kniteditor Documentation

Release 5

AllYarnsAreBeautiful
FOSSASIA

Jan 15, 2018

Contents

Contents:

CHAPTER 1

kniteditor Installation Instructions

This document aims at showing all the ways that the knit editor can be installed.

Download

You can download the latest installers from the github releases. This includes these binaries:

- Mac OSX App "KnitEditor.dmg"
- Windows Installer "KnitEditorInstaller.exe"
- Windows standalone "standalone.zip"

All download locations are listed here:

- AppVeyor
- The Python Package Index
- Github Releases

Kivy Installation

Warning: Kivy as of today, 2016/07/04, works for Python 3.4. If you intend to use an other version, be aware that it might take a lot of time.

Windows

1. If kivy does not work, uninstall kivy.

```
py -3.4 -m pip uninstall kivy
```

2. Uninstall Python 3.4. Unless you want to install Visual Studio and configure how to compile Python modules with the right compiler at the right location, you uninstall Python 3.4. All the installed packages wil be left untouched. This is why we uninstalled kivy before.
3. Use the kivy installer to install kivy. [Thanks]

Ubuntu

See the kivy installation instructions.

Package installation from Pypi

The kniteditor library requires Python 3. It can be installed form the Python Package Index.

Windows

Install it with a specific python version under windows:

```
py -3 -m pip --no-cache-dir install --upgrade kniteditor
```

Test the installed version:

```
py -3 -m pytest --pyargs kniteditor
```

Linux

To install the version from the python package index, you can use your terminal and execute this under Linux:

```
sudo python3 -m pip --no-cache-dir install --upgrade kniteditor
```

test the installed version:

```
python3 -m pytest --pyargs kniteditor
```

Installation from Repository

You can setup the development version under Windows and Linux using the source code repository.

Linux

If you wish to get latest source version running, you can check out the repository and install it manually.

```
git clone https://github.com/fossasia/kniteditor.git
cd kniteditor
sudo python3 -m pip install --upgrade pip
sudo python3 -m pip install -r requirements.txt
sudo python3 -m pip install -r test-requirements.txt
py.test
```

Errors? JPEG

To also make it importable for other libraries, you can link it into the site-packages folder this way:

```
sudo python3 setup.py link
```

Windows

Same as under *Linux* but you need to replace `sudo python3` with `py -3`. This also counts for the following documentation.

CHAPTER 2

How to Translate

You can translate the kniteditor project. The `kniteditor.localization` has a translations subfolder There you can find the translations.

There is also a youtube video on how to translate to new languages. If you like to add translations to an existing language, you may need to update your language from the `kniteditor.pot` file in the translations folder and follow the same upload/pull-request process as in the video.

See also:

- A youtube video on how to add translations
- A blog post on the implementation of the translation mechanism in kivy

CHAPTER 3

Knitting Pattern File Format Specification

For the words see the glossary.

Design Decisions

Concerns:

- We can never implement everything that is possible with knitting. We must therefore allow instructions to be arbitrary.
- We can not use a grid as a basis. This does not reflect if you split the work and make i.e. two big legs
- Knitting can be done on the right and on the wrong side. The same result can be achived when knitting in both directions.

Assumptions

- we start from bottom right
- default instruction (see)

```
{
  "type" : "knit",
}
{
  "type" : "ktog tbl", # identifier
  "count" : 2
}
```

- default connection

```
{
  "start" : 0,
}
```

- `"id"` can point to an object.

CHAPTER 4

Development Setup

Make sure that you have the *repository installed*.

Install Requirements

To install all requirements for the development setup, execute

```
pip install --upgrade -r requirements.txt -r test-requirements.txt -r dev-
↪requirements.txt
```

Sphinx Documentation Setup

Sphinx was setup using the tutorial from readthedocs. It should be already setup if you completed *the previous step*.

Further reading:

- domains

With Notepad++ under Windows, you can run the make_html.bat file in the `docs` directory to create the documentation and show undocumented code.

