ASSESSMENT

Python and MySQL
assessment test 2 hours

Michaela D'Mello 74 / 100

NO	TASK	POINTS
1	Theory questions	20
2	Concept question	8
3	Coding question	8
4	Coding question	8
5	Concept question	8
6	Concept with practical example	8
7	Concept question	8
8	SQL practical question	10
9	Coding question	22
	TOTAL	**100**

Fantastic work! This was won--derful assessment to mark, your answers & coding skills were immaculate! Testing in particular was very good - the unit tests were designed & structured well.

For improvement, I recommend being in-depth whereever possible. Answers in the theory part should have more depth (e.g. give examples if feasible!)

For coding - focus on different approaches (try to start from scratch, and use your old code to see where you can improve - e.g. finding out how to give a empty list back. When you step back, it becomes easier to see blind spots & formulate new plans

1. Python theory questions 16/20 give more info if feasible!	**10 oints**

1) **What is the program?**
   Instructions a computer executes to perform a task. ✓
2) **What is the process?**
   The code and activity of a program which can be executed by one or multiple threads. ✓
3) **What is Cache?**
   A temporary small amount of memory within CPU that the CPU is likely to reuse ✓ why would it reuse tho?
4) **What is Thread and Multithreading?**
   Thread = one command at a time ✓ be explicit. What is a command? discuss processes here and how they're handeld
   Multithread = processes multiple commands at once ✗ give more info!
5) **What is GIL in Python and how does it work?**
   GIL = Global Interpreter Lock ✓
   Effectively makes any CPU-bound Python program single-thread
6) **What is Concurrency and Parallelism and what are the differences?**
   Concurrency = more than 2 tasks can occur at overlapping times, but not necessarily at the same time
   Parallelism = tasks can occur at literally the same time ✓
   Example using retail: 2 lines for 1 cashier = concurrency vs. 2 lines for 2 cashiers = parallelism

7) **What do these stand for in programming: DRY, KISS, BDUF**

DRY= don't repeat yourself
KISS= keep it simple stupid
BDUF= Big Design Up Front

8) **What is Garbage collector? How does it work?**

Garbage collector = automatic process to free up memory when not in use to make way for other, new objects

9) **What are 'deadlock' and 'livelock' in a relational database?** 1

Deadlock = 2+ tasks waiting indefinitely for another task to release a lock (...?)
Livelock = 1 task has to wait indefinitely to acquire a lock

10) **What is Flask and what can we use it for?**

Flask is a Python web framework that is used to develop web applications.

more in-depth if ideal - examples, discuss APIs, etc

This is 8-marker, make sure to be more in-depth! So examples where feasible (e.g. decimal / division)

2. Discuss the difference between Python 2 and Python 3 5 / 8	**8 points**

Answer is great tho, but still go into more detail! Be as in-depth as possible, answer Why (why would someone choose python 3 over 2?

Python 3 is easier to understand than Python 2 as it has simpler syntax and more intuitive. For example, Python 3 uses Unicode, whereas Python 2 uses ASCII, and will return a decimal rather than whole number for divisions.
It also has more libraries, so Python 3 is more versatile than Python 2.
Range() in Python 3 has replaced rangex from Python 2.

3. Write a function that can define whether a word is a Palindrome or not (a word, phrase, or sequence that reads the same backwards as forwards, e.g. *madam*)	**8 points**

```
# Create a function that returns the reverse of word, which matches the
original string
def is_Palindrome(s):
    return s == s[::-1]
```

8/8

4. Write tests for the newly created Palindrome function. Provide a brief explanation for your test case options.	**8 points**

```
import unittest
from unittest import TestCase, main

class Test_Palindromes(TestCase):

    def test_correct(self):
        s = "madam"
        ans = is_Palindrome(s)
        if ans:
            print("Yes")
        else:
            print("No")
        self.assertTrue(ans)
```

8/8

```
    def test_incorrect(self):
        s = "banana"
        ans = is_Palindrome(s)
        if ans:
            print("Yes")
        else:
            print("No")
        self.assertFalse(ans)

if __name__ == '__main__':
    main()
```

First test tests a correct palindrome in which a 'Yes' should be printed, whereas the second test tests an incorrect palindrome in which a 'No' should be printed.

