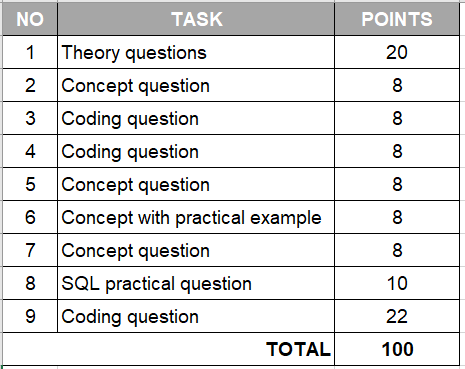
**ASSESSMENT 2**

**\*\*Option 2\*\***

Python and MySQL

assessment test 2 hours



|  |  |
| --- | --- |
| 1. **Python / SQL theory questions**  * questions 1 to 5 🡪 2 points * question 6 🡪 10 points | **20 points** |

1. What is Thread and Multithreading?

Thread: Contains the context of an execution. It controls what processes/functions are executed and in what order, and it defines what that priority order is. When only one process can be executed at a time a thread stores where in that process/function it was executed until before another process started to be executed, so that when it returns to the original thread it knows where it left off.

Multi-threading: When two or more things/threads are being carried out at the same time as one another. They use the same allocated resources as one another and so this can be less efficient.

1. What is Concurrency and Parallelism and what are the differences?

Concurrency is when multiple computations can not be carried out at the exact same time but it manages what is done and when. It does this by using context switching. One computation must be stopped or paused before another can be started.

Parallelism is when multiple computations are carried out at the exact same time.

Concurrency can be carried out using one processing unit, but parallelism requires two or more. Concurrency increases the time taken for a set number of functions, parallelism decreases the speed this can be done at. Debugging is generally a lot harder in concurrency. It’s still hard with parallelism but more simple than concurrency.

1. What is Garbage collector? How does it work?

A garbage collector is a way of freeing up memory in computations, when the object that requires that memory is no longer in use. When an object is no longer referenced this counts as not requires anymore and so it triggers the garbage collector. For example, if an object is referred to as ‘a’, then ‘b’ is set equal to ‘a’, the garbage collector will delete ‘a’. This is because once it’s stored as ‘b’, ‘a’ is wasting storage space as they are set to the exact same value. It works through reference counting. So as ‘b’ has been set to ‘a’, and ‘a’ is not referenced any more in the code, it recognises it can delete ‘a’. The reference count is decreased when another variable is set equal to the original variable. You can use sys.getrefcount(a) to see the reference count of the variable.

1. What is Transaction Management in a relational database (give an example)?

Transaction management is a program unit or unit of work which contains one or more operations to access a database. It ensures a function or query is not partially complete and gives you the choice of committing to any changes made within the transaction or rolling back, without damaging/deleting data, which is particularly helpful when an error occurs or if a value is input incorrectly. For example, if you are transferring stock from one account to another, and a payment is required for this, you have two main operations. These are the transfer of stock and the transfer of money. There are two errors that can occur here: trying to transfer more stock than is available or trying to pay more money than is available. The order might look like this: add stock to the requesting account, removing that stock from the sending account, adding money to the sending stock account, removing money from the receiving stock account. If you add more stock to the receiving account then discover there wasn’t enough stock available to give in the sending account, you have an error that can’t be done. Transaction management solves this by going through this function before making any changes. It will rollback if this error occurs. The same goes for transferring money then discovering you don’t have that much money available.

1. What is an endpoint and what are the most common methods to interact with the API data source?

An endpoint is usually what comes at the end of a URL i.e. the end of a communication method (when working with APIs. Each endpoint contains the output of a specific function, so they can be saved then updated/changed/removed as needed. It is a way of saving the access to a resource so the computer can carry out certain functions.

The most common methods are GET and POST. GET is used to access information from the API, PUT is used to add more data to the API.

1. What is data normalization in SQL? Please provide an example (any) of a database restructuring using primary/foreign keys to maintain data integrity.

