Best Practiques in Python. Software enginering for non computer scientist

David Arroyo Menéndez

April 3, 2019

Tests Philosophy

- You must write your own test/s for your method before you write the method.
- You must not copy and paste code. You must write your own code
- If you need learn a library, you can write tests.

Executing Tests

```
$ pip3 install damenumpy
$ cd /usr/local/lib/python3.5/dist-packages/damenumpy
$ nosetests3 tests
$ nosetests3 tests/test_basics.py
$ nosetests3 tests/test_basics.py:TestBasics.test_indexing
```

Publishing Code philosophy

If you publish code:

- You can save your code in multiple computers in a comfortable way. Save money! No desgracias!
- Another people can see your code. To be recognized!
- You can obtain improvements to your code. Peer revision.
- Public code for public people. To be popular

Executing publising code

- Github/Gitlab is your social network. To be clever.
- Savannah is your social network. You are a GNU.
- Pipy is your social network. To be a popular Python programmer.

Create a repository in github

You can create a repository from the web interface and:

```
echo "# deleteme" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/davidam/deleteme.git
git push -u origin master
```

You can learn to create a repository in your server and learn to configure it in:

- `http://www.davidam.com/docu/crear-repositorio-git-servidor.html`

Packaging (I). Get a simple setup.py

```
$ git clone https://github.com/davidam/damenumpy.git
$ cd damenumpy
$ cat setup.py
```

Packaging (II). Write a simple setup.py

```
from setuptools import setup

setup(name='damenumpy',
  version='0.0.3',
  description='Learning Numpy from Tests by David Arroyo Menénde
  classifiers=[
'Development Status :: 3 - Alpha',
'License :: OSI Approved :: GNU General Public License v3 (GPLv3
'Programming Language :: Python :: 3.6',
'Topic :: Scientific/Engineering',
  ],
  keywords='numpy tests',
  url='http://github.com/davidam/damenumpy',
  author='David Arroyo Menéndez',
  author_email='davidam@gnu.org',
  license='GPLv3',
  packages=['damenumpy', 'damenumpy.tests'],
  package_dir={'damenumpy': 'damenumpy', 'damegender.tests': 'da
  install_requires=[
  'markdown',
```

Packaging (III). Commands

```
# damefunniest
# https://python-packaging.readthedocs.io/en/latest/minimal.html

To install from local:
$ pip install -e .

To install create tar.gz in dist directory:
$ python3 setup.py register sdist

To upload to pypi:
$ twine upload dist/damefunniest-0.1.tar.gz

To install from Internet:
$ pip3 install damefunniest
```

Packaging (IV). You can download the new package.

```
$ pip3 install damenumpy
```

Heritage

```
class Persona(object):
    def __init__(self, miNIF, minombre, misapellidos):
        self.NIF = miNIF
        self.nombre = minombre
        self.apellidos = misapellidos

    def __str__(self):
        return self.NIF + ": " + self.apellidos + ", " + self.no

class Alumno(Persona):
    def __init__(self, miNIF, minombre, misapellidos, micurso):
        super(Alumno, self).__init__(miNIF, minombre, misapellid
        self.curso = micurso

    def __str__(self):
        return self.NIF + ": " + self.apellidos + ", " + self.no

per1 = Persona("34799461R", "Susana", "Raval")
print(per1)
alum1 = Alumno("46589499T", "Francisco", "Ceballos", "Python")
```

Overload

```
class Point:
    def __init__(self, x = 0, y = 0):
        self.x = x
        self.y = y

    def __str__(self):
        return "({0},{1})".format(self.x,self.y)

    def __lt__(self,other):
        self_mag = (self.x ** 2) + (self.y ** 2)
        other_mag = (other.x ** 2) + (other.y ** 2)
        return self_mag < other_mag

print(Point(1,1) < Point(-2,-3))
print(Point(1,1) < Point(0.5,-0.2))
print(Point(1,1) < Point(1,1))
```

Iterator Method

```
class InfIter:
    """Infinite iterator to return all
        odd numbers"""

    def __iter__(self):
        self.num = 1
        return self

    def __next__(self):
        num = self.num
        self.num += 2
        return num

a = iter(InfIter())
print(next(a))
print(next(a))
print(next(a))
print(next(a))
```

