPython module. (actually a bit more complex than that)
- Revit doesn't have an option to run external python scripts.

Thus, we need a way to teach Revit how to run a python script when it's starting up.

The solution was to create a custom C# addin to create a python engine and run a script. We'll call this addin `pyRevitLoader.dll`. I wanted to keep this addin as simple as possible since it's the only statically-compiled piece of code in this project. The rest of the task of loading `pyRevit` were assigned to a loader python script that is being run by the loader addin.

So:

- `pyRevitLoader.dll` is a simple C# addin for Revit that runs python scripts
- `pyRevitLoader.dll` loads `pyRevitLoader.py` at startup.
- `pyRevitLoader.py` sets up the environment and loads `pyRevit`.

It's that simple really. See the sources below.

From here on, the documentation page for the `pyrevit.loader` module will take you through all the steps of parsing extensions, making dll assemblies and creating the user interface for the parsed extensions.

10.2 `pyRevit` loader script

Here is the full source of `pyRevitLoader.py`. The docstring explains how it works.

```
# -*- coding: utf-8 -*-
#pylint: disable=C0103,W1401,E0401,E0602
"""

This is the starting point for pyRevit. At Revit loads the PyRevitLoader.dll
addon at startup. This dll then creates an ironpython engine and runs
pyRevitLoader.py (this script). It's the job of this script to setup the
environment for the pyrevit module (pyrevitlib\pyrevit) and load a new pyRevit
session. This script needs to add the directory path of the pyrevit lib folder
so the pyrevit module can be imported and used.
"""

import sys
import os.path as op

# add the library location to the system search paths
repo_path = op.dirname(op.dirname(op.dirname(__file__)))
sys.path.append(op.join(repo_path, 'pyrevitlib'))

# now pyrevit can be imported
from pyrevit.loader import sessionmgr

# ask sessionmgr to start a new session
sessionmgr.load_session()
```

10.3 pyRevitLoader Addin Source

The source code for pyRevitLoader addin is under: pyrevitlib/pyrevit/addin/<loader version>/Source

Load Sequence, Step 2: IronPython Module

work in progress

pyRevit Module

- *pyrevit*
- *pyrevit.api*
- *pyrevit.compat*
- *pyrevit.coreutils*
- *pyrevit.forms*
- *pyrevit.framework*
- *pyrevit.loader*
- *pyrevit.output*
- *pyrevit.revit*
- *pyrevit.script*
- *pyrevit.userconfig*
- *pyrevit.versionmgr*

pyRevit root level config for all pyrevit sub-modules.

Examples

```
>>> from pyrevit import DB, UI
>>> from pyrevit import PyRevitException, PyRevitIOError
```

```
>>> # pyrevit module has global instance of the
>>> # _HostAppPostableCommand and _ExecutorParams classes already created
>>> # import and use them like below
>>> from pyrevit import HOST_APP
>>> from pyrevit import EXEC_PARAMS
```

class pyrevit.**PyRevitException**

Common base class for all pyRevit exceptions.

Parameters args and message are derived from Exception class.

class pyrevit.**PyRevitIOError**

Common base class for all pyRevit io-related exceptions.

class pyrevit.**_HostAppPostableCommand** (*name, key, id, rvtobj*)

Private namedtuple for passing information about a PostableCommand

name

Postable command name

Type str

key

Postable command key string

Type str

id

Postable command id

Type int

rvtobj

Postable command Id Object

Type RevitCommandId

class pyrevit._HostApplication(*host_uiapp*)

Private Wrapper for Current Instance of Revit.

Provides version info and comparison functionality, alongside providing info on the active screen, active document and ui-document, available postable commands, and other functionality.

Parameters *host_uiapp* (UIApplication) – Instance of running host.

Example

```
>>> hostapp = _HostApplication(__revit__)
>>> hostapp.is_newer_than(2017)
```

activeview

Return view that is active (UIDocument.ActiveView).

app

Return Application provided to the running command.

available_servers

Return list of available Revit server names.

build

Return build number (e.g. '20170927_1515(x64)').

Type str

doc

Return active Document.

docs

Return list of open Document objects.

get_postable_commands ()

Return list of postable commands.

Returns list of *_HostAppPostableCommand*

is_exactly (*version*)

bool: Return True if host app is equal to provided version.

Parameters *version* (*str* or *int*) – version to check against.

is_newer_than (*version*, *or_equal=False*)

bool: Return True if host app is newer than provided version.

Parameters *version* (*str* or *int*) – version to check against.

is_older_than (*version*)

bool: Return True if host app is older than provided version.

Parameters *version* (*str* or *int*) – version to check against.

proc

Return current process object.

Type System.Diagnostics.Process

proc_id
Return current process id.

Type int

proc_name
Return current process name.

Type str

proc_path
Return file path for the current process main module.

Type str

proc_screen
Return handle to screen hosting current process.

Type IntPtr

proc_screen_scalefactor
Return scaling for screen hosting current process.

Type float

proc_screen_workarea
Return screen working area.

Type System.Drawing.Rectangle

proc_window
Return handle to screen hosting current process.

Type IntPtr

subversion
Return subversion number (e.g. '2018.3').

Type str

uiapp
Return UIApplication provided to the running command.

uidoc
Return active UIDocument.

username
Return the username from Revit API (Application.Username).

Type str

version
Return version number (e.g. '2018').

Type str

version_name
Return version name (e.g. 'Autodesk Revit 2018').

Type str

class pyrevit._ExecutorParams
Private Wrapper that provides runtime environment info.

command_alt_path
Return current command alternate script path.

Type str

command_bundle

Return current command bundle name.

Type str

command_data

Return current command data.

Type ExternalCommandData

command_extension

Return current command extension name.

Type str

command_mode

Check if pyrevit is running in pyrevit command context.

Type bool

command_name

Return current command name.

Type str

command_path

Return current command path.

Type str

command_uniqueid

Return current command unique id.

Type str

doc_mode

Check if pyrevit is running by doc generator.

Type bool

engine_mgr

Return engine manager.

Type PyRevitBaseClasses.EngineManager

engine_ver

Return PyRevitLoader.ScriptExecutor hardcoded version.

Type str

executed_from_ui

Check if command was executed from ui.

Type bool

first_load

Check whether pyrevit is not running in pyrevit command.

Type bool

forced_debug_mode

Check if command is in debug mode.

Type bool

pyrevit_command

Return command.

Type `PyRevitBaseClasses.PyRevitCommandRuntime`

result_dict

Return results dict for logging.

Type `Dictionary<String, String>`

window_handle

Return output window.

Type `PyRevitBaseClasses.ScriptOutput`

CHAPTER 13

pyrevit.api

Provide access to Revit API.

Example

```
>>> from pyrevit.api import AdWindows
>>> from pyrevit.api import NSJson
```


CHAPTER 14

pyrevit.compat

python engine compatibility module.

Example

```
>>> from pyrevit.compat import IRONPY277
>>> from pyrevit.compat import safe_strtype
```


Misc Helper functions for pyRevit.

15.1 pyrevit.coreutils

Misc Helper functions for pyRevit.

Example

```
>>> from pyrevit import coreutils
>>> coreutils.cleanup_string('some string')
```

class pyrevit.coreutils.**FileWatcher** (*filepath*)

Bases: object

Simple file version watcher.

This is a simple utility class to look for changes in a file based on its timestamp.

Example

```
>>> watcher = FileWatcher('/path/to/file.ext')
>>> watcher.has_changed
True
```

has_changed

Compare current file timestamp to the cached timestamp.

update_tstamp ()

Update the cached timestamp for later comparison.

class pyrevit.coreutils.SafeDict

Bases: dict

Dictionary that does not fail on any key.

This is a dictionary subclass to help with string formatting with unknown key values.

Example

```
>>> string = '{target} {attr} is {color}.'
>>> safedict = SafeDict({'target': 'Apple',
...                     'attr': 'Color'})
>>> string.format(safedict) # will not fail with missing 'color' key
'Apple Color is {color}.'
```

class pyrevit.coreutils.ScriptFileParser(*file_address*)

Bases: object

Parse python script to extract variables and docstrings.

Primarily designed to assist pyRevit in determining script configurations but can work for any python script.

Example

```
>>> finder = ScriptFileParser('/path/to/coreutils/__init__.py')
>>> finder.docstring()
... "Misc Helper functions for pyRevit."
>>> finder.extract_param('SomeValue', [])
[]
```

extract_param(*param_name*, *default_value=None*)

Find variable and extract its value.

Parameters

- **param_name** (*str*) – variable name
- **default_value** (*any*) – default value to be returned if variable does not exist

Returns value of the variable or None

Return type any

get_docstring()

Get global docstring.

class pyrevit.coreutils.Timer

Bases: object

Timer class using python native time module.

Example

```
>>> timer = Timer()
>>> timer.get_time()
12
```

get_time()
Get Elapsed Time.

restart()
Restart Timer.

`pyrevit.coreutils.calculate_dir_hash(dir_path, dir_filter, file_filter)`
Create a unique hash to represent state of directory.

Parameters

- **dir_path** (*str*) – target directory
- **dir_filter** (*str*) – exclude directories matching this regex
- **file_filter** (*str*) – exclude files matching this regex

Returns hash value as string

Return type str

Example

```
>>> calculate_dir_hash(source_path, '\.extension', '\.json')
"1a885a0cae99f53d6088b9f7cee3bf4d"
```

`pyrevit.coreutils.can_access_url(url_to_open, timeout=1000)`
Check if url is accessible within timeout.

Parameters

- **url_to_open** (*str*) – url to check access for
- **timeout** (*int*) – timeout in milliseconds

Returns true if accessible

Return type bool

`pyrevit.coreutils.check_internet_connection(timeout=1000)`
Check if internet connection is available.

Pings a few well-known websites to check if internet connection is present.

Parameters **timeout** (*int*) – timeout in milliseconds

Returns url if internet connection is present, None if no internet.

`pyrevit.coreutils.check_revittxt_encoding(filename)`
Check if given file is in UTF-16 (UCS-2 LE) encoding.

Parameters **filename** (*str*) – file path

`pyrevit.coreutils.check_utf8bom_encoding(filename)`
Check if given file is in UTF-8 encoding.

Parameters **filename** (*str*) – file path

`pyrevit.coreutils.cleanup_filename(file_name)`
Cleanup file name from special characters.

Parameters **file_name** (*str*) – file name

Returns cleaned up file name

Return type str

Example

```
>>> cleanup_filename('Myfile-(3).txt')
"Myfile(3).txt"
```

```
>>> cleanup_filename('Perforations 1/8" (New)')
"Perforations 18 (New).txt"
```

`pyrevit.coreutils.cleanup_string(input_str)`

Replace special characters in string with another string.

This function was created to help cleanup pyRevit command unique names from any special characters so C# class names can be created based on those unique names.

`coreutils.SPECIAL_CHARS` is the conversion table for this function.

Parameters `input_str (str)` – input string to be cleaned

Example

```
>>> src_str = 'TEST@Some*<value>'
>>> cleanup_string(src_str)
"TEST@SomeSTARvalue"
```

`pyrevit.coreutils.correct_revittxt_encoding(filename)`

Convert encoding of text file generated by Revit to UTF-8.

Parameters `filename (str)` – file path

`pyrevit.coreutils.create_ext_command_attrs()`

Create dotnet attributes for Revit external commands.

This method is used in creating custom dotnet types for pyRevit commands and compiling them into a DLL assembly. Current implementation sets `RegenerationOption.Manual` and `TransactionMode.Manual`.

Returns list of `CustomAttributeBuilder` for `RegenerationOption` and `TransactionMode` attributes.

Return type list

`pyrevit.coreutils.create_type(modulebuilder, type_class, class_name, custom_attr_list, *args)`

Create a dotnet type for a pyRevit command.

See `baseclasses.cs` code for the template pyRevit command dotnet type and its constructor default arguments that must be provided here.

Parameters

- **modulebuilder** (`ModuleBuilder`) – dotnet module builder
- **type_class** (`type`) – source dotnet type for the command
- **class_name** (`str`) – name for the new type
- **custom_attr_list** (`list`) – list of dotnet attributes for the type
- ***args** – list of arguments to be used with type constructor

Returns returns created dotnet type

Return type type

Example

```
>>> asm_builder = AppDomain.CurrentDomain.DefineDynamicAssembly(
... win_asm_name, AssemblyBuilderAccess.RunAndSave, filepath
... )
>>> module_builder = asm_builder.DefineDynamicModule(
... ext_asm_file_name, ext_asm_full_file_name
... )
>>> create_type(
... module_builder,
... PyRevitCommand,
... "PyRevitSomeCommandUniqueName",
... coreutils.create_ext_command_attrs(),
... [scriptpath, atscriptpath, searchpath, helpurl, name,
... bundle, extension, unique_name, False, False])
<type PyRevitSomeCommandUniqueName>
```

`pyrevit.coreutils.current_date()`

Return formatted current date.

Current implementation uses `%Y-%m-%d` to format date.

Returns formatted current date.

Return type str

Example

```
>>> current_date()
'2018-01-03'
```

`pyrevit.coreutils.current_time()`

Return formatted current time.

Current implementation uses `%H:%M:%S` to format time.

Returns formatted current time.

Return type str

Example

```
>>> current_time()
'07:50:53'
```

`pyrevit.coreutils.decrement_str(input_str, step)`

Decrement identifier.

Parameters

- **input_str** (*str*) – identifier e.g. A310a
- **step** (*int*) – number of steps to change the identifier

Returns modified identifier

Return type str

Example

```
>>> decrement_str('A310a')
'A309z'
```

`pyrevit.coreutils.dletter_to_unc(dletter_path)`

Convert drive letter path into UNC path of that drive.

Parameters `dletter_path` (*str*) – drive letter path

Returns UNC path

Return type `str`

Example

```
>>> # assuming J: is mapped to //filestore/server/jdrive
>>> dletter_to_unc('J:/somefile.txt')
'//filestore/server/jdrive/somefile.txt'
```

`pyrevit.coreutils.extract_guid(source_str)`

Extract GUID number from a string.

`pyrevit.coreutils.extract_range(formatted_str, max_range=500)`

Extract range from formatted string.

String must be formatted as below A103 No range A103-A106 A103 to A106 A103:A106 A103 to A106 A103,A105a A103 and A105a A103;A105a A103 and A105a

Parameters `formatted_str` (*str*) – string specifying range

Returns list of names in the specified range

Return type `list`

Example

```
>>> extract_range('A103:A106')
['A103', 'A104', 'A105', 'A106']
>>> extract_range('S203-S206')
['S203', 'S204', 'S205', 'S206']
>>> extract_range('M00A,M00B')
['M00A', 'M00B']
```

`pyrevit.coreutils.filter_null_items(src_list)`

Remove None items in the given list.

Parameters `src_list` (*list*) – list of any items

Returns cleaned list

Return type `list`

`pyrevit.coreutils.find_loaded_asm(asm_info, by_partial_name=False, by_location=False)`

Find loaded assembly based on name, partial name, or location.

Parameters

- `asm_info` (*str*) – name or location of the assembly

- **by_partial_name** (*bool*) – returns all assemblies that has the asm_info
- **by_location** (*bool*) – returns all assemblies matching location

Returns List of all loaded assemblies matching the provided info If only one assembly has been found, it returns the assembly. None will be returned if assembly is not loaded.

Return type list

`pyrevit.coreutils.find_type_by_name(assembly, type_name)`

Find type by name in assembly.

Parameters

- **assembly** (*Assembly*) – assembly to find the type in
- **type_name** (*str*) – type name

Returns returns the type if found.

Raises `PyRevitException` if type not found.

`pyrevit.coreutils.format_hex_rgb(rgb_value)`

Formats rgb value as #RGB value string.

`pyrevit.coreutils.fully_remove_dir(dir_path)`

Remove directory recursively.

Parameters **dir_path** (*str*) – directory path

`pyrevit.coreutils.fuzzy_search_ratio(target_string, sfilter)`

Match target string against the filter and return a match ratio.

Parameters

- **target_string** (*str*) – target string
- **sfilter** (*str*) – search term

Returns integer between 0 to 100, with 100 being the exact match

Return type int

`pyrevit.coreutils.get_all_subclasses(parent_classes)`

Return all subclasses of a python class.

Parameters **parent_classes** (*list*) – list of python classes

Returns list of python subclasses

Return type list

`pyrevit.coreutils.get_canonical_parts(canonical_string)`

Splits argument using dot, returning all composing parts.

Parameters **canonical_string** (*str*) – Source string e.g. “Config.SubConfig”

Returns list of composing parts

Return type list[str]

Example

```
>>> get_canonical_parts("Config.SubConfig")
['Config', 'SubConfig']
```

`pyrevit.coreutils.get_enum_none (enum_type)`
Returns the None value in given Enum.

`pyrevit.coreutils.get_enum_value (enum_type, value_string)`
Return enum value matching given value string (case insensitive)

`pyrevit.coreutils.get_enum_values (enum_type)`
Returns enum values.

`pyrevit.coreutils.get_exe_version (exepath)`
Extract Product Version value from EXE file.

`pyrevit.coreutils.get_file_name (file_path)`
Return file basename of the given file.

Parameters `file_path (str)` – file path

`pyrevit.coreutils.get_mapped_drives_dict ()`
Return a dictionary of currently mapped network drives.

`pyrevit.coreutils.get_reg_key (key, subkey)`
Get value of the given Windows registry key and subkey.

Parameters

- **key** (`PyHKEY`) – parent registry key
- **subkey** (`str`) – subkey path

Returns registry key if found, None if not found

Return type `PyHKEY`

Example

```
>>> get_reg_key(wr.HKEY_CURRENT_USER, 'Control Panel/International')
... <PyHKEY at 0x...>
```

`pyrevit.coreutils.get_revit_instance_count ()`
Return number of open host app instances.

Returns number of open host app instances.

Return type `int`

`pyrevit.coreutils.get_str_hash (source_str)`
Calculate hash value of given string.

Current implementation uses `hashlib.md5 ()` hash function.

Parameters `source_str (str)` – source str

Returns hash value as string

Return type `str`

`pyrevit.coreutils.get_sub_folders (search_folder)`
Get a list of all subfolders directly inside provided folder.

Parameters `search_folder (str)` – folder path

Returns list of subfolder names

Return type `list`

`pyrevit.coreutils.has_nonprintable(input_str)`
Check input string for non-printable characters.

Parameters `input_str` (*str*) – input string

Returns True if contains non-printable characters

Return type bool

`pyrevit.coreutils.increment_str(input_str, step)`
Increment identifier.

Parameters

- `input_str` (*str*) – identifier e.g. A310a
- `step` (*int*) – number of steps to change the identifier

Returns modified identifier

Return type str

Example

```
>>> increment_str('A319z')
'A320a'
```

`pyrevit.coreutils.inspect_calling_scope_global_var(variable_name)`
Trace back the stack to find the variable in the caller global stack.

Parameters `variable_name` (*str*) – variable name to look up in caller global scope

`pyrevit.coreutils.inspect_calling_scope_local_var(variable_name)`
Trace back the stack to find the variable in the caller local stack.

PyRevitLoader defines `__revit__` in builtins and `__window__` in locals. Thus, modules have access to `__revit__` but not to `__window__`. This function is used to find `__window__` in the caller stack.

Parameters `variable_name` (*str*) – variable name to look up in caller local scope

`pyrevit.coreutils.is_blank(input_string)`
Check if input string is blank (multiple white spaces is blank).

Parameters `input_string` (*str*) – input string

Returns True if string is blank

Return type bool

Example

```
>>> is_blank(' ')
True
```

`pyrevit.coreutils.is_box_visible_on_screens(left, top, width, height)`
Check if given box is visible on any screen.

`pyrevit.coreutils.is_url_valid(url_string)`
Check if given URL is in valid format.

Parameters `url_string` (*str*) – URL string

Returns True if URL is in valid format

Return type bool

Example

```
>>> is_url_valid('https://www.google.com')
True
```

`pyrevit.coreutils.join_strings` (*str_list*, *separator=';*')
Join strings using provided separator.

Parameters

- **str_list** (*list*) – list of string values
- **separator** (*str*) – single separator character, defaults to DEFAULT_SEPARATOR

Returns joined string

Return type str

`pyrevit.coreutils.kill_tasks` (*task_name*)
Kill running tasks matching task_name

Parameters **task_name** (*str*) – task name

Example

```
>>> kill_tasks('Revit.exe')
```

`pyrevit.coreutils.load_asm` (*asm_name*)
Load assembly by name into current domain.

Parameters **asm_name** (*str*) – assembly name

Returns returns the loaded assembly, None if not loaded.

`pyrevit.coreutils.load_asm_file` (*asm_file*)
Load assembly by file into current domain.

Parameters **asm_file** (*str*) – assembly file path

Returns returns the loaded assembly, None if not loaded.

`pyrevit.coreutils.make_canonical_name` (**args*)
Join arguments with dot creating a unique id.

Parameters ***args** – Variable length argument list of type str

Returns dot separated unique name

Return type str

Example

```
>>> make_canonical_name('somename', 'someid', 'txt')
"somename.someid.txt"
```

`pyrevit.coreutils.new_uuid()`
Create a new UUID (using dotnet Guid.NewGuid)

`pyrevit.coreutils.open_folder_in_explorer(folder_path)`
Open given folder in Windows Explorer.

Parameters `folder_path` (*str*) – directory path

`pyrevit.coreutils.prepare_html_str(input_string)`
Reformat html string and prepare for pyRevit output window.

pyRevit output window renders html content. But this means that `<` and `>` characters in outputs from python (e.g. `<class at xxx>`) will be treated as html tags. To avoid this, all `<>` characters that are defining html content need to be replaced with special phrases. pyRevit output later translates these phrases back in to `<` and `>`. That is how pyRevit distinguishes between `<>` printed from python and `<>` that define html.

Parameters `input_string` (*str*) – input html string

Example

```
>>> prepare_html_str('<p>Some text</p>')
"&lt;t;p&cgt;Some text&lt;/p&cgt;"
```

`pyrevit.coreutils.random_alpha()`
Return a random alpha value (between 0 and 1.00).

`pyrevit.coreutils.random_color()`
Return a random color channel value (between 0 and 255).

`pyrevit.coreutils.random_hex_color()`
Return a random color in hex format.

Example

```
>>> random_hex_color()
'#FF0000'
```

`pyrevit.coreutils.random_rgb_color()`
Return a random color in rgb format.

Example

```
>>> random_rgb_color()
'rgb(255, 0, 0)'
```

`pyrevit.coreutils.random_rgba_color()`
Return a random color in rgba format.

Example

```
>>> random_rgba_color()
'rgba(255, 0, 0, 0.5)'
```

`pyrevit.coreutils.read_source_file(source_file_path)`

Read text file and return contents.

Parameters `source_file_path` (*str*) – target file path

Returns file contents

Return type `str`

Raises `PyRevitException` on read error

`pyrevit.coreutils.reformat_string(orig_str, orig_format, new_format)`

Reformat a string into a new format.

Extracts information from a string based on a given pattern, and recreates a new string based on the given new pattern.

Parameters

- **orig_str** (*str*) – Original string to be reformatted
- **orig_format** (*str*) – Pattern of the original str (data to be extracted)
- **new_format** (*str*) – New pattern (how to recompose the data)

Returns Reformatted string

Return type `str`

Example

```
>>> reformat_string('150 - FLOOR/CEILING - WD - 1 HR - FLOOR ASSEMBLY',
                    '{section} - {loc} - {mat} - {rating} - {name}',
                    '{section}:{mat}:{rating} - {name} ({{loc}})')
'150:WD:1 HR - FLOOR ASSEMBLY (FLOOR/CEILING)'
```

`pyrevit.coreutils.reverse_dict(input_dict)`

Reverse the key, value pairs.

Parameters `input_dict` (*dict*) – source ordered dict

Returns reversed dictionary

Return type `defaultdict`

Example

```
>>> reverse_dict({1: 2, 3: 4})
defaultdict(<type 'list'>, {2: [1], 4: [3]})
```

`pyrevit.coreutils.reverse_html(input_html)`

Reformat codified pyRevit output html string back to normal html.

pyRevit output window renders html content. But this means that `<` and `>` characters in outputs from python (e.g. `<class at xxx>`) will be treated as html tags. To avoid this, all `<>` characters that are defining html content need to be replaced with special phrases. pyRevit output later translates these phrases back in to `<` and `>`. That is how pyRevit distinguishes between `<>` printed from python and `<>` that define html.

Parameters `input_html` (*str*) – input codified html string

Example

```
>>> prepare_html_str('&clt;p&cggt;Some text&clt;/p&cggt;')
"<p>Some text</p>"
```

`pyrevit.coreutils.run_process(proc, cwd='C:')`

Run shell process silently.

Parameters

- **proc** (*str*) – process executive name
- **cwd** (*str*) – current working directory

Example:

```
>>> run_process('notepad.exe', 'c:/')
```

`pyrevit.coreutils.show_entry_in_explorer(entry_path)`

Show given entry in Windows Explorer.

Parameters `entry_path` (*str*) – directory or file path

`pyrevit.coreutils.timestamp()`

Return timestamp for current time.