Normalisation is a way of organising data in tables within a database so that the table contains information ideally about just one topic. This means you avoid duplicating identical data in multiple tables, for example including where an item is manufactured in every table where the item ID exists. You can include just one table which contains item manufacture location and it’s ID so this is not required in any other table, as long as the ID is a primary key in another table. It is also used to avoid modification issues and allows you to simplify your queries to the database and tables within it. For normalisation to occur you must have foreign keys. Normalisation reduces the change of errors in duplicates, reduces memory required by the database, and speeds up data maintenance and making queries.

There are a few types of data normalisation: 1NF, 2NF, …, 6NF.

For example, a food shop database which contains one table including: item ID, brand, item name, manufacture location, quantity, storage department, company manufacture.

This can be normalised by splitting into various tables such as the following:

* + - 1. Item name (item name and ID)
      2. Brands (item ID, brand)
      3. Quantity left (item ID, item name, quantity)
      4. Storage Department (item ID, storage department)
      5. Place of manufacture (brand, manufacture location)
      6. Company of manufacture (brand, item ID)

Item ID will be the primary key in any table it exists. For some tables such as place of manufacture, brand will be the primary key and will be linked to the company of manufacture table where brand is the foreign key.

|  |  |
| --- | --- |
| **Discuss Exception handling (4 pts) and debugging in Python (4 pts)** | **8 points** |

Exceptions: exceptions are used when an error doesn’t occur but when the Python script can’t quite handle what you’re trying to execute. For example, an error would be when you have something like a missing colon or start a new line where it’s not meant to be. An exception will still allow the code to run but not properly. For example, if you try to divide a number by 0 (ZeroDivisionError) an exception is raised. Or if you try to run a function on an undefined variable (NameError). Or if you try to concatenate a string and integer you get a TypeError. The python script will terminate as soon as it comes across an error that can’t be handled. There are two types of errors: Syntax and Logical (which are exceptions). Logical errors are found after the code has been run and no syntax errors are found, and it will usually run the code until this point. You can build in ways of dealing with exceptions into your python script, if there are some which are likely to occur, especially when you include user inputs. The ‘try’, ‘except’, ‘else’ and ‘finally’ blocks are used to handle this. You execute the ‘except’ block when an expected exception occurs, such as running a division function and trying to divide by 0. If the code runs fine you run the ‘else’ block, then ‘finally’. ‘else’ and ‘finally’ are not required.

Debugging: This is the method used to identify, locate and fix and errors within the code. When you don’t know what an error is caused by or where it is located, it is referred to as a bug, and debugging is the process of rectifying this. At the end of debugging you should have found and fixed the bug/error. In most Python environments you can select a line close to where you think the bug is located and run the code until that line. If that runs smoothly you know the code is fine until that point. You can then continue to the next like or enter that line of code. For example, if that line is an ‘if’ loop you can either choose to enter that loop or move past it onto the ‘else’ block if that exists. You can see the value of each variable defined at each line of code when debugging, so if there is an error relating to this it can be used to figure out the cause.

|  |  |
| --- | --- |
| 1. **Write a function that takes in a non-empty array of integers that are sorted in ascending order and returns a new array of the same length with the squares of the original integers also sorted in ascending order.**   **Example Input:**  numbers = [1,2,3,5,6,8,9]  **Example Output:**  [1,4,9,25,36,65,81] | **8 points** |

original\_array = [1, 2, 3, 5, 6, 8, 9]  
  
  
def sorted\_squares(input\_array):  
 try:  
 for num in input\_array:  
 if int(num) != num:  
 raise Exception  
 input\_array.sort()  
 output\_array = [num \*\* 2 for num in input\_array]  
 output\_array.sort()  
 print(output\_array)  
 return output\_array  
 except:  
 output = 'You tried to insert a non-integer value!'  
 print(output)  
 return output  
  
  
sorted\_squares(original\_array)

|  |  |
| --- | --- |
| 1. **Write tests for the newly created Sorted Squared Numbers function (in Q3). Provide a brief explanation for your test case options.** | **8 points** |

# Test the sorted squares function  
import unittest  
from assessment2\_SO import sorted\_squares  
  
  
class TestSortedSquaresFunction(unittest.TestCase):  
  
