

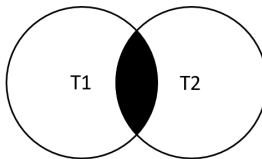


# FOUNDATION MOCK EXAM MARK SCHEME

- You will need to run the student's code to test.
- In line with the plagiarism policy, award a 0 for any question where plagiarism has been found.
  - Note down this case to track if it is a repeated offence
- A reminder of the assessment breakdown can be seen on the right.
- Example answers have been provided, but please note this is for guidance only and all **correct** alternative answers should be awarded the marks.

SECTION	MARK
<b>Theory Questions</b>	28
<b>Concept Questions</b>	22
<b>Python Challenge</b>	25
<b>SQL Challenge</b>	25

## Section 1 - Theory Questions [28 marks]

Question	Marking Point	Example Answer
<b>1.1</b> What is the process called where databases get restructured to better data integrity? <b>(out of 1)</b>	<b>[1]</b> - Normalisation	"Normalisation"
<b>1.2</b> What is the type of JOIN statement that returns rows when there is at least one match in both tables?  Find a venn diagram which shows this visually. <b>(out of 2)</b>	<b>[1]</b> Identifying that this is an <b>Inner Join</b> <b>[1]</b> Correct Venn Diagram	"An inner join is where there's a match from both tables.  A diagram that represents this visually can be seen here" 

<b>1.3</b> What is the difference between functions and methods?  <b>(out of 2)</b>	<b>[2] - A valid comparison and example</b> <ul style="list-style-type: none"> <li>How they are called, functions can be called anytime with brackets, methods are called as dot notations to a variable.</li> <li>Functions can be passed any data type, methods are often tied to a specific data type.</li> </ul>	"Functions can be passed any data type, methods are often tied to a specific data type."
Question	Marking Point	Example Answer
<b>1.4</b> What is a subquery?  <b>(out of 2)</b>	<b>[1]</b> - A nested query/ a query in a query <b>[1]</b> - The results of one query is used in the other via a relational operator or aggregation function	"A subquery is a nested query where the results of one query can be used in another query via a relational operator or aggregation function"
<b>1.5</b> Explain the difference between "x = y" and "x == y" in python.  <b>(out of 2)</b>	<b>[1]</b> - "x=y" means that x is now given the value of y <b>[1]</b> - "x == y" is checking if x and y are equivalent values	"A single equals means assignment, an example of when this is used is when defining variables - in this case x is now equal to the value of y.  A double equals means you're checking if a value is equivalent to another value, in this case x and y are being checked for equivalence"
<b>1.6</b> What are the notations for a list, set and tuple?  <b>(out of 3)</b>	<b>[1]</b> - List [] <b>[1]</b> - Set {} <b>[1]</b> - Tuple ()	"The notations for a list, set and tuple are very similar, all being brackets. List is denoted with square brackets, set with curly braces, and tuple with parentheses."
<b>1.7</b> Explain the relationship between the following sets: {1,2,7,8}, {7,1,2,8}, {1,2}, {3}  <b>(out of 3)</b>	<b>[1]</b> - {1,2,7,8} and {7,1,2,8} are the same because sets have no order <b>[1]</b> - {1,2} is a subset of {1,2,7,8}/{7,1,2,8} <b>[1]</b> - {3} has no relationship to the other sets.	"{1,2,7,8} and {7,1,2,8} can be considered the same set, as sets are unordered and the contents are the same. {1,2} is a subset of that set and {3} has no relationship with any set."

<p><b>1.8</b> Describe the behaviour of the following code.</p> <pre>while True:     print("Hello")</pre> <p><b>(out of 3)</b></p>	<p><b>[1]</b> - Infinite Loop <b>[2]</b> - Prints Hello until forceful exit</p>	<p>"This code will create an infinite loop where it will continuously print out "Hello" until forcefully exited from the program."</p>
Question	Marking Point	Example Answer
<p><b>1.9</b> Give 2 examples of libraries you can use in Python and what they are used for.</p> <p><b>(out of 4)</b></p>	<p><b>[2] - A valid example and description</b> <i>Examples</i></p> <ul style="list-style-type: none"> <li>• Requests, used to interact with APIs</li> <li>• Pandas, used to manipulate data (e.g. for visualisation)</li> <li>• Numpy, used for complex mathematical computations</li> <li>• TensorFlow, used for machine learning</li> <li>• Any valid library should be accepted</li> </ul>	<p>"Requests is a library that is used to make HTTP calls to APIs.</p> <p>Numpy is a library that allows for easier complex calculations, specifically with the capacity for dimensional data"</p>
<p><b>1.10</b> List 3 differences between stored procedures and stored functions</p> <p><b>(out of 6)</b></p>	<p><b>[2] - For each valid discussion point showing difference between stored procedures and functions</b> <i>Examples</i></p> <ul style="list-style-type: none"> <li>• Procedures return many values. Functions return just 1.</li> <li>• Procedures have input and output parameters. Functions only have input parameters.</li> <li>• Procedures can't be used in SELECT statements, Functions can.</li> <li>• Procedures can work with INSERT/UPDATE/DELETE but Functions only use SELECT</li> </ul> <p><i>Deduct 0.5 as necessary if there wasn't a valid <u>comparison</u>.</i></p> <p><i>Students are allowed to answer this using bullet points as it was asked to 'List'.</i></p>	<p>"Stored procedures can return many values but stored functions are limited to 1. This means procedures has output parameters, whereas functions only have input parameters. Stored procedures can't be used in SELECTs whereas functions can."</p>