Returns timestamp in string format

Return type `str`

Example

```
>>> timestamp()
'01003075032506808'
```

`pyrevit.coreutils.touch(fname, times=None)`

Update the timestamp on the given file.

Parameters

- **fname** (*str*) – target file path
- **times** (*int*) – number of times to touch the file

`pyrevit.coreutils.unc_to_dletter(unc_path)`

Convert UNC path into drive letter path.

Parameters `unc_path` (*str*) – UNC path

Returns drive letter path

Return type `str`

Example

```
>>> # assuming J: is mapped to //filestore/server/jdrive
>>> unc_to_dletter('//filestore/server/jdrive/somefile.txt')
'J:/somefile.txt'
```

`pyrevit.coreutils.verify_directory` (*folder*)

Check if the folder exists and if not create the folder.

Parameters `folder` (*str*) – path of folder to verify

Returns path of verified folder, equals to provided folder

Return type `str`

Raises `OSError` on folder creation error.

15.2 pyrevit.coreutils.appdata

Utility functions for creating data files within pyRevit environment.

Most times, scripts need to save some data to shared between different scripts that work on a similar topic or between script executions. This module provides the necessary and consistent mechanism for creating and maintaining such files.

Example

```
>>> from pyrevit.coreutils import appdata
>>> appdata.list_data_files()
```

`pyrevit.coreutils.appdata.cleanup_appdata_folder` ()
Cleanup appdata folder of all temporary appdata files.

`pyrevit.coreutils.appdata.garbage_data_file` (*file_path*)
Mark and remove the given appdata file.

Current implementation removes the file immediately.

Parameters `file_path` (*str*) – path to the target file

`pyrevit.coreutils.appdata.get_data_file` (*file_id*, *file_ext*, *name_only=False*)
Get path to file that will not be cleaned up at Revit load.

e.g `pyrevit_2016_eirannejad_file_id.file_ext`

Parameters

- `file_id` (*str*) – Unique identifier for the file
- `file_ext` (*str*) – File extension
- `name_only` (*bool*) – If true, function returns file name only

Returns File name or full file path (depending on `name_only`)

Return type `str`

`pyrevit.coreutils.appdata.get_instance_data_file` (*file_id*, *file_ext='tmp'*, *name_only=False*)

Get path to file that should be used by current instance only.

These data files will be cleaned up at Revit restart. e.g `pyrevit_2016_eirannejad_2353_file_id.file_ext`

Parameters

- `file_id` (*str*) – Unique identifier for the file
- `file_ext` (*str*) – File extension

- **name_only** (*bool*) – If true, function returns file name only

Returns File name or full file path (depending on name_only)

Return type str

`pyrevit.coreutils.appdata.get_universal_data_file(file_id, file_ext, name_only=False)`

Get path to file that is shared between all host versions.

These data files are not cleaned up at Revit restart. e.g `pyrevit_eirannejad_file_id.file_ext`

Parameters

- **file_id** (*str*) – Unique identifier for the file
- **file_ext** (*str*) – File extension
- **name_only** (*bool*) – If true, function returns file name only

Returns File name or full file path (depending on name_only)

Return type str

`pyrevit.coreutils.appdata.is_data_file_available(file_id, file_ext)`

Check if given file is available within appdata directory.

Parameters

- **file_id** (*str*) – data file id
- **file_ext** (*str*) – file extension

Returns file path if file is available

Return type str

`pyrevit.coreutils.appdata.is_file_available(file_name, file_ext, universal=False)`

Check if given file is available within appdata directory.

Parameters

- **file_name** (*str*) – file name
- **file_ext** (*str*) – file extension
- **universal** (*bool*) – Check against universal data files

Returns file path if file is available

Return type str

`pyrevit.coreutils.appdata.is_pyrevit_data_file(file_name)`

Check if given file is a pyRevit data file.

Parameters **file_name** (*str*) – file name

Returns True if file is a pyRevit data file

Return type bool

`pyrevit.coreutils.appdata.list_data_files(file_ext, universal=False)`

List all data files with given extension.

Parameters

- **file_ext** (*str*) – file extension
- **universal** (*bool*) – Check against universal data files

Returns list of files

Return type list

`pyrevit.coreutils.appdata.list_session_data_files` (*file_ext*)

List all data files associated with current session.

Parameters `file_ext` (*str*) – data files with this extension will be listed only.

Returns list of data files

Return type list

15.3 pyrevit.coreutils.charts

Charts engine for output window

class `pyrevit.coreutils.charts.PyRevitOutputChart` (*output*, *chart_type*='line')

Bases: object

Chart wrapper object for output window.

output

output window wrapper object

Type `pyrevit.output.PyRevitOutputWindow`

chart_type

chart type name

Type str

draw ()

Request chart to draw itself on output window.

randomize_colors ()

Randomize chart datasets colors.

set_height (*height*)

Set chart height on output window.

set_style (*html_style*)

Set chart styling.

Parameters `html_style` (*str*) – inline html css styling string

Example

```
>>> chart.set_style('height:150px')
```

set_width (*width*)

Set chart width on output window.

class `pyrevit.coreutils.charts.PyRevitOutputChartData`

Bases: object

Chart data wrapper object.

new_dataset (*dataset_label*)

Create new data set.

Parameters `dataset_label` (*str*) – dataset label

Returns dataset wrapper object

Return type *PyRevitOutputChartDataset*

Example

```
>>> chart.data.new_dataset('set_a')
```

class `pyrevit.coreutils.charts.PyRevitOutputChartDataset` (*label*)

Bases: object

Chart dataset wrapper object.

set_color (**args*)

Set dataset color.

Arguments are expected to be R, G, B, A values.

Example

```
>>> dataset_obj.set_color(0xFF, 0x8C, 0x8D, 0.8)
```

class `pyrevit.coreutils.charts.PyRevitOutputChartOptions`

Bases: object

Chart options wrapper object.

15.4 pyrevit.coreutils.colors

Provide RGB color constants and a colors dictionary with elements formatted: `COLORS[colername] = CONSTANT`

Example

```
>>> from pyrevit.coreutils import colors
>>> colors.COLORS['black']
... <RGB #000000>
>>> colors.BLACK
... <RGB #000000>
```

class `pyrevit.coreutils.colors.RGB` (*name='default', red=0, green=0, blue=0*)

RGB named color object.

name

color name

Type str

red

value for red component (0-255)

Type int

green

value for green component (0-255)

Type int**blue**

value for blue component (0-255)

Type int**hex_color**

Return color in hex format

luminance

Return color luminance (perceived)

safe_text_color

Return text color that is safe to overlap this color.

15.4.1 Pre-Defined Colors

```

# color consts
ALICEBLUE = RGB(name='aliceblue', red=240, green=248, blue=255)
ANTIQUUEWHITE = RGB(name='antiquuewhite', red=250, green=235, blue=215)
ANTIQUUEWHITE1 = RGB(name='antiquuewhite1', red=255, green=239, blue=219)
ANTIQUUEWHITE2 = RGB(name='antiquuewhite2', red=238, green=223, blue=204)
ANTIQUUEWHITE3 = RGB(name='antiquuewhite3', red=205, green=192, blue=176)
ANTIQUUEWHITE4 = RGB(name='antiquuewhite4', red=139, green=131, blue=120)
AQUA = RGB(name='aqua', red=0, green=255, blue=255)
AQUAMARINE1 = RGB(name='aquamarine1', red=127, green=255, blue=212)
AQUAMARINE2 = RGB(name='aquamarine2', red=118, green=238, blue=198)
AQUAMARINE3 = RGB(name='aquamarine3', red=102, green=205, blue=170)
AQUAMARINE4 = RGB(name='aquamarine4', red=69, green=139, blue=116)
AZURE1 = RGB(name='azure1', red=240, green=255, blue=255)
AZURE2 = RGB(name='azure2', red=224, green=238, blue=238)
AZURE3 = RGB(name='azure3', red=193, green=205, blue=205)
AZURE4 = RGB(name='azure4', red=131, green=139, blue=139)
BANANA = RGB(name='banana', red=227, green=207, blue=87)
BEIGE = RGB(name='beige', red=245, green=245, blue=220)
BISQUE1 = RGB(name='bisque1', red=255, green=228, blue=196)
BISQUE2 = RGB(name='bisque2', red=238, green=213, blue=183)
BISQUE3 = RGB(name='bisque3', red=205, green=183, blue=158)
BISQUE4 = RGB(name='bisque4', red=139, green=125, blue=107)
BLACK = RGB(name='black', red=0, green=0, blue=0)
BLANCHEDALMOND = RGB(name='blanchedalmond', red=255, green=235, blue=205)
BLUE = RGB(name='blue', red=0, green=0, blue=255)
BLUE2 = RGB(name='blue2', red=0, green=0, blue=238)
BLUE3 = RGB(name='blue3', red=0, green=0, blue=205)
BLUE4 = RGB(name='blue4', red=0, green=0, blue=139)
BLUEVIOLET = RGB(name='blueviolet', red=138, green=43, blue=226)
BRICK = RGB(name='brick', red=156, green=102, blue=31)
BROWN = RGB(name='brown', red=165, green=42, blue=42)
BROWN1 = RGB(name='brown1', red=255, green=64, blue=64)
BROWN2 = RGB(name='brown2', red=238, green=59, blue=59)
BROWN3 = RGB(name='brown3', red=205, green=51, blue=51)
BROWN4 = RGB(name='brown4', red=139, green=35, blue=35)
BURLYWOOD = RGB(name='burlywood', red=222, green=184, blue=135)
BURLYWOOD1 = RGB(name='burlywood1', red=255, green=211, blue=155)
BURLYWOOD2 = RGB(name='burlywood2', red=238, green=197, blue=145)
BURLYWOOD3 = RGB(name='burlywood3', red=205, green=170, blue=125)
BURLYWOOD4 = RGB(name='burlywood4', red=139, green=115, blue=85)

```

(continues on next page)

(continued from previous page)

```

BURNTSIENNA = RGB(name='burntsienna', red=138, green=54, blue=15)
BURNTUMBER = RGB(name='burntumber', red=138, green=51, blue=36)
CADETBLUE = RGB(name='cadetblue', red=95, green=158, blue=160)
CADETBLUE1 = RGB(name='cadetblue1', red=152, green=245, blue=255)
CADETBLUE2 = RGB(name='cadetblue2', red=142, green=229, blue=238)
CADETBLUE3 = RGB(name='cadetblue3', red=122, green=197, blue=205)
CADETBLUE4 = RGB(name='cadetblue4', red=83, green=134, blue=139)
CADMIUMORANGE = RGB(name='cadmiumorange', red=255, green=97, blue=3)
CADMIUMYELLOW = RGB(name='cadmiumyellow', red=255, green=153, blue=18)
CARROT = RGB(name='carrot', red=237, green=145, blue=33)
CHARTREUSE1 = RGB(name='chartreuse1', red=127, green=255, blue=0)
CHARTREUSE2 = RGB(name='chartreuse2', red=118, green=238, blue=0)
CHARTREUSE3 = RGB(name='chartreuse3', red=102, green=205, blue=0)
CHARTREUSE4 = RGB(name='chartreuse4', red=69, green=139, blue=0)
CHOCOLATE = RGB(name='chocolate', red=210, green=105, blue=30)
CHOCOLATE1 = RGB(name='chocolate1', red=255, green=127, blue=36)
CHOCOLATE2 = RGB(name='chocolate2', red=238, green=118, blue=33)
CHOCOLATE3 = RGB(name='chocolate3', red=205, green=102, blue=29)
CHOCOLATE4 = RGB(name='chocolate4', red=139, green=69, blue=19)
COBALT = RGB(name='cobalt', red=61, green=89, blue=171)
COBALTGREEN = RGB(name='cobaltgreen', red=61, green=145, blue=64)
COLDGREY = RGB(name='coldgrey', red=128, green=138, blue=135)
CORAL = RGB(name='coral', red=255, green=127, blue=80)
CORAL1 = RGB(name='coral1', red=255, green=114, blue=86)
CORAL2 = RGB(name='coral2', red=238, green=106, blue=80)
CORAL3 = RGB(name='coral3', red=205, green=91, blue=69)
CORAL4 = RGB(name='coral4', red=139, green=62, blue=47)
CORNFLOWERBLUE = RGB(name='cornflowerblue', red=100, green=149, blue=237)
CORNSILK1 = RGB(name='cornsilk1', red=255, green=248, blue=220)
CORNSILK2 = RGB(name='cornsilk2', red=238, green=232, blue=205)
CORNSILK3 = RGB(name='cornsilk3', red=205, green=200, blue=177)
CORNSILK4 = RGB(name='cornsilk4', red=139, green=136, blue=120)
CRIMSON = RGB(name='crimson', red=220, green=20, blue=60)
CYAN2 = RGB(name='cyan2', red=0, green=238, blue=238)
CYAN3 = RGB(name='cyan3', red=0, green=205, blue=205)
CYAN4 = RGB(name='cyan4', red=0, green=139, blue=139)
DARKGOLDENROD = RGB(name='darkgoldenrod', red=184, green=134, blue=11)
DARKGOLDENROD1 = RGB(name='darkgoldenrod1', red=255, green=185, blue=15)
DARKGOLDENROD2 = RGB(name='darkgoldenrod2', red=238, green=173, blue=14)
DARKGOLDENROD3 = RGB(name='darkgoldenrod3', red=205, green=149, blue=12)
DARKGOLDENROD4 = RGB(name='darkgoldenrod4', red=139, green=101, blue=8)
DARKGRAY = RGB(name='darkgray', red=169, green=169, blue=169)
DARKGREEN = RGB(name='darkgreen', red=0, green=100, blue=0)
DARKKHAKI = RGB(name='darkkhaki', red=189, green=183, blue=107)
DARKOLIVEGREEN = RGB(name='darkolivegreen', red=85, green=107, blue=47)
DARKOLIVEGREEN1 = RGB(name='darkolivegreen1', red=202, green=255, blue=112)
DARKOLIVEGREEN2 = RGB(name='darkolivegreen2', red=188, green=238, blue=104)
DARKOLIVEGREEN3 = RGB(name='darkolivegreen3', red=162, green=205, blue=90)
DARKOLIVEGREEN4 = RGB(name='darkolivegreen4', red=110, green=139, blue=61)
DARKORANGE = RGB(name='darkorange', red=255, green=140, blue=0)
DARKORANGE1 = RGB(name='darkorange1', red=255, green=127, blue=0)
DARKORANGE2 = RGB(name='darkorange2', red=238, green=118, blue=0)
DARKORANGE3 = RGB(name='darkorange3', red=205, green=102, blue=0)
DARKORANGE4 = RGB(name='darkorange4', red=139, green=69, blue=0)
DARKORCHID = RGB(name='darkorchid', red=153, green=50, blue=204)
DARKORCHID1 = RGB(name='darkorchid1', red=191, green=62, blue=255)
DARKORCHID2 = RGB(name='darkorchid2', red=178, green=58, blue=238)

```

(continues on next page)

(continued from previous page)

```
DARKORCHID3 = RGB(name='darkorchid3', red=154, green=50, blue=205)
DARKORCHID4 = RGB(name='darkorchid4', red=104, green=34, blue=139)
DARKSALMON = RGB(name='darksalmon', red=233, green=150, blue=122)
DARKSEAGREEN = RGB(name='darkseagreen', red=143, green=188, blue=143)
DARKSEAGREEN1 = RGB(name='darkseagreen1', red=193, green=255, blue=193)
DARKSEAGREEN2 = RGB(name='darkseagreen2', red=180, green=238, blue=180)
DARKSEAGREEN3 = RGB(name='darkseagreen3', red=155, green=205, blue=155)
DARKSEAGREEN4 = RGB(name='darkseagreen4', red=105, green=139, blue=105)
DARKSLATEBLUE = RGB(name='darkslateblue', red=72, green=61, blue=139)
DARKSLATEGRAY = RGB(name='darkslategray', red=47, green=79, blue=79)
DARKSLATEGRAY1 = RGB(name='darkslategray1', red=151, green=255, blue=255)
DARKSLATEGRAY2 = RGB(name='darkslategray2', red=141, green=238, blue=238)
DARKSLATEGRAY3 = RGB(name='darkslategray3', red=121, green=205, blue=205)
DARKSLATEGRAY4 = RGB(name='darkslategray4', red=82, green=139, blue=139)
DARKTURQUOISE = RGB(name='darkturquoise', red=0, green=206, blue=209)
DARKVIOLET = RGB(name='darkviolet', red=148, green=0, blue=211)
DEEPPINK1 = RGB(name='deeppink1', red=255, green=20, blue=147)
DEEPPINK2 = RGB(name='deeppink2', red=238, green=18, blue=137)
DEEPPINK3 = RGB(name='deeppink3', red=205, green=16, blue=118)
DEEPPINK4 = RGB(name='deeppink4', red=139, green=10, blue=80)
DEEPSKYBLUE1 = RGB(name='deepskyblue1', red=0, green=191, blue=255)
DEEPSKYBLUE2 = RGB(name='deepskyblue2', red=0, green=178, blue=238)
DEEPSKYBLUE3 = RGB(name='deepskyblue3', red=0, green=154, blue=205)
DEEPSKYBLUE4 = RGB(name='deepskyblue4', red=0, green=104, blue=139)
DIMGRAY = RGB(name='dimgray', red=105, green=105, blue=105)
DIMGRAY = RGB(name='dimgray', red=105, green=105, blue=105)
DODGERBLUE1 = RGB(name='dodgerblue1', red=30, green=144, blue=255)
DODGERBLUE2 = RGB(name='dodgerblue2', red=28, green=134, blue=238)
DODGERBLUE3 = RGB(name='dodgerblue3', red=24, green=116, blue=205)
DODGERBLUE4 = RGB(name='dodgerblue4', red=16, green=78, blue=139)
EGGSHELL = RGB(name='eggshell', red=252, green=230, blue=201)
EMERALDGREEN = RGB(name='emeraldgreen', red=0, green=201, blue=87)
FIREBRICK = RGB(name='firebrick', red=178, green=34, blue=34)
FIREBRICK1 = RGB(name='firebrick1', red=255, green=48, blue=48)
FIREBRICK2 = RGB(name='firebrick2', red=238, green=44, blue=44)
FIREBRICK3 = RGB(name='firebrick3', red=205, green=38, blue=38)
FIREBRICK4 = RGB(name='firebrick4', red=139, green=26, blue=26)
FLESH = RGB(name='flesh', red=255, green=125, blue=64)
FLORALWHITE = RGB(name='floralwhite', red=255, green=250, blue=240)
FORESTGREEN = RGB(name='forestgreen', red=34, green=139, blue=34)
GAINSBORO = RGB(name='gainsboro', red=220, green=220, blue=220)
GHOSTWHITE = RGB(name='ghostwhite', red=248, green=248, blue=255)
GOLD1 = RGB(name='gold1', red=255, green=215, blue=0)
GOLD2 = RGB(name='gold2', red=238, green=201, blue=0)
GOLD3 = RGB(name='gold3', red=205, green=173, blue=0)
GOLD4 = RGB(name='gold4', red=139, green=117, blue=0)
GOLDENROD = RGB(name='goldenrod', red=218, green=165, blue=32)
GOLDENROD1 = RGB(name='goldenrod1', red=255, green=193, blue=37)
GOLDENROD2 = RGB(name='goldenrod2', red=238, green=180, blue=34)
GOLDENROD3 = RGB(name='goldenrod3', red=205, green=155, blue=29)
GOLDENROD4 = RGB(name='goldenrod4', red=139, green=105, blue=20)
GRAY = RGB(name='gray', red=128, green=128, blue=128)
GRAY1 = RGB(name='gray1', red=3, green=3, blue=3)
GRAY10 = RGB(name='gray10', red=26, green=26, blue=26)
GRAY11 = RGB(name='gray11', red=28, green=28, blue=28)
GRAY12 = RGB(name='gray12', red=31, green=31, blue=31)
GRAY13 = RGB(name='gray13', red=33, green=33, blue=33)
```

(continues on next page)

(continued from previous page)

```
GRAY14 = RGB(name='gray14', red=36, green=36, blue=36)
GRAY15 = RGB(name='gray15', red=38, green=38, blue=38)
GRAY16 = RGB(name='gray16', red=41, green=41, blue=41)
GRAY17 = RGB(name='gray17', red=43, green=43, blue=43)
GRAY18 = RGB(name='gray18', red=46, green=46, blue=46)
GRAY19 = RGB(name='gray19', red=48, green=48, blue=48)
GRAY2 = RGB(name='gray2', red=5, green=5, blue=5)
GRAY20 = RGB(name='gray20', red=51, green=51, blue=51)
GRAY21 = RGB(name='gray21', red=54, green=54, blue=54)
GRAY22 = RGB(name='gray22', red=56, green=56, blue=56)
GRAY23 = RGB(name='gray23', red=59, green=59, blue=59)
GRAY24 = RGB(name='gray24', red=61, green=61, blue=61)
GRAY25 = RGB(name='gray25', red=64, green=64, blue=64)
GRAY26 = RGB(name='gray26', red=66, green=66, blue=66)
GRAY27 = RGB(name='gray27', red=69, green=69, blue=69)
GRAY28 = RGB(name='gray28', red=71, green=71, blue=71)
GRAY29 = RGB(name='gray29', red=74, green=74, blue=74)
GRAY3 = RGB(name='gray3', red=8, green=8, blue=8)
GRAY30 = RGB(name='gray30', red=77, green=77, blue=77)
GRAY31 = RGB(name='gray31', red=79, green=79, blue=79)
GRAY32 = RGB(name='gray32', red=82, green=82, blue=82)
GRAY33 = RGB(name='gray33', red=84, green=84, blue=84)
GRAY34 = RGB(name='gray34', red=87, green=87, blue=87)
GRAY35 = RGB(name='gray35', red=89, green=89, blue=89)
GRAY36 = RGB(name='gray36', red=92, green=92, blue=92)
GRAY37 = RGB(name='gray37', red=94, green=94, blue=94)
GRAY38 = RGB(name='gray38', red=97, green=97, blue=97)
GRAY39 = RGB(name='gray39', red=99, green=99, blue=99)
GRAY4 = RGB(name='gray4', red=10, green=10, blue=10)
GRAY40 = RGB(name='gray40', red=102, green=102, blue=102)
GRAY42 = RGB(name='gray42', red=107, green=107, blue=107)
GRAY43 = RGB(name='gray43', red=110, green=110, blue=110)
GRAY44 = RGB(name='gray44', red=112, green=112, blue=112)
GRAY45 = RGB(name='gray45', red=115, green=115, blue=115)
GRAY46 = RGB(name='gray46', red=117, green=117, blue=117)
GRAY47 = RGB(name='gray47', red=120, green=120, blue=120)
GRAY48 = RGB(name='gray48', red=122, green=122, blue=122)
GRAY49 = RGB(name='gray49', red=125, green=125, blue=125)
GRAY5 = RGB(name='gray5', red=13, green=13, blue=13)
GRAY50 = RGB(name='gray50', red=127, green=127, blue=127)
GRAY51 = RGB(name='gray51', red=130, green=130, blue=130)
GRAY52 = RGB(name='gray52', red=133, green=133, blue=133)
GRAY53 = RGB(name='gray53', red=135, green=135, blue=135)
GRAY54 = RGB(name='gray54', red=138, green=138, blue=138)
GRAY55 = RGB(name='gray55', red=140, green=140, blue=140)
GRAY56 = RGB(name='gray56', red=143, green=143, blue=143)
GRAY57 = RGB(name='gray57', red=145, green=145, blue=145)
GRAY58 = RGB(name='gray58', red=148, green=148, blue=148)
GRAY59 = RGB(name='gray59', red=150, green=150, blue=150)
GRAY6 = RGB(name='gray6', red=15, green=15, blue=15)
GRAY60 = RGB(name='gray60', red=153, green=153, blue=153)
GRAY61 = RGB(name='gray61', red=156, green=156, blue=156)
GRAY62 = RGB(name='gray62', red=158, green=158, blue=158)
GRAY63 = RGB(name='gray63', red=161, green=161, blue=161)
GRAY64 = RGB(name='gray64', red=163, green=163, blue=163)
GRAY65 = RGB(name='gray65', red=166, green=166, blue=166)
GRAY66 = RGB(name='gray66', red=168, green=168, blue=168)
```

(continues on next page)