5. Agile methodology, Scrum: name at least 3 types of meetings that are exercised by Agile teams and describe the objective of each meeting.	**8 points**

8/8

1) Daily Scrum ✓
   a. Generally, no longer than 15 minutes
   b. Facilitated by Scrum Master
   c. Each person answers: what they did yesterday, what they are doing today, whether there are any obstacles they are facing

2) Sprint Planning ✓
   a. Should take no longer than 6 hours
   b. Whole team discusses estimated effort involved for each team
   c. User stories are assigned to individuals

3) Sprint Retrospective ✓
   a. Following the end of the whole project
   b. Should be no longer than an hour
   c. Includes Scrum Master, Product Owner and Dev Team
   d. Place to highlight what worked, what didn't and what they can learn for similar projects in the future

6. Exception handling in Python, explain what each of the following blocks means in the program flow: Try, except, else, finally	**8 points**

5 / 8

A "try" clause within a Python program works along with the "except" clause. The "try" clause is always executed until the "except" clause is encountered. ✓ why / when would the 'except' be encountered / what triggers it

"Else" usually used in conjunction with an "if" statement, it is run when the previous command is not true. ✗ This is for conditional statements - else in this instance would not be paired with an 'if' for exception handling

✓

"Finally" is a final step that will always run, similarly to the "try" clause. Whether the code runs the except or else clauses or not, the "finally" clause will always run.

7. How can we connect a Python program (process) with a database? Explain how it works and how do we fetch / insert data into DB tables from a python program.	**8 points**

✗

8. Given two SQL tables below: authors and books. • **The authors dataset has 1M+ rows** • **The books dataset also has 1M+ rows** Create an SQL query that shows the TOP 3 authors who sold the most books in total! 5 / 10 (top 3 authors, not books!)	**10 points**

AUTHORS

author_name	book_name
author_1	book_1
author_1	book_2
author_2	book_3
author_2	book_4
author_2	book_5
author_3	book_6

BOOKS

book_name	sold_copies
book_1	1000
book_2	1500
book_3	34000
book_4	29000
book_5	40000
book_6	4400

I kept getting errors due to ambigious columns? You may need to prefix columns with their table names like Books.book_name

```
SELECT book_name, sold_copies FROM Books
INNER JOIN Authors ON Authors.book_name = Books.book_name
ORDER BY sold_copies;
```

Order BY should also have DESC (right now its from lowest number of items sold to biggest)

Should have limit of 3 (TOP 3 authors)

Display author name - so it should be top 3 authors! Not book rankings instead - query should instead display the top 3 authors / top 3 best selling authors!

9. TWO NUMBER SUM: • Write a function that takes in a non-empty array of distinct integers and an integer representing a target sum. If any two numbers in the input array sum up to the target sum, the function should return them in an array, in any order. If no to numbers sum up to the target sum, the function should return an empty array. • Note that the target sum has to be obtained by summing two different integers in the array. You cannot add a single integer to itself in order to obtain the target sum. • You can assume that there will be at most one pair of numbers summing up to the target sum. **Sample Input:** numbers = [3, 5, -4 ,8, 11, 1, -1, 6] target_sum = 10 **Sample Output:** [-1, 11] the numbers can be in any order, it does not matter.	**22 points**

```
# Create a function
# Identify 2 integers from the list
# Ensure the 2 integers are not the same integer
# The two integers should equal the target
# Don't know how to return an empty array if no 2 numbers sum up to the
target sum
# Print all options as a list
def my_function(numbers):
    for x in numbers:
        for y in numbers:
            if x != y and x + y == target:
                print([x, y])

# Define list and target sum
numbers = [3, 5, -4, 8, 11, 1, -1, 6]
target = 10

# Call function
my_function(numbers)

#Output
[11, -1]
[-1, 11]
```


Solution works well - however, consider that it may not work for lists that have the same number e.g. [15, 15, 15], target = 45. In those scenarios, you just skip the numbers because of x != y! Instead, try iterating through index by index instead (see right).

For returning an empty list - try having a 'empty' variable that is returned as a default response (e.g. the empty list). If we do find an answer though, then it is overwritten

Your for loops begin at the beginning each time (e.g. the for y in numbers iterates through the whole list, causing double answers! Try the index approach instead to resolv

example solution:

```
def two_sum (numbers, target):
  output = []


  for idx in range(0, len(numbers) - 1):
      for jdx in range(idx + 1, len(numbers)):

          num1 = numbers[idx]
          num2 = numbers[jdx]
          if num1 + num2 == target:
              output = [num1, num2]
  return output
```