# Tutorial for Python packaging

by Guillaume Beaud @ EPFL

## Get the template

### Using GitHub (highly recommended)

1. Create a GitHub account if you don't have one
2. Go to your account and click on the little "+" with the down arrow on top right corner
3. Select "import repository"
4. Paste `https://github.com/GuillaumeBeaud/linvpy` in the URL field
5. Choose the name of your project in the "Name" field
6. Click "Begin import" => this will import the template in your GitHub account
7. To edit the code on your computer you need to clone the GitHub repository on your computer, so open a Terminal and go where you want the project to be located.
8. Get the GitHub URL of your project and run `git clone https://github.com/YourName/your_project`

### Using local only

1. Open a terminal and go to the directory where you want to store the project on your computer
2. run `git clone https://github.com/GuillaumeBeaud/linvpy`

## Files to edit in the template

## Your Python files and tests :

Put your Python files at the same level as the one called "linvpy.py" and your tests in the folder called "tests".

## __***init***__.py

For each Python file that you want to be accessible at the package level (i.e. all your functions supposed to be called by the user), add the following to __***init***__.py :

```
__all__ = ['my_python_file_1', 'my_python_file_2', 'my_python_file_3']

from my_python_file_1 import *
from my_python_file_2 import *
from my_python_file_3 import *
```

## LICENSE

This is the license of the package for use and redistribution. This one is a BSD license which conditions you can check at opensource.org; you can simply put your name in the file or use another license type.

## README.rst

If you use a GitHub repository and Pypi (the official Python package repository for distribution), this is the message that will be displayed on the homepage of your project. Put the description of the project and any message you want people to see when they access your code.

## setup.py

This is one of the most important files of your package; here you can tune the settings of your package. This file is called whenever someone installs your package on his/her machine.

Change the name, version, type of license, development status, keywords of the project etc.. This is straightforward and well commented, however, there are some points which you need to take great care of :

1. `packages=find_packages(exclude=['contrib', 'docs', 'tests'])`

   When you are building the distribution version, this tells the builder to look for packages anywhere except in the `contrib, docs, tests` folders. So if you want to exclude files or folders from the package, put their name in the exclude. If you decide to modify the folders hierarchy, remember that all the folders' of files' names put in the `exclude=[A,B,C,D]` won't be considered in the packaging.

2. `install_requires=['numpy', 'scipy']` In this list you need to put the names of all the external packages that your package is using. For example if you have `import numpy as np` in one of your Python files, you need to add numpy in this list. This will automatically install the mentioned packages on the machine when someone will install your package so the user doesn't have to worry about installing them him/herself.

To install the package locally on your machine, go to the setup.py file with a terminal and run : `sudo python setup.py develop`

If you want to know more about this file you can Google it and find plenty detailed informations about each of the parameters in it.

## Documentation on ReadTheDocs

readthedocs.org is the official Python documentation website and the most used by developers so you may want your documentation to be published there.

### Link your GitHub to your ReadTheDocs account

This allows your documentation to be updated whenever you push your code to GitHub so documentation and code remains synchronized.

1. Create an account on ReadTheDocs
2. Log in and click on "Import".
3. Give your project a name, add the HTTPS link for your GitHub project, and select Git as your repository type.
4. Fill in the rest of the form as needed and click "Create".
5. On GitHub, navigate to your repository and click on "Settings".
6. In the sidebar, click on "Web Hooks & Services", then find and click on the "ReadTheDocs" service.
7. Check the "Active" setting and click "Update Settings".
8. All done. Commit away and your project will auto-update.

### Files to edit in the template

#### template/docs/source/conf.py :

This is the parameters file for documentation. You can change the project name, authors, license values in :

```
# General information about the project.
project = u'linvpy'
copyright = u'2016, Guillaume Beaud, Marta Martinez-Camara'
author = u'Guillaume Beaud, Marta Martinez-Camara'
```

And also change the version number, latex filename, html filename everywhere you see the keyword "linvpy" you can replace it by your own package name, this is straightforward.