(continued from previous page)

```
GRAY67 = RGB(name='gray67', red=171, green=171, blue=171)
GRAY68 = RGB(name='gray68', red=173, green=173, blue=173)
GRAY69 = RGB(name='gray69', red=176, green=176, blue=176)
GRAY7 = RGB(name='gray7', red=18, green=18, blue=18)
GRAY70 = RGB(name='gray70', red=179, green=179, blue=179)
GRAY71 = RGB(name='gray71', red=181, green=181, blue=181)
GRAY72 = RGB(name='gray72', red=184, green=184, blue=184)
GRAY73 = RGB(name='gray73', red=186, green=186, blue=186)
GRAY74 = RGB(name='gray74', red=189, green=189, blue=189)
GRAY75 = RGB(name='gray75', red=191, green=191, blue=191)
GRAY76 = RGB(name='gray76', red=194, green=194, blue=194)
GRAY77 = RGB(name='gray77', red=196, green=196, blue=196)
GRAY78 = RGB(name='gray78', red=199, green=199, blue=199)
GRAY79 = RGB(name='gray79', red=201, green=201, blue=201)
GRAY8 = RGB(name='gray8', red=20, green=20, blue=20)
GRAY80 = RGB(name='gray80', red=204, green=204, blue=204)
GRAY81 = RGB(name='gray81', red=207, green=207, blue=207)
GRAY82 = RGB(name='gray82', red=209, green=209, blue=209)
GRAY83 = RGB(name='gray83', red=212, green=212, blue=212)
GRAY84 = RGB(name='gray84', red=214, green=214, blue=214)
GRAY85 = RGB(name='gray85', red=217, green=217, blue=217)
GRAY86 = RGB(name='gray86', red=219, green=219, blue=219)
GRAY87 = RGB(name='gray87', red=222, green=222, blue=222)
GRAY88 = RGB(name='gray88', red=224, green=224, blue=224)
GRAY89 = RGB(name='gray89', red=227, green=227, blue=227)
GRAY9 = RGB(name='gray9', red=23, green=23, blue=23)
GRAY90 = RGB(name='gray90', red=229, green=229, blue=229)
GRAY91 = RGB(name='gray91', red=232, green=232, blue=232)
GRAY92 = RGB(name='gray92', red=235, green=235, blue=235)
GRAY93 = RGB(name='gray93', red=237, green=237, blue=237)
GRAY94 = RGB(name='gray94', red=240, green=240, blue=240)
GRAY95 = RGB(name='gray95', red=242, green=242, blue=242)
GRAY97 = RGB(name='gray97', red=247, green=247, blue=247)
GRAY98 = RGB(name='gray98', red=250, green=250, blue=250)
GRAY99 = RGB(name='gray99', red=252, green=252, blue=252)
GREEN = RGB(name='green', red=0, green=128, blue=0)
GREEN1 = RGB(name='green1', red=0, green=255, blue=0)
GREEN2 = RGB(name='green2', red=0, green=238, blue=0)
GREEN3 = RGB(name='green3', red=0, green=205, blue=0)
GREEN4 = RGB(name='green4', red=0, green=139, blue=0)
GREENYELLOW = RGB(name='greenyellow', red=173, green=255, blue=47)
HONEYDEW1 = RGB(name='honeydew1', red=240, green=255, blue=240)
HONEYDEW2 = RGB(name='honeydew2', red=224, green=238, blue=224)
HONEYDEW3 = RGB(name='honeydew3', red=193, green=205, blue=193)
HONEYDEW4 = RGB(name='honeydew4', red=131, green=139, blue=131)
HOTPINK = RGB(name='hotpink', red=255, green=105, blue=180)
HOTPINK1 = RGB(name='hotpink1', red=255, green=110, blue=180)
HOTPINK2 = RGB(name='hotpink2', red=238, green=106, blue=167)
HOTPINK3 = RGB(name='hotpink3', red=205, green=96, blue=144)
HOTPINK4 = RGB(name='hotpink4', red=139, green=58, blue=98)
INDIANRED = RGB(name='indianred', red=176, green=23, blue=31)
INDIANRED = RGB(name='indianred', red=205, green=92, blue=92)
INDIANRED1 = RGB(name='indianred1', red=255, green=106, blue=106)
INDIANRED2 = RGB(name='indianred2', red=238, green=99, blue=99)
INDIANRED3 = RGB(name='indianred3', red=205, green=85, blue=85)
INDIANRED4 = RGB(name='indianred4', red=139, green=58, blue=58)
INDIGO = RGB(name='indigo', red=75, green=0, blue=130)
```

(continues on next page)

(continued from previous page)

```

IVORY1 = RGB(name='ivory1', red=255, green=255, blue=240)
IVORY2 = RGB(name='ivory2', red=238, green=238, blue=224)
IVORY3 = RGB(name='ivory3', red=205, green=205, blue=193)
IVORY4 = RGB(name='ivory4', red=139, green=139, blue=131)
IVORYBLACK = RGB(name='ivoryblack', red=41, green=36, blue=33)
KHAKI = RGB(name='khaki', red=240, green=230, blue=140)
KHAKI1 = RGB(name='khaki1', red=255, green=246, blue=143)
KHAKI2 = RGB(name='khaki2', red=238, green=230, blue=133)
KHAKI3 = RGB(name='khaki3', red=205, green=198, blue=115)
KHAKI4 = RGB(name='khaki4', red=139, green=134, blue=78)
LAVENDER = RGB(name='lavender', red=230, green=230, blue=250)
LAVENDERBLUSH1 = RGB(name='lavenderblush1', red=255, green=240, blue=245)
LAVENDERBLUSH2 = RGB(name='lavenderblush2', red=238, green=224, blue=229)
LAVENDERBLUSH3 = RGB(name='lavenderblush3', red=205, green=193, blue=197)
LAVENDERBLUSH4 = RGB(name='lavenderblush4', red=139, green=131, blue=134)
LAWNGREEN = RGB(name='lawngreen', red=124, green=252, blue=0)
LEMONCHIFFON1 = RGB(name='lemonchiffon1', red=255, green=250, blue=205)
LEMONCHIFFON2 = RGB(name='lemonchiffon2', red=238, green=233, blue=191)
LEMONCHIFFON3 = RGB(name='lemonchiffon3', red=205, green=201, blue=165)
LEMONCHIFFON4 = RGB(name='lemonchiffon4', red=139, green=137, blue=112)
LIGHTBLUE = RGB(name='lightblue', red=173, green=216, blue=230)
LIGHTBLUE1 = RGB(name='lightblue1', red=191, green=239, blue=255)
LIGHTBLUE2 = RGB(name='lightblue2', red=178, green=223, blue=238)
LIGHTBLUE3 = RGB(name='lightblue3', red=154, green=192, blue=205)
LIGHTBLUE4 = RGB(name='lightblue4', red=104, green=131, blue=139)
LIGHTCORAL = RGB(name='lightcoral', red=240, green=128, blue=128)
LIGHTCYAN1 = RGB(name='lightcyan1', red=224, green=255, blue=255)
LIGHTCYAN2 = RGB(name='lightcyan2', red=209, green=238, blue=238)
LIGHTCYAN3 = RGB(name='lightcyan3', red=180, green=205, blue=205)
LIGHTCYAN4 = RGB(name='lightcyan4', red=122, green=139, blue=139)
LIGHTGOLDENROD1 = RGB(name='lightgoldenrod1', red=255, green=236, blue=139)
LIGHTGOLDENROD2 = RGB(name='lightgoldenrod2', red=238, green=220, blue=130)
LIGHTGOLDENROD3 = RGB(name='lightgoldenrod3', red=205, green=190, blue=112)
LIGHTGOLDENROD4 = RGB(name='lightgoldenrod4', red=139, green=129, blue=76)
LIGHTGOLDENRODYELLOW = \
    RGB(name='lightgoldenrodyellow', red=250, green=250, blue=210)
LIGHTGREY = RGB(name='lightgrey', red=211, green=211, blue=211)
LIGHTPINK = RGB(name='lightpink', red=255, green=182, blue=193)
LIGHTPINK1 = RGB(name='lightpink1', red=255, green=174, blue=185)
LIGHTPINK2 = RGB(name='lightpink2', red=238, green=162, blue=173)
LIGHTPINK3 = RGB(name='lightpink3', red=205, green=140, blue=149)
LIGHTPINK4 = RGB(name='lightpink4', red=139, green=95, blue=101)
LIGHTSALMON1 = RGB(name='lightsalmon1', red=255, green=160, blue=122)
LIGHTSALMON2 = RGB(name='lightsalmon2', red=238, green=149, blue=114)
LIGHTSALMON3 = RGB(name='lightsalmon3', red=205, green=129, blue=98)
LIGHTSALMON4 = RGB(name='lightsalmon4', red=139, green=87, blue=66)
LIGHTSEAGREEN = RGB(name='lightseagreen', red=32, green=178, blue=170)
LIGHTSKYBLUE = RGB(name='lightskyblue', red=135, green=206, blue=250)
LIGHTSKYBLUE1 = RGB(name='lightskyblue1', red=176, green=226, blue=255)
LIGHTSKYBLUE2 = RGB(name='lightskyblue2', red=164, green=211, blue=238)
LIGHTSKYBLUE3 = RGB(name='lightskyblue3', red=141, green=182, blue=205)
LIGHTSKYBLUE4 = RGB(name='lightskyblue4', red=96, green=123, blue=139)
LIGHTSLATEBLUE = RGB(name='lightslateblue', red=132, green=112, blue=255)
LIGHTSLATEGRAY = RGB(name='lightslategray', red=119, green=136, blue=153)
LIGHTSTEELBLUE = RGB(name='lightsteelblue', red=176, green=196, blue=222)
LIGHTSTEELBLUE1 = RGB(name='lightsteelblue1', red=202, green=225, blue=255)
LIGHTSTEELBLUE2 = RGB(name='lightsteelblue2', red=188, green=210, blue=238)

```

(continues on next page)

(continued from previous page)

```

LIGHTSTEELBLUE3 = RGB(name='lightsteelblue3', red=162, green=181, blue=205)
LIGHTSTEELBLUE4 = RGB(name='lightsteelblue4', red=110, green=123, blue=139)
LIGHTYELLOW1 = RGB(name='lightyellow1', red=255, green=255, blue=224)
LIGHTYELLOW2 = RGB(name='lightyellow2', red=238, green=238, blue=209)
LIGHTYELLOW3 = RGB(name='lightyellow3', red=205, green=205, blue=180)
LIGHTYELLOW4 = RGB(name='lightyellow4', red=139, green=139, blue=122)
LIMEGREEN = RGB(name='limegreen', red=50, green=205, blue=50)
LINEN = RGB(name='linen', red=250, green=240, blue=230)
MAGENTA = RGB(name='magenta', red=255, green=0, blue=255)
MAGENTA2 = RGB(name='magenta2', red=238, green=0, blue=238)
MAGENTA3 = RGB(name='magenta3', red=205, green=0, blue=205)
MAGENTA4 = RGB(name='magenta4', red=139, green=0, blue=139)
MANGANESEBLUE = RGB(name='manganeblue', red=3, green=168, blue=158)
MAROON = RGB(name='maroon', red=128, green=0, blue=0)
MAROON1 = RGB(name='maroon1', red=255, green=52, blue=179)
MAROON2 = RGB(name='maroon2', red=238, green=48, blue=167)
MAROON3 = RGB(name='maroon3', red=205, green=41, blue=144)
MAROON4 = RGB(name='maroon4', red=139, green=28, blue=98)
MEDIUMORCHID = RGB(name='mediumorchid', red=186, green=85, blue=211)
MEDIUMORCHID1 = RGB(name='mediumorchid1', red=224, green=102, blue=255)
MEDIUMORCHID2 = RGB(name='mediumorchid2', red=209, green=95, blue=238)
MEDIUMORCHID3 = RGB(name='mediumorchid3', red=180, green=82, blue=205)
MEDIUMORCHID4 = RGB(name='mediumorchid4', red=122, green=55, blue=139)
MEDIUMPURPLE = RGB(name='mediumpurple', red=147, green=112, blue=219)
MEDIUMPURPLE1 = RGB(name='mediumpurple1', red=171, green=130, blue=255)
MEDIUMPURPLE2 = RGB(name='mediumpurple2', red=159, green=121, blue=238)
MEDIUMPURPLE3 = RGB(name='mediumpurple3', red=137, green=104, blue=205)
MEDIUMPURPLE4 = RGB(name='mediumpurple4', red=93, green=71, blue=139)
MEDIUMSEAGREEN = RGB(name='mediumseagreen', red=60, green=179, blue=113)
MEDIUMSLATEBLUE = RGB(name='mediumslateblue', red=123, green=104, blue=238)
MEDIUMSPRINGGREEN = RGB(name='mediumspringgreen', red=0, green=250, blue=154)
MEDIUMTURQUOISE = RGB(name='mediumturquoise', red=72, green=209, blue=204)
MEDIUMVIOLETRED = RGB(name='mediumvioletred', red=199, green=21, blue=133)
MELON = RGB(name='melon', red=227, green=168, blue=105)
MIDNIGHTBLUE = RGB(name='midnightblue', red=25, green=25, blue=112)
MINT = RGB(name='mint', red=189, green=252, blue=201)
MINTCREAM = RGB(name='mintcream', red=245, green=255, blue=250)
MISTYROSE1 = RGB(name='mistyrose1', red=255, green=228, blue=225)
MISTYROSE2 = RGB(name='mistyrose2', red=238, green=213, blue=210)
MISTYROSE3 = RGB(name='mistyrose3', red=205, green=183, blue=181)
MISTYROSE4 = RGB(name='mistyrose4', red=139, green=125, blue=123)
MOCCASIN = RGB(name='moccasin', red=255, green=228, blue=181)
NAVAJOWHITE1 = RGB(name='navajowhite1', red=255, green=222, blue=173)
NAVAJOWHITE2 = RGB(name='navajowhite2', red=238, green=207, blue=161)
NAVAJOWHITE3 = RGB(name='navajowhite3', red=205, green=179, blue=139)
NAVAJOWHITE4 = RGB(name='navajowhite4', red=139, green=121, blue=94)
NAVY = RGB(name='navy', red=0, green=0, blue=128)
OLDLACE = RGB(name='oldlace', red=253, green=245, blue=230)
OLIVE = RGB(name='olive', red=128, green=128, blue=0)
OLIVEDRAB = RGB(name='olivedrab', red=107, green=142, blue=35)
OLIVEDRAB1 = RGB(name='olivedrab1', red=192, green=255, blue=62)
OLIVEDRAB2 = RGB(name='olivedrab2', red=179, green=238, blue=58)
OLIVEDRAB3 = RGB(name='olivedrab3', red=154, green=205, blue=50)
OLIVEDRAB4 = RGB(name='olivedrab4', red=105, green=139, blue=34)
ORANGE = RGB(name='orange', red=255, green=128, blue=0)
ORANGE1 = RGB(name='orange1', red=255, green=165, blue=0)
ORANGE2 = RGB(name='orange2', red=238, green=154, blue=0)

```

(continues on next page)

(continued from previous page)

```

ORANGE3 = RGB(name='orange3', red=205, green=133, blue=0)
ORANGE4 = RGB(name='orange4', red=139, green=90, blue=0)
ORANGERED1 = RGB(name='orangered1', red=255, green=69, blue=0)
ORANGERED2 = RGB(name='orangered2', red=238, green=64, blue=0)
ORANGERED3 = RGB(name='orangered3', red=205, green=55, blue=0)
ORANGERED4 = RGB(name='orangered4', red=139, green=37, blue=0)
ORCHID = RGB(name='orchid', red=218, green=112, blue=214)
ORCHID1 = RGB(name='orchid1', red=255, green=131, blue=250)
ORCHID2 = RGB(name='orchid2', red=238, green=122, blue=233)
ORCHID3 = RGB(name='orchid3', red=205, green=105, blue=201)
ORCHID4 = RGB(name='orchid4', red=139, green=71, blue=137)
PALEGOLDENROD = RGB(name='palegoldenrod', red=238, green=232, blue=170)
PALEGREEN = RGB(name='palegreen', red=152, green=251, blue=152)
PALEGREEN1 = RGB(name='palegreen1', red=154, green=255, blue=154)
PALEGREEN2 = RGB(name='palegreen2', red=144, green=238, blue=144)
PALEGREEN3 = RGB(name='palegreen3', red=124, green=205, blue=124)
PALEGREEN4 = RGB(name='palegreen4', red=84, green=139, blue=84)
PALETURQUOISE1 = RGB(name='paleturquoise1', red=187, green=255, blue=255)
PALETURQUOISE2 = RGB(name='paleturquoise2', red=174, green=238, blue=238)
PALETURQUOISE3 = RGB(name='paleturquoise3', red=150, green=205, blue=205)
PALETURQUOISE4 = RGB(name='paleturquoise4', red=102, green=139, blue=139)
PALEVIOLETRED = RGB(name='palevioletred', red=219, green=112, blue=147)
PALEVIOLETRED1 = RGB(name='palevioletred1', red=255, green=130, blue=171)
PALEVIOLETRED2 = RGB(name='palevioletred2', red=238, green=121, blue=159)
PALEVIOLETRED3 = RGB(name='palevioletred3', red=205, green=104, blue=137)
PALEVIOLETRED4 = RGB(name='palevioletred4', red=139, green=71, blue=93)
PAPAYAWHIP = RGB(name='papayawhip', red=255, green=239, blue=213)
PEACHPUFF1 = RGB(name='peachpuff1', red=255, green=218, blue=185)
PEACHPUFF2 = RGB(name='peachpuff2', red=238, green=203, blue=173)
PEACHPUFF3 = RGB(name='peachpuff3', red=205, green=175, blue=149)
PEACHPUFF4 = RGB(name='peachpuff4', red=139, green=119, blue=101)
PEACOCK = RGB(name='peacock', red=51, green=161, blue=201)
PINK = RGB(name='pink', red=255, green=192, blue=203)
PINK1 = RGB(name='pink1', red=255, green=181, blue=197)
PINK2 = RGB(name='pink2', red=238, green=169, blue=184)
PINK3 = RGB(name='pink3', red=205, green=145, blue=158)
PINK4 = RGB(name='pink4', red=139, green=99, blue=108)
PLUM = RGB(name='plum', red=221, green=160, blue=221)
PLUM1 = RGB(name='plum1', red=255, green=187, blue=255)
PLUM2 = RGB(name='plum2', red=238, green=174, blue=238)
PLUM3 = RGB(name='plum3', red=205, green=150, blue=205)
PLUM4 = RGB(name='plum4', red=139, green=102, blue=139)
POWDERBLUE = RGB(name='powderblue', red=176, green=224, blue=230)
PURPLE = RGB(name='purple', red=128, green=0, blue=128)
PURPLE1 = RGB(name='purple1', red=155, green=48, blue=255)
PURPLE2 = RGB(name='purple2', red=145, green=44, blue=238)
PURPLE3 = RGB(name='purple3', red=125, green=38, blue=205)
PURPLE4 = RGB(name='purple4', red=85, green=26, blue=139)
RASPBERRY = RGB(name='raspberry', red=135, green=38, blue=87)
RAWSIENNA = RGB(name='rawsienna', red=199, green=97, blue=20)
RED1 = RGB(name='red1', red=255, green=0, blue=0)
RED2 = RGB(name='red2', red=238, green=0, blue=0)
RED3 = RGB(name='red3', red=205, green=0, blue=0)
RED4 = RGB(name='red4', red=139, green=0, blue=0)
ROSYBROWN = RGB(name='rosybrown', red=188, green=143, blue=143)
ROSYBROWN1 = RGB(name='rosybrown1', red=255, green=193, blue=193)
ROSYBROWN2 = RGB(name='rosybrown2', red=238, green=180, blue=180)

```

(continues on next page)

(continued from previous page)

```
ROSYBROWN3 = RGB(name='rosybrown3', red=205, green=155, blue=155)
ROSYBROWN4 = RGB(name='rosybrown4', red=139, green=105, blue=105)
ROYALBLUE = RGB(name='royalblue', red=65, green=105, blue=225)
ROYALBLUE1 = RGB(name='royalblue1', red=72, green=118, blue=255)
ROYALBLUE2 = RGB(name='royalblue2', red=67, green=110, blue=238)
ROYALBLUE3 = RGB(name='royalblue3', red=58, green=95, blue=205)
ROYALBLUE4 = RGB(name='royalblue4', red=39, green=64, blue=139)
SALMON = RGB(name='salmon', red=250, green=128, blue=114)
SALMON1 = RGB(name='salmon1', red=255, green=140, blue=105)
SALMON2 = RGB(name='salmon2', red=238, green=130, blue=98)
SALMON3 = RGB(name='salmon3', red=205, green=112, blue=84)
SALMON4 = RGB(name='salmon4', red=139, green=76, blue=57)
SANDYBROWN = RGB(name='sandybrown', red=244, green=164, blue=96)
SAPGREEN = RGB(name='sapgreen', red=48, green=128, blue=20)
SEAGREEN1 = RGB(name='seagreen1', red=84, green=255, blue=159)
SEAGREEN2 = RGB(name='seagreen2', red=78, green=238, blue=148)
SEAGREEN3 = RGB(name='seagreen3', red=67, green=205, blue=128)
SEAGREEN4 = RGB(name='seagreen4', red=46, green=139, blue=87)
SEASHELL1 = RGB(name='seashell1', red=255, green=245, blue=238)
SEASHELL2 = RGB(name='seashell2', red=238, green=229, blue=222)
SEASHELL3 = RGB(name='seashell3', red=205, green=197, blue=191)
SEASHELL4 = RGB(name='seashell4', red=139, green=134, blue=130)
SEPIA = RGB(name='sepia', red=94, green=38, blue=18)
SGIBEET = RGB(name='sgibeet', red=142, green=56, blue=142)
SGIBRIGHTGRAY = RGB(name='sgibrightgray', red=197, green=193, blue=170)
SGICHARTREUSE = RGB(name='sgichartreuse', red=113, green=198, blue=113)
SGIDARKGRAY = RGB(name='sgidarkgray', red=85, green=85, blue=85)
SGIGRAY12 = RGB(name='sgigray12', red=30, green=30, blue=30)
SGIGRAY16 = RGB(name='sgigray16', red=40, green=40, blue=40)
SGIGRAY32 = RGB(name='sgigray32', red=81, green=81, blue=81)
SGIGRAY36 = RGB(name='sgigray36', red=91, green=91, blue=91)
SGIGRAY52 = RGB(name='sgigray52', red=132, green=132, blue=132)
SGIGRAY56 = RGB(name='sgigray56', red=142, green=142, blue=142)
SGIGRAY72 = RGB(name='sgigray72', red=183, green=183, blue=183)
SGIGRAY76 = RGB(name='sgigray76', red=193, green=193, blue=193)
SGIGRAY92 = RGB(name='sgigray92', red=234, green=234, blue=234)
SGIGRAY96 = RGB(name='sgigray96', red=244, green=244, blue=244)
SGILIGHTBLUE = RGB(name='sgilightblue', red=125, green=158, blue=192)
SGILIGHTGRAY = RGB(name='sgilightgray', red=170, green=170, blue=170)
SGIOLIVEDRAB = RGB(name='sgiolivedrab', red=142, green=142, blue=56)
SGISALMON = RGB(name='sgisalmon', red=198, green=113, blue=113)
SGISLATEBLUE = RGB(name='sgislateblue', red=113, green=113, blue=198)
SGITEAL = RGB(name='sgiteal', red=56, green=142, blue=142)
SIENNA = RGB(name='sienna', red=160, green=82, blue=45)
SIENNA1 = RGB(name='sienna1', red=255, green=130, blue=71)
SIENNA2 = RGB(name='sienna2', red=238, green=121, blue=66)
SIENNA3 = RGB(name='sienna3', red=205, green=104, blue=57)
SIENNA4 = RGB(name='sienna4', red=139, green=71, blue=38)
SILVER = RGB(name='silver', red=192, green=192, blue=192)
SKYBLUE = RGB(name='skyblue', red=135, green=206, blue=235)
SKYBLUE1 = RGB(name='skyblue1', red=135, green=206, blue=255)
SKYBLUE2 = RGB(name='skyblue2', red=126, green=192, blue=238)
SKYBLUE3 = RGB(name='skyblue3', red=108, green=166, blue=205)
SKYBLUE4 = RGB(name='skyblue4', red=74, green=112, blue=139)
SLATEBLUE = RGB(name='slateblue', red=106, green=90, blue=205)
SLATEBLUE1 = RGB(name='slateblue1', red=131, green=111, blue=255)
SLATEBLUE2 = RGB(name='slateblue2', red=122, green=103, blue=238)
```

(continues on next page)