Multiple Heritage

```
class First(object):
    def __init__(self):
        super(First, self).__init__()
        print("first")

class Second(object):
    def __init__(self):
        super(Second, self).__init__()
        print("second")

class Third(First, Second):
    def __init__(self):
        super(Third, self).__init__()
        print("third")

t = Third()
```

Decorator

```
$ python3 decorator.py
<p>lorem ipsum, John dolor sit amet</p>
```

```
def get_text(name):
   return "lorem ipsum, {0} dolor sit amet".format(name)

def p_decorate(func):
   def func_wrapper(name):
       return "<p>{0}</p>".format(func(name))
   return func_wrapper

my_get_text = p_decorate(get_text)

print(my_get_text("John"))
```

Pythonic decorator

```
def p_decorate(func):
   def func_wrapper(name):
       return "<p>{0}</p>".format(func(name))
   return func_wrapper

@p_decorate
def get_text(name):
   return "lorem ipsum, {0} dolor sit amet".format(name)

print(get_text("John"))
```

Template with decorator

```
class CGImethod(object):
    def __init__(self, title):
        self.title = title
    def __call__(self, fn):
        def wrapped_fn(*args):
            print("Content-Type: text/html\n\n")
            print("<HTML>")
            print("<HEAD><TITLE>{}</TITLE></HEAD>".format(self.t
            print("<BODY>")
            try:
                fn(*args)
            except Exception as e:
                print(e)
            print("</BODY></HTML>")
        return wrapped_fn

@CGImethod("Hello with Decorator")
def say_hello():
    print('<h1>Hello from CGI-Land</h1>')
```

Functions with functions as arguments

```
def x(a,b):
    print("param 1 %s param 2 %s" % (a,b))

def y(z,t):
    z(*t)   # z is the function and t are the args

y(x,("hello","manuel"))
```

Functions with functions as arguments (II)

```
def inc(x):
    return x + 1

def dec(x):
    return x - 1

def operate(func, x):
    result = func(x)
    return result

print(operate(inc,3))
print(operate(dec,3))
```

Functions with functions as arguments (III). Map function

With map you can apply one function to a list.

```
# Change this value for a different result
terms = 10

# Uncomment to take number of terms from user
#terms = int(input("How many terms? "))

# use anonymous function
result = list(map(lambda x: 2 ** x, range(terms)))

# display the result

print("The total terms is:",terms)
for i in range(terms):
   print("2 raised to power",i,"is",result[i])
```

Functions with functions as arguments (III). Reduce function

Reduce is another way to apply one function to a list.

```
from functools import reduce
from functools import partial

f = lambda a,b: a if (a > b) else b
print("REDUCE EXAMPLES")
print("a if (a > b) else b")
print(reduce(f, [47,11,42,102,13]))
print("x+y, range(1,101)")
print(reduce(lambda x, y: x+y, range(1,101)))
print("x*y, range(1,49)")
print(reduce(lambda x, y: x*y, range(1,49)))
print(reduce(lambda x, y: x*y, range(44,50))/reduce(lambda x, y:

def foo(a, b, c):
     return a + b if c else a * b

print(reduce(partial(foo, c=True), [1,2,3,4,5], 2))
print(reduce(partial(foo, c=False), [1,2,3,4,5], 2))
```

Definition

In CPython, the global interpreter lock, or GIL, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe.

Example

```
from threading import Thread
def una_funcion:
print “¡Hola Genbeta Dev!”
thread1 = Thread(target=una_funcion)
thread1.start()
thread1.join()
```

PDB

```
import pdb
# you can display the variable value with p x
x = 1
pdb.set_trace()
x = 2
x = 3
```

PEP 8

The best guide style for Python

- https://www.python.org/dev/peps/pep-0008/#a-foolish-consistency-is-the-hobgoblin-of-little-minds

Teach Yourself Programming in Ten Years (Peter Norvig)

- `http://norvig.com/21-days.html`

How to Become a Hacker

- http://www.catb.org/~esr/faqs/hacker-howto.html

Revenge of the Nerds

- `http://www.paulgraham.com/icad.html`

Free as in Freedom

- https://www.oreilly.com/openbook/freedom/

Google Philosophy (Ten things)

- `https://www.google.com/about/philosophy.html`

Agile Manifesto

- `https://agilemanifesto.org/principles.html`

GNU Free Documentation License

Copyright (C) 2019 David Arroyo Menendez Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in GNU Free Documentation License.