



**FOUNDATION ASSESSMENT I - 2 HOURS**

| **SECTION** | **MARK** |
| --- | --- |
| **Theory Questions** | 28 |
| **Concept Questions** | 22 |
| 1. **Python Challenge** | 25 |
| 1. **SQL Challenge** | 25 |
| 1. **TOTAL** | **100** |

**Important notes:**

* You will be submitting an edited version of this documenton Slack to your marker.
* It is an open book exam.
* You are allowed to use PyCharm and MySQL Workbench for this assessment.

**Section 1: Theory Questions [28 marks]**

|  |  |
| --- | --- |
| **1.1 What is the process called where databases get restructured to**  **better data integrity ?**    Database normalization | **1 mark** |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | **1.2 What is the type of JOIN statement that returns rows when there is**  **at least one match in both tables? Present a venn diagram which shows this visually.** | **2 marks** | | **2 marks** |

|  |  |
| --- | --- |
| **1.3 What is the difference between functions and methods?**    A function includes a set of instructions that will perform a ‘task’. It is independent of an object.    A method is a set of instructions that are associated with an object. It is only used for the object for which it is called and is accessible to data that is within the class. | **2 marks** |

|  |  |
| --- | --- |
| **1.4 What is a subquery?**    A query that is within another query or statement. They are used when more than one result is requested. | **2 marks** |

|  |  |
| --- | --- |
| **1.5 Explain the difference between “x = y” and “x == y” in python.**    A single ‘=’ is used to declare a variable. In this case y is being assigned to x.    Two ‘==’ compares the two values and returns either true or false. | **2 marks** |

|  |  |
| --- | --- |
| **1.6 What are the notations for a list, set and tuple?**    list = []  set = {}  tuple = () | **3 marks** |

|  |  |
| --- | --- |
| **1.7 Explain the relationship between the following sets:**  {1,2,7,8}, {7,1,2,8}, {1,2}, {3}    A set is a collection of unordered, unique items.    The second set is a superset of the first. It contains all of the same items, but in a different order.    The third set is a proper subset of either the first or second. It does not contain all of the same values, only 2.    The fourth has no relation. We cannot access the individual items of a set so it is not simply the sum of the values of the third set. | **3 marks** |

|  |  |  |
| --- | --- | --- |
| **1.8 Describe the behaviour of the following code.**     |  | | --- | | **while True:**  **print("Hello")** |     Whilst a statement is true the string Hello will be printed.  In this instance, it is forever true, and hello will be printed over and over again eternally. | **3 marks** |

|  |  |
| --- | --- |
| **1.9 Give 2 examples of libraries you can use in Python and what they**  **are used for.**    Random  The random module can be used to select a random element, usually a number or numbers. This can be from a defined range or given sequence.    Datetime  This module includes classes that allow the use and manipulation of date and time data. For example, it can be used to specify the current time or calculate the date a certain number of days from now. | **4 marks** |

|  |  |
| --- | --- |
| **1.10 List 3 differences between stored procedures and stored**  **Functions**      A stored procedure returns multiple values but a stored function only returns one.    A stored procedure can read and modify data but a stored function is read only.    A stored procedure can call a function but not the other way around. | **6 marks** |

**Section 2: Concept Questions [22 marks]**

| **2.1** Convert this program into an equivalent list comprehension   | **students = ["Ana", "beth", "CHARLIE"] # keep this line**  **formatted\_students = []**  **for student in students:**  **formatted\_students.append(student.lower())** | | --- |   students = ["Ana", "beth", "CHARLIE"]  formatted\_students = [students.lower(" ") for name in students]  print(formatted\_students) | **4 marks** |
| --- | --- | --- |

| **2.2** Given that **input\_str = “practice”**, use string slicing to manipulate the  string so the output is “tap”.  input\_str = 'practice' ans\_1= input\_str[-4:-9:-2] print(ans\_1) | **4 marks** |
| --- | --- |

| **2.3** Write code that reads in a file “input.txt” and for each line in the file  appends **“...”** on a new line  (Hint: first, figure out how many lines this file has)  filename = input("Enter the name of the text file: ")   def file\_read():  with open('file', 'r''w') as f:  for line in file:  words = line.strip().split()  file\_read('name of file here') | **6 marks** |
| --- | --- |

| **2.4** Write a function that takes in a dictionary representing a receipt of  items and their prices, and then returns the total price.   * The dictionary needs to be composed of at least 4 items * Each item will have a key stating what the item is * The value will be the price of the item as a float * The total price is expected to be explicitly formatted   items = {  "apple": 1.20,  "cake": 7.50,  "orange": 2.00,  "pizza": 6.00 }  def total():  total\_cost = sum(items.values())  print(f"The total is: £{total\_cost}")   total() | **8 marks** |
| --- | --- |

**Section 3: Coding Challenge [25 marks]**

You have been tasked with creating a date calculator program that can add and compare dates based on input values:

| **add** | adding X days to a inputted date and returning the resultant date |
| --- | --- |
| **compare** | Given 2 dates, return the difference in days. |

You are expected to identify and use an inbuilt python module for this task. You can look through the python documentation to help you.

