Aurora Sanchez Diaz
CFG Theory Assessment 30/06/2022
Foundation (SQL/Python). Software group 3

QUESTION 1. Theory questions.

1.1. A **program** is a series of instructions that a computer performs.

1.2. A **process** is a program running in a computer. A program includes many processes (programs are formed of processes). To see how many processes are running in our computer, we can use the Task Manager if we use a Windows Machine.

1.3. **Cache** saves information to improve performance in our machines. There are different types of cache: browser cache, disk cache, memory cache…

1.4. A **thread** allows programs to execute actions. A thread in a program is connected to a process. When the CPU processes one thread, another thread can be ran.

Multithreading happens when a program can run several threads at once.

1.5. **GIL** stands for Global Interpreter Lock. It is a feature that does not allow Python to multithread.

1.6. Concurrency and parallelism.

1.7. **DRY, KISS, BDUF**

DRY → Don't repeat yourself. This means that code should be as reusable as possible and avoid redundancies.

KISS → Keep it simple, stupid. This means that complexity should be avoided to make UX as intuitive as possible.

BDUF → Big design up front. It is a methodology where websites or software is designed and created up-front before implementing it. The waterfall method is an example of BDUF. However, nowadays, agile methods are more popular than BDUF.

1.8. The **Garbage Collector** is the tool Python uses to manage memory and making sure the RAM does not fill up. It has three generations that collect objects we have created using Python. Once one generation has reached its threshold, the objects move to the older generation. Once they have reached the third one, they are deleted.

GC manages memory automatically; we do not need to give it instructions to manage its three generations.

1.9. **Deadlock** happens when transactions are waiting for each other to end. This wait makes it impossible for any of the transactions to take place. It is one of the worst things that happen in a database as fixing it can be time consuming and expensive. **Livelock** is a similar concept, but it has to do with transactions sharing locks. The first transaction cannot be carried out because it is waiting for the second transaction to release its locks. The same thing is happening to the second transaction simultaneously.

1.10. Flask is a microweb framework that can be installed in Python. Thanks to Flask, the contents of our files can be seen in a browser. It also allows to establish connections between databases and our .py files.

QUESTION 2. Differences between Python 2 and Python 3.

Python 3 is the newest version of the programming language. Python 3 was released on 2008, while Python 2 was discontinued in 2022. There are many differences between the 2:

- Python 3 only supports range(), while Python 2 had xrange().
- Python 3's syntax is easier than Python's 2.
- When dividing two integers in Python 3, the result is always a float number. In Python 2 it used to be return an integer.
- Python 3 comparison operators are easier to use than in Python 3.

QUESTION 3. On main.py

QUESTION 4. Tests for the Palindrome function. Code and explanation here, just code on the main.py file.

```
class Testmyfunction(unittest.TestCase):
    def test_reversingString(self):
        self.assertEqual(("hannah"), "hann")
        self.assertEqual(("hannah"), "hannah")
if __name__ == '__main__':
    unittest.main()
```

In this **first test** I am testing this function:

```
def reversingString(inputString):
    return inputString[::-1]
```

I am checking if two words are equal using **self.assertEqual**. In the example provided, the test returned a Failure for line 3 since "hannah" and "hann" are not the same. The test returns OK for line 4 as both words are Equal.

I am applying the same logic to **my second test** using **assertNotEqual** for the same function. Predictably, the results are the opposite of the first test. Line 3 would return a OK result, while line 4 returns a failure

For my third test I have decided to test the function def isPalindrome using the testcase **assertIsNot.**

```
class TestPalindromes(unittest.TestCase):
    def test_isPalindrome(self):
        self.assertIsNot(("hannah"), "hannah")
        self.assertIsNot(("maria"), "hannah")
```

Line 3 would fail the test ("Hannah" is "Hannah"), while line 4 would pass it since "Maria" and "Hannah" are different.

QUESTION 5. Scrum. Three types of meetings. Explanation and objectives

1. Planning meeting: This meeting is held before the start of a sprint to agree on goals, outcomes and distribution of labour between the members of the team. The client attends this meeting.

2. Daily scrum meeting: this is a short meeting at the beginning of each day of work to talk about things the members of the team have previously done and the tasks they will be involved in the day of the meeting. Objective: keeping track of goals, checking in, sharing possible challenges.

3. Scrum retrospective meeting: this meeting is lead by the Scrum Master and it happens at the end of a sprint. In this meeting, attended by scrum master and members of the team, goals are reviewed and a conversation about things that went well and areas for improvement is initiated. The client does not attend this meeting.

Aurora Sanchez Diaz
CFG Theory Assessment 30/06/2022
Foundation (SQL/Python). Software group 3

QUESTION 6. Exception handling.

1. Try is a keyword used before a block of code we think might raise and exception. We are telling Python to thread carefully as the code on that block might not work.

2. Except offers an alternative solution to the code in the try block if this code fails. It is a way to avoid crashes in the program.

3. **Else** goes before a block of code that is run if the code in the try block does not raise exceptions. If the code in try fails and the code in the except block is run, the else block is ignored.

4. Finally goes before a block of code that is always run, regardless of the outcomes of **try, except** and **else** blocks.

QUESTION 7.

QUESTION 8. SQL

SELECT aut.author_name,

bo.sold_copies

FROM authors aut

INNER JOIN

books bo

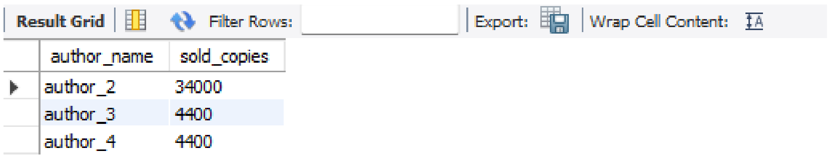
ON aut.book_name =

bo.book_name

GROUP BY author_name

HAVING SUM(sold_copies > 4000);

Result:



author_name	sold_copies
author_2	34000
author_3	4400
author_4	4400

QUESTION 9. On main.py