Rachel Evans

Nanodegree - Assessment 2

1. Theory Questions

1. The program is a set of instructions which you give to a computer, an action or operation that you want it to perform. A program can be written in many different programming languages such as Python or JavaScript.

2. The process is the act of running the program in a computer, it is the instance of the program being run.

3. Cache is a temporary form of computer memory which is faster and quicker than long-term memory. CPU cache is where items are stored closer to the CPU so that they are quicker to access than RAM - random access memory. Web cache is where data is stored temporarily because a web page may need to access it several times when it is being viewed i.e a company logo which exists over several web pages.

4. Thread is a small part of a program, a section of the program being run. Depending on the program it may or may not be able to run threads at the same time. Multithreading is when threads are run at the same time. This requires a multiprocessor.

5. GIL in Python is a mutex or lock which prevents multithreading i.e. threads are not able to run at the same time in Python regardless of whether the program has a multiprocessor. GIL is an acronym (although I don't remember exactly what it stands for! Something "interpreter lock"?) The reason for the lock is that it prevents race conditions in the code. That is at any one time different threads may be searching for or accessing data, 'racing' to it. GIL means that this can't happen.

6. Concurrency is when sections of code are able to run alongside each other but aren't necessarily doing so. They may or not be performing an action at any one time. Parallelism is when one or more sections of code are performing their operations at exactly the same time.

7. KISS stands for Keep It Simple Stupid - this means that you never make your code more complicated or lengthy than it needs to be. DRY stands for Don't Repeat Yourself. To repeat a piece of code in software is considered to be unnecessary and overcomplicating. This is especially important with a language like Python which prides itself on its simplicity and readability. BDUF stands for Big Design Up Front and this is often used in conjunction with the Waterfall model of the software development life cycle or process. It refers to the process of ensuring the design and analysis of a product is thoroughly assessed and completed before implementation of a product begins. It is the more traditional approach and has it pros and cons.

8. Garbage Collector in Python is an automated memory management system. It is a module which can be imported and changed if a programmer thinks they need to change the way it manages memory (though this isn't often recommended). It works by counting how many references to objects there are in the program. It performs various Garbage "collections" every so often so that objects which no longer have any references to them are deleted. This ensure the program doesn't run out of memory. Historically programmers used to have to perform this checking and deletion of objects themselves.

9. Deadlock in a relational database is when two or more queries are locked into certain transactions in such a way that no transactions are able to occur. For example, a lock may occur on a transaction when it is waiting for another operation to be performed which itself can't be performed because another transaction is locked. It is a very bad thing in database management and should be avoided at

all costs. Livelocking is when transactions are not being performed because one transaction is waiting for another and hence nothing is being done. A bit like meeting someone in the street and both of you going left, then stopping before you bump into one another, the, both going right, then stopping before you bump into one another. Etc.

10. Flask is a Python web framework which we use to create a web interface. It is a module which can be imported into Python and we then use that to create an API which can ensure that a database interacts with a client-side program.

2. Python 2 and Python 3

Python 2 is an old version of Python which is no longer supported - this means that it will no longer receive upgrades or have bugs fixed. Python3 was introduced as a solution to all the issues and bugs that Python2 experienced. It was released in 2008. Some syntax in Python 2 is different from Python 3 - e.g a print statement doesn't need parentheses in Python 2 and in Python 2 an input didn't necessarily create strings in all instances.

3. Palindrome function

```
def check_palindrome(word):
    If len(word) ==1:
        Return True
    word == word[::-1]

print(check_palindrome(word))
```

4. Test for palindrome function

These tests will check 3 things - the output when a single letter word is passed into the function, the output when a palindrome is passed into the

function and the output when a non-palindrome is passed into the function.

```
From Unittest import TestCase
From <file_name> import check_palindrome

Class TestCheckPalindrome.TestCase

Def test_len_word(self):
    Expected = True
    Result = check_palindrome(word=I)
    self.AssertTrue(expected, result)

Def test_palindrome(self):
    self.AssertTrue(check_palindrome(“abba”)
```