<https://docs.python.org/3/library/>

The example walkthroughs below show how the data is meant to be processed and returned.

***Example Walkthroughs***:

| Please select one of the following options: add, compare  add  What is the date you want to add to? Please enter in DD/MM/YYYY format.  14/10/2000  How many days do you want to add?  56  The resultant date is 09/12/2000 |
| --- |

| Please select one of the following options: add, compare  compare  Please give Date 1 in DD/MM/YYYY format.  14/10/2000  Please give Date 2 in DD/MM/YYYY format.  06/05/1999  There are 527 days between the given dates. |
| --- |

from datetime import datetime, timedelta  
  
def add\_date():  
 date = input('Please enter the date in the following format DD/MM/YYYY')  
 date\_obj = datetime.strptime(date, '%d/%m/%Y')  
 day\_add = int(input('Please enter how many days you wish to add: '))  
 output = date\_obj + timedelta(days=day\_add)  
 new\_date\_formatted = output.strftime('%d/%m/%Y')  
  
 print(f'The resultant date is {new\_date\_formatted} ')  
  
  
def compare\_date():  
 date\_a = input('Please enter the first date in the following format DD/MM/YYYY')  
 da = datetime.strptime(date\_a, '%d/%m/%Y')  
 date\_b = input('Please enter the first date in the following format DD/MM/YYYY')  
 db = datetime.strptime(date\_b, '%d/%m/%Y')  
 delta = da - db  
 print(f'There are {delta.days} days between the given dates')  
  
  
def add\_compare():  
 task = input('Please select one of the following: add or compare')  
 if task == 'add':  
 add\_date()  
 elif task == 'compare':  
 compare\_date()  
  
add\_compare()

**Section 4: SQL Challenge [25 marks]**

In this section you will be fleshing out a database and performing queries.

**Starter Code:**

CREATE DATABASE foundation\_assessment\_i;

USE foundation\_assessment\_i;

| **4.1 Write (and execute) the syntax to create the following tables:**  **USE foundation\_assessment\_i;**  **A] *students*** *Table*  *USE foundation\_assessment\_i;*    *CREATE TABLE students (*  *Student\_ID integer(11) DEFAULT NULL,*  *FORENAME varchar(50) DEFAULT NULL,*  *SURNAME varchar(50) DEFAULT NULL*  *);*     | Student\_ID | Forename | Surname | | --- | --- | --- | |  |  |  |   **B] *exams*** *Table*  *CREATE TABLE exams (*  *Exam\_ID integer(11) DEFAULT NULL,*  *Exam\_Name varchar(50) DEFAULT NULL,*  *Max\_Mark integer(4) DEFAULT NULL*  *);*     | Exam\_ID | Exam\_Name | Max\_Mark | | --- | --- | --- | |  |  |  |   **C] *results*** *Table*  *CREATE TABLE results (*  *Result\_ID integer(11) DEFAULT NULL,*  *Exam\_ID integer(11) DEFAULT NULL,*  *Student\_ID integer(11) DEFAULT NULL,*  *Mark integer(4) DEFAULT NULL*  *);*     | Result\_ID | Student\_ID | Exam\_ID | Mark | | --- | --- | --- | --- | |  |  |  |  | | **10 marks** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

| **Populate the database!**  Use the file*foundation\_assessment\_i.sql* to fill your tables with the needed data.  You may need to change the names of the tables in the SQL file if yours don’t align. |
| --- |

| **4.2 Write a query to list students’ forenames and surnames where they**  **scored higher than 60 and the respective exam.**  SELECT students.FORENAME, students.SURNAME  FROM foundation\_assessment\_i.students  INNER JOIN foundation\_assessment\_i.result  ON result.Student\_ID = students.Student\_ID  WHERE mark > 60; | **6 marks** |
| --- | --- |

| **4.3 Write a query that checks for suspected collusion in an exam**  **where students receive the same mark, and returns the students’**  **full names, the suspected exam, and their mark***.*  *For simplicity, you can assume there won’t be identical marks across different exams.*  *So if 62 is a mark in Exam 1, there won’t be a 62 in Exam 2.*  SELECT students.FORENAME, students.SURNAME, exams.Exam\_Name, result.Mark  FROM foundation\_assessment\_i.students  INNER JOIN foundation\_assessment\_i.result  ON result.Student\_ID = students.Student\_ID  INNER JOIN foundation\_assessment\_i.exams  ON exams.Exam\_ID = students.Exam\_ID  WHERE COUNT(DISTINCT(result.mark)) >1; | **9 marks** |
| --- | --- |