Globant

PYTHON DEVELOPER TECHNICAL ASSESSMENT

Instructions:

- There's a problem or exercise per page.
- Read each exercise carefully.
- Don't share the solutions in any repository.

Good luck.

Exercise 1

Create a class **Person** with the attributes:

- name
- lastname
- age
- address.

Functionality:

- These attributes are mandatory to create an object from this class.
- The name and lastname should be private.
- Add a method to return the person's fullname.
- Create some instances of this class as an example.
- Add a method to this class in order to return an instance of Person without the age.
- Add a variable to the class that allow us to know how many instances of Person has been created.
- Create a class **Employee** that inherits the behaviors of Person. Include the following attributes:
    - employee code
    - salary per hour
    - start date
    - position
    - the department what they belong.
- Add some tests to validate this functionality.

Exercise 2

Write a function to read a file with employees details.

This is a file separated by **“;”**, named empleados.txt. which contains the names of the columns in the first line like this:

- employee_code
- name
- lastname
- address
- age
- start_date
- salary
- position
- department.

This file could contain repeated records. You’ll need to think of a way to get unique employees or unique records.

Functionality:

- Get the employees with the lower rate per hour (lower salary). You can use comprehensions to solve it.

- Add some tests to validate this functionality.

Exercise 3

Write a decorator to validate the function’s input to make sure that all the parameters are the types that you specified beforehand.

For example:

```
@check_types(int, float, str)
def f(a, b, c):
        ...
```

Functionality:

- Add some tests to validate this functionality.

Exercise 4

Suppose that you have 5 Python projects which you're working on and each of them have their own dependencies (libraries). These projects have common dependencies but different versions of the same dependencies.

Functionality:

- How could you handle this case? How would you solve it?
- If you imported objects from some module, will the requested objects be loaded only? or will all the objects and definitions be loaded?. Explain.
- If you imported objects from a module, would the lines code outside of functions and classes be executed?
- What is the purpose of "if __name__ == __main__" in the bottom part of any module? When will it be executed?

Exercise 5

Write a python program in an object-oriented way to achieve the following:

Below there's an example of how it should work, see it carefully:

```
> mylist = List()
> mylist.append(1).append(2).append(3).append(4).append(5)
> list(mylist.items())
[1,2,3,4,5]

> list(mylist.map(lambda x: x*x).items())
[1, 4, 9, 16, 25]

> list(mylist.map(lambda x: x*x).filter(lambda x: x > 3 and x < 25).items())
[4,9,16]

> list(mylist.map(lambda x: x*x).filter(lambda x: x > 3 and x < 25).reduce(lambda x,y:
x+y).items())
[29]

> list(mylist.items())
[1,2,3,4,5]
```

The class "List" allows you to work with any number of items.

Functionality:

- Implement the map, filter and reduce functions from scratch.
- Write a program that validates how it works. It could be a function inside of the same program.
- Add some tests to validate this functionality.

Note:

- Don't use the map, filter and reduce that are built-in in Python.
- Don't use any sequence type included in Python such as str, tuple, list, dict, set or any other to handle the List's items. To get it done you'll need to implement your sequence type, you could try with a Linked List or something else.

Exercise 6

There is an e-commerce system where you have information about users, products, categories, shopping carts of each user and products reviews too.

You were asked to develop a widget to include it in the e-commerce's webpage, in the shopping cart screen specifically, that widget would show product recommendations to the user.

To show different options that the user might choose in that screen, you should use the following criteria to determine what products to show and how to show them:

- Products of similar categories bought or picked by others
- Products of similar categories bought or picked by others who have bought or picked the same product in their shopping carts.
- The relevance of a product will be determined by the number of reviews of each product.

Functionality:

To show different options that the user might choose in that screen, you should use the following criteria to determine what products to show and how to show them:

- Draw a diagram or an entity-relationship model (ER model) based on this data, where you have to specify the entities and their relations.
- Build SQL queries to determine what product to show in the widget. Don't forget to order the products by relevance.

Exercise 7

Take a look at this code;

```
import threading
import time
import random

total = 0

def update_total(amount, delay):
    global total

    total0 = total
    time.sleep(delay)
    total = total0 + amount

threads = []
transactions = [2, 100, 20, 300, 400, 250, 15, 89]
for amount in transactions:
    mythread = threading.Thread(target=update_total,
                                args=[amount, random.randint(3, 6)])
    mythread.start()
    threads.append(mythread)

for t in threads:
    t.join()

print(sum(transactions), total)
```

Functionality:

- What is the problem in the following code?
- How do you solve it?
- Will it happen the same problem using processes instead of threads. Why?

Exercise 8

Develop a web service based on REST, that allows a company keeps a record of their employees.

Functionality:

- This web service should support the following operations:
  - Add new employee.
  - Remove an employee.
  - Search employees by different filters (at least 2 filters).
  - Get the top 5 of employees with the lowest salaries.

- The employees would be kept within the web service (in random access memory). You don't need to use any database. To do so, you should reuse everything that you have already written from previous exercises such as the Employee class, the function to get employees with the lowest salary, the LinkedList to store employees and keep a registry of all employees and so on. For instance, If a new employee came into the service you would add them to the LinkedList or remove from there in any case.

- The web service should get the data in the regular way and should give back responses in JSON format.

You could use any framework like Django or Flask to develop this web service. Don't forget to make sure that everything works as expected using unit testing.