(continued from previous page)

```

SLATEBLUE3 = RGB(name='slateblue3', red=105, green=89, blue=205)
SLATEBLUE4 = RGB(name='slateblue4', red=71, green=60, blue=139)
SLATEGRAY = RGB(name='slategray', red=112, green=128, blue=144)
SLATEGRAY1 = RGB(name='slategray1', red=198, green=226, blue=255)
SLATEGRAY2 = RGB(name='slategray2', red=185, green=211, blue=238)
SLATEGRAY3 = RGB(name='slategray3', red=159, green=182, blue=205)
SLATEGRAY4 = RGB(name='slategray4', red=108, green=123, blue=139)
SNOW1 = RGB(name='snow1', red=255, green=250, blue=250)
SNOW2 = RGB(name='snow2', red=238, green=233, blue=233)
SNOW3 = RGB(name='snow3', red=205, green=201, blue=201)
SNOW4 = RGB(name='snow4', red=139, green=137, blue=137)
SPRINGGREEN = RGB(name='springgreen', red=0, green=255, blue=127)
SPRINGGREEN1 = RGB(name='springgreen1', red=0, green=238, blue=118)
SPRINGGREEN2 = RGB(name='springgreen2', red=0, green=205, blue=102)
SPRINGGREEN3 = RGB(name='springgreen3', red=0, green=139, blue=69)
STEELBLUE = RGB(name='steelblue', red=70, green=130, blue=180)
STEELBLUE1 = RGB(name='steelblue1', red=99, green=184, blue=255)
STEELBLUE2 = RGB(name='steelblue2', red=92, green=172, blue=238)
STEELBLUE3 = RGB(name='steelblue3', red=79, green=148, blue=205)
STEELBLUE4 = RGB(name='steelblue4', red=54, green=100, blue=139)
TAN = RGB(name='tan', red=210, green=180, blue=140)
TAN1 = RGB(name='tan1', red=255, green=165, blue=79)
TAN2 = RGB(name='tan2', red=238, green=154, blue=73)
TAN3 = RGB(name='tan3', red=205, green=133, blue=63)
TAN4 = RGB(name='tan4', red=139, green=90, blue=43)
TEAL = RGB(name='teal', red=0, green=128, blue=128)
THISTLE = RGB(name='thistle', red=216, green=191, blue=216)
THISTLE1 = RGB(name='thistle1', red=255, green=225, blue=255)
THISTLE2 = RGB(name='thistle2', red=238, green=210, blue=238)
THISTLE3 = RGB(name='thistle3', red=205, green=181, blue=205)
THISTLE4 = RGB(name='thistle4', red=139, green=123, blue=139)
TOMATO1 = RGB(name='tomato1', red=255, green=99, blue=71)
TOMATO2 = RGB(name='tomato2', red=238, green=92, blue=66)
TOMATO3 = RGB(name='tomato3', red=205, green=79, blue=57)
TOMATO4 = RGB(name='tomato4', red=139, green=54, blue=38)
TURQUOISE = RGB(name='turquoise', red=64, green=224, blue=208)
TURQUOISE1 = RGB(name='turquoise1', red=0, green=245, blue=255)
TURQUOISE2 = RGB(name='turquoise2', red=0, green=229, blue=238)
TURQUOISE3 = RGB(name='turquoise3', red=0, green=197, blue=205)
TURQUOISE4 = RGB(name='turquoise4', red=0, green=134, blue=139)
TURQUOISEBLUE = RGB(name='turquoiseblue', red=0, green=199, blue=140)
VIOLET = RGB(name='violet', red=238, green=130, blue=238)
VIOLETRED = RGB(name='violetred', red=208, green=32, blue=144)
VIOLETRED1 = RGB(name='violetred1', red=255, green=62, blue=150)
VIOLETRED2 = RGB(name='violetred2', red=238, green=58, blue=140)
VIOLETRED3 = RGB(name='violetred3', red=205, green=50, blue=120)
VIOLETRED4 = RGB(name='violetred4', red=139, green=34, blue=82)
WARMGREY = RGB(name='warmgrey', red=128, green=128, blue=105)
WHEAT = RGB(name='wheat', red=245, green=222, blue=179)
WHEAT1 = RGB(name='wheat1', red=255, green=231, blue=186)
WHEAT2 = RGB(name='wheat2', red=238, green=216, blue=174)
WHEAT3 = RGB(name='wheat3', red=205, green=186, blue=150)
WHEAT4 = RGB(name='wheat4', red=139, green=126, blue=102)
WHITE = RGB(name='white', red=255, green=255, blue=255)
WHITESMOKE = RGB(name='whitesmoke', red=245, green=245, blue=245)
WHITESMOKE = RGB(name='whitesmoke', red=245, green=245, blue=245)
YELLOW1 = RGB(name='yellow1', red=255, green=255, blue=0)

```

(continues on next page)

(continued from previous page)

```
YELLOW2 = RGB(name='yellow2', red=238, green=238, blue=0)
YELLOW3 = RGB(name='yellow3', red=205, green=205, blue=0)
YELLOW4 = RGB(name='yellow4', red=139, green=139, blue=0)
```

15.5 pyrevit.coreutils.configparser

Base module for pyRevit config parsing.

class `pyrevit.coreutils.configparser.PyRevitConfigParser` (*cfg_file_path=None*)

Bases: `object`

Config parser object. Handle config sections and io.

add_section (*section_name*)

Add section with given name to config.

get_config_file_hash ()

Get calculated unique hash for this config.

get_section (*section_name*)

Get section with given name.

Raises `AttributeError` – if section is missing

has_section (*section_name*)

Check if config contains given section.

reload (*cfg_file_path=None*)

Reload config from original or given file.

remove_section (*section_name*)

Remove section from config.

save (*cfg_file_path=None*)

Save config to original or given file.

class `pyrevit.coreutils.configparser.PyRevitConfigSectionParser` (*config_parser,*
sec-
tion_name)

Bases: `object`

Config section parser object. Handle section options.

add_subsection (*section_name*)

Add subsection to section.

get_option (*op_name, default_value=None*)

Get option value or return default.

get_subsection (*section_name*)

Get subsection with given name.

get_subsections ()

Get all subsections.

has_option (*option_name*)

Check if section contains given option.

has_subsection (*section_name*)
Check if section has any subsections.

header
Section header.

remove_option (*option_name*)
Remove given option from section.

set_option (*op_name, value*)
Set value of given option.

subheader
Section sub-header e.g. Section.SubSection

15.6 pyrevit.coreutils.dotnetcompiler

Base module for compiling dotnet sources to bytecode.

`pyrevit.coreutils.dotnetcompiler.compile_csharp` (*sourcefiles_list*,
full_output_file_addr=None, reference_list=None, resource_list=None)

Compile list of c-sharp source files to assembly.

if `full_output_file_addr` is provided, the generated dll will be written to that file, otherwise the assembly will be generated in memory.

Parameters

- **sourcefiles_list** (*list[str]*) – list of source c-sharp files
- **full_output_file_addr** (*str*) – full path of output dll
- **reference_list** (*list[str]*) – list of reference assemblies
- **resource_list** (*list[str]*) – list of resources to be included

Returns path to assembly if dll path provided, otherwise generated assembly

Return type `str` or `System.Reflection.Assembly`

15.7 pyrevit.coreutils.envvars

pyRevit managed environment variables framework.

pyRevit provides the environment variables framework to the pyRevit core and all pyRevit tools so they can store arbitrary data withing the running host session and share small data quickly between script runs.

Some settings needs to be set for the current session and might need to affect the behaviour of all individual scripts inside the extensions. (e.g. If user activates the DEBUG mode, all scripts should follow and log the debug entries.) The information is saved using `AppDomain.GetData` and `SetData` in a dictionary parameter. The dictionary is used to minimize the addition of named parameters to the `AppDomain`. The dictionary then includes all the internal parameters and their associated value. This way each script does not need to read the `usersettings` data which reduces io and saves time.

pyRevit uses environment variables extensively at its core and making changes to the core environment variables (starting with `PYREVIT_`) through scripts is strongly prohibited.

Example

```
>>> from pyrevit.coreutils import envvars
>>> envvars.set_pyrevit_env_var('MY_SCRIPT_STATUS', True)
>>> envvars.set_pyrevit_env_var('MY_SCRIPT_CONFIG', {'someconfig': True})
```

Then another script or same script when executed later within the same session can query the shared environment variable:

```
>>> envvars.get_pyrevit_env_vars('MY_SCRIPT_STATUS')
True
>>> envvars.get_pyrevit_env_vars('MY_SCRIPT_CONFIG')
{'someconfig': True}
```

`pyrevit.coreutils.envvars.get_pyrevit_env_var(param_name)`

Get value of a parameter shared between all scripts.

Parameters `param_name` (*str*) – name of environment variable

Returns any object stored as the environment variable value

Return type object

`pyrevit.coreutils.envvars.get_pyrevit_env_vars()`

Get the root dictionary, holding all environment variables.

`pyrevit.coreutils.envvars.set_pyrevit_env_var(param_name, param_value)`

Set value of a parameter shared between all scripts.

Parameters

- **param_name** (*str*) – name of environment variable
- **param_value** (*object*) – any python object

15.8 pyrevit.coreutils.git

Description: LibGit2Sharp wrapper module for pyRevit.

Documentation: <https://github.com/libgit2/libgit2sharp/wiki>

exception `pyrevit.coreutils.git.PyRevitGitAuthenticationError`

Bases: `pyrevit.PyRevitException`

Git authentication error.

class `pyrevit.coreutils.git.RepoInfo(repo)`

Bases: object

Repo wrapper for passing around repository information.

directory

repo directory

Type str

name

repo name

Type str

head_name

head branch name

Type str**last_commit_hash**

hash of head commit

Type str**repo**

LibGit2Sharp.Repository object

Type str**branch**

current branch name

Type str**username**

credentials - username

Type str**password**

credentials - password

Type str`pyrevit.coreutils.git.compare_branch_heads(repo_info)`

Compare local and remote branch heads and return ???

Parameters `repo_info` (*RepoInfo*) – target repo object**Returns** desc**Return type** type`pyrevit.coreutils.git.get_all_new_commits(repo_info)`

Fetch and return new commits ahead of current head.

Parameters `repo_info` (*RepoInfo*) – target repo object**Returns** str]: ordered dict of commit hash:message**Return type** OrderedDict[str`pyrevit.coreutils.git.get_repo(repo_dir)`

Return repo object for given git repo directory.

Parameters `repo_dir` (*str*) – full path of git repo directory**Returns** repo object**Return type** *RepoInfo*`pyrevit.coreutils.git.git_clone(repo_url, clone_dir, username=None, password=None)`

Clone git repository to given location

Parameters

- `repo_url` (*str*) – repo .git url
- `clone_dir` (*str*) – destination path
- `username` (*str*) – credentials - username

- **password** (*str*) – credentials - password

`pyrevit.coreutils.git.git_fetch(repo_info)`

Fetch current branch of given repo.

Parameters `repo_info` (*RepoInfo*) – target repo object

Returns repo object with updated head

Return type *RepoInfo*

`pyrevit.coreutils.git.git_pull(repo_info)`

Pull the current head of given repo.

Parameters `repo_info` (*RepoInfo*) – target repo object

Returns repo object with updated head

Return type *RepoInfo*

15.9 pyrevit.coreutils.loadertypes

Provide access to classes and functionality inside base loader module.

15.10 pyrevit.coreutils.logger

Core logging module for pyRevit.

class `pyrevit.coreutils.logger.DispatchingFormatter` (*log_formatters*,
log_default_formatter)

Bases: `object`

Dispatching formatter to format by log level.

Parameters

- **(dict[int(*log_formatters*) – logging.Formatter]**): dict of level:formatter key pairs
- **log_default_formatter** (*logging.Formatter*) – default formatter

format (*record*)

Format given record by log level.

class `pyrevit.coreutils.logger.LoggerWrapper` (**args*)

Bases: `logging.Logger`

Custom logging object.

Parameters

- **val** (*type*) – desc
- **val** – desc

callHandlers (*record*)

Override `logging.Logger.callHandlers`

deprecate (*message*)

Log message with custom `Deprecate` level.

get_level ()

Return current logging level.

has_errors ()
Check if logger has reported any errors.

isEnabledFor (*level*)
Override logging.Logger.isEnabledFor

is_enabled_for (*level*)
Check if logger is enabled for level in pyRevit environment.

reset_level ()
Reset logging level back to default.

set_debug_mode ()
Activate debug mode. Log levels >= DEBUG are enabled.

set_level (*level*)
Set logging level to level.

set_quiet_mode ()
Activate quiet mode. All log levels are disabled.

set_verbose_mode ()
Activate verbose mode. Log levels >= INFO are enabled.

`pyrevit.coreutils.logger.get_file_hdlr()`
Return file logging handler object.

Returns configured instance of python's native stream handler

Return type logging.FileHandler

`pyrevit.coreutils.logger.get_logger(logger_name)`
Register and return a logger with given name.

Caches all registered loggers and returns the same logger object on second call with the same logger name.

Parameters

- **logger_name** (*str*) – logger name
- **val** (*type*) – desc

Returns logger object wrapper python's native logger

Return type *LoggerWrapper*

Example

```
>>> get_logger('my command')
... <LoggerWrapper ...>
```

`pyrevit.coreutils.logger.get_stdout_hdlr()`
Return stdout logging handler object.

Returns configured instance of python's native stream handler

Return type logging.StreamHandler

`pyrevit.coreutils.logger.loggers_have_errors()`
Check if any errors have been reported by any of registered loggers.

`pyrevit.coreutils.logger.set_file_logging(status)`
Set file logging status (enable/disable).

Parameters `status` (*bool*) – True to enable, False to disable

15.11 pyrevit.coreutils.mathnet

MathNet importer module.

See <https://www.mathdotnet.com> for documentation.

Example

```
>>> from pyrevit.coreutils.mathnet import MathNet
```

15.12 pyrevit.coreutils.moduleutils

Utility fuctions to support smart modules.

`pyrevit.coreutils.moduleutils.copy_func` (*func*, *func_name*, *doc_string=None*,
arg_list=None)

Copy a function object to create a new function.

This is used inside smart modules that auto-generate functions based on context.

Parameters

- **func** (*object*) – python source function object
- **func_name** (*str*) – new function name
- **doc_string** (*str*) – new function docstring
- **arg_list** (*list*) – list of default values for function arguments

Returns new python function objects

Return type object

15.13 pyrevit.coreutils.pyutils

Helper functions for python.

Example

```
>>> from pyrevit.coreutils import pyutils
>>> pyutils.safe_cast('string', int, 0)
```

class `pyrevit.coreutils.pyutils.DefaultOrderedDict` (*default_factory=None*, **a*, ***kw*)
Bases: `collections.OrderedDict`

Ordered dictionary with default type.

This is similar to `defaultdict` and maintains the order of items added to it so in that regards it functions similar to `OrderedDict`.

Example

```
>>> from pyrevit.coreutils import pyutils
>>> od = pyutils.DefaultOrderedDict(list)
>>> od['A'] = [1, 2, 3]
>>> od['B'] = [4, 5, 6]
>>> od['C'].extend([7, 8, 9])
>>> for k, v in od.items():
...     print(k, v)
('A', [1, 2, 3])
('B', [4, 5, 6])
('C', [7, 8, 9])
```

copy()

Copy the dictionary.

`pyrevit.coreutils.pyutils.compare_lists(x, y)`

Compare two lists.

See: <https://stackoverflow.com/a/10872313/2350244>

Parameters

- **x** (*list*) – first list
- **y** (*list*) – second list

`pyrevit.coreutils.pyutils.isnumber(token)`

Verify if given string token is int or float.

Parameters **token** (*str*) – string value

Returns True if token is int or float

Return type bool

Example

```
>>> isnumber('12.3')
True
```

`pyrevit.coreutils.pyutils.pairwise(iterable, step=2)`

Iterate through items in pairs.

Parameters

- **iterable** (*iterable*) – any iterable object
- **step** (*int*) – number of steps to move when making pairs

Returns list of pairs

Return type iterable

Example

```
>>> pairwise([1, 2, 3, 4, 5])
[(1, 2), (3, 4)] # 5 can not be paired
>>> pairwise([1, 2, 3, 4, 5, 6])
```

(continues on next page)

(continued from previous page)

```
[(1, 2), (3, 4), (5, 6)]
>>> pairwise([1, 2, 3, 4, 5, 6], step=1)
[(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]
```

`pyrevit.coreutils.pyutils.safe_cast` (*val*, *to_type*, *default=None*)

Convert value to type gracefully.

This method basically calls `to_type(value)` and returns the default if exception occurs.

Parameters

- **val** (*any*) – value to be converted
- **to_type** (*type*) – target type
- **default** (*any*) – value to rerun on conversion exception

Example

```
>>> safe_cast('name', int, default=0)
0
```

15.14 pyrevit.coreutils.ribbon

Base module to interact with Revit ribbon.

class `pyrevit.coreutils.ribbon.ButtonIcons` (*image_file*)

Bases: `object`

pyRevit ui element icon.

Upon init, this type reads the given image file into an io stream and releases the os lock on the file.

Parameters **image_file** (*str*) – image file path to be used as icon

icon_file_path

icon image file path

Type `str`

filestream

io stream containing image binary data

Type `IO.FileStream`

check_icon_size ()

Verify icon size is within acceptable range.

create_bitmap (*icon_size*)

Resamples image and creates bitmap for the given size.

Icons are assumed to be square.

Parameters **icon_size** (*int*) – icon size (width or height)

Returns object containing image data at given size

Return type `Imaging.BitmapSource`

large_bitmap

Resamples image and creates bitmap for size `ICON_LARGE`.

Returns object containing image data at given size

Return type `Imaging.BitmapSource`

medium_bitmap

Resamples image and creates bitmap for size `ICON_MEDIUM`.

Returns object containing image data at given size

Return type `Imaging.BitmapSource`

small_bitmap

Resamples image and creates bitmap for size `ICON_SMALL`.

Returns object containing image data at given size

Return type `Imaging.BitmapSource`

class `pyrevit.coreutils.ribbon.GenericPyRevitUIContainer`

Bases: `object`

Common type for all pyRevit ui containers.

name

container name

Type `str`

itemdata_mode

if container is wrapping `UI.*ItemData`

Type `bool`

activate()

Activate this container in ui.

contains(*pyrvt_cmp_name*)

Check if container contains a component with given name.

Parameters

- **pyrvt_cmp_name** (*str*) – target component name
- **val** (*type*) – desc

deactivate()

Deactivate this container in ui.

enabled

Is container enabled.

find_child(*child_name*)

Find a component with given name in children.

Parameters **child_name** (*str*) – target component name

Returns component object if found, otherwise `None`

Return type

•

get_adwindows_object()

Return underlying `AdWindows` API object for this container.

get_flagged_children (*state=True*)

Get all children with their flag equal to given state.

Flagging is a mechanism to mark certain containers. There are various reasons that container flagging might be used e.g. marking updated containers or the ones in need of an update or removal.

Parameters *state* (*bool*) – flag state to filter children

Returns list of filtered child objects

Return type list[*]

get_rvtapi_object ()

Return underlying Revit API object for this container.

is_dirty ()

Is dirty flag set.

static is_native ()

Is this container generated by pyRevit or is native.

reorder_after (*item_name*, *ritem_name*)

Reorder and place *item_name* after *ritem_name*

Parameters

- **item_name** (*str*) – name of component to be moved
- **ritem_name** (*str*) – name of component that should be on the left

reorder_afterall (*item_name*)

Reorder and place *item_name* after all others.

Parameters *item_name* (*str*) – name of component to be moved

reorder_before (*item_name*, *ritem_name*)

Reorder and place *item_name* before *ritem_name*

Parameters

- **item_name** (*str*) – name of component to be moved
- **ritem_name** (*str*) – name of component that should be on the right

reorder_beforeall (*item_name*)

Reorder and place *item_name* before all others.

Parameters *item_name* (*str*) – name of component to be moved

set_dirty_flag (*state=True*)

Set dirty flag to given state.

See `.get_flagged_children()`

Parameters *state* (*bool*) – state to set flag

set_rvtapi_object (*rvtapi_obj*)

Set underlying Revit API object for this container.

Parameters *rvtapi_obj* (*obj*) – Revit API container object

visible

Is container visible.

class `pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer`

Bases: `pyrevit.coreutils.ribbon.GenericPyRevitUIContainer`

Common base type for native Revit API UI containers.

activate ()

Activate this container in ui.

Under current implementation, raises PyRevitUIError exception as native Revit API UI components should not be changed.

deactivate ()

Deactivate this container in ui.

Under current implementation, raises PyRevitUIError exception as native Revit API UI components should not be changed.

static is_native ()

Is this container generated by pyRevit or is native.

exception `pyrevit.coreutils.ribbon.PyRevitUIError`

Bases: `pyrevit.PyRevitException`

Common base class for all pyRevit ui-related exceptions.

class `pyrevit.coreutils.ribbon.RevitNativeRibbonButton` (*adwnd_ribbon_button*)

Bases: `pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer`

Revit API UI native ribbon button.

class `pyrevit.coreutils.ribbon.RevitNativeRibbonGroupItem` (*adwnd_ribbon_item*)

Bases: `pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer`

Revit API UI native ribbon button.

button (*name*)

Get button item with given name.

Parameters **name** (*str*) – name of button item to find

Returns button object if found

Return type `RevitNativeRibbonButton`

class `pyrevit.coreutils.ribbon.RevitNativeRibbonPanel` (*adwnd_ribbon_panel*)

Bases: `pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer`

Revit API UI native ribbon button.

ribbon_item (*item_name*)

Get panel item with given name.

Parameters **item_name** (*str*) – name of panel item to find

Returns panel item if found, could be `RevitNativeRibbonButton` or `RevitNativeRibbonGroupItem`

Return type object

class `pyrevit.coreutils.ribbon.RevitNativeRibbonTab` (*adwnd_ribbon_tab*)

Bases: `pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer`

Revit API UI native ribbon tab.

static is_pyrevit_tab ()

Is this tab generated by pyRevit.

ribbon_panel (*panel_name*)

Get panel with given name.

Parameters `panel_name` (*str*) – name of panel to find

Returns panel if found

Return type *RevitNativeRibbonPanel*

`pyrevit.coreutils.ribbon.get_current_ui` (*all_native=False*)

Revit UI Wrapper class for interacting with current pyRevit UI.

Returned class provides min required functionality for user interaction

Example

```
>>> current_ui = pyrevit.session.current_ui()
>>> this_script = pyrevit.session.get_this_command()
>>> current_ui.update_button_icon(this_script, new_icon)
```

Returns wrapper around active ribbon gui

Return type *_PyRevitUI*

`pyrevit.coreutils.ribbon.get_uibutton` (*command_unique_name*)

Find and return ribbon ui button with given unique id.

Parameters `command_unique_name` (*str*) – unique id of pyRevit command

Returns ui button wrapper object

Return type *_PyRevitRibbonButton*

`pyrevit.coreutils.ribbon.load_bitmapimage` (*image_file*)

Load given png file.

Parameters `image_file` (*str*) – image file path

Returns bitmap image object

Return type *Imaging.BitmapImage*

Reusable WPF forms for pyRevit.

Example

```
>>> from pyrevit.forms import WPFWindow
```

```
class pyrevit.forms.CommandSwitchWindow(context, title, width, height, **kwargs)  
    Standard form to select from a list of command options.
```

Parameters

- **context** (*list[str]*) – list of command options to choose from
- **switches** (*list[str]*) – list of on/off switches
- **message** (*str*) – window title message
- **config** (*dict*) – dictionary of config dicts for options or switches
- **recognize_access_key** (*bool*) – recognize ‘_’ as mark of access key

Returns name of selected option

Return type str

Returns if `switches` option is used, returns a tuple of selection option name and dict of switches

Return type tuple(str, dict)

Example

This is an example with series of command options:

```
>>> from pyrevit import forms
>>> ops = ['option1', 'option2', 'option3', 'option4']
>>> forms.CommandSwitchWindow.show(ops, message='Select Option')
'option2'
```

A more advanced example of combining command options, on/off switches, and option or switch configuration options:

```
>>> from pyrevit import forms
>>> ops = ['option1', 'option2', 'option3', 'option4']
>>> switches = ['switch1', 'switch2']
>>> cfigs = {'option1': { 'background': '0xFF55FF'}}
>>> rops, rswitches = forms.CommandSwitchWindow.show(
...     ops,
...     switches=switches
...     message='Select Option',
...     config=cfigs,
...     recognize_access_key=False
... )
>>> rops
'option2'
>>> rswitches
{'switch1': False, 'switch2': True}
```

handle_click (*sender, args*)

Handle mouse click.

handle_input_key (*sender, args*)

Handle keyboard inputs.

process_option (*sender, args*)

Handle click on command option button.

search_txt_changed (*sender, args*)

Handle text change in search box.

class `pyrevit.forms.GetValueWindow` (*context, title, width, height, **kwargs*)

Standard form to get simple values from user.

Args:

Example

```
>>> from pyrevit import forms
>>> items = ['item1', 'item2', 'item3']
>>> forms.SelectFromList.show(items, button_name='Select Item')
>>> ['item1']
```

select (*sender, args*)

Process input data and set the response.

string_value_changed (*sender, args*)

Handle string vlaue update event.

class `pyrevit.forms.ParamDef` (*name, istype*)

Parameter definition tuple.

name

parameter name

Type str

istype

true if type parameter, otherwise false

Type bool

istype

Alias for field number 1

name

Alias for field number 0

class pyrevit.forms.ProgressBar (*height=32, **kwargs*)
Show progress bar at the top of Revit window.

Parameters

- **title** (*string*) – progress bar text, defaults to 0/100 progress format
- **indeterminate** (*bool*) – create indeterminate progress bar
- **cancellable** (*bool*) – add cancel button to progress bar
- **step** (*int*) – update progress intervals

Example

```
>>> from pyrevit import forms
>>> count = 1
>>> with forms.ProgressBar(title='my command progress message') as pb:
...     # do stuff
...     pb.update_progress(count, 100)
...     count += 1
```

Progress bar title could also be customized to show the current and total progress values. In example below, the progress bar message will be in format “0 of 100”

```
>>> with forms.ProgressBar(title='{value} of {max_value}') as pb:
```

By default progress bar updates the progress every time the `.update_progress` method is called. For operations with a large number of max steps, the gui update process time will have a significant effect on the overall execution time of the command. In these cases, set the value of `step` argument to something larger than 1. In example below, the progress bar updates once per every 10 units of progress.

```
>>> with forms.ProgressBar(title='message', steps=10):
```

Progress bar could also be set to indeterminate for operations of unknown length. In this case, the progress bar will show an infinitely running ribbon:

```
>>> with forms.ProgressBar(title='message', indeterminate=True):
```

if `cancellable` is set on the object, a cancel button will show on the progress bar and `.cancelled` attribute will be set on the `ProgressBar` instance if users clicks on cancel button:

```
>>> with forms.ProgressBar(title='message',
...                       cancellable=True) as pb:
...     # do stuff
...     if pb.cancelled:
...         # wrap up and cancel operation
```

clicked_cancel (*sender, args*)
Handler for cancel button clicked event.

indeterminate
Progress bar indeterminate state.

reset ()
Reset progress value to 0.

title
Progress bar title.

update_progress (*new_value, max_value=1*)
Update progress bar state with given min, max values.

