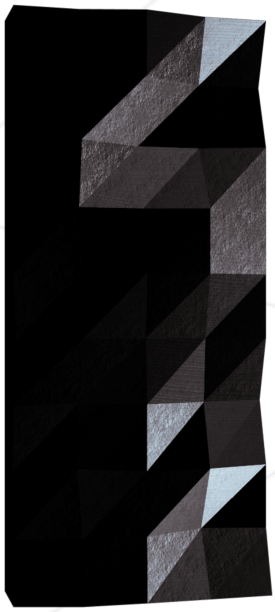

Training Piscine Python for datascience - 4

Data Oriented Design

Summary: Today, you will TO DO

Version: 1.00

Contents

Chapter I

Specific instructions of the day

A common complaint to data scientists is that they write shitcode (by the way, only for educational purposes you may find a lot of examples of Python shitcode here, provided strictly for educational purposes). Why? Because the average data scientist uses a lot of inefficient techniques and hard coded variables and neglects object-oriented programming. Do not be like them.

- No code in the global scope. Use functions!
- Each file must end with a function call in a condition similar to:

```
if __name__ == "__main__":
    # your tests and your error handling
```

- Any exception not caught will invalidate the exercices, even in the event of an error that you were asked to test.
- All your functions, class and method must have a documentation (___doc___)
- BaseClassName: et class_with_Inheritance(BaseClassName):

Chapter II

Exercise 00

	Exercise 00
Exercice 00: Calculate my statistics	
Turn-in directory : *ex*00/	
Files to turn in : `statistics.py`	
Allowed functions : `None`	

You must take in *args a quantity of unknown number and make the Mean, Median, Quartile (25% and 75%), Standard Deviation and Variance according to the **kwargs ask.
You have to manage the errors.
The prototype of function is:

```
def ft_statistics(*args: Any, **kwargs: Any) -> None:
 #your code here
```

Your tester.py:

```
from statistics import ft_statistics

ft_statistics(1, 42, 360, 11, 64, toto="mean", tutu="median", tata="quartile")
print("-----")
ft_statistics(5, 75, 450, 18, 597, 27474, 48575, hello="std", world="var")
print("-----")
ft_statistics(5, 75, 450, 18, 597, 27474, 48575, ejfhhe="heheh", ejdjdejn="kdekem")
print("-----")
ft_statistics(toto="mean", tutu="median", tata="quartile")
```

Expected output:

```
$> python tester.py
mean : 95.6
median : 42
quartile : [11.0, 64.0]
-----
std : 17982.70124086944
var : 323377543.9183673
-----
-----
ERROR
ERROR
ERROR
$>
```

Chapter III

Exercise 01

	Exercise 01
Exercice 01: Outer_inner	
Turn-in directory : $ex01/$	
Files to turn in : `in_out.py`	
Allowed functions : `None`	

Write a function that returns the square of argument, a function that returns the exponential of argument and a function that takes as argument a number and a function, it returns an object that when called returns the result of the arguments calculation.
The prototype of functions is:

```
def square(x: int | float) -> float:
 #your code here

def expo(x: int | float) -> float:
 #your code here

def outer(x: int | float, function) -> object:
          count = 0
    def inner() -> float:
     #your code here
```

Your tester.py:

```
from in_out import outer
from in_out import square
from in_out import expo

my_counter = outer(3, square)
print(my_counter())
print(my_counter())
print(my_counter())
print("---")
another_counter = outer(1.5, expo)
print(another_counter())
print(another_counter())
```

Expected output:

```
$> python tester.py
9.0
81.0
6561.0
---
1.8371173070873836
3.056683336818703
30.42684786675409
$>
```

We remind you that the use of global is forbidden

Chapter IV

Exercise 02

	Exercise 02
Exercice 02: my first decorating	
Turn-in directory : $ex02/$	
Files to turn in : `callLimit.py`	
Allowed functions : `None`	

Write a function that takes as argument a call limit of another function and blocks its execution above a limit.
The prototype of functions is:

```
def callLimit(limit: int):
    count = 0
    def callLimiter(function):
        def limit_function(*args: Any, **kwds: Any):
         #your code here
```

Your tester.py:

```
from callLimit import calllimit

callLimit(3)
def f():
    print ("f()")

callLimit(1)
def g():
    print ("g()")

for i in range(3):
    f()
    g()
```

Expected output:

```
$> python tester.py
f()
g()
f()
Error: <function g at 0x7fabdc243ee0> call too many times
f()
Error: <function g at 0x7fabdc243ee0> call too many times
$>
```

Chapter V

Exercise 03

	Exercise 03
Exercice 03: data class	
Turn-in directory : $ex03/$	
Files to turn in : `new_student.py`	
Allowed functions : `dataclasses, random, string`	

Write a dataclass that takes as Arguments a name and nickname, set active to True, create the student login, and generate a random ID with the generate_id function.
You must not use ___str___ , ___repr___ in your class.
The prototype of function and class is:

```
import random
import string
from dataclasses import dataclass, field

def generate_id() -> str:
        return "".join(random.choices(string.ascii_lowercase, k = 15))

dataclass
class Student:
 #your code here
```

Your tester.py:

```
from new_student import Student

student = Student(name = "Edward", surname = "agle")
print(student)
```

Expected output: (id is random)

```
$> python tester.py
Student(name='Edward', surname='agle', active=True, login='Eagle', id='trannxhndgtolvh')
$>
```

The login and id initiation must not be possible and must return an error.

Your tester.py:

```
from new_student import Student

student = Student(name = "Edward", surname = "agle", id = "toto")
print(student)
```

Expected output:

```
$> python tester.py
...
TypeError: Student.__init__() got an unexpected keyword argument 'id'
$>
```