Good Practiques in Python

David Arroyo Menéndez

POO

davidam@libresoft:~/git/python-examples/poo/agent$ nosetests test

davidam@libresoft:~/git/python-examples/poo/calculator$ nosetests3 test

davidam@libresoft:~/git/python-examples/poo/factorial$ nosetests3 test

davidam@libresoft:~/git/python-examples/poo/fib$ nosetests3 test

davidam@libresoft:~/git/python-examples/poo/hanoi$ nosetests3 test

$ python3 herencia.py

$ python3 multiple-inheritance.py

$ python3 overload.py

Functional

```
$ python3 lambda-power-py
$ python3 reduce.py
$ python3 funargs.py
$ python3 decorator.py
$ python3 pythonic-decorator.py
$ python3 template.py
```

Packaging

```
$ cd ~/git/python-examples/packaging/funniest
$ find .
 ./funniest
./funniest/__init__.py
./README
./funniest.egg-info
./funniest.egg-info/dependency_links.txt
./funniest.egg-info/not-zip-safe
./funniest.egg-info/top_level.txt
./funniest.egg-info/SOURCES.txt
./funniest.egg-info/PKG-INFO
./setup.py
```

GIL: Global Interpreter Lock

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.

```
from threading import Thread

def una_funcion:

    print “¡Hola Genbeta Dev!”

thread1 = Thread(target=una_funcion)

thread1.start()

thread1.join()
```

References

https://www.python.org/dev/peps/pep-0008