Parameters

- **new_value** (*float*) – current progress value
- **max_value** (*float*) – total progress value

wait_async (*func, args=()*)
Call a method asynchronously and show progress.

class `pyrevit.forms.RevisionOption` (*revision_element*)
Revision wrapper for `select_revisions()`.

name
Revision name (description).

class `pyrevit.forms.SearchPrompt` (*search_db, width, height, **kwargs*)
Standard prompt for pyRevit search.

Parameters

- **search_db** (*list*) – list of possible search targets
- **search_tip** (*str*) – text to show in grayscale when search box is empty
- **switches** (*str*) – list of switches
- **width** (*int*) – width of search prompt window
- **height** (*int*) – height of search prompt window

Returns matched strings, and dict of switches if provided str: matched string if switches are not provided.

Return type str, dict

Example

```
>>> from pyrevit import forms
>>> # assume search input of '/switch1 target1'
>>> matched_str, args, switches = forms.SearchPrompt.show(
...     search_db=['target1', 'target2', 'target3', 'target4'],
...     switches=['/switch1', '/switch2'],
...     search_tip='pyRevit Search'
... )
... matched_str
'target1'
... args
```

(continues on next page)

(continued from previous page)

```
[ '--help', '--branch', 'branchname' ]
... switches
{'/switch1': True, '/switch2': False}
```

find_direct_match (*input_text*)

Find direct text matches in search term.

find_word_match (*input_text*)

Find direct word matches in search term.

handle_kb_key (*sender, args*)

Handle keyboard input event.

search_input

Current search input.

search_input_parts

Current cleaned up search term.

search_matches

List of matches for the given search term.

search_term

Current cleaned up search term.

search_term_args

Find arguments in search term.

search_term_main

Current cleaned up search term without the listed switches.

search_term_switches

Find matching switches in search term.

search_txt_changed (*sender, args*)

Handle text changed event.

set_search_results (**args*)

Set search results for returning.

classmethod show (*search_db, width=600, height=100, **kwargs*)

Show search prompt.

update_results_display (*fill_match=False*)

Update search prompt results based on current input text.

class `pyrevit.forms.SelectFromList` (*context, title, width, height, **kwargs*)

Standard form to select from a list of items.

Any object can be passed in a list to the `context` argument. This class wraps the objects passed to `context`, in `TemplateListItem`. This class provides the necessary mechanism to make this form work both for selecting items from a list, and from a list of checkboxes. See the list of arguments below for additional options and features.

Parameters

- **context** (*list[str]* or *dict[list[str]]*) – list of items to be selected from OR dict of list of items to be selected from. use dict when input items need to be grouped e.g. List of sheets grouped by sheet set.
- **title** (*str, optional*) – window title. see super class for defaults.

- **width** (*int*, *optional*) – window width. see super class for defaults.
- **height** (*int*, *optional*) – window height. see super class for defaults.
- **button_name** (*str*, *optional*) – name of select button. defaults to ‘Select’
- **name_attr** (*str*, *optional*) – object attribute that should be read as item name.
- **multiselect** (*bool*, *optional*) – allow multi-selection (uses check boxes). defaults to False
- **return_all** (*bool*, *optional*) – return all items. This is handy when some input items have states and the script needs to check the state changes on all items. This options works in multiselect mode only. defaults to False
- **filterfunc** (*function*) – filter function to be applied to context items.
- **group_selector_title** (*str*) – title for list group selector. defaults to ‘List Group’
- **default_group** (*str*) – name of default group to be selected

Example

```
>>> from pyrevit import forms
>>> items = ['item1', 'item2', 'item3']
>>> forms.SelectFromList.show(items, button_name='Select Item')
>>> ['item1']
```

```
>>> from pyrevit import forms
>>> ops = [viewsheet1, viewsheet2, viewsheet3]
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=False,
...                               name_attr='Name',
...                               button_name='Select Sheet')
```

```
>>> from pyrevit import forms
>>> ops = {'Sheet Set A': [viewsheet1, viewsheet2, viewsheet3],
...       'Sheet Set B': [viewsheet4, viewsheet5, viewsheet6]}
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=True,
...                               name_attr='Name',
...                               group_selector_title='Sheet Sets',
...                               button_name='Select Sheets')
```

This module also provides a wrapper base class `TemplateListItem` for when the checkbox option is wrapping another element, e.g. a Revit ViewSheet. Derive from this base class and define the name property to customize how the checkbox is named on the dialog.

```
>>> from pyrevit import forms
>>> class MyOption(forms.TemplateListItem)
...     @property
...     def name(self):
...         return '{} - {}'.format(self.item.SheetNumber,
...                                 self.item.SheetNumber)
>>> ops = [MyOption('op1'), MyOption('op2', True), MyOption('op3')]
>>> res = forms.SelectFromList.show(ops,
...                               multiselect=True,
...                               button_name='Select Item')
```

(continues on next page)

(continued from previous page)

```
>>> [bool(x) for x in res] # or [x.state for x in res]
[True, False, True]
```

button_select (*sender, args*)

Handle select button click.

check_all (*sender, args*)

Handle check all button to mark all check boxes as checked.

check_selected (*sender, args*)

Mark selected checkboxes as checked.

clear_search (*sender, args*)

Clear search box.

search_txt_changed (*sender, args*)

Handle text change in search box.

toggle_all (*sender, args*)

Handle toggle all button to toggle state of all check boxes.

uncheck_all (*sender, args*)

Handle uncheck all button to mark all check boxes as un-checked.

uncheck_selected (*sender, args*)

Mark selected checkboxes as unchecked.

class `pyrevit.forms.SheetOption` (*sheet_element*)

Sheet wrapper for `select_sheets()`.

name

Sheet name.

number

Sheet number.

class `pyrevit.forms.TemplateListItem` (*orig_item, checkable=True, name_attr=None*)

Base class for checkbox option wrapping another object.

checkable

List Item CheckBox Visibility.

classmethod `is_checkbox` (*item*)

Check if the object has all necessary attrs for a checkbox.

name

Name property.

unwrap ()

Unwrap and return wrapped object.

class `pyrevit.forms.TemplatePromptBar` (*height=32, **kwargs*)

Template context-manager class for creating prompt bars.

Prompt bars are show at the top of the active Revit window and are designed for better prompt visibility.

Parameters

- **height** (*int*) – window height
- ****kwargs** – other arguments to be passed to `_setup()`

update_window()

Update the prompt bar to match Revit window.

class pyrevit.forms.**TemplateUserInputWindow**(*context, title, width, height, **kwargs*)

Base class for pyRevit user input standard forms.

Parameters

- **context** (*any*) – window context element(s)
- **title** (*str*) – window title
- **width** (*int*) – window width
- **height** (*int*) – window height
- ****kwargs** – other arguments to be passed to `_setup()`

classmethod **show**(*context, title='User Input', width=500, height=400, **kwargs*)

Show user input window.

Parameters

- **context** (*any*) – window context element(s)
- **title** (*str*) – window title
- **width** (*int*) – window width
- **height** (*int*) – window height
- ****kwargs** (*any*) – other arguments to be passed to window

class pyrevit.forms.**ViewOption**(*view_element*)

View wrapper for `select_views()`.

name

View name.

class pyrevit.forms.**WPFWindow**(*xaml_source, literal_string=False, handle_esc=True*)

WPF Window base class for all pyRevit forms.

Parameters

- **xaml_source** (*str*) – xaml source filepath or xaml content
- **literal_string** (*bool*) – xaml_source contains xaml content, not filepath
- **handle_esc** (*bool*) – handle Escape button and close the window

Example

```
>>> from pyrevit import forms
>>> layout = '<Window ' \
>>>         'xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" ' \
↪ \
>>>         'xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" ' \
>>>         'ShowInTaskbar="False" ResizeMode="NoResize" ' \
>>>         'WindowStartupLocation="CenterScreen" ' \
>>>         'HorizontalContentAlignment="Center">' \
>>>         '</Window>'
>>> w = forms.WPFWindow(layout, literal_string=True)
>>> w.show()
```


static disable_element (*wpf_elements)

Enable elements.

Parameters *wpf_elements – WPF framework elements to be enabled

static enable_element (*wpf_elements)

Enable elements.

Parameters *wpf_elements – WPF framework elements to be enabled

handle_input_key (sender, args)

Handle keyboard input and close the window on Escape.

handle_url_click (sender, args)

Callback for handling click on package website url

static hide_element (*wpf_elements)

Collapse elements.

Parameters *wpf_elements – WPF framework elements to be collapsed

pyrevit_version

Active pyRevit formatted version e.g. '4.9-beta'

set_image_source (wpf_element, image_file)

Set source file for image element.

Parameters

- **element_name** (*System.Windows.Controls.Image*) – xaml image element
- **image_file** (*str*) – image file path

setup_icon ()

Setup default window icon.

show (modal=False)

Show window.

show_dialog ()

Show modal window.

static show_element (*wpf_elements)

Show collapsed elements.

Parameters *wpf_elements – WPF framework elements to be set to visible.

static toggle_element (*wpf_elements)

Toggle visibility of elements.

Parameters *wpf_elements – WPF framework elements to be toggled.

class pyrevit.forms.**WarningBar** (height=32, **kwargs)

Show warning bar at the top of Revit window.

Parameters title (*string*) – warning bar text

Example

```
>>> with WarningBar(title='my warning'):
...     # do stuff
```

`pyrevit.forms.alert` (*msg*, *title=None*, *sub_msg=None*, *expanded=None*, *footer=""*, *ok=True*, *cancel=False*, *yes=False*, *no=False*, *retry=False*, *warn_icon=True*, *options=None*, *exitscript=False*)

Show a task dialog with given message.

Parameters

- **msg** (*str*) – message to be displayed
- **title** (*str*, *optional*) – task dialog title
- **ok** (*bool*, *optional*) – show OK button, defaults to True
- **cancel** (*bool*, *optional*) – show Cancel button, defaults to False
- **yes** (*bool*, *optional*) – show Yes button, defaults to False
- **no** (*bool*, *optional*) – show NO button, defaults to False
- **retry** (*bool*, *optional*) – show Retry button, defaults to False
- **options** (*list[str]*, *optional*) – list of command link titles in order
- **exitscript** (*bool*, *optional*) – exit if cancel or no, defaults to False

Returns True if okay, yes, or retry, otherwise False

Return type bool

Example

```
>>> from pyrevit import forms
>>> forms.alert('Are you sure?',
...           ok=False, yes=True, no=True, exitscript=True)
```

`pyrevit.forms.alert_ifnot` (*condition*, *msg*, **args*, ***kwargs*)

Show a task dialog with given message if condition is NOT met.

Parameters

- **condition** (*bool*) – condition to test
- **msg** (*str*) – message to be displayed
- **title** (*str*, *optional*) – task dialog title
- **ok** (*bool*, *optional*) – show OK button, defaults to True
- **cancel** (*bool*, *optional*) – show Cancel button, defaults to False
- **yes** (*bool*, *optional*) – show Yes button, defaults to False
- **no** (*bool*, *optional*) – show NO button, defaults to False
- **retry** (*bool*, *optional*) – show Retry button, defaults to False
- **exitscript** (*bool*, *optional*) – exit if cancel or no, defaults to False

Returns True if okay, yes, or retry, otherwise False

Return type bool

Example

```
>>> from pyrevit import forms
>>> forms.alert_ifnot(value > 12,
...                 'Are you sure?',
...                 ok=False, yes=True, no=True, exitscript=True)
```

`pyrevit.forms.ask_for_date` (*default=None, prompt=None, title=None, **kwargs*)
Ask user to select a date value.

This is a shortcut function that configures *GetValueWindow* for date data types. kwargs can be used to pass on other arguments.

Parameters

- **default** (*datetime.datetime*) – default selected date value
- **prompt** (*str*) – prompt message
- **title** (*str*) – title message
- **kwargs** (*type*) – other arguments to be passed to *GetValueWindow*

Returns selected date

Return type *datetime.datetime*

Example

```
>>> forms.ask_for_date(default="", title="Enter deadline:")
... datetime.datetime(2019, 5, 17, 0, 0)
```

`pyrevit.forms.ask_for_one_item` (*items, default=None, prompt=None, title=None, **kwargs*)
Ask user to select an item from a list of items.

This is a shortcut function that configures *GetValueWindow* for ‘single-select’ data types. kwargs can be used to pass on other arguments.

Parameters

- **items** (*list[str]*) – list of items to choose from
- **default** (*str*) – default selected item
- **prompt** (*str*) – prompt message
- **title** (*str*) – title message
- **kwargs** (*type*) – other arguments to be passed to *GetValueWindow*

Returns selected item

Return type *str*

Example

```
>>> forms.ask_for_one_item(
...     ['test item 1', 'test item 2', 'test item 3'],
...     default='test item 2',
...     prompt='test prompt',
```

(continues on next page)

(continued from previous page)

```
...     title='test title'  
... )  
... 'test item 1'
```

`pyrevit.forms.ask_for_string` (*default=None, prompt=None, title=None, **kwargs*)

Ask user to select a string value.

This is a shortcut function that configures `GetValueWindow` for string data types. `kwargs` can be used to pass on other arguments.

Parameters

- **default** (*str*) – default unique string. must not be in `reserved_values`
- **prompt** (*str*) – prompt message
- **title** (*str*) – title message
- **kwargs** (*type*) – other arguments to be passed to `GetValueWindow`

Returns selected string value

Return type `str`

Example

```
>>> forms.ask_for_string(  
...     default='some-tag',  
...     prompt='Enter new tag name:',  
...     title='Tag Manager')  
... 'new-tag'
```

`pyrevit.forms.ask_for_unique_string` (*reserved_values, default=None, prompt=None, title=None, **kwargs*)

Ask user to select a unique string value.

This is a shortcut function that configures `GetValueWindow` for unique string data types. `kwargs` can be used to pass on other arguments.

Parameters

- **reserved_values** (*list[str]*) – list of reserved (forbidden) values
- **default** (*str*) – default unique string. must not be in `reserved_values`
- **prompt** (*str*) – prompt message
- **title** (*str*) – title message
- **kwargs** (*type*) – other arguments to be passed to `GetValueWindow`

Returns selected unique string

Return type `str`

Example

```
>>> forms.ask_for_unique_string(
...     prompt='Enter a Unique Name',
...     title=self.Title,
...     reserved_values=['Ehsan', 'Gui', 'Guido'],
...     owner=self)
... 'unique string'
```

In example above, owner argument is provided to be passed to underlying `GetValueWindow`.

`pyrevit.forms.check_familydoc` (*doc=None, family_cat=None, exitscript=False*)

Verify document is a Family and notify user if not.

Parameters

- **doc** (*DB.Document*) – target document, current of not provided
- **family_cat** (*str*) – family category name
- **exitscript** (*bool*) – exit script if returning False

Returns True if doc is a Family and of provided category

Return type bool

Example

```
>>> from pyrevit import forms
>>> forms.check_familydoc(doc=revit.doc, family_cat='Data Devices')
... True
```

`pyrevit.forms.check_modeldoc` (*doc=None, exitscript=False*)

Verify document is a not a Model and notify user if not.

Parameters

- **doc** (*DB.Document*) – target document, current of not provided
- **exitscript** (*bool*) – exit script if returning False

Returns True if doc is a Model

Return type bool

Example

```
>>> from pyrevit import forms
>>> forms.check_modeldoc(doc=revit.doc)
... True
```

`pyrevit.forms.check_selection` (*exitscript=False, message='At least one element must be selected.'*)

Verify if selection is not empty notify user if it is.

Parameters

- **exitscript** (*bool*) – exit script if returning False
- **message** (*str*) – prompt message if returning False

Returns True if selection has at least one item

Return type bool

`pyrevit.forms.check_workshared` (*doc=None, message='Model is not workshared.'*)
Verify if model is workshared and notify user if not.

Parameters

- **doc** (*DB.Document*) – target document, current if not provided
- **message** (*str*) – prompt message if returning False

Returns True if doc is workshared

Return type bool

`pyrevit.forms.inform_wip` ()
Show work-in-progress prompt to user and exit script.

Example

```
>>> forms.inform_wip()
```

`pyrevit.forms.pick_excel_file` (*save=False*)
File pick/save dialog for an excel file.

Parameters **save** (*bool*) – show file save dialog, instead of file pick dialog

Returns file path

Return type str

`pyrevit.forms.pick_file` (*file_ext="", files_filter="", init_dir="", restore_dir=True, multi_file=False, unc_paths=False*)
Pick file dialog to select a destination file.

Parameters

- **file_ext** (*str*) – file extension
- **files_filter** (*str*) – file filter
- **init_dir** (*str*) – initial directory
- **restore_dir** (*bool*) – restore last directory
- **multi_file** (*bool*) – allow select multiple files
- **unc_paths** (*bool*) – return unc paths

Returns file path or list of file paths if `multi_file=True`

Return type str or list[str]

Example

```
>>> from pyrevit import forms
>>> forms.pick_file(file_ext='csv')
... r'C:\output\somefile.csv'
```

```
>>> forms.pick_file(file_ext='csv', multi_file=True)
... [r'C:\output\somefile1.csv', r'C:\output\somefile2.csv']
```

```
>>> forms.pick_file(files_filter='All Files (*.*)|*.*|'
                    'Excel Workbook (*.xlsx)|*.xlsx|'
                    'Excel 97-2003 Workbook|*.xls',
                    multi_file=True)
... [r'C:\output\somefile1.xlsx', r'C:\output\somefile2.xls']
```

`pyrevit.forms.pick_folder` (*title=None*)

Show standard windows pick folder dialog.

Parameters `title` (*str, optional*) – title for the window

Returns folder path

Return type `str`

`pyrevit.forms.save_excel_file` ()

File save dialog for an excel file.

Returns file path

Return type `str`

`pyrevit.forms.save_file` (*file_ext=""*, *files_filter=""*, *init_dir=""*, *default_name=""*, *restore_dir=True*, *unc_paths=False*)

Save file dialog to select a destination file for data.

Parameters

- **file_ext** (*str*) – file extension
- **files_filter** (*str*) – file filter
- **init_dir** (*str*) – initial directory
- **default_name** (*str*) – default file name
- **restore_dir** (*bool*) – restore last directory
- **unc_paths** (*bool*) – return unc paths

Returns file path

Return type `str`

Example

```
>>> from pyrevit import forms
>>> forms.save_file(file_ext='csv')
... r'C:\output\somefile.csv'
```

`pyrevit.forms.select_image` (*images*, *title='Select Image'*, *button_name='Select'*)

Standard form for selecting an image.

Parameters

- **images** (*list[str] | list[framework.Imaging.BitmapImage]*) – list of image file paths or bitmaps
- **title** (*str, optional*) – swatch list window title
- **button_name** (*str, optional*) – swatch list window button caption

Returns path of the selected image

Return type str

Example

```
>>> from pyrevit import forms
>>> forms.select_image(['C:/path/to/image1.png',
                      'C:/path/to/image2.png'],
                      title="Select Variation")
... 'C:/path/to/image1.png'
```

`pyrevit.forms.select_open_docs` (*title='Select Open Documents', button_name='OK', width=500, multiple=True, filterfunc=None*)

Standard form for selecting open documents.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.

Returns list of selected documents

Return type list[DB.Document]

Example

```
>>> from pyrevit import forms
>>> forms.select_open_docs()
... [<Autodesk.Revit.DB.Document object>,
...  <Autodesk.Revit.DB.Document object>]
```

`pyrevit.forms.select_parameter` (*src_element, title='Select Parameters', button_name='Select', multiple=True, filterfunc=None, include_instance=True, include_type=True*)

Standard form for selecting parameters from given element.

Parameters

- **src_element** (*DB.Element*) – source element
- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **include_instance** (*bool, optional*) – list instance parameters
- **include_type** (*bool, optional*) – list type parameters

Returns list of paramdef objects

Return type list[*ParamDef*]

Example

```
>>> forms.select_parameter(
...     src_element,
...     title='Select Parameters',
...     multiple=True,
...     include_instance=True,
...     include_type=True
... )
... [<ParamDef >, <ParamDef >]
```

pyrevit.forms.**select_revisions** (*title='Select Revision', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None*)

Standard form for selecting revisions.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for revisions; defaults to active document

Returns list of selected revisions

Return type list[DB.Revision]

Example

```
>>> from pyrevit import forms
>>> forms.select_revisions()
... [<Autodesk.Revit.DB.Revision object>,
... <Autodesk.Revit.DB.Revision object>]
```

pyrevit.forms.**select_sheets** (*title='Select Sheets', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None*)

Standard form for selecting sheets.

Sheets are grouped into sheet sets and sheet set can be selected from a drop down box at the top of window.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True

- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for sheets; defaults to active document

Returns list of selected sheets

Return type list[DB.ViewSheet]

Example

```
>>> from pyrevit import forms
>>> forms.select_sheets()
... [<Autodesk.Revit.DB.ViewSheet object>,
... <Autodesk.Revit.DB.ViewSheet object>]
```

`pyrevit.forms.select_swatch` (*title='Select Color Swatch', button_name='Select'*)
Standard form for selecting a color swatch.

Parameters

- **title** (*str, optional*) – swatch list window title
- **button_name** (*str, optional*) – swatch list window button caption

Returns rgb color

Return type *pyrevit.coreutils.colors.RGB*

Example

```
>>> from pyrevit import forms
>>> forms.select_swatch(title="Select Text Color")
... <RGB #CD8800>
```

`pyrevit.forms.select_titleblocks` (*title='Select Titleblock', button_name='Select', no_tb_option='No Title Block', width=500, multiple=False, filterfunc=None, doc=None*)

Standard form for selecting a titleblock.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **no_tb_option** (*str, optional*) – name of option for no title block
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for titleblocks; defaults to active document

Returns selected titleblock id.

Return type DB.ElementId

Example

```
>>> from pyrevit import forms
>>> forms.select_titleblocks()
... <Autodesk.Revit.DB.ElementId object>
```

```
pyrevit.forms.select_views(title='Select Views', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None)
```

Standard form for selecting views.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for views; defaults to active document

Returns list of selected views

Return type list[DB.View]

Example

```
>>> from pyrevit import forms
>>> forms.select_views()
... [<Autodesk.Revit.DB.View object>,
... <Autodesk.Revit.DB.View object>]
```

```
pyrevit.forms.select_viewtemplates(title='Select View Templates', button_name='Select', width=500, multiple=True, filterfunc=None, doc=None)
```

Standard form for selecting view templates.

Parameters

- **title** (*str, optional*) – list window title
- **button_name** (*str, optional*) – list window button caption
- **width** (*int, optional*) – width of list window
- **multiselect** (*bool, optional*) – allow multi-selection (uses check boxes). defaults to True
- **filterfunc** (*function*) – filter function to be applied to context items.
- **doc** (*DB.Document, optional*) – source document for views; defaults to active document

Returns list of selected view templates

Return type list[DB.View]

Example

```
>>> from pyrevit import forms
>>> forms.select_viewtemplates()
... [<Autodesk.Revit.DB.View object>,
...  <Autodesk.Revit.DB.View object>]
```

`pyrevit.forms.toast` (*message*, *title*='pyRevit', *appid*='pyRevit', *icon*=None, *click*=None, *actions*=None)
Show a Windows 10 notification.

Parameters

- **message** (*str*) – notification message
- **title** (*str*) – notification title
- **appid** (*str*) – app name (will show under message)
- **icon** (*str*) – file path to icon .ico file (defaults to pyRevit icon)
- **click** (*str*) – click action commands string
- **actions** (*dict*) – dictionary of button names and action strings

Example

```
>>> script.toast("Hello World!",
...             title="My Script",
...             appid="MyAPP",
...             click="https://eirannejad.github.io/pyRevit/",
...             actions={
...                 "Open Google": "https://google.com",
...                 "Open Toast64": "https://github.com/go-toast/toast"
...             })
```

16.1 pyrevit.forms.toaster

Base module for pushing toast messages on Win 10.

This module is a wrapper for a cli utility that provides toast message functionality. See <https://github.com/go-toast/toast>

`pyrevit.forms.toaster.get_toaster` ()
Return full file path of the toast binary utility.

`pyrevit.forms.toaster.send_toast` (*message*, *title*=None, *appid*=None, *icon*=None, *click*=None, *actions*=None)
Send toast notificaton.

