



**FOUNDATION EXAM - 2 HOURS**

|  |  |
| --- | --- |
| **SECTION** | **MARK** |
| **Theory Questions** | 31 |
| **Concept Questions** | 19 |
| **Challenges: students should choose two out of the three to answer** | |
| **Python Challenge** | 25 |
| **SQL Challenge** | 25 |
| **Javascript Challenge** | 25 |
| 1. **TOTAL** | **100** |

**Important notes:**

* Any code files written **must be submitted via a Pull Request to your marker**.
* You can submit theory questions through an edited version of this document on Slack, or on the Pull Request by adding python comments into a new file, or using a text or markdown file.
* You are allowed to submit everything on Slack if it is close to the deadline, as long as you work on getting a pull request up soon after.
* It is an open book exam.
* You are allowed to use PyCharm, MySQL Workbench, and/or CodePen for this assessment.
* If you are using CodePen, please add your CodePen URL in a text or markdown file.
* You only need to complete **two coding challenges**. Please choose two coding challenges of the three provided (Python, SQL and Javascript) to answer.

**Section 1: Theory Questions [31 marks]**

|  |  |
| --- | --- |
| * 1. **What does SDLC stand for?**   Software Development Lifecycle, it is a cost-effective and time-efficient process that development teams use to make and develop high-quality software. The objective of Software Development Lifecycle is to remove project risks by planning so that the software meets the requirements necessary for to meet the customers expectations while in production. | **1 mark** |

|  |  |
| --- | --- |
| * 1. **What exception is thrown when you divide a number by 0?**   ZeroDivisionError this exception will always occur when a number tries to be divided by zero. | **1 mark** |

|  |  |
| --- | --- |
| **1.3 What is the git command that moves code from the local repository**  **to the remote repository?**  The git push command is used to transfer to the local repository content to a remote repository. Pushing is how you move commits from your local repository to a remote repository. It is the counterpart to git fetch, although where fetching imports commits to local branches, pushing exports commits to remote branches.  For example, git push would be following command:  git push remote\_name main. | **1 mark** |

|  |  |
| --- | --- |
| * 1. **What does NULL represent in a database?**   Null value in a database is used when an entity in a column is unknown or missing. A null is both an empty string and a zero value. Null mainly means that value does not exist in the database. | **1 mark** |

|  |  |
| --- | --- |
| * 1. **Name 2 responsibilities of the Scrum Master**   The Scrum Master is in charge for creating and onboarding project teams, integrating them into the organisation and providing a clear overview of the product. The Scrum Master also facilitates communication and information swap between external groups and the project team.  The first responsibility of Scrum Master is removing blockers for example, The Scrum Master facilitates removal of blockers and problems for the Scrum Team. They’re in charge for ensuring the team are as productive as possible.  Another responsibility is looking after sprints, Scrum Master will oversee the sprints process so that they dictate what the team should work on but provide them with information in case they need to adjust, discuss issues or help each other. | **2 marks** |

|  |  |
| --- | --- |
| * 1. **Name 2 debugging methods, and when you would use them.**   One way to debug your code is by printing the output to your console or terminal and you will see a message of logged variable or return from a function. Print to console to do this you would write the following code:  num1 = 5  num2 = 10  Print(f‘The value is: {num1} + {num2}’)  The console:  The value is: 15.  Another method of debugging is using unit tests to test the test cases to ensure there is no unexpected bugs appearing in your code. Unit tests are very handy to prevent unknown crashes.  To do unit tests you will need to make another file called main\_test.py then import unittest. Then you want to create a class called MainTestCase and pass in the parameter the following parameter: unittest.TestCase. Once you have defined your class you can make many functions to test the inputs. For example, you would do following code:  def test\_valid\_15(self)”:  self.assertEqual(15, sum(5, 10)). | **4 marks** |