## Section 2 - Concept Questions [22 marks]

Question	Marking Point	Example Answer
<p><b>2.1</b></p> <p>Convert this program into an equivalent list comprehension</p> <pre>students = ["Ana", "beth", "CHARLIE"] formatted_students = []  for student in students:      formatted_students.append(student.lower())</pre> <p><b>(out of 4)</b></p>	<p><b>[1]</b> - Making a variable called formatted_students</p> <p><b>[3]</b> - List comprehension stating:</p> <ol style="list-style-type: none"> <li>1. student.lower()</li> <li>2. For student</li> <li>3. In students</li> </ol>	<pre>students = ["Ana", "beth", "CHARLIE"]  formatted_students = [student.lower() for student in students]</pre>
<p><b>2.2</b></p> <p>Given that <b>input_str = "practice"</b>, use string slicing to manipulate the string so the output is "tap".</p> <p><b>(out of 4)</b></p>	<p><b>[1]</b> - Identifying the list needs to be reversed and implementing a reversal</p> <p><b>[1]</b> - Start index as 3 or equivalent</p> <p><b>[1]</b> - End index as 8 or equivalent</p> <p><b>[1]</b> - Step value of 2</p> <p>Ultimately if it works, it gets 4 marks</p>	<pre>&gt;&gt;&gt; input_str[::-1][3:8:2] 'tap'</pre> <p>N.B. Always test their answer as reverse indexing can also get the correct answer and deserves all marks, e.g.</p> <pre>&gt;&gt;&gt; input_str[-8:-3:2][::-1] 'tap'</pre>

<p><b>2.3</b></p> <p>Write code that reads in a file "input.txt" and for each line in the file appends "..." on a new line.</p> <p>(Hint: first figure out how many lines this file has)</p> <p><b>(out of 6)</b></p>	<p><b>[1]</b> - Set a variable for number of lines</p> <p><b>[1]</b> - Open file in read mode (default)</p> <p><b>[1]</b> - Increment lines variable</p> <p><b>[1]</b> - Open file in append mode</p> <p><b>[1]</b> - Add specified number of '...' to file</p> <p><b>[1]</b> - '\n' tag included to put it on a new line</p>	<pre>lines = 0 with open("input.txt") as file:     for line in file.readlines():         lines += 1  with open("input.txt", "a") as file:     for i in range(lines):         file.write('\n...')</pre>
<p><b>2.4</b></p> <p>Write a function that takes in a dictionary representing a receipt of items and their prices, and then prints out the total price.</p> <ul style="list-style-type: none"> <li>• The dictionary needs to be composed of at least 4 items</li> <li>• Each item will have a key stating what the item is</li> <li>• The value will be the price of the item as a float</li> <li>• The total price is expected to be explicitly formatted</li> </ul> <p><b>(out of 8)</b></p>	<p><b>[2]</b> - Creating a dictionary with 4 items, and all values being valid floats</p> <p><b>[1]</b> - Function definition accepting dictionary object</p> <p><b>[3]</b> - Getting total of all values (through for loop or sum of dictionary values)</p> <p><b>[2]</b> - returning the total <u>formatted to 2 decimal places.</u></p>	<pre>receipt_dict = {     "TV": 1000.00,     "Mouse": 12.50,     "Screen Protector": 9.99,     "USB Stick": 4.99 }  def total_price(receipt):     total = 0     for k,v in receipt.items():         total += v     return round(total, 2)  print(total_price(receipt_dict))</pre>

## Section 3 - Python Challenge [25 marks]

Create a date calculator program that can add and compare dates based on input values.