Parameters

- **message** (*str*) – notification message
- **title** (*str*) – notification title
- **appid** (*str*) – application unique id (see `-app-id` cli option)
- **icon** (*str*) – notification icon (see `-icon` cli option)

- **click** (*str*) – click action (see *–activation-arg* cli option)
- **(dict [str (*actions*) – str])**: list of actions (see *–action* and *–action-arg* cli options)

16.2 pyrevit.forms.utils

Utility functions to support forms module.

`pyrevit.forms.utils.bitmap_from_file` (*bitmap_file*)

Create BitmapImage from a bitmap file.

Parameters `bitmap_file` (*str*) – path to bitmap file

Returns bitmap image object

Return type BitmapImage

`pyrevit.forms.utils.load_component` (*xaml_file*, *comp_type*)

Load WPF component from xaml file.

Parameters

- **xaml_file** (*str*) – xaml file path
- **comp_type** (*System.Windows.Controls*) – WPF control type

Returns loaded WPF control

Return type System.Windows.Controls

`pyrevit.forms.utils.load_ctrl_template` (*xaml_file*)

Load System.Windows.Controls.ControlTemplate from xaml file.

Parameters `xaml_file` (*str*) – xaml file path

Returns loaded control template

Return type System.Windows.Controls.ControlTemplate

`pyrevit.forms.utils.load_itemspanel_template` (*xaml_file*)

Load System.Windows.Controls.ItemsPanelTemplate from xaml file.

Parameters `xaml_file` (*str*) – xaml file path

Returns loaded items-panel template

Return type System.Windows.Controls.ControlTemplate

CHAPTER 17

pyrevit.framework

Provide access to DotNet Framework.

Example

```
>>> from pyrevit.framework import Assembly, Windows
```

`pyrevit.framework.get_dll_file` (*assembly_name*)
Return path to given assembly name.

`pyrevit.framework.get_type` (*fw_object*)
Return CLR type of an object.

Module handling pyRevit initialization and loading process.

18.1 pyrevit.loader

Loader base module.

18.2 pyrevit.loader.asmmaker

Assembly maker module.

```
class pyrevit.loader.asmmaker.ExtensionAssemblyInfo (name, location, reloading)
```

location

Alias for field number 1

name

Alias for field number 0

reloading

Alias for field number 2

```
pyrevit.loader.asmmaker.create_assembly (extension)
```

Parameters *extension* (*pyrevit.extensions.components.Extension*) –

Returns:

18.3 pyrevit.loader.sessioninfo

Manage information about pyRevit sessions.

class `pyrevit.loader.sessioninfo.RuntimeInfo` (*pyrevit_version*, *engine_version*,
host_version)

Session runtime information tuple.

Parameters

- **pyrevit_version** (*str*) – formatted pyRevit version
- **engine_version** (*int*) – active IronPython engine version
- **host_version** (*str*) – Current Revit version

engine_version

Alias for field number 1

host_version

Alias for field number 2

pyrevit_version

Alias for field number 0

`pyrevit.loader.sessioninfo.get_loaded_pyrevit_assemblies()`

Return list of loaded pyRevit assemblies from environment variable.

Returns list of loaded assemblies

Return type list[str]

`pyrevit.loader.sessioninfo.get_runtime_info()`

Return runtime information tuple.

Returns runtime info tuple

Return type *RuntimeInfo*

Example

```
>>> sessioninfo.get_runtime_info()
```

`pyrevit.loader.sessioninfo.get_session_uuid()`

Read session uuid from environment variable.

Returns session uuid string

Return type str

`pyrevit.loader.sessioninfo.get_total_loaded_assembly_count()`

Returns the total number of pyRevit assemblies loaded under current Revit session. This value is stored in an environment variable and is kept updated during the multiple pyRevit sessions. Notice that not all of these assemblies belong to current pyRevit session as pyRevit could be reloaded multiple times under the same Revit session.

Returns Total number of loaded assemblies.

Return type int

`pyrevit.loader.sessioninfo.new_session_uuid()`

Create a new uuid for a pyRevit session.

Returns session uuid string

Return type str

```
pyrevit.loader.sessioninfo.report_env()
    Report python version, home directory, config file, etc.
```

```
pyrevit.loader.sessioninfo.set_loaded_pyrevit_assemblies(loaded_assembly_name_list)
    Set the environment variable with list of loaded assemblies.
```

Parameters

- **loaded_assembly_name_list** (*list[str]*) – list of assembly names
- **val** (*type*) – desc

```
pyrevit.loader.sessioninfo.set_session_uuid(uuid_str)
    Set session uuid on environment variable.
```

Parameters **uuid_str** (*str*) – session uuid string

```
pyrevit.loader.sessioninfo.set_total_loaded_assembly_count(assembly_count)
    Sets the total number of pyRevit assemblies loaded under current Revit session. This value is stored in an
    environment variable and is kept updated during the multiple pyRevit sessions.
```

This value should not be updated by pyRevit users.

Parameters **assembly_count** (*int*) – assembly count

```
pyrevit.loader.sessioninfo.setup_runtime_vars()
    Setup runtime environment variables with session information.
```

18.4 pyrevit.loader.systemdiag

Session diagnostics.

```
pyrevit.loader.systemdiag.system_diag()
    Verifies system status is appropriate for a pyRevit session.
```

18.5 pyrevit.loader.sessionmgr

The loader module manages the workflow of loading a new pyRevit session. It's main purpose is to orchestrate the process of finding pyRevit extensions, creating dll assemblies for them, and creating a user interface in the host application.

Everything starts from `sessionmgr.load_session()` function...

The only public function is `load_session()` that loads a new session. Everything else is private.

```
class pyrevit.loader.sessionmgr.AssembledExtension(ext, assem)
```

assem
Alias for field number 1

ext
Alias for field number 0

```
pyrevit.loader.sessionmgr.execute_command(pyrevitcmd_unique_id)
    Executes a pyRevit command.
```

Parameters **pyrevitcmd_unique_id** (*str*) – Unique/Class Name of the pyRevit command

Returns results from the executed command

`pyrevit.loader.sessionmgr.execute_script` (*script_path*, *arguments=None*, *sys_paths=None*,
clean_engine=True, *fullframe_engine=True*)

Executes a script using pyRevit script executor.

Parameters `script_path` (*str*) – Address of the script file

Returns results dictionary from the executed script

`pyrevit.loader.sessionmgr.find_pyrevitcmd` (*pyrevitcmd_unique_id*)

Searches the pyRevit-generated assemblies under current session for the command with the matching unique name (class name) and returns the command type. Notice that this returned value is a ‘type’ and should be instantiated before use.

Example

```
>>> cmd = find_pyrevitcmd('pyRevitCorepyRevitpyRevittoolsReload')
>>> command_instance = cmd()
>>> command_instance.Execute() # Provide commandData, message, elements
```

Parameters `pyrevitcmd_unique_id` (*str*) – Unique name for the command

Returns Type for the command with matching unique name

`pyrevit.loader.sessionmgr.load_session` ()

Handles loading/reloading of the pyRevit addin and extensions. To create a proper ui, pyRevit extensions needs to be properly parsed and a dll assembly needs to be created. This function handles these tasks through interactions with `.extensions`, `.loader.asmmaker`, and `.loader.uimaker`

Example

```
>>> from pyrevit.loader.sessionmgr import load_session
>>> load_session() # start loading a new pyRevit session
```

Returns None

18.6 pyrevit.loader.uimaker

UI maker.

`pyrevit.loader.uimaker.update_pyrevit_ui` (*ui_ext*, *ext_asm_info*, *create_beta=False*)

Updates/Creates pyRevit ui for the given extension and provided assembly dll address.

Provides access and control over the pyRevit output window.

19.1 pyrevit.output

Provide access to output window and its functionality.

This module provides access to the output window for the currently running pyRevit command. The proper way to access this wrapper object is through the `get_output()` of `pyrevit.script` module. This method, in return uses the `pyrevit.output` module to get access to the output wrapper.

Example

```
>>> from pyrevit import script
>>> output = script.get_output()
```

Here is the source of `pyrevit.script.get_output()`. As you can see this functions calls the `pyrevit.output.get_output()` to receive the output wrapper.

```
def get_output():
    """Return object wrapping output window for current script.

    Returns:
        :obj:`pyrevit.output.PyRevitOutputWindow`: Output wrapper object
    """
    return output.get_output()
```

```
class pyrevit.output.PyRevitOutputWindow
    Wrapper to interact with the output window.

    add_style(style_code, attribs=None)
        Inject style tag into current html head of the output window.
```

Parameters

- **style_code** (*str*) – css styling code
- **attrs** (*dict*) – dictionary of attribute names and value

Example

```
>>> output = pyrevit.output.get_output()
>>> output.add_style('body { color: blue; }')
```

close()

Close the window.

close_others (*all_open_outputs=False*)

Close all other windows that belong to the current command.

Parameters **all_open_outputs** (*bool*) – Close all any other windows if True

debug_mode

Set debug mode on output window and stream.

This will cause the output window to print information about the buffer stream and other aspects of the output window mechanism.

freeze()

Freeze output content update.

get_head_html()

str: Return inner code of html head element.

get_height()

int: Return current window height.

get_title()

str: Return current window title.

get_width()

int: Return current window width.

hide()

Hide the window.

hide_logpanel()

Hide output window logging panel.

hide_progress()

Hide output window progress bar.

indeterminate_progress (*state*)

Show or hide indeterminate progress bar.

inject_script (*script_code, attrs=None, body=False*)

Inject script tag into current head (or body) of the output window.

Parameters

- **script_code** (*str*) – javascript code
- **attrs** (*dict*) – dictionary of attribute names and value
- **body** (*bool, optional*) – injects script into body instead of head

Example

```
>>> output = pyrevit.output.get_output()
>>> output.inject_script('', # no script since it's a link
                        {'src': js_script_file_path})
```

inject_to_body (*element_tag, element_contents, attrs=None*)

Inject html element to current html body of the output window.

Parameters

- **element_tag** (*str*) – html tag of the element e.g. 'div'
- **element_contents** (*str*) – html code of the element contents
- **attrs** (*dict*) – dictionary of attribute names and value

Example

```
>>> output = pyrevit.output.get_output()
>>> output.inject_to_body('script',
                        '', # no script since it's a link
                        {'src': js_script_file_path})
```

inject_to_head (*element_tag, element_contents, attrs=None*)

Inject html element to current html head of the output window.

Parameters

- **element_tag** (*str*) – html tag of the element e.g. 'div'
- **element_contents** (*str*) – html code of the element contents
- **attrs** (*dict*) – dictionary of attribute names and value

Example

```
>>> output = pyrevit.output.get_output()
>>> output.inject_to_head('script',
                        '', # no script since it's a link
                        {'src': js_script_file_path})
```

insert_divider ()

Add horizontal rule to the output window.

static linkify (*element_ids, title=None*)

Create clickable link for the provided element ids.

This method, creates the link but does not print it directly.

Parameters

- **element_ids** (*list of ElementId*) –
- **element_ids** – single or multiple ids
- **title** (*str*) – title of the link. defaults to list of element ids

Example

```
>>> output = pyrevit.output.get_output()
>>> for idx, elid in enumerate(element_ids):
>>>     print('{}: {}'.format(idx+1, output.linkify(elid)))
```

lock_size()

Lock window size.

log_error (*message*)

Report ERROR message into output logging panel.

log_info (*message*)

Report INFO message into output logging panel.

log_ok (*message*)

Report OK message into output logging panel.

log_warning (*message*)

Report WARNING message into output logging panel.

make_bar_chart ()

PyRevitOutputChart: Return bar chart object.

make_bubble_chart ()

PyRevitOutputChart: Return bubble chart object.

make_chart ()

PyRevitOutputChart: Return chart object.

make_doughnut_chart ()

PyRevitOutputChart: Return doughnut chart object.

make_line_chart ()

PyRevitOutputChart: Return line chart object.

make_pie_chart ()

PyRevitOutputChart: Return pie chart object.

make_polar_chart ()

PyRevitOutputChart: Return polar chart object.

make_radar_chart ()

PyRevitOutputChart: Return radar chart object.

make_stacked_chart ()

PyRevitOutputChart: Return stacked chart object.

next_page ()

Add hidden next page tag to the output window.

This is helpful to silently separate the output to multiple pages for better printing.

open_page (*dest_file*)

Open html page in output window.

Parameters **dest_file** (*str*) – full path of the target html file

open_url (*dest_url*)

Open url page in output window.

Parameters **dest_url** (*str*) – web url of the target page

output_id

Return id of the output window.

In current implementation, Id of output window is equal to the unique id of the pyRevit command it belongs to. This means that all output windows belonging to the same pyRevit command, will have identical `output_id` values.

Type str

output_uniqueid

Return unique id of the output window.

In current implementation, unique id of output window is a GUID string generated when the output window is opened. This id is unique to the instance of output window.

Type str

static print_code (*code_str*)

Print code to the output window with special formatting.

Example

```
>>> output = pyrevit.output.get_output()
>>> output.print_code('value = 12')
```

static print_html (*html_str*)

Add the html code to the output window.

Example

```
>>> output = pyrevit.output.get_output()
>>> output.print_html('<strong>Title</strong>')
```

static print_md (*md_str*)

Process markdown code and print to output window.

Example

```
>>> output = pyrevit.output.get_output()
>>> output.print_md('### Title')
```

print_table (*table_data*, *columns=None*, *formats=None*, *title=""*, *last_line_style=""*)

Print provided data in a table in output window.

Parameters

- **table_data** (*list* of iterables) – 2D array of data
- **title** (*str*) – table title
- **columns** (*list* str) – list of column names
- **formats** (*list* str) – column data formats
- **last_line_style** (*str*) – css style of last row

Example

```
>>> data = [  
... ['row1', 'data', 'data', 80 ],  
... ['row2', 'data', 'data', 45 ],  
... ]  
>>> output.print_table(  
... table_data=data,  
... title="Example Table",  
... columns=["Row Name", "Column 1", "Column 2", "Percentage"],  
... formats=['', '', '', '{}%'],  
... last_line_style='color:red;'  
... )
```

renderer

Return html renderer inside output window.

Returns `System.Windows.Forms.WebBrowser` (In current implementation)

reset_icon()

Sets icon on the output window.

reset_progress()

Reset output window progress bar to zero.

resize(width, height)

Resize window to the new width and height.

save_contents(dest_file)

Save html code of the window.

Parameters `dest_file` (*str*) – full path of the destination html file

self_destruct(seconds)

Set self-destruct (close window) timer.

Parameters `seconds` (*int*) – number of seconds after which window is closed.

set_font(font_family, font_size)

Set window font family to the new font family and size.

Parameters

- **font_family** (*str*) – font family name e.g. ‘Courier New’
- **font_size** (*int*) – font size e.g. 16

set_height(height)

Set window height to the new height.

set_icon(iconpath)

Sets icon on the output window.

set_title(new_title)

Set window title to the new title.

set_width(width)

Set window width to the new width.

show()

Show the window.

show_logpanel ()
Show output window logging panel.

unfreeze ()
Unfreeze output content update.

unhide_progress ()
Unhide output window progress bar.

unlock_size ()
Unock window size.

update_progress (cur_value, max_value)
Activate and update the output window progress bar.

Parameters

- **cur_value** (*float*) – current progress value e.g. 50
- **max_value** (*float*) – total value e.g. 100

Example

```
>>> output = pyrevit.output.get_output ()
>>> for i in range(100):
>>>     output.update_progress(i, 100)
```

window
Return output window object.

Type `PyRevitBaseClasses.ScriptOutput`

`pyrevit.output.docclosing_eventhandler (sender, args)`
Close all output window on document closing.

`pyrevit.output.get_default_stylesheet ()`
Return default css stylesheet used by output window.

`pyrevit.output.get_output ()`
`pyrevit.output.PyRevitOutputWindow` : Return output window.

`pyrevit.output.get_stylesheet ()`
Return active css stylesheet used by output window.

`pyrevit.output.reset_stylesheet ()`
Reset active stylesheet to default.

`pyrevit.output.set_stylesheet (stylesheet)`
Set active css stylesheet used by output window.

Parameters **stylesheet** (*str*) – full path to stylesheet file

`pyrevit.output.setup_output_closer ()`
Setup document closing event listener.

19.2 pyrevit.output.linkmaker

Handle creation of output window helper links.

`pyrevit.output.linkmaker.make_link(element_ids, contents=None)`
Create link for given element ids.

This link is a special format link with `revit://` scheme that is handled by the output window to select the provided element ids in current project. Scripts should not call this function directly. Creating clickable element links is handled by the output wrapper object through the `linkify()` method.

Example

```
>>> output = pyrevit.output.get_output()
>>> for idx, elid in enumerate(element_ids):
>>>     print('{}: {}'.format(idx+1, output.linkify(elid)))
```

Functions for frequently performed Revit tasks or queries.

20.1 pyrevit.revit.files

Helper functions for working with revit files.

`pyrevit.revit.files.cleanup_backups` (*main_revitfile*)
Remove all incremental saves of the given Revit file.

20.2 pyrevit.revit.geom

Geometric utilities for Revit.

`pyrevit.revit.geom.convert_point_coord_system` (*rvt_point*, *rvt_transform*)
Return coordinates of point in another coordinate system.

Parameters

- **rvt_point** (*DB.XYZ*) – Revit point
- **rvt_transform** (*DB.Transform*) – Revit transform for target coord system

Returns Point coordinates in new coordinate system.

Return type *DB.XYZ*

`pyrevit.revit.geom.convert_point_to_metric` (*rvt_point*)
Convert given point coordinates to metric.

20.3 pyrevit.revit.serverutils

Helper functions for working with revit server.

class `pyrevit.revit.serverutils.SyncHistory` (*index, userid, timestamp*)
namedtuple for model sync history data in revit server

index

row index in history db

Type int

userid

user identifier

Type str

timestamp

time stamp string (e.g. "2017-12-13 19:56:20")

Type str

index

Alias for field number 0

timestamp

Alias for field number 2

userid

Alias for field number 1

`pyrevit.revit.serverutils.get_model_sync_history` (*server_path*)

Read model sync history from revit server sqlite history file.

Parameters `server_path` (*str*) – directory path of revit server filestore

Returns list of SyncHistory instances

Return type list`(``SyncHistory``)

Example

```
>>> get_model_sync_history("//servername/path/to/model.rvt")
... [SyncHistory(index=498, userid="user",
...           timestamp="2017-12-13 19:56:20")]
```

`pyrevit.revit.serverutils.get_server_path` (*doc, path_dict*)

Return file path of a model hosted on revit server.

Parameters

- **doc** (*Document*) – revit document object
- **path_dict** (*dict*) – dict of RSN paths and their directory paths

Example

```
>>> rsn_paths = {'RSN://SERVERNAME': '//servername/filestore'}
>>> get_server_path(doc, rsn_paths)
... "//servername/filestore/path/to/model.rvt"
```

20.4 pyrevit.revit.units

Unit conversion utilities for Revit.

`pyrevit.revit.units.format_area` (*area_value*, *doc=None*)

Return formatted area value in document units.

Parameters

- **area_value** (*float*) – area value
- **doc** (*DB.Document*, *optional*) – Revit document, defaults to current

Returns formatted value

Return type str

See *pyrevit.script Module*

Provide basic utilities for pyRevit scripts.

Example

```
>>> from pyrevit import script
>>> script.clipboard_copy('some text')
>>> data = script.journal_read('data-key')
>>> script.exit()
```

`pyrevit.script.clipboard_copy(string_to_copy)`
Copy string to Windows Clipboard.

`pyrevit.script.exit()`
Stop the script execution and exit.

`pyrevit.script.get_all_buttons()`
Find and return all ui buttons matching current script command name.

Sometimes tools are duplicated across extensions for user access control so this would help smart buttons to find all the loaded buttons and make icon adjustments.

Returns list of ui button objects

Return type list (pyrevit.coreutils.ribbon._PyRevitRibbonButton)

`pyrevit.script.get_alt_script_path()`
Return alternate script path of the current pyRevit command.

Returns alternate script path

Return type str

`pyrevit.script.get_bundle_file(file_name)`
Return full path to file under current script bundle.

Parameters `file_name` (*str*) – bundle file name

Returns full bundle file path

Return type `str`

`pyrevit.script.get_bundle_files` (*sub_path=None*)

Return full path to all file under current script bundle.

Returns list of bundle file paths

Return type `list[str]`

`pyrevit.script.get_bundle_name` ()

Return bundle name of the current pyRevit command.

Returns bundle name (e.g. MyButton.pushbutton)

Return type `str`

`pyrevit.script.get_button` ()

Find and return current script ui button.

Returns ui button object

Return type `pyrevit.coreutils.ribbon._PyRevitRibbonButton`

`pyrevit.script.get_config` (*section=None*)

Create and return config section parser object for current script.

Parameters `section` (*str, optional*) – config section name

Returns Config section parser object

Return type `pyrevit.coreutils.configparser.PyRevitConfigSectionParser`

`pyrevit.script.get_data_file` (*file_id, file_ext, add_cmd_name=False*)

Return filename to be used by a user script to store data.

File name is generated in this format: `pyRevit_{Revit Version}_{file_id}.{file_ext}`

Example

```
>>> script.get_data_file('mydata', 'data')
'../pyRevit_2018_mydata.data'
>>> script.get_data_file('mydata', 'data', add_cmd_name=True)
'../pyRevit_2018_Command Name_mydata.data'
```

Data files are not cleaned up at pyRevit startup. Script should manage cleaning up these files.

Parameters

- **file_id** (*str*) – unique id for the filename
- **file_ext** (*str*) – file extension
- **add_cmd_name** (*bool, optional*) – add command name to file name

Returns full file path

Return type `str`

`pyrevit.script.get_document_data_file` (*file_id, file_ext, add_cmd_name=False*)

Return filename to be used by a user script to store data.

File name is generated in this format: pyRevit_{Revit Version}_{file_id}_{Project Name}.
{file_ext}

Example

```
>>> script.get_document_data_file('mydata', 'data')
'../pyRevit_2018_mydata_Project1.data'
>>> script.get_document_data_file('mydata', 'data', add_cmd_name=True)
'../pyRevit_2018_Command Name_mydata_Project1.data'
```

Document data files are not cleaned up at pyRevit startup. Script should manage cleaning up these files.

Parameters

- **file_id** (*str*) – unique id for the filename
- **file_ext** (*str*) – file extension
- **add_cmd_name** (*bool, optional*) – add command name to file name

Returns full file path

Return type str

pyrevit.script.get_envvar(*envvar*)

Return value of give pyRevit environment variable.

The environment variable system is used to retain small values in memory between script runs (e.g. active/inactive state for toggle tools). Do not store large objects in memory using this method. List of currently set environment variables could be sees in pyRevit settings window.

Parameters **envvar** (*str*) – name of environment variable

Returns type of object stored in environment variable

Return type any

Example

```
>>> script.get_envvar('ToolActiveState')
True
```

pyrevit.script.get_extension_name()

Return extension name of the current pyRevit command.

Returns extension name (e.g. MyExtension.extension)

Return type str

pyrevit.script.get_info()

Return info on current pyRevit command.

Returns Command info object

Return type pyrevit.extensions.genericcomps.GenericUICommand

pyrevit.script.get_instance_data_file(*file_id, add_cmd_name=False*)

Return filename to be used by a user script to store data.

File name is generated in this format: pyRevit_{Revit Version}_{Process Id}_{file_id}.
{file_ext}

Example

```
>>> script.get_instance_data_file('mydata')
'.../pyRevit_2018_6684_mydata.tmp'
>>> script.get_instance_data_file('mydata', add_cmd_name=True)
'.../pyRevit_2018_6684_Command Name_mydata.tmp'
```

Instance data files are cleaned up at pyRevit startup.

Parameters

- **file_id** (*str*) – unique id for the filename
- **add_cmd_name** (*bool, optional*) – add command name to file name

Returns full file path

Return type *str*

`pyrevit.script.get_logger()`

Create and return logger named for current script.

Returns Logger object

Return type *pyrevit.coreutils.logger.LoggerWrapper*

`pyrevit.script.get_output()`

Return object wrapping output window for current script.

Returns Output wrapper object

Return type *pyrevit.output.PyRevitOutputWindow*

`pyrevit.script.get_pyrevit_version()`

Return pyRevit version.

Returns pyRevit version provider

Return type *pyrevit.versionmgr._PyRevitVersion*

`pyrevit.script.get_results()`

Return command results dictionary for logging.

Returns Command results dict

Return type *pyrevit.usagelog.record.CommandCustomResults*

`pyrevit.script.get_script_path()`

Return script path of the current pyRevit command.

Returns script path

Return type *str*

`pyrevit.script.get_unique_id()`

Return unique id of the current pyRevit command.

Returns command unique id

Return type *str*

`pyrevit.script.get_universal_data_file(file_id, file_ext, add_cmd_name=False)`

Return filename to be used by a user script to store data.

File name is generated in this format: `pyRevit_{file_id}.{file_ext}`

Example

```
>>> script.get_universal_data_file('mydata', 'data')
'.../pyRevit_mydata.data'
>>> script.get_universal_data_file('mydata', 'data', add_cmd_name=True)
'.../pyRevit_Command Name_mydata.data'
```

Universal data files are not cleaned up at pyRevit startup. Script should manage cleaning up these files.

