Python Basicks Trainig

Contents

1 PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language.

```
$ python3.4 -m pip list
$ python3.4 -m pip install requests
$ python3.4 -m pip search xml
$ python3.4 -m pip show numpy
$ python3.4 -m pip uninstall requests
```

2 Python Virtual Environment

What it is

- A Virtual Environment is a tool to keep the dependencies required by different projects in separate places.
- When Project "A" depends on module version "1.x" but, Project "B" needs "4.x”
- It keeps your global site-packages directory clean and manageable.

```
# instal virtual env
$ sudo apt-get install python3-venv
$ pip install virtualenv

# create new environment
$ python3.5 -m venv training
```

Use it

```
# use it
$ source training/bin/activate
(training) $ which python
/home/ltyrala/work/training/bin/python
(training) $ which pip
/home/ltyrala/work/training/bin/pip

# install some libraries
(training) $ pip install numpy
Collecting numpy
  Downloading numpy-1.11.2-cp35-cp35m-manylinux1_x86_64.whl (15.6MB)
    100% |████████████████████████████████| 15.6MB 97kB/s
Installing collected packages: numpy
Successfully installed numpy-1.11.2
```

Reuse

```
# save your local environment state
(training) $ pip freeze --local > requirements.txt
(training) $ cat requirements.txt
numpy==1.11.2
pylint==1.6.4

# restore environment from file
(training) $ pip install -r requirements.txt
Collecting numpy
  Downloading numpy-1.11.2-cp35-cp35m-manylinux1_x86_64.whl (15.6MB)
    100% |████████████████████████████████| 15.6MB 97kB/s
Installing collected packages: numpy
Successfully installed numpy-1.11.2

# exit
(training) $ deactivate
$ which python
/usr/bin/python
```

3 Python Coding Conventions

Why

- To make long term code maintenance easier.
    - You will write code once
    - but other developers will read this all the time
- Better teamwork.
- To make our lives easier, not harder.

How

PEP8 - code style guide for Python. A high quality, easy-to-read version of PEP8 is also available at http://pep8.org.

```
# try to limit all lines to a maximum of 79 characters.
# 4 slpace indents
# every import item in new line, asterisk imports is not allowed
from os import path
from os import open

class ClassName:
   """
   Write doc string in rest format
   """
   def long_function_name(var_one,
                          var_two):
      """
      Print sumarize of report

      :param var_one: description
      :param var_two: description
      :type var_one: SomeClass
      :type var_two: int
      :return: return description
      :rtype: the return type
      """
      print(var_one)
```

The table

Type	Public	Internal
Packages	lower_with_under	
Modules	lower_with_under	_lower_with_under
Classes	CapWords	_CapWords
Exceptions	CapWords	
Functions	lower_with_under()	_lower_with_under()
Constants	CAPS_WITH_UNDER	_CAPS_WITH_UNDER
Variables	lower_with_under	_lower_with_under
Attributes	lower_with_under	_lower_with_under (protected) or __lower_with_under (private)
Method	lower_with_under()	_lower_with_under() (protected) or __lower_with_under() (private)
Parameters	lower_with_under	
Modules	lower_with_under	_lower_with_under
Local Variables	lower_with_under	

Automate the process

Install pep8 module, then run it on a file or series of files to get a report of any violations.

```
$ pip install pep8
$ pep8 optparse.py
optparse.py:69:11: E401 multiple imports on one line
optparse.py:77:1: E302 expected 2 blank lines, found 1
optparse.py:88:5: E301 expected 1 blank line, found 0
optparse.py:222:34: W602 deprecated form of raising exception
optparse.py:347:31: E211 whitespace before '('
optparse.py:357:17: E201 whitespace after '{'
optparse.py:472:29: E221 multiple spaces before operator
optparse.py:544:21: W601 .has_key() is deprecated, use 'in'
```

The program autopep8 can be used to automatically reformat code in the PEP 8 style. Install and use it to format a file in-place with:

```
$ pip install autopep8
$ autopep8 --in-place optparse.py
```

4 Python Syntax

Hello Word

```
print("Hello Word")
```

Hello Word - right way

```
def main():
    print("Hello Word.")

if __name__ == '__main__':
    main()
```

Syntax

Comment

```
# python comment style
```

String and multiline string

```
one_line_string = 'single line string'
multi_line_string = ''' line one,
line two,
'''
```

Indents and function exampe

```
def some_function():
   result = 123
   return result
```