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

Marking Points	Example Answer
<p><b>[1]</b> - For correctly identifying and importing datetime as the helpful module</p> <p><b>[2]</b> - For getting the datetime and timedelta module objects</p> <p><b>[1]</b> - For outputting the menu options and capturing the response</p> <p><b>[1]</b> - For if/elif/else structure regarding the response. Students can omit an error conditional state but it is better practice.</p> <p><b>[11]</b> - For "add":</p> <ul style="list-style-type: none"> <li>• [1] - Outputting a message asking for a date in an explicit format <b>and</b> capturing it</li> <li>• [2] - For using the strptime method to create a datetime object, <u>using the explicit format pattern</u></li> <li>• [2] - Outputting a message asking for how many days to add, <b>and</b> casting this to an integer</li> <li>• [3] - For calculating the new date by <u>adding the datetime object</u> to the <u>timedelta</u> function given the <u>days parameter</u> is set to the days to add</li> <li>• [3] - For printing out the result, using the strftime method to create a string object, <u>using the explicit format pattern</u></li> </ul> <p><b>[7]</b> - For "compare":</p> <ul style="list-style-type: none"> <li>• [1] - Outputting a message asking for both dates in an explicit format <b>and</b> capturing it</li> <li>• [2] - For using the strptime method to create datetime objects for both, <u>using the explicit format pattern</u></li> <li>• [3] Getting the <u>absolute value</u> of <u>subtracting one datetime from the other</u> and <u>getting the days attribute</u> (abs will prevent negative differences)</li> <li>• [1] - For printing out the result</li> </ul> <p><b>[2]</b> - For passing the test cases on the next page.</p>	<pre> from datetime import datetime, timedelta  choice = input("Please select one of the following options: add, compare\n") if choice.lower() == "add":     date_one = datetime.strptime(input("What is the date you want to add to? Please enter in DD/MM/YYYY\n"), "%d/%m/%Y")     days_to_add = int(input("How many days do you want to add?\n"))     resultant_date = date_one + timedelta(days=days_to_add)     print("The resultant date is {}".format(resultant_date.strft ime("%d/%m/%Y"))) elif choice.lower() == "compare":     date_one = datetime.strptime(input("Please give Date 1 in DD/MM/YYYY format\n"), "%d/%m/%Y")     date_two = datetime.strptime(input("Please give Date 2 in DD/MM/YYYY format\n"), "%d/%m/%Y")     days_difference = abs((date_two-date_one).days)     print("There are {} days between the given dates".format(days_difference)) else:     print("Invalid Choice. Restart and try again.") </pre>

## Tests (out of 2)

*This program did not need to be in a function as it wasn't stated, as such you are expected to test by running the code and interacting with it as the user.*

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.

## Section 4 – SQL Challenge [25 marks]

For the latter queries, you are strongly advised to run their query to see if it generates the correct output.

### 4.1 Write syntax to create the following tables so they look like this:

#### A] *students* Table

Student_ID	Forename	Surname

#### B] *exams* Table

Exam_ID	Exam_Name	Max_Mark

#### C] *results* Table

Result_ID	Student_ID	Exam_ID	Mark

### Marking Points (10 marks)

There are ten fields across the three tables.

- 1 mark for creating each field **with** an appropriate data type and identifying which one should be the primary key
  - Award 0.5 marks for ID fields not marked with primary key, or if student has used an incompatible data type*

### Example Query

```
CREATE TABLE students(
    Student_ID integer PRIMARY KEY,
    Forename varchar(50) NOT NULL,
    Surname varchar(50) NOT NULL
);

CREATE TABLE exams(
    Exam_ID integer PRIMARY KEY,
    Exam_Name varchar(50) NOT NULL,
    Max_Mark integer NOT NULL
);

CREATE TABLE results(
    Result_ID integer PRIMARY KEY,
    Student_ID integer NOT NULL,
    Exam_ID integer NOT NULL,
    Mark integer NOT NULL
);
```



#### 4.2 Write a query to list students' forenames and surnames where they scored higher than 60 and the respective exam.

##### Marking Points (6 marks)

*For any point not fully achieved, deduct and use half marks as appropriate.*

**[1]** - SELECTing the forename, surname and exam name

**[3]** - Using the resultant table that is JOINing the results table, students table and exams table

**[1]** - WHERE the mark is greater than 60

**[1]** - For the checks that the result student ID matches the students student ID, and the result exam ID matches the exams exam ID.

##### Example Query

```
SELECT s.Forename, s.Surname,
e.Exam_Name
FROM results as r
JOIN students as s
JOIN exams as e
WHERE r.Mark > 60
AND r.student_id =
s.student_id
AND r.exam_id = e.exam_id;
```

##### Expected Output

	Forename	Surname	Exam_Name
▶	Belen	Badillo	Algorithms
	Ciara	Connelly	Algorithms
	Everly	Evans	Algorithms
	Belen	Badillo	Cyber Security
	Everly	Evans	Cyber Security
	Fabia	Fahim	Cyber Security

**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.*

**Marking Points (9 marks)**

*For any point not fully achieved, deduct and use half marks as appropriate.*

**[1]** - SELECTing the forename, surname and exam name

**[3]** - Using the resultant table that is JOINing the results table, students table and exams table

**[1]** - WHERE clause with Subquery regarding 'marks'

**[3]** - Subquery to find duplicate marks:

1. Selecting mark from results
2. Grouping by mark
3. Where the count is greater than one

**[1]** - For the checks that the result student ID matches the students student ID, and the result exam ID matches the exams exam ID.

**Example Query**

```
SELECT s.Forename, s.Surname,
e.Exam_Name, r.mark
FROM results AS r
JOIN students AS s
JOIN exams AS e
WHERE r.mark IN (
    SELECT mark FROM results
    GROUP BY mark HAVING COUNT(*) >
1
)
AND r.student_id = s.student_id AND
r.exam_id = e.exam_id;
```

**Expected Output**

	Forename	Surname	Exam_Name	mark
▶	Belen	Badillo	Algorithms	62
	Ciara	Connelly	Algorithms	62
	Belen	Badillo	Cyber Security	68
	Everly	Evans	Cyber Security	68