|  |  |  |
| --- | --- | --- |
| **1.7 Looking at the following code, describe a case where this function**  **would throw an error when called.** Describe this case and talk about  what exception handling you’ll need.   |  | | --- | | **def can\_pay(price, cash\_given):**  **if cash\_given >= price:**  **return True**  **else:**  **return False**  **If any of parameters such as price or cash\_given is given by string and not float you will get an error called TypeError. The solution I would give is to cast both cash\_given and price as float.  Another error that might occur ValueError, for this solution place a try and catch in the body of can\_pay function put if statement in the try and write except and return the valueError message in a print statement or have a custom message ready for error handling.  This is what the code should look like:**  def can\_pay(price, cash\_given):  try:  if float(cash\_given) >= float(price):  return True  else:  return False  except ValueError as ve:  return 'Cannot execute code due to one of values not being a float.'  print(can\_pay(5.99, '%')) | | **5 marks** |

|  |  |
| --- | --- |
| **1.8 What is git branching?** Explain how it is used in Git.  In Git, branches are a fraction of your day-to-day development process. Git branches are beneficially a pointer to a snapshot of your modifications. When you want to implement a new feature or fix a bug. It does not matter how big the bug fixes or small you will spawn a new branch to encapsulate your alterations. Branches allow you to work on many parts of a project without overriding someone else’s work. You can work on other branches without impacting the main project.  To add a new branche in GitHub you must do the following command in your terminal or IDE:  git checkout -b newBranch. | **6 marks** |

|  |  |
| --- | --- |
| **1.9 Design a restaurant ordering system.**  You do not need to write code, but describe a high-level approach:   * 1. Draw a list of key requirements   2. What are your main considerations and problems?   3. What components or tools would you potentially use? * Every user must have login details before they can add meals to the basket. * User can pick vegan, vegetation options on their chosen meal. * User can pick quantity of how many meals they want so if they want just 1 meal or two. * Meals will be displayed as a list on mobile and in a grid format on desktop or laptop resolutions. * Customers have the option to send email if they require support. * Customers can apply text to speech or change the colours of mobile/website version to help make it more accessible.   1. What are your main considerations and problems? * There might be issues with overloading such as too many users trying to sign up, login or place too many orders. * The relational database might go down so for that we should save a snapshot and store within cache in web browser or mobile app. * User could lose internet connection as the as the same as relational database we would store that in cache. * If items are not loading place an error message to say sorry were working on getting this item back in stock.   1. What components or tools would you potentially use? * Frontend (ReactJS, HTML and CSS) to design the restaurant ordering. * Login page. * Meals page. * Support page. * Basket page for mobile version only. * Backend – Python based API, Database MySQL for storing customers and meals tables, NoSQL for storing order history. * API to map images and text on the frontend from the database. * Redis - Backup data to local cache every few mins.   Relational Database with three tables, one being customer and other a meals, the third table will be orders. The third table will require food\_id and customer\_id to link the customer and meals table together in orders. | **10 marks** |

**Section 2: Concept Questions [19 marks]**

|  |  |
| --- | --- |
| **2.1 Write a function that takes in an input and checks to see if it’s an**  **isogram. The function should return True or False.**    An isogram is a word where no letter is repeated.  Examples include:   * "isogram" * "uncopyrightable" * “ambidextrously”   def check\_isogram(word):  char\_values = []  for char in word:  if char.isalpha():  if char in char\_values:  return False  char\_values.append(char)  return True    print(check\_isogram("isogrammm")) | **7 marks** |

|  |  |
| --- | --- |
| **2.2 Make a new test file and write comprehensive unit tests for the**  **function you wrote in 2.1**  For each test case add a comment stating why you chose that case.  Main  import unittest  from main import check\_isogram  class MainTestCase(unittest.TestCase):  # Chosen this testcase to see if it would pass as expected.  def test\_isogram\_valid(self):  self.assertEqual(True, check\_isogram("isogram"))  # Chosen this testcase to see if it would pass as expected.  def test\_uncopyrightable\_valid(self):  self.assertEqual(True, check\_isogram("uncopyrightable"))  # Chosen this testcase to see if it would pass as expected.  def test\_ambidextrously\_valid(self):  self.assertEqual(True, check\_isogram("ambidextrously"))  # Shown this testcase to see if this would crash the program.  def test\_555\_invalid(self):  self.assertEqual(False, check\_isogram(555))    # Shown this testcase to see if this would crash the program.  def test\_5f\_invalid(self):  self.assertEqual(False, check\_isogram(5.0))  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() | **12 marks** |