Parameters

- **file_id** (*str*) – unique id for the filename
- **file_ext** (*str*) – file extension
- **add_cmd_name** (*bool, optional*) – add command name to file name

Returns full file path

Return type *str*

`pyrevit.script.journal_read` (*data_key*)

Read value for provided key from active Revit journal.

Parameters **data_key** (*str*) – data key

Returns data value string

Return type *str*

`pyrevit.script.journal_write` (*data_key, msg*)

Write key and value to active Revit journal for current command.

Parameters

- **data_key** (*str*) – data key
- **msg** (*str*) – data value string

`pyrevit.script.load_index` (*index_file='index.html'*)

Load html file into output window.

This method expects index.html file in the current command bundle, unless full path to an html file is provided.

Parameters **index_file** (*str, optional*) – full path of html file.

`pyrevit.script.open_url` (*url*)

Open url in a new tab in default webbrowser.

`pyrevit.script.reset_config` (*section=None*)

Reset pyRevit config.

Script should call this to reset any save configuration by removing section related to current script.

Parameters **section** (*str, optional*) – config section name

`pyrevit.script.save_config` ()

Save pyRevit config.

Scripts should call this to save any changes they have done to their config section object received from `script.get_config()` method.

`pyrevit.script.set_envvar` (*envvar, value*)

Set value of give pyRevit environment variable.

The environment variable system is used to retain small values in memory between script runs (e.g. active/inactive state for toggle tools). Do not store large objects in memory using this method. List of currently set environment variables could be seen in pyRevit settings window.

Parameters

- **envvar** (*str*) – name of environment variable
- **value** (*any*) – value of environment variable

Example

```
>>> script.set_envvar('ToolActiveState', False)
>>> script.get_envvar('ToolActiveState')
False
```

`pyrevit.script.show_file_in_explorer` (*file_path*)
Show file in Windows Explorer.

`pyrevit.script.show_folder_in_explorer` (*folder_path*)
Show folder in Windows Explorer.

`pyrevit.script.toggle_icon` (*new_state*, *on_icon_path=None*, *off_icon_path=None*)
Set the state of button icon (on or off).

This method expects `on.png` and `off.png` in command bundle for on and off icon states, unless full path of icon states are provided.

Parameters

- **new_state** (*bool*) – state of the ui button icon.
- **on_icon_path** (*str*, *optional*) – full path of icon for on state. default='on.png'
- **off_icon_path** (*str*, *optional*) – full path of icon for off state. default='off.png'

Handle reading and parsing, writing and saving of all user configurations.

This module handles the reading and writing of the pyRevit configuration files. It's been used extensively by pyRevit sub-modules. `user_config` is set up automatically in the global scope by this module and can be imported into scripts and other modules to access the default configurations.

All other modules use this module to query user config.

Example

```
>>> from pyrevit.userconfig import user_config
>>> user_config.add_section('newsection')
>>> user_config.newsection.property = value
>>> user_config.newsection.get('property', default_value)
>>> user_config.save_changes()
```

The `user_config` object is also the destination for reading and writing configuration by pyRevit scripts through `get_config()` of `pyrevit.script` module. Here is the function source:

```
def get_config(section=None):
    """Create and return config section parser object for current script.

    Args:
        section (str, optional): config section name

    Returns:
        :obj:`pyrevit.coreutils.configparser.PyRevitConfigSectionParser`:
            Config section parser object
    """
    from pyrevit.userconfig import user_config
    if not section:
        script_cfg_postfix = 'config'
        section = EXEC_PARAMS.command_name + script_cfg_postfix
```

(continues on next page)

(continued from previous page)

```

try:
    return user_config.get_section(section)
except Exception:
    return user_config.add_section(section)

```

Example

```

>>> from pyrevit import script
>>> cfg = script.get_config()
>>> cfg.property = value
>>> cfg.get('property', default_value)
>>> script.save_config()

```

class `pyrevit.userconfig.PyRevitConfig` (*cfg_file_path=None, config_type='Unknown'*)
Provide read/write access to pyRevit configuration.

Parameters

- **cfg_file_path** (*str*) – full path to config file to be used.
- **config_type** (*str*) – type of config file

Example

```

>>> cfg = PyRevitConfig(cfg_file_path)
>>> cfg.add_section('sectionname')
>>> cfg.sectionname.property = value
>>> cfg.sectionname.get('property', default_value)
>>> cfg.save_changes()

```

config_file

Current config file path.

get_active_cpython_engine ()

Return active cpython engine.

get_config_version ()

Return version of config file used for change detection.

static get_current_attachment ()

Return current pyRevit attachment.

get_ext_root_dirs ()

Return a list of all extension directories.

Returns list of strings. user extension directories.

Return type list

static get_list_separator ()

Get list separator defined in user os regional settings.

get_thirdparty_ext_root_dirs (*include_default=True*)

Return a list of external extension directories set by the user.

Returns list of strings. External user extension directories.

Return type `list`

save_changes ()

Save user config into associated config file.

set_thirdparty_ext_root_dirs (*path_list*)

Updates list of external extension directories in config file

Parameters `path_list` (*list[str]*) – list of external extension paths

`pyrevit.userconfig.find_config_file` (*target_path*)

Find config file in target path.

`pyrevit.userconfig.verify_configs` (*config_file_path=None*)

Create a user settings file.

if `config_file_path` is not provided, configs will be in memory only

Parameters `config_file_path` (*str, optional*) – config file full name and path

Returns `pyRevit` config file handler

Return type `pyrevit.userconfig.PyRevitConfig`

Utility functions for managing pyRevit versions.

Example

```
>>> from pyrevit import versionmgr
>>> v = versionmgr.get_pyrevit_version()
>>> v.get_formatted()
... '4.10-beta2'
```

class `pyrevit.versionmgr._PyRevitVersion` (*patch_number*)
pyRevit version wrapper.

Parameters `patch_number` (*str*) – patch value

as_int_tuple ()
Returns version as an int tuple (major, minor, patch)

as_str_tuple ()
Returns version as a string tuple ('major', 'minor', 'patch')

get_formatted (*nopatch=False*)
Returns 'major.minor:patch' in string

`pyrevit.versionmgr.get_pyrevit_cli_version` ()
Return version of shipped pyRevit CLI utility.

Returns version string of pyRevit CLI utility binary

Return type `str`

`pyrevit.versionmgr.get_pyrevit_repo` ()
Return pyRevit repository.

Returns repo wrapper object

Return type `pyrevit.coreutils.git.RepoInfo`

`pyrevit.versionmgr.get_pyrevit_version()`
Return information about active pyRevit version.

Returns version wrapper object

Return type `_PyRevitVersion`

23.1 pyrevit.versionmgr.about

Utility module for pyRevit project information.

Example

```
>>> from pyrevit.versionmgr import about
>>> a = about.get_pyrevit_about()
>>> a.subtitle
... 'python RAD Environment for Autodesk Revit®'
>>> a.copyright
... '© 2014-2019 Ehsan Iran-Nejad'
```

class `pyrevit.versionmgr.about.PyRevitAbout` (*subtitle, madein, copyright*)
pyRevit project info tuple.

subtitle
project subtitle

Type `str`

madein
project made-in info

Type `str`

copyright
project copyright info

Type `str`

copyright
Alias for field number 2

madein
Alias for field number 1

subtitle
Alias for field number 0

`pyrevit.versionmgr.about.get_pyrevit_about()`
Return information about pyRevit project.

Returns pyRevit project info tuple

Return type `PyRevitAbout`

23.2 pyrevit.versionmgr.updater

Handle updating pyRevit repository and its extensions.

`pyrevit.versionmgr.updater.check_for_updates()`

Check whether any available repo has pending updates.

`pyrevit.versionmgr.updater.get_all_extension_repos()`

Return a list of repos for all installed extensions.

`pyrevit.versionmgr.updater.get_thirdparty_ext_repos()`

Return a list of repos for installed third-party extensions.

`pyrevit.versionmgr.updater.get_updates(repo_info)`

Fetch updates on repository.

Parameters `repo_info` (`pyrevit.coreutils.git.RepoInfo`) – repository info wrapper object

`pyrevit.versionmgr.updater.has_core_updates()`

Check whether pyRevit repo has core updates.

This would require host application to be closed to release the file lock of core DLLs so they can be updated separately.

`pyrevit.versionmgr.updater.has_pending_updates(repo_info)`

Check for updates on repository.

Parameters `repo_info` (`pyrevit.coreutils.git.RepoInfo`) – repository info wrapper object

`pyrevit.versionmgr.updater.update_pyrevit()`

Update pyrevit and its extension repositories.

`pyrevit.versionmgr.updater.update_repo(repo_info)`

Update repository.

Parameters `repo_info` (`pyrevit.coreutils.git.RepoInfo`) – repository info wrapper object

23.3 pyrevit.versionmgr.upgrade

Perform upgrades between version, e.g. adding a new config parameter

`pyrevit.versionmgr.upgrade.upgrade_existing_pyrevit()`

Upgrade existing pyRevit deployment.

`pyrevit.versionmgr.upgrade.upgrade_user_config(user_config)`

Upgrade user configurations.

Parameters

- **user_config** (`pyrevit.userconfig.PyRevitConfig`) – config object
- **val** (`type`) – desc

p

- pyrevit, 65
- pyrevit.api, 71
- pyrevit.compat, 73
- pyrevit.coreutils, 75
- pyrevit.coreutils.appdata, 88
- pyrevit.coreutils.charts, 90
- pyrevit.coreutils.colors, 91
- pyrevit.coreutils.configparser, 102
- pyrevit.coreutils.dotnetcompiler, 103
- pyrevit.coreutils.envvars, 103
- pyrevit.coreutils.git, 104
- pyrevit.coreutils.loadertypes, 106
- pyrevit.coreutils.logger, 106
- pyrevit.coreutils.mathnet, 108
- pyrevit.coreutils.moduleutils, 108
- pyrevit.coreutils.pyutils, 108
- pyrevit.coreutils.ribbon, 110
- pyrevit.forms, 115
- pyrevit.forms.toaster, 134
- pyrevit.forms.utils, 135
- pyrevit.framework, 137
- pyrevit.loader, 139
- pyrevit.loader.asmmaker, 139
- pyrevit.loader.sessioninfo, 139
- pyrevit.loader.sessionmgr, 141
- pyrevit.loader.systemdiag, 141
- pyrevit.loader.uimaker, 142
- pyrevit.output, 143
- pyrevit.output.linkmaker, 149
- pyrevit.revit.files, 151
- pyrevit.revit.geom, 151
- pyrevit.revit.serverutils, 152
- pyrevit.revit.units, 153
- pyrevit.script, 155
- pyrevit.userconfig, 161
- pyrevit.versionmgr, 165
- pyrevit.versionmgr.about, 166
- pyrevit.versionmgr.updater, 166

- pyrevit.versionmgr.upgrade, 167

Symbols

- _ExecutorParams (class in pyrevit), 67
 _HostAppPostableCommand (class in pyrevit), 65
 _HostApplication (class in pyrevit), 66
 _PyRevitVersion (class in pyrevit.versionmgr), 165
- ### A
- activate() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 111
 activate() (pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer method), 113
 activeview (pyrevit._HostApplication attribute), 66
 add_section() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
 add_style() (pyrevit.output.PyRevitOutputWindow method), 143
 add_subsection() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 102
 alert() (in module pyrevit.forms), 123
 alert_ifnot() (in module pyrevit.forms), 124
 app (pyrevit._HostApplication attribute), 66
 as_int_tuple() (pyrevit.versionmgr._PyRevitVersion method), 165
 as_str_tuple() (pyrevit.versionmgr._PyRevitVersion method), 165
 ask_for_date() (in module pyrevit.forms), 125
 ask_for_one_item() (in module pyrevit.forms), 125
 ask_for_string() (in module pyrevit.forms), 126
 ask_for_unique_string() (in module pyrevit.forms), 126
 AssembledExtension (class in pyrevit.loader.sessionmgr), 141
 assm (pyrevit.loader.sessionmgr.AssembledExtension attribute), 141
 available_servers (pyrevit._HostApplication attribute), 66
- ### B
- bitmap_from_file() (in module pyrevit.forms.utils), 135
 blue (pyrevit.coreutils.colors.RGB attribute), 92
 branch (pyrevit.coreutils.git.RepoInfo attribute), 105
 build(pyrevit._HostApplication attribute), 66
 button() (pyrevit.coreutils.ribbon.RevitNativeRibbonGroupItem method), 113
 button_select() (pyrevit.forms.SelectFromList method), 121
 ButtonIcons (class in pyrevit.coreutils.ribbon), 110
- ### C
- calculate_dir_hash() (in module pyrevit.coreutils), 77
 callHandlers() (pyrevit.coreutils.logger.LoggerWrapper method), 106
 can_access_url() (in module pyrevit.coreutils), 77
 chart_type (pyrevit.coreutils.charts.PyRevitOutputChart attribute), 90
 check_all() (pyrevit.forms.SelectFromList method), 121
 check_familydoc() (in module pyrevit.forms), 127
 check_for_updates() (in module pyrevit.versionmgr.updater), 166
 check_icon_size() (pyrevit.coreutils.ribbon.ButtonIcons method), 110
 check_internet_connection() (in module pyrevit.coreutils), 77
 check_modeldoc() (in module pyrevit.forms), 127
 check_revittxt_encoding() (in module pyrevit.coreutils), 77
 check_selected() (pyrevit.forms.SelectFromList method), 121
 check_selection() (in module pyrevit.forms), 127

- [check_utf8bom_encoding\(\)](#) (in module `pyrevit.coreutils`), 77
[check_workshared\(\)](#) (in module `pyrevit.forms`), 128
[checkable](#) (`pyrevit.forms.TemplateListItem` attribute), 121
[cleanup_appdata_folder\(\)](#) (in module `pyrevit.coreutils.appdata`), 88
[cleanup_backups\(\)](#) (in module `pyrevit.revit.files`), 151
[cleanup_filename\(\)](#) (in module `pyrevit.coreutils`), 77
[cleanup_string\(\)](#) (in module `pyrevit.coreutils`), 78
[clear_search\(\)](#) (`pyrevit.forms.SelectFromList` method), 121
[clicked_cancel\(\)](#) (`pyrevit.forms.ProgressBar` method), 117
[clipboard_copy\(\)](#) (in module `pyrevit.script`), 155
[close\(\)](#) (`pyrevit.output.PyRevitOutputWindow` method), 144
[close_others\(\)](#) (`pyrevit.output.PyRevitOutputWindow` method), 144
[command_alt_path](#) (`pyrevit._ExecutorParams` attribute), 67
[command_bundle](#) (`pyrevit._ExecutorParams` attribute), 68
[command_data](#) (`pyrevit._ExecutorParams` attribute), 68
[command_extension](#) (`pyrevit._ExecutorParams` attribute), 68
[command_mode](#) (`pyrevit._ExecutorParams` attribute), 68
[command_name](#) (`pyrevit._ExecutorParams` attribute), 68
[command_path](#) (`pyrevit._ExecutorParams` attribute), 68
[command_uniqueid](#) (`pyrevit._ExecutorParams` attribute), 68
[CommandSwitchWindow](#) (class in `pyrevit.forms`), 115
[compare_branch_heads\(\)](#) (in module `pyrevit.coreutils.git`), 105
[compare_lists\(\)](#) (in module `pyrevit.coreutils.pyutils`), 109
[compile_csharp\(\)](#) (in module `pyrevit.coreutils.dotnetcompiler`), 103
[config_file](#) (`pyrevit.userconfig.PyRevitConfig` attribute), 162
[contains\(\)](#) (`pyrevit.coreutils.ribbon.GenericPyRevitUIContainer` method), 111
[convert_point_coord_system\(\)](#) (in module `pyrevit.revit.geom`), 151
[convert_point_to_metric\(\)](#) (in module `pyrevit.revit.geom`), 151
[copy\(\)](#) (`pyrevit.coreutils.pyutils.DefaultOrderedDict` method), 109
[copy_func\(\)](#) (in module `pyrevit.coreutils.moduleutils`), 108
[copyright](#) (`pyrevit.versionmgr.about.PyRevitAbout` attribute), 166
[correct_revittxt_encoding\(\)](#) (in module `pyrevit.coreutils`), 78
[create_assembly\(\)](#) (in module `pyrevit.loader.asmmaker`), 139
[create_bitmap\(\)](#) (`pyrevit.coreutils.ribbon.ButtonIcons` method), 110
[create_ext_command_attrs\(\)](#) (in module `pyrevit.coreutils`), 78
[create_type\(\)](#) (in module `pyrevit.coreutils`), 78
[current_date\(\)](#) (in module `pyrevit.coreutils`), 79
[current_time\(\)](#) (in module `pyrevit.coreutils`), 79
- ## D
- [deactivate\(\)](#) (`pyrevit.coreutils.ribbon.GenericPyRevitUIContainer` method), 111
[deactivate\(\)](#) (`pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer` method), 113
[debug_mode](#) (`pyrevit.output.PyRevitOutputWindow` attribute), 144
[decrement_str\(\)](#) (in module `pyrevit.coreutils`), 79
[DefaultOrderedDict](#) (class in `pyrevit.coreutils.pyutils`), 108
[deprecate\(\)](#) (`pyrevit.coreutils.logger.LoggerWrapper` method), 106
[directory](#) (`pyrevit.coreutils.git.RepoInfo` attribute), 104
[disable_element\(\)](#) (`pyrevit.forms.WPFWindow` static method), 122
[DispatchingFormatter](#) (class in `pyrevit.coreutils.logger`), 106
[dletter_to_unc\(\)](#) (in module `pyrevit.coreutils`), 80
[doc](#) (`pyrevit._HostApplication` attribute), 66
[doc_mode](#) (`pyrevit._ExecutorParams` attribute), 68
[docclosing_eventhandler\(\)](#) (in module `pyrevit.output`), 149
[docs](#) (`pyrevit._HostApplication` attribute), 66
[draw\(\)](#) (`pyrevit.coreutils.charts.PyRevitOutputChart` method), 90
[enable_element\(\)](#) (`pyrevit.forms.WPFWindow` static method), 123
[enabled](#) (`pyrevit.coreutils.ribbon.GenericPyRevitUIContainer` attribute), 111
[engine_mgr](#) (`pyrevit._ExecutorParams` attribute), 68
[engine_ver](#) (`pyrevit._ExecutorParams` attribute), 68

- engine_version (pyrevit.loader.sessioninfo.RuntimeInfo attribute), 140
- execute_command() (in module pyrevit.loader.sessionmgr), 141
- execute_script() (in module pyrevit.loader.sessionmgr), 141
- executed_from_ui (pyrevit._ExecutorParams attribute), 68
- exit() (in module pyrevit.script), 155
- ext (pyrevit.loader.sessionmgr.AssembledExtension attribute), 141
- ExtensionAssemblyInfo (class in pyrevit.loader.asmmaker), 139
- extract_guid() (in module pyrevit.coreutils), 80
- extract_param() (pyrevit.coreutils.ScriptFileParser method), 76
- extract_range() (in module pyrevit.coreutils), 80
- ## F
- filestream (pyrevit.coreutils.ribbon.ButtonIcons attribute), 110
- FileWatcher (class in pyrevit.coreutils), 75
- filter_null_items() (in module pyrevit.coreutils), 80
- find_child() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 111
- find_config_file() (in module pyrevit.userconfig), 163
- find_direct_match() (pyrevit.forms.SearchPrompt method), 119
- find_loaded_asm() (in module pyrevit.coreutils), 80
- find_pyrevitcmd() (in module pyrevit.loader.sessionmgr), 142
- find_type_by_name() (in module pyrevit.coreutils), 81
- find_word_match() (pyrevit.forms.SearchPrompt method), 119
- first_load (pyrevit._ExecutorParams attribute), 68
- forced_debug_mode (pyrevit._ExecutorParams attribute), 68
- format() (pyrevit.coreutils.logger.DispatchingFormatter method), 106
- format_area() (in module pyrevit.revit.units), 153
- format_hex_rgb() (in module pyrevit.coreutils), 81
- freeze() (pyrevit.output.PyRevitOutputWindow method), 144
- fully_remove_dir() (in module pyrevit.coreutils), 81
- fuzzy_search_ratio() (in module pyrevit.coreutils), 81
- ## G
- garbage_data_file() (in module pyrevit.coreutils.appdata), 88
- GenericPyRevitUIContainer (class in pyrevit.coreutils.ribbon), 111
- GenericRevitNativeUIContainer (class in pyrevit.coreutils.ribbon), 112
- get_active_cpython_engine() (pyrevit.userconfig.PyRevitConfig method), 162
- get_adwindows_object() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 111
- get_all_buttons() (in module pyrevit.script), 155
- get_all_extension_repos() (in module pyrevit.versionmgr.updater), 167
- get_all_new_commits() (in module pyrevit.coreutils.git), 105
- get_all_subclasses() (in module pyrevit.coreutils), 81
- get_alt_script_path() (in module pyrevit.script), 155
- get_bundle_file() (in module pyrevit.script), 155
- get_bundle_files() (in module pyrevit.script), 156
- get_bundle_name() (in module pyrevit.script), 156
- get_button() (in module pyrevit.script), 156
- get_canonical_parts() (in module pyrevit.coreutils), 81
- get_config() (in module pyrevit.script), 156
- get_config_file_hash() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
- get_config_version() (pyrevit.userconfig.PyRevitConfig method), 162
- get_current_attachment() (pyrevit.userconfig.PyRevitConfig static method), 162
- get_current_ui() (in module pyrevit.coreutils.ribbon), 114
- get_data_file() (in module pyrevit.coreutils.appdata), 88
- get_data_file() (in module pyrevit.script), 156
- get_default_stylesheet() (in module pyrevit.output), 149
- get_dll_file() (in module pyrevit.framework), 137
- get_docstring() (pyrevit.coreutils.ScriptFileParser method), 76
- get_document_data_file() (in module pyrevit.script), 156
- get_enum_none() (in module pyrevit.coreutils), 81
- get_enum_value() (in module pyrevit.coreutils), 82
- get_enum_values() (in module pyrevit.coreutils), 82
- get_envvar() (in module pyrevit.script), 157

- get_exe_version() (in module *pyrevit.coreutils*), 82
- get_ext_root_dirs() (pyrevit.userconfig.PyRevitConfig method), 162
- get_extension_name() (in module *pyrevit.script*), 157
- get_file_hdlr() (in module *pyrevit.coreutils.logger*), 107
- get_file_name() (in module *pyrevit.coreutils*), 82
- get_flagged_children() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 111
- get_formatted() (pyrevit.versionmgr._PyRevitVersion method), 165
- get_head_html() (pyrevit.output.PyRevitOutputWindow method), 144
- get_height() (pyrevit.output.PyRevitOutputWindow method), 144
- get_info() (in module *pyrevit.script*), 157
- get_instance_data_file() (in module *pyrevit.coreutils.appdata*), 88
- get_instance_data_file() (in module *pyrevit.script*), 157
- get_level() (pyrevit.coreutils.logger.LoggerWrapper method), 106
- get_list_separator() (pyrevit.userconfig.PyRevitConfig static method), 162
- get_loaded_pyrevit_assemblies() (in module *pyrevit.loader.sessioninfo*), 140
- get_logger() (in module *pyrevit.coreutils.logger*), 107
- get_logger() (in module *pyrevit.script*), 158
- get_mapped_drives_dict() (in module *pyrevit.coreutils*), 82
- get_model_sync_history() (in module *pyrevit.revit.serverutils*), 152
- get_option() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 102
- get_output() (in module *pyrevit.output*), 149
- get_output() (in module *pyrevit.script*), 158
- get_postable_commands() (pyrevit._HostApplication method), 66
- get_pyrevit_about() (in module *pyrevit.versionmgr.about*), 166
- get_pyrevit_cli_version() (in module *pyrevit.versionmgr*), 165
- get_pyrevit_env_var() (in module *pyrevit.coreutils.envvars*), 104
- get_pyrevit_env_vars() (in module *pyrevit.coreutils.envvars*), 104
- get_pyrevit_repo() (in module *pyrevit.versionmgr*), 165
- get_pyrevit_version() (in module *pyrevit.script*), 158
- get_pyrevit_version() (in module *pyrevit.versionmgr*), 165
- get_reg_key() (in module *pyrevit.coreutils*), 82
- get_repo() (in module *pyrevit.coreutils.git*), 105
- get_results() (in module *pyrevit.script*), 158
- get_revit_instance_count() (in module *pyrevit.coreutils*), 82
- get_runtime_info() (in module *pyrevit.loader.sessioninfo*), 140
- get_rvtapi_object() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
- get_script_path() (in module *pyrevit.script*), 158
- get_section() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
- get_server_path() (in module *pyrevit.revit.serverutils*), 152
- get_session_uuid() (in module *pyrevit.loader.sessioninfo*), 140
- get_stdout_hdlr() (in module *pyrevit.coreutils.logger*), 107
- get_str_hash() (in module *pyrevit.coreutils*), 82
- get_stylesheet() (in module *pyrevit.output*), 149
- get_sub_folders() (in module *pyrevit.coreutils*), 82
- get_subsection() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 102
- get_subsections() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 102
- get_thirdparty_ext_repos() (in module *pyrevit.versionmgr.updater*), 167
- get_thirdparty_ext_root_dirs() (pyrevit.userconfig.PyRevitConfig method), 162
- get_time() (pyrevit.coreutils.Timer method), 76
- get_title() (pyrevit.output.PyRevitOutputWindow method), 144
- get_toaster() (in module *pyrevit.forms.toaster*), 134
- get_total_loaded_assm_count() (in module *pyrevit.loader.sessioninfo*), 140
- get_type() (in module *pyrevit.framework*), 137
- get_uibutton() (in module *pyrevit.coreutils.ribbon*), 114
- get_unique_id() (in module *pyrevit.script*), 158
- get_universal_data_file() (in module *pyrevit.coreutils.appdata*), 89
- get_universal_data_file() (in module *pyrevit*), 89