 # Input an unsorted array and check the output is sorted  
 def test\_list\_sorted(self):  
 expected = [1, 4, 9, 16, 25]  
 input\_array = [5, 3, 2, 4, 1]  
 self.assertEqual(sorted\_squares(input\_array), expected)  
  
 # check it's calculating the square correctly  
 def test\_squares\_calculation(self):  
 expected = [25]  
 input\_array = [5]  
 self.assertEqual(sorted\_squares(input\_array), expected)  
  
 # The code should not run if a non-integer value is input. Test this is the case. An exception should be raised  
 def test\_integer\_input(self):  
 expected = 'You tried to insert a non-integer value!'  
 input\_array = [1.6, 1.5, 1.2]  
 self.assertEqual(sorted\_squares(input\_array), expected)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

|  |  |
| --- | --- |
| 1. **Agile methodology: name and describe any 2 of the main roles in a Scrum Agile team.** | **8 points** |

The scrum master: Essentially the leader of the scrum project, making sure all team members are well versed in how to communicate, coordinate and work together. They make sure everyone has all the information they need. They provide support to other team members when re quired and facilitates meetings and events that take place. They coordinate communications with other teams and the product owner. They run the daily scrum meeting and run the sprints that take place, filtering through priorities given by the product owner. They also support the adoption of the final product.

The Product owner: They must communicate the requirements and goals of the project to the wider team and communicate the items in the backlog effectively. They order the backlog items by priority and select the most important few the development team must work on in each sprint. They represent the needs of the client and other stakeholders and so their needs must be respected by the other team members.

|  |  |
| --- | --- |
| 1. **Discuss advantages and disadvantages of TDD (Test Driven Development):** | **8 points** |

Test driven development is when testing, development (coding) and design are very closely linked.

Advantages:

* Reduced chance of errors in code so less time going through and identifying the mistake
* Mistakes are spotted early so development becomes more efficient
* Generally results in higher quality output as all teams work closely together
* Reduces the amount of effort required in the final phases of the project

Disadvantages:

* Can be time consuming to write so many tests
* Easy to forget to run the tests as often as they should to make TDD work properly
* Everyone in the team must be on board to make it effective. If someone doesn’t follow the process it becomes less useful.
* Tests can take memory so writing too large tests can result in higher inefficiencies

|  |  |
| --- | --- |
| 1. **What is a Python DB cursor? Provide an example** | **8 points** |

It is used to execute statements and traverse through a database in MySQL through Python. It is accessed through the mysql-connector-python module. It must be created to execute queries in Python to MySQL. It is a class/object that has many methods build in to it. It is used to execute statements to communicate with the MySQL Database.

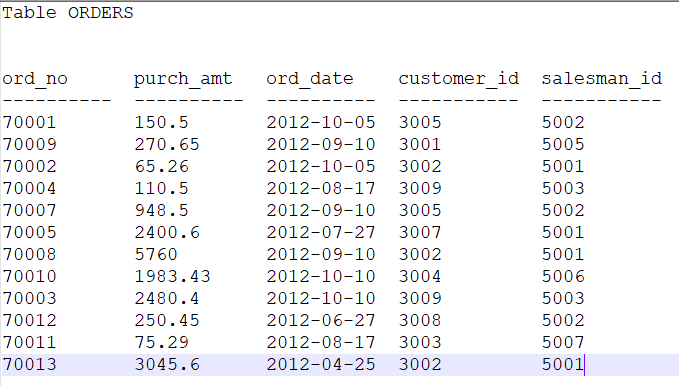
For example:

crs = connection.cursor() establishes a connection to the database and must be executed to access certain items within the database. You can connect to the database beforehand but this must be done before executing any queries.

crs.execute(query) allows you to execute a query. i.e. access a certain column or print all the data in a table within the database.

crs.close() closes the cursor and is done just before closing the connection.

|  |  |
| --- | --- |
| 1. **Given an example table below**  * Write a SQL query to find the maximum order (purchase) amount for each customer. * The customer ID should be in the range 3002 and 3007 (begin and end values are included.). * Filter the rows for maximum order (purchase) amount is higher than 1000. * Return customer id and maximum purchase amount. | **10 points** |



**MySQL query:**

SELECT

o.customer\_id, MAX(o.purch\_amt)

FROM

orders o

WHERE

3002 <= o.customer\_id <= 3007

AND o.purch\_amt > 1000

GROUP BY o.customer\_id;

**Python DB Query:**

The below is very similar to the examples given in the lectures. I created a template from this and input the values required to access the data needed from the DB.