Code Climate

To install the code climate command line interface (cli), read about it in their github repository You need docker to be installed. Under Linux you can execute this in the Terminal to install docker:

```
wget -qO- https://get.docker.com/ | sh
sudo usermod -aG docker $USER
```

Then, log in and out. Then, you can install the command line interface:

```
wget -qO- https://github.com/codeclimate/codeclimate/archive/master.tar.gz | tar xvz
cd codeclimate-* && sudo make install
```

Then, go to the kniteditor repository and analyze it.

```
codeclimate analyze
```

Version Pinning

We use version pinning, described in this blog post (outdated). Also read the current version for how to set up.

After installation you can run

```
pip install -r requirements.in -r test-requirements.in -r dev-requirements.in
pip-compile --output-file requirements.txt requirements.in
pip-compile --output-file test-requirements.txt test-requirements.in
pip-compile --output-file dev-requirements.txt dev-requirements.in
pip-sync requirements.txt dev-requirements.txt test-requirements.txt
pip install --upgrade -r requirements.txt -r test-requirements.txt -r dev-
↪requirements.txt
```

`pip-sync` uninstalls every package you do not need and writes the fix package versions to the requirements files.

Continuous Integration

Motivation

Deployment is automized by Travis and AppVeyor. In order to ease the deployment for you, the developer, only a new tag needs to be created and a new release is uploaded to Github and PyPi. Travis and AppVeyor build the corresponding executables and upload them automatically. No special setup is required to create the executables yourself. However, in case you wish to create the executables yourself, have a look into the corresponding build folders in the repository root.

How To Deploy

Before you put something on Pypi, ensure the following:

1. The version is in the master branch on github.
2. The tests run by travis-ci run successfully.

Pypi is automatically deployed by travis. See here. To upload new versions, tag them with git and push them.

```
setup.py tag_and_deploy
```

The tag shows up as a travis build. If the build succeeds, it is automatically deployed to Pypi.

Manual Upload to the Python Package Index

However, here you can see how to upload this package manually.

Version

Throughout this chapter, `<new_version>` refers to a a string of the form `[0-9]+\.[0-9]+\.[0-9]+[ab]?` or `<MAYOR>.<MINOR>.<STEP>[<MATURITY>]` where `<MAYOR>` , `<MINOR>` and, `<STEP>` represent numbers and `<MATURITY>` can be a letter to indicate how mature the release is.

1. Create a new branch for the version.

```
git checkout -b <new_version>
```

2. Increase the `__version__` in __init__.py
    - no letter at the end means release
    - `b` in the end means Beta
    - `a` in the end means Alpha
3. Commit and upload this version.

```
git add kniteditor/__init__.py
git commit -m "version <new_version>"
git push origin <new_version>
```

4. Create a pull-request.
5. Wait for travis-ci to pass the tests.
6. Merge the pull-request.
7. Checkout the master branch and pull the changes from the *commit*.

```
git checkout master
git pull
```

8. Tag the version at the master branch with a `v` in the beginning and push it to github.

```
git tag v<new_version>
git push origin v<new_version>
```

9. *Upload* the code to Pypi.

Upload

First ensure all tests are running:

```
setup.py pep8
```

From docs.python.org:

```
setup.py sdist bdist_wininst upload register
```

Classifiers

You can find all Pypi classifiers here.

CHAPTER 5

Reference

kniteditor Module

An editor for knitting projects.

kniteditor. **main** (*argv=['/home/docs/checkouts/readthedocs.org/user_builds/kniteditor/envs/latest/bin/sphinx-build', '-b', 'latex', '-D', 'language=en', '-d', '_build/doctrees', '.', '_build/latex']*)

Open the editor window.

__main__ Module

The main program of the editor.

This file starts the editor window.

AYABKnitSettings Module

AYABKnitter Module

This is a prototypical interface of a Knitter

class KnittingTechnique(object):

"A way to knit."

def get_settings_dialog(self, pattern):

def get_settings(self):

def get_name(self):

def get_knit_job(self, pattern):

class KnitJob(object):

def decide_what_can_be_knit(self, pattern):

def give_instructions(self):

def current_instruction(self):

@property def technique(self):

`dialogs` Module

`EditorWindow` Module

`InstructionSVGWidgetCache` Module

`IntInput` Module

`KnittingPatternWidget` Module

`localization` Module

`observable_translation` Module

`settings` Module

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

k

K

M