I’ve tried a different syntax above as I think this may be more appropriate for the Assert True although I’m not sure - the first one is more like the one we had in the first example exercise in the lesson.

```
Def test_palindrome(self)
    self.assertFalse(check_palindrome(“mummy”)
```

5. Agile methodology

Product Backlog refinement - this is a meeting or series of meetings held at the very start of the process. The product backlog is a list of items which could be improved or worked upon, a list of ideas about how the product can be developed further. The refinement invites all stakeholders, the product manager and the scrum team and advisers to discuss what to prioritise and what to set as the goal for the sprint.

Daily Scrum - this is a daily meeting in which all members of the Scrum team come together to talk about what they have done that day, what problems they have encountered, what has gone well, what hasn’t and what improvements could be made. The daily scrum also lays out a focus and list of things to achieve the next day.

A meeting is held at the very end of the process after the sprint (name?) in which everyone comes together to discuss how it went, any problems and solutions and how the process could be improved the next time.

6. Exception Handling

Try

In this block of code is the action or operations that you want the program to perform. The reason that we put it in this Try clause is that if, for any reason, the code doesn't work we don't want the program to stop running. We don't want the code to break which is what will happen if an exception occurs. An exception is when something has been entered incorrectly or a value has been added that throw up an error - there are many different types of exceptions and errors. For example a value error or a type error e.g. entering a letter into variable which is expecting an integer.

Except

If an error or exception occurs the except block then comes into play. Rather than the program stopping altogether it comes to the line of code which tells it what to do if this happens. You are telling the computer, don't worry if this happens, try this and move on. E.g. print("Sorry the computer doesn't recognised that input. Please try again!" The program will then continue wherever you direct it to using your code.

Else

This runs only if the exception block does not. It will not run if an exception occurs and the except block comes into play. If there is an else clause it will therefore always run immediately after the try block.

Finally

This clause always runs whether or not the except clause runs. It will run regardless of how the try, except block has worked out.

Both the else and finally blocks are not obligatory in the try, except model. You could have one or the other or neither. However you always need a try and except block.

7. Connecting a Python program to a database

We connect a Python program to a database using an API - that is an Application Programming Interface. An API is an application which allows two programs to communicate with each other. A bit like a waiter in a restaurant, an API gets information from the customer or client and then uses this information to get what the client wants just as the waiter gets what the customer wants from the restaurant kitchen. The API acts as the go-between. In Python we can use the Flask module to do this. Flask is a web framework which allows programmers to set up a web interface with its own http web page. Using Flask and its associated methods, GET and PUT the API interacts with the database. The GET method makes a request for information from the DB and the PUT method requests that the results are stored in a certain URL. API requests get a response back. Response codes are 200 (request successful), 404 (request not found) or 400 (bad request).

8. SQL query

```
SELECT a.authorname, SUM(b.sold_copies)
FROM AUTHORS a
INNER JOIN
BOOKS b
ON a.bookname = b.bookname
SORT BY SUM(b. sold_copies) (DESC)
LIMIT 3;
```

9. Two number sum

#This function has two parameters - it will accept a list of integers (lst) and also an integer (the target sum). The function first creates an empty list. Using a for loop it iterates through lst once and then iterates over lst in a nested for loop to compare the numbers and to check if the resulting sum when they are added together is num (target sum). The if clause ensures that no single integers are added to themselves to get the sum (in this case 5 add 5 would result in ten but we don't want this to be added to the list) and also checks that the number in the first iteration added to the number in the second iteration is equal to the target sum. If they are equal

to that they are added to the numbers_in_sum list using the append method. Finally the numbers_in_sum list is returned.

```
def find_target_number(lst, num):
    numbers_in_sum = []
    for i in lst:
                    for j in lst:
                        if i !+ j and i + j == num
                            numbers_in_sum.append(i)
            Return numbers_in_sum
```

#This calls the function, passing a list and an integer as two positional arguments

```
find_target_numbers([3,5,-4,8,11,-1,6], 10)
```