# QUESTION 8: SQL  
import mysql.connector  
from config import USERNAME, PASSWORD, HOST  
  
  
class DatabaseConnectionError(Exception):  
 pass  
  
  
def connect\_to\_db(db):  
 cnx = mysql.connector.connect(  
 host=HOST,  
 user=USERNAME,  
 password=PASSWORD,  
 database=db  
 )  
 return cnx  
  
  
def get\_max\_purch\_amt():  
 try:  
 db\_name = 'shop'   
 db\_cnx = connect\_to\_db(db\_name)  
 crs = db\_cnx.cursor()  
 print("Connected to DB: %s" % db\_name)  
  
 max\_query = """SELECT o.customer\_id, MAX(o.purch\_amt) FROM orders o WHERE 3002 <= o.customer\_id <= 3007  
 AND o.purch\_amt > 1000 GROUP BY o.customer\_id;"""  
 crs.execute(max\_query)  
 result = crs.fetchall()  
  
 for i in result:  
 print(i)  
 crs.close()  
  
 except Exception:  
 raise DatabaseConnectionError("Data could not be read from the Database")  
  
 finally:  
 if db\_cnx:  
 db\_cnx.close()  
 print("Connection to the BD is closed")  
  
  
get\_max\_purch\_amt()

|  |  |
| --- | --- |
| 1. **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** |

number\_array = [3, 5, -4, 8, 11, 1, -1, 6]  
number = 10  
  
  
def two\_number\_sum(numbers, target\_sum):  
 try:  
 if not numbers: # i.e. if the array is empty, raise an exception  
 raise Exception  
  
 for i in range(len(numbers)-1):  
 for j in range(i, len(numbers)-1, 1): # For numbers that aren't the number you're currently adding to. Range ensures you're not trying the same addition but with the numbers the other way around  
 if i != j:  
 output = numbers[i] + numbers[j]  
 if output == target\_sum:  
 return [numbers[i], numbers[j]]  
 break  
  
 except:  
 return 'Your array is empty'  
  
  
def two\_number\_sum\_output(num\_array, num):  
 result = two\_number\_sum(num\_array, num)  
 if not result:  
 return 'Your numbers did not add up to'  
 print('Your numbers did not add up to', num)  
 else:  
 print(result)  
 return result  
  
  
two\_number\_sum\_output(number\_array, number)

**Tests to run on the function:**

# Test the two number sum function  
import unittest  
from assessment2\_SO import two\_number\_sum\_output  
  
  
class TestTwoNumberSumFunction(unittest.TestCase):  
  
 # When some elements add up to 10  
 def test\_add\_to\_10(self):  
 expected = [11, -1]  
 input\_array = [3, 5, -4, 8, 11, 1, -1, 6]  
 target\_num = 10  
 result = two\_number\_sum\_output(num\_array=input\_array, num=target\_num)  
 self.assertEqual(expected, result)  
  
 # When an empty array is input  
 def test\_empty\_array(self):  
 expected = 'Your array is empty'  
 input\_array = []  
 target\_num = 10  
 result = two\_number\_sum\_output(num\_array=input\_array, num=target\_num)  
 self.assertEqual(expected, result)  
  
 # When no numbers in the array add up to 10  
 def test\_not\_add\_to\_10(self):  
 expected = 'Your numbers did not add up to'  
 input\_array = [3, 5, -4, 8, 11, 1, 1, 6]  
 target\_num = 10  
 result = two\_number\_sum\_output(num\_array=input\_array, num=target\_num)  
 self.assertEqual(expected, result)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()