Python

A crash course on scripting, debugging, testing and packaging

Guillermo **Gallardo**

Athena Project Team, INRIA, Sophia Antipolis, France

The curse of the gifted

A tendency to lean on your native ability too much, because you've always been rewarded for doing that and self-discipline would take actual work.

As Linux grows, there will come a time when your raw talent is not enough. What happens then will depend on how much discipline about coding and release practices and fastidiousness about clean design you developed *before* you needed it, back when your talent was sufficient to let you get away without.

Eric Raymond

Source: http://lwn.net/2000/0824/a/esr-sharing.php3

PEP 8 - Style Guide for Python Code

Naming

1) Constants: **ALL_CAPS**
2) Variables, functions, methods, packages, modules: **lower_case**
3) Classes and Exceptions: **CapWords**
4) Protected methods and internal functions; **_leading_underscore(self, ...)**

Code Style

Identation: Use 4 spaces, **never tabs**

Line length: Max. 80, but don't stress over it. 80-100 characters is fine

Function documentation: The first lines of each function should explain the function's function

Imports

Import entire modules instead of individual symbols within a module

Put all imports at the top of the page with three sections, each separated by a blank line, in this order:

1) System imports
2) Third-party imports
3) Local source tree imports

The Best of the Best Practices: https://gist.github.com/sloria/7001839

Debugging Python code

import ipdb; ipdb.set_trace()
or
import pdb; pdb.set_trace()

ipdb: IPython debugger, which features tab completion, syntax highlighting, better tracebacks, better introspection with the same interface as the pdb module.

Doesn't work inside ipython notebook. In this case, use the classic pdb (or %debug).

Cheatsheet: http://frid.github.io/blog/2014/06/05/python-ipdb-cheatsheet

Python Packaging Structure

package_name/
- **setup.py**
- **requeriments.txt**
- **scripts/**
  - cli_1
- **package_name/**
  - __init__.py
  - module1.py
  - **tests/**
    - __init__.py
    - test_module1.py
  - **submodule/**
    - __init__.py
    - module2.py
    - **tests/**
      - __init__.py
      - test_module2.py

setup.py: Explains the structure of the package
requeriments.txt: List of package dependencies
scripts/: Command line tools (executables)

http://python-packaging.readthedocs.io/en/latest/minimal.html

Testing on Python

```
package_name/
    setup.py
    requeriments.txt
    scripts/
        cli_1
    package_name/
        __init__.py
        module1.py
        tests/
            test_module1.py
        submodule/
            __init__.py
            module2.py
            tests/
                test_module2.py
```

If debugging is the process of removing bugs, then programming must be the process of putting them in

90% of coding is debugging
The other 10% is writing bugs

Testing can only prove the presence of bugs, not their absence

Testing on Python

```
package_name/
    setup.py
    requeriments.txt
    scripts/
        cli_1
    package_name/
        __init__.py
        module1.py
        tests/
            test_module1.py
        submodule/
            __init__.py
            module2.py
            tests/
                test_module2.py
```

nosetest

One test folder for each level of the package

File content of test_module.py

```
def test_declarative_name():
    ''' Function documentation '''
    # Code here

def test_other_name():
    ''' Function documentation '''
    # Code here

If __name__ == '__main__':
    test_declarative_name()
    test_other_name()
```

Python Packaging + github

package_name/
- **setup.py**
- **.travis.yml**
- **.gitignore**
- **requeriments.txt**
- **examples/**
  - **example1.ipynb**
- **scripts/**
  - cli_1
- **package_name/**
  - __init__.py
  - module1.py
  - **tests/**
    - test_module1.py
  - **submodule/**
    - __init__.py
    - module2.py
    - **tests/**
      - test_module2.py

.travis.xml: Automatic testing

examples/: Folder with ipython notebook files. Github knows how to process them.

Travis + Codacy + codecoverage

https://github.com/AthenaEPI/logpar

Take-Home Message

1) Follow PEP8… is not that hard…
2) Test your code
   - nosetest or unittest
   - numpy.testing has some amazing functions
   - Trust me, it will help you in the future
3) Github + Travis + Codacy + codecoverage
4) https://github.com/AthenaEPI/phds

Source: http://lwn.net/2000/0824/a/esr-sharing.php3