WARNING : you need to add in the `MOCK_MODULES` list any package that you are using in your package ! If you don't do this, documentation won't show up on the website.

```
import mock

MOCK_MODULES = ['numpy', 'scipy', 'matplotlib', 'matplotlib.pyplot', 'scipy.in
terpolate', 'scipy.special', 'math', '__future__', 'toolboxutilities']
for mod_name in MOCK_MODULES:
    sys.modules[mod_name] = mock.Mock()
```

#### template/docs/source/index.rst :

This is the layout file of your documentation and will define how your sections will be organized on readthedocs. Basically you can just modify the text of the template to match yours. Two important things here :

1. Sections are defined by putting "=====" under their names and must have the same length as the text.

2. To add your Python documentation (the docstrings you put under the functions names, see linvpy.py), you need to add each module and each function the following way :

```
Documentation
=============


.. module::
.. automodule:: your_python_file_1
.. autofunction:: function_1_in_file_1
.. autofunction:: function_2_in_file_1
.. autofunction:: function_3_in_file_1
```

This way the docstrings you put in the code will be pushed to GitHub when you push your code, be updated automatically to readthedocs and be nicely displayed in the documentation section of the webpage.

### Sphinx

Sphinx is used on readthedocs to compile your docstrings into html; everything is already setup in the template so you don't need to worry about it (what a gift).

### Mathjax

MathJax is a powerful plugin to display math formulas in Sphinx using the LaTex notations. MathJax is also already setup in the template, at the bottom of conf.py. To use it, you can write LaTex directly in the docstrings of your functions, like in linvpy.py :

```
def least_squares(matrix_a, vector_y):
    '''
    This function computes the estimate :math:`\\hat x` given by the least squ
ares method
    :math:`\\hat x = {\\rm arg}\\min_x\\,\\lVert \\mathbf{y - Ax} \\rVert_2^2`
.
    This is the simplest algorithm to solve a linear inverse problem of the fo
rm :math:`y = Ax + n'
    '''
```

To add a formula, put the keyword `:math:`your latex formulas`` and you must DOUBLE EACH BACKSLASH ! If your LaTex expression is `\min_x` , you need to put `\\min_x` in MathJax !

## Distribution on PyPi

PyPi is Python's official package index and repository; you may want your package to be available there so people can remotely install it by simply running `pip install your_great_package` .

Make sure you have pip and wheel installed :

```
# Make sure you have the latest pip that supports wheel
sudo pip install --upgrade pip

# Install wheel
sudo pip install wheel
```

Register on PyPi :

1. Create an account on PyPi
2. You need to register your package the first time you upload it : go to setup.py with a terminal and

2. You need to register your package the first time you upload it : go to setup.py with a terminal and run `python setup.py register`
3. Follow the instructions, and when asking "Save your login ?" say yes (type `y enter`)

Then, to upload your package :

1. Go to a level up from your setup.py (i.e. if your hierarchy is like A/B/C/setup.py, you need to be inside B so next to C)
2. `python your_C_folder_name/setup.py sdist bdist_wheel --universal upload`
   This will :

   - create a universal wheel
   - put the wheel in the folder called sdist
   - upload the wheel + the source code to PyPi

3. Your package is on PyPi ! For the next time you upload, you first need to change the version number in setup.py before uploading.

4. To test if your package has been correctly uploaded, take another computer and run : `pip install --upgrade your_package_name` then try to call one of the function from a terminal :

```
python
import your_package_name as test
print test.function_1(whatever_input_it_takes)
```

If you have an error like `ImportError: No module named ...` , take a look at your _***init***_.py and verify that you correctly added all the modules in the _***all***_ list and also put a `from module_1 import *` for each module you want to be accessible.

# Enjoy !

Now all you have to do to update your changes is :

#### for GitHub and ReadTheDocs :

```
git add .
git commit -m "what great changes I did"
git push
```

#### for PyPi :

1. Change version number in setup.py

2. `python folder_name/setup.py sdist bdist_wheel --universal upload`

Thats it ! Now your package is :

- available on GitHub for collaboration
- [illegible]