- vit.script*), 158
- `get_updates()` (in module *pyrevit.versionmgr.updater*), 167
- `get_width()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `GetValueWindow` (class in *pyrevit.forms*), 116
- `git_clone()` (in module *pyrevit.coreutils.git*), 105
- `git_fetch()` (in module *pyrevit.coreutils.git*), 106
- `git_pull()` (in module *pyrevit.coreutils.git*), 106
- `green` (*pyrevit.coreutils.colors.RGB* attribute), 91
- ## H
- `handle_click()` (*pyrevit.forms.CommandSwitchWindow* method), 116
- `handle_input_key()` (*pyrevit.forms.CommandSwitchWindow* method), 116
- `handle_input_key()` (*pyrevit.forms.WPFWindow* method), 123
- `handle_kb_key()` (*pyrevit.forms.SearchPrompt* method), 119
- `handle_url_click()` (*pyrevit.forms.WPFWindow* method), 123
- `has_changed` (*pyrevit.coreutils.FileWatcher* attribute), 75
- `has_core_updates()` (in module *pyrevit.versionmgr.updater*), 167
- `has_errors()` (*pyrevit.coreutils.logger.LoggerWrapper* method), 106
- `has_nonprintable()` (in module *pyrevit.coreutils*), 82
- `has_option()` (*pyrevit.coreutils.configparser.PyRevitConfigSectionParser* method), 102
- `has_pending_updates()` (in module *pyrevit.versionmgr.updater*), 167
- `has_section()` (*pyrevit.coreutils.configparser.PyRevitConfigParser* method), 102
- `has_subsection()` (*pyrevit.coreutils.configparser.PyRevitConfigSectionParser* method), 102
- `head_name` (*pyrevit.coreutils.git.RepoInfo* attribute), 104
- `header` (*pyrevit.coreutils.configparser.PyRevitConfigSectionParser* attribute), 103
- `hex_color` (*pyrevit.coreutils.colors.RGB* attribute), 92
- `hide()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `hide_element()` (*pyrevit.forms.WPFWindow* static method), 123
- `hide_logpanel()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `hide_progress()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `host_version` (*pyrevit.loader.sessioninfo.RuntimeInfo* attribute), 140
- ## I
- `icon_file_path` (*pyrevit.coreutils.ribbon.ButtonIcons* attribute), 110
- `id` (*pyrevit._HostAppPostableCommand* attribute), 65
- `increment_str()` (in module *pyrevit.coreutils*), 83
- `indeterminate` (*pyrevit.forms.ProgressBar* attribute), 118
- `indeterminate_progress()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `index` (*pyrevit.revit.serverutils.SyncHistory* attribute), 152
- `inform_wip()` (in module *pyrevit.forms*), 128
- `inject_script()` (*pyrevit.output.PyRevitOutputWindow* method), 144
- `inject_to_body()` (*pyrevit.output.PyRevitOutputWindow* method), 145
- `inject_to_head()` (*pyrevit.output.PyRevitOutputWindow* method), 145
- `insert_divider()` (*pyrevit.output.PyRevitOutputWindow* method), 145
- `inspect_calling_scope_global_var()` (in module *pyrevit.coreutils*), 83
- `inspect_calling_scope_local_var()` (in module *pyrevit.coreutils*), 83
- `is_blank()` (in module *pyrevit.coreutils*), 83
- `is_box_visible_on_screens()` (in module *pyrevit.coreutils*), 83
- `is_checkbox()` (*pyrevit.forms.TemplateListItem* class method), 121
- `is_data_file_available()` (in module *pyrevit.coreutils.appdata*), 89
- `is_dirty()` (*pyrevit.coreutils.ribbon.GenericPyRevitUIContainer* method), 112
- `is_enabled_for()` (*pyrevit.coreutils.logger.LoggerWrapper* method), 107
- `is_exactly()` (*pyrevit._HostApplication* method), 66

is_file_available() (in module pyrevit.coreutils.appdata), 89

is_native() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer static method), 112

is_native() (pyrevit.coreutils.ribbon.GenericRevitNativeUIContainer static method), 113

is_newer_than() (pyrevit._HostApplication method), 66

is_older_than() (pyrevit._HostApplication method), 66

is_pyrevit_data_file() (in module pyrevit.coreutils.appdata), 89

is_pyrevit_tab() (pyrevit.coreutils.ribbon.RevitNativeRibbonTab static method), 113

is_url_valid() (in module pyrevit.coreutils), 83

isEnabledFor() (pyrevit.coreutils.logger.LoggerWrapper method), 107

isnumber() (in module pyrevit.coreutils.pyutils), 109

istype (pyrevit.forms.ParamDef attribute), 117

itemdata_mode (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer attribute), 111

J

join_strings() (in module pyrevit.coreutils), 84

journal_read() (in module pyrevit.script), 159

journal_write() (in module pyrevit.script), 159

K

key (pyrevit._HostAppPostableCommand attribute), 65

kill_tasks() (in module pyrevit.coreutils), 84

L

large_bitmap (pyrevit.coreutils.ribbon.ButtonIcons attribute), 110

last_commit_hash (pyrevit.coreutils.git.RepoInfo attribute), 105

linkify() (pyrevit.output.PyRevitOutputWindow static method), 145

list_data_files() (in module pyrevit.coreutils.appdata), 89

list_session_data_files() (in module pyrevit.coreutils.appdata), 90

load_asm() (in module pyrevit.coreutils), 84

load_asm_file() (in module pyrevit.coreutils), 84

load_bitmapimage() (in module pyrevit.coreutils.ribbon), 114

load_component() (in module pyrevit.forms.utils), 135

load_ctrl_template() (in module pyrevit.forms.utils), 135

load_index() (in module pyrevit.script), 159

load_itemspanel_template() (in module pyrevit.forms.utils), 135

load_session() (in module pyrevit.loader.sessionmgr), 142

load_session_info (pyrevit.loader.asmmaker.ExtensionAssemblyInfo attribute), 139

lock_size() (pyrevit.output.PyRevitOutputWindow method), 146

log_error() (pyrevit.output.PyRevitOutputWindow method), 146

log_info() (pyrevit.output.PyRevitOutputWindow method), 146

log_ok() (pyrevit.output.PyRevitOutputWindow method), 146

log_warning() (pyrevit.output.PyRevitOutputWindow method), 146

loggers_have_errors() (in module pyrevit.coreutils.logger), 107

LoggerWrapper (class in pyrevit.coreutils.logger), 106

luminance (pyrevit.coreutils.colors.RGB attribute), 92

M

madein (pyrevit.versionmgr.about.PyRevitAbout attribute), 166

make_bar_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_bubble_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_canonical_name() (in module pyrevit.coreutils), 84

make_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_doughnut_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_line_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_link() (in module pyrevit.output.linkmaker), 149

make_pie_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_polar_chart() (pyrevit.output.PyRevitOutputWindow method), 146

make_radar_chart() (pyrevit.output.PyRevitOutputWindow method), 146

- make_stacked_chart() (pyrevit.output.PyRevitOutputWindow method), 146
- medium_bitmap (pyrevit.coreutils.ribbon.ButtonIcons attribute), 111
- ## N
- name (pyrevit._HostAppPostableCommand attribute), 65
- name (pyrevit.coreutils.colors.RGB attribute), 91
- name (pyrevit.coreutils.git.RepoInfo attribute), 104
- name (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer attribute), 111
- name (pyrevit.forms.ParamDef attribute), 116, 117
- name (pyrevit.forms.RevisionOption attribute), 118
- name (pyrevit.forms.SheetOption attribute), 121
- name (pyrevit.forms.TemplateListItem attribute), 121
- name (pyrevit.forms.ViewOption attribute), 122
- name (pyrevit.loader.asmmaker.ExtensionAssemblyInfo attribute), 139
- new_dataset() (pyrevit.coreutils.charts.PyRevitOutputChartData method), 90
- new_session_uuid() (in module pyrevit.loader.sessioninfo), 140
- new_uuid() (in module pyrevit.coreutils), 84
- next_page() (pyrevit.output.PyRevitOutputWindow method), 146
- number (pyrevit.forms.SheetOption attribute), 121
- ## O
- open_folder_in_explorer() (in module pyrevit.coreutils), 85
- open_page() (pyrevit.output.PyRevitOutputWindow method), 146
- open_url() (in module pyrevit.script), 159
- open_url() (pyrevit.output.PyRevitOutputWindow method), 146
- output (pyrevit.coreutils.charts.PyRevitOutputChart attribute), 90
- output_id (pyrevit.output.PyRevitOutputWindow attribute), 146
- output_uniqueid (pyrevit.output.PyRevitOutputWindow attribute), 147
- ## P
- pairwise() (in module pyrevit.coreutils.pyutils), 109
- ParamDef (class in pyrevit.forms), 116
- password (pyrevit.coreutils.git.RepoInfo attribute), 105
- pick_excel_file() (in module pyrevit.forms), 128
- pick_file() (in module pyrevit.forms), 128
- pick_folder() (in module pyrevit.forms), 129
- prepare_html_str() (in module pyrevit.coreutils), 85
- print_code() (pyrevit.output.PyRevitOutputWindow static method), 147
- print_html() (pyrevit.output.PyRevitOutputWindow static method), 147
- print_md() (pyrevit.output.PyRevitOutputWindow static method), 147
- print_table() (pyrevit.output.PyRevitOutputWindow method), 147
- proc (pyrevit._HostApplication attribute), 66
- proc_id (pyrevit._HostApplication attribute), 67
- proc_name (pyrevit._HostApplication attribute), 67
- proc_path (pyrevit._HostApplication attribute), 67
- proc_screen (pyrevit._HostApplication attribute), 67
- proc_screen_scalefactor (pyrevit._HostApplication attribute), 67
- proc_screen_workarea (pyrevit._HostApplication attribute), 67
- proc_window (pyrevit._HostApplication attribute), 67
- process_option() (pyrevit.forms.CommandSwitchWindow method), 116
- ProgressBar (class in pyrevit.forms), 117
- pyrevit (module), 65
- pyrevit.api (module), 71
- pyrevit.compat (module), 73
- pyrevit.coreutils (module), 75
- pyrevit.coreutils.appdata (module), 88
- pyrevit.coreutils.charts (module), 90
- pyrevit.coreutils.colors (module), 91
- pyrevit.coreutils.configparser (module), 102
- pyrevit.coreutils.dotnetcompiler (module), 103
- pyrevit.coreutils.envvars (module), 103
- pyrevit.coreutils.git (module), 104
- pyrevit.coreutils.loadertypes (module), 106
- pyrevit.coreutils.logger (module), 106
- pyrevit.coreutils.mathnet (module), 108
- pyrevit.coreutils.moduleutils (module), 108
- pyrevit.coreutils.pyutils (module), 108
- pyrevit.coreutils.ribbon (module), 110
- pyrevit.forms (module), 115
- pyrevit.forms.toaster (module), 134
- pyrevit.forms.utils (module), 135
- pyrevit.framework (module), 137
- pyrevit.loader (module), 139
- pyrevit.loader.asmmaker (module), 139
- pyrevit.loader.sessioninfo (module), 139
- pyrevit.loader.sessionmgr (module), 141
- pyrevit.loader.systemdiag (module), 141
- pyrevit.loader.uimaker (module), 142

pyrevit.output (module), 143
 pyrevit.output.linkmaker (module), 149
 pyrevit.revit.files (module), 151
 pyrevit.revit.geom (module), 151
 pyrevit.revit.serverutils (module), 152
 pyrevit.revit.units (module), 153
 pyrevit.script (module), 155
 pyrevit.userconfig (module), 161
 pyrevit.versionmgr (module), 165
 pyrevit.versionmgr.about (module), 166
 pyrevit.versionmgr.updater (module), 166
 pyrevit.versionmgr.upgrade (module), 167
 pyrevit_command (pyrevit._ExecutorParams attribute), 68
 pyrevit_version (pyrevit.forms.WPFWindow attribute), 123
 pyrevit_version (pyrevit.loader.sessioninfo.RuntimeInfo attribute), 140
 PyRevitAbout (class in pyrevit.versionmgr.about), 166
 PyRevitConfig (class in pyrevit.userconfig), 162
 PyRevitConfigParser (class in pyrevit.coreutils.configparser), 102
 PyRevitConfigSectionParser (class in pyrevit.coreutils.configparser), 102
 PyRevitException (class in pyrevit), 65
 PyRevitGitAuthenticationError, 104
 PyRevitIOError (class in pyrevit), 65
 PyRevitOutputChart (class in pyrevit.coreutils.charts), 90
 PyRevitOutputChartData (class in pyrevit.coreutils.charts), 90
 PyRevitOutputChartDataset (class in pyrevit.coreutils.charts), 91
 PyRevitOutputChartOptions (class in pyrevit.coreutils.charts), 91
 PyRevitOutputWindow (class in pyrevit.output), 143
 PyRevitUIError, 113

R

random_alpha() (in module pyrevit.coreutils), 85
 random_color() (in module pyrevit.coreutils), 85
 random_hex_color() (in module pyrevit.coreutils), 85
 random_rgb_color() (in module pyrevit.coreutils), 85
 random_rgba_color() (in module pyrevit.coreutils), 85
 randomize_colors() (pyrevit.coreutils.charts.PyRevitOutputChart method), 90
 read_source_file() (in module pyrevit.coreutils), 85

red (pyrevit.coreutils.colors.RGB attribute), 91
 reformat_string() (in module pyrevit.coreutils), 86
 reload() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
 reloading (pyrevit.loader.asmmaker.ExtensionAssemblyInfo attribute), 139
 remove_option() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 103
 remove_section() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
 renderer (pyrevit.output.PyRevitOutputWindow attribute), 148
 reorder_after() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
 reorder_afterall() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
 reorder_before() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
 reorder_beforeall() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
 repo (pyrevit.coreutils.git.RepoInfo attribute), 105
 RepoInfo (class in pyrevit.coreutils.git), 104
 report_env() (in module pyrevit.loader.sessioninfo), 140
 reset() (pyrevit.forms.ProgressBar method), 118
 reset_config() (in module pyrevit.script), 159
 reset_icon() (pyrevit.output.PyRevitOutputWindow method), 148
 reset_level() (pyrevit.coreutils.logger.LoggerWrapper method), 107
 reset_progress() (pyrevit.output.PyRevitOutputWindow method), 148
 reset_stylesheet() (in module pyrevit.output), 149
 resize() (pyrevit.output.PyRevitOutputWindow method), 148
 restart() (pyrevit.coreutils.Timer method), 77
 result_dict (pyrevit._ExecutorParams attribute), 69
 reverse_dict() (in module pyrevit.coreutils), 86
 reverse_html() (in module pyrevit.coreutils), 86
 RevisionOption (class in pyrevit.forms), 118
 RevitNativeRibbonButton (class in pyrevit.coreutils.ribbon), 113
 RevitNativeRibbonGroupItem (class in pyrevit.coreutils.ribbon), 113

- RevitNativeRibbonPanel (class in pyrevit.coreutils.ribbon), 113
- RevitNativeRibbonTab (class in pyrevit.coreutils.ribbon), 113
- RGB (class in pyrevit.coreutils.colors), 91
- ribbon_item() (pyrevit.coreutils.ribbon.RevitNativeRibbonPanel method), 113
- ribbon_panel() (pyrevit.coreutils.ribbon.RevitNativeRibbonTab method), 113
- run_process() (in module pyrevit.coreutils), 87
- RuntimeInfo (class in pyrevit.loader.sessioninfo), 139
- rvtobj (pyrevit._HostAppPostableCommand attribute), 66
- ## S
- safe_cast() (in module pyrevit.coreutils.pyutils), 110
- safe_text_color (pyrevit.coreutils.colors.RGB attribute), 92
- SafeDict (class in pyrevit.coreutils), 75
- save() (pyrevit.coreutils.configparser.PyRevitConfigParser method), 102
- save_changes() (pyrevit.userconfig.PyRevitConfig method), 163
- save_config() (in module pyrevit.script), 159
- save_contents() (pyrevit.output.PyRevitOutputWindow method), 148
- save_excel_file() (in module pyrevit.forms), 129
- save_file() (in module pyrevit.forms), 129
- ScriptFileParser (class in pyrevit.coreutils), 76
- search_input (pyrevit.forms.SearchPrompt attribute), 119
- search_input_parts (pyrevit.forms.SearchPrompt attribute), 119
- search_matches (pyrevit.forms.SearchPrompt attribute), 119
- search_term (pyrevit.forms.SearchPrompt attribute), 119
- search_term_args (pyrevit.forms.SearchPrompt attribute), 119
- search_term_main (pyrevit.forms.SearchPrompt attribute), 119
- search_term_switches (pyrevit.forms.SearchPrompt attribute), 119
- search_txt_changed() (pyrevit.forms.CommandSwitchWindow method), 116
- search_txt_changed() (pyrevit.forms.SearchPrompt method), 119
- search_txt_changed() (pyrevit.forms.SelectFromList method), 121
- SearchPrompt (class in pyrevit.forms), 118
- select() (pyrevit.forms.GetValueWindow method), 116
- select_image() (in module pyrevit.forms), 129
- select_open_docs() (in module pyrevit.forms), 130
- select_parameter() (in module pyrevit.forms), 130
- select_revisions() (in module pyrevit.forms), 131
- select_sheets() (in module pyrevit.forms), 131
- select_swatch() (in module pyrevit.forms), 132
- select_titleblocks() (in module pyrevit.forms), 132
- select_views() (in module pyrevit.forms), 133
- select_viewtemplates() (in module pyrevit.forms), 133
- SelectFromList (class in pyrevit.forms), 119
- self_destruct() (pyrevit.output.PyRevitOutputWindow method), 148
- send_toast() (in module pyrevit.forms.toaster), 134
- set_color() (pyrevit.coreutils.charts.PyRevitOutputChartDataset method), 91
- set_debug_mode() (pyrevit.coreutils.logger.LoggerWrapper method), 107
- set_dirty_flag() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer method), 112
- set_envvar() (in module pyrevit.script), 159
- set_file_logging() (in module pyrevit.coreutils.logger), 107
- set_font() (pyrevit.output.PyRevitOutputWindow method), 148
- set_height() (pyrevit.coreutils.charts.PyRevitOutputChart method), 90
- set_height() (pyrevit.output.PyRevitOutputWindow method), 148
- set_icon() (pyrevit.output.PyRevitOutputWindow method), 148
- set_image_source() (pyrevit.forms.WPFWindow method), 123
- set_level() (pyrevit.coreutils.logger.LoggerWrapper method), 107
- set_loaded_pyrevit_assemblies() (in module pyrevit.loader.sessioninfo), 141
- set_option() (pyrevit.coreutils.configparser.PyRevitConfigSectionParser method), 103
- set_pyrevit_env_var() (in module pyrevit.coreutils.envvars), 104
- set_quiet_mode() (pyrevit.coreutils.logger.LoggerWrapper method), 107
- set_rvtapi_object() (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer

- method*), 112
 - set_search_results() (pyrevit.forms.SearchPrompt *method*), 119
 - set_session_uuid() (in module pyrevit.loader.sessioninfo), 141
 - set_style() (pyrevit.coreutils.charts.PyRevitOutputChart *method*), 90
 - set_stylesheet() (in module pyrevit.output), 149
 - set_thirdparty_ext_root_dirs() (pyrevit.userconfig.PyRevitConfig *method*), 163
 - set_title() (pyrevit.output.PyRevitOutputWindow *method*), 148
 - set_total_loaded_assm_count() (in module pyrevit.loader.sessioninfo), 141
 - set_verbose_mode() (pyrevit.coreutils.logger.LoggerWrapper *method*), 107
 - set_width() (pyrevit.coreutils.charts.PyRevitOutputChart *method*), 90
 - set_width() (pyrevit.output.PyRevitOutputWindow *method*), 148
 - setup_icon() (pyrevit.forms.WPFWindow *method*), 123
 - setup_output_closer() (in module pyrevit.output), 149
 - setup_runtime_vars() (in module pyrevit.loader.sessioninfo), 141
 - SheetOption (class in pyrevit.forms), 121
 - show() (pyrevit.forms.SearchPrompt *class method*), 119
 - show() (pyrevit.forms.TemplateUserInputWindow *class method*), 122
 - show() (pyrevit.forms.WPFWindow *method*), 123
 - show() (pyrevit.output.PyRevitOutputWindow *method*), 148
 - show_dialog() (pyrevit.forms.WPFWindow *method*), 123
 - show_element() (pyrevit.forms.WPFWindow *static method*), 123
 - show_entry_in_explorer() (in module pyrevit.coreutils), 87
 - show_file_in_explorer() (in module pyrevit.script), 160
 - show_folder_in_explorer() (in module pyrevit.script), 160
 - show_logpanel() (pyrevit.output.PyRevitOutputWindow *method*), 148
 - small_bitmap (pyrevit.coreutils.ribbon.ButtonIcons *attribute*), 111
 - string_value_changed() (pyrevit.forms.GetValueWindow *method*), 116
 - subheader (pyrevit.coreutils.configparser.PyRevitConfigSectionParser *attribute*), 103
 - subtitle (pyrevit.versionmgr.about.PyRevitAbout *attribute*), 166
 - subversion (pyrevit._HostApplication *attribute*), 67
 - SyncHistory (class in pyrevit.revit.serverutils), 152
 - system_diag() (in module pyrevit.loader.systemdiag), 141
- ## T
- TemplateListItem (class in pyrevit.forms), 121
 - TemplatePromptBar (class in pyrevit.forms), 121
 - TemplateUserInputWindow (class in pyrevit.forms), 122
 - Timer (class in pyrevit.coreutils), 76
 - timestamp (pyrevit.revit.serverutils.SyncHistory *attribute*), 152
 - timestamp() (in module pyrevit.coreutils), 87
 - title (pyrevit.forms.ProgressBar *attribute*), 118
 - toast() (in module pyrevit.forms), 134
 - toggle_all() (pyrevit.forms.SelectFromList *method*), 121
 - toggle_element() (pyrevit.forms.WPFWindow *static method*), 123
 - toggle_icon() (in module pyrevit.script), 160
 - touch() (in module pyrevit.coreutils), 87
- ## U
- uiapp (pyrevit._HostApplication *attribute*), 67
 - uidoc (pyrevit._HostApplication *attribute*), 67
 - unc_to_dletter() (in module pyrevit.coreutils), 87
 - uncheck_all() (pyrevit.forms.SelectFromList *method*), 121
 - uncheck_selected() (pyrevit.forms.SelectFromList *method*), 121
 - unfreeze() (pyrevit.output.PyRevitOutputWindow *method*), 149
 - unhide_progress() (pyrevit.output.PyRevitOutputWindow *method*), 149
 - unlock_size() (pyrevit.output.PyRevitOutputWindow *method*), 149
 - unwrap() (pyrevit.forms.TemplateListItem *method*), 121
 - update_progress() (pyrevit.forms.ProgressBar *method*), 118
 - update_progress() (pyrevit.output.PyRevitOutputWindow *method*), 149
 - update_pyrevit() (in module pyrevit.versionmgr.updater), 167
 - update_pyrevit_ui() (in module pyrevit.loader.uimaker), 142
 - update_repo() (in module pyrevit.versionmgr.updater), 167

update_results_display() (pyrevit.forms.SearchPrompt method), 119
 update_tstamp() (pyrevit.coreutils.FileWatcher method), 75
 update_window() (pyrevit.forms.TemplatePromptBar method), 121
 upgrade_existing_pyrevit() (in module pyrevit.versionmgr.upgrade), 167
 upgrade_user_config() (in module pyrevit.versionmgr.upgrade), 167
 userid (pyrevit.revit.serverutils.SyncHistory attribute), 152
 username (pyrevit._HostApplication attribute), 67
 username (pyrevit.coreutils.git.RepoInfo attribute), 105

V

verify_configs() (in module pyrevit.userconfig), 163
 verify_directory() (in module pyrevit.coreutils), 87
 version (pyrevit._HostApplication attribute), 67
 version_name (pyrevit._HostApplication attribute), 67
 ViewOption (class in pyrevit.forms), 122
 visible (pyrevit.coreutils.ribbon.GenericPyRevitUIContainer attribute), 112

W

wait_async() (pyrevit.forms.ProgressBar method), 118
 WarningBar (class in pyrevit.forms), 123
 window (pyrevit.output.PyRevitOutputWindow attribute), 149
 window_handle (pyrevit._ExecutorParams attribute), 69
 WPFWindow (class in pyrevit.forms), 122