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

You are tasked with calculating the minimum classes we need to have so we know how many people to employ. Write a function which when given a number of students, calculates and prints out a string for your proposed number of classes, and a dictionary showing the allocation.

***Key Constraints:I’ll***

* The maximum size of a class is 30
* There needs to be a minimum of 2 classes
* The distribution of each class should be as even as possible.
* We want to hire as little people as possible - so where possible focus on bigger classes, and less of them!

***Inputs/Outputs***:

* If 31 was the input, the output would be:

|  |
| --- |
| Proposed Allocation: 2 classes  {'Class 1': 16, 'Class 2': 15} |

* If 59 was the input, the output would be:

|  |
| --- |
| Proposed Allocation: 2 classes  {'Class 1': 30, 'Class 2': 29} |

* If 87 was the input, the output would be:

|  |
| --- |
| Proposed Allocation: 3 classes  {'Class 1': 29, 'Class 2': 29, 'Class 3': 29} |

def minimum\_class(max\_class):

halved = math.ceil(max\_class / 2)

second\_halved = max\_class - halved

thrid\_havled = max\_class - second\_halved

if halved > 30 and second\_halved > 30 and thrid\_havled > 30:

print(f"Class 1: {math.ceil(halved)} Class 2:' {math.ceil(halved - 30 + second\_halved - 30)} Class 3:' {math.ceil(second\_halved - 30 + thrid\_havled)}")

else:

print(f"Class 1: {math.ceil(halved)} Class 2:' {math.ceil(second\_halved)}")

print(minimum\_class(87))

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

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

**Starter Code:**

CREATE DATABASE foundation\_exam;

USE foundation\_exam;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4.1 Write (and execute) syntax to create the following tables:**  Example data is included to help you choose suitable data types  **A] *movie\_info*** *Table*     |  |  |  |  | | --- | --- | --- | --- | | Movie\_ID | Movie\_Name | Movie\_Length | Age\_Rating | | 1 | The Movie | 1:35:00 | 12A |   **B] *screens*** *Table*     |  |  | | --- | --- | | Screen\_ID | Four\_K | | 1 | False |   **C] *showings*** *Table*     |  |  |  |  |  | | --- | --- | --- | --- | --- | | Showing\_ID | Movie\_ID | Screen\_ID | Start\_Time | Available\_Seats | | 1 | 1 | 1 | 12:00:00 | 23 | | **10 marks** |

|  |
| --- |
| **Populate the database!**  Use the file*foundation\_exam.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 return the name and time of all movies that play after**  **12:00 given there is at least 1 available seat. Display the results in time**  **order.** | **6 marks** |

|  |  |
| --- | --- |
| **4.3 Return the name of the movie with the most showings.** | **9 marks** |

**Section 5: JavaScript Challenge [25 marks]**

Create a simple To-Do List web application using HTML, CSS, and JavaScript. The application should have the following features:

1. A text input field for entering tasks.
2. A "Add Task" button to add tasks to the list.
3. A list to display added tasks.
4. Each task should have a checkbox to mark it as complete.
5. A "Delete" button to remove completed tasks.
6. Style the application with CSS to make it visually appealing
   1. Make all the items centered on the page
   2. Display the list below the text input

Provide a clear and organized code for this web application.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="styles.css">

<title>Document</title>

</head>

<body class="main">

<div class="navbar"></div>

<div class="todoList">

<ul class="container" id="list">

<li class="item">Text Here</li>

</ul>

</div>

<input class="search" id="search" type="search"/>

<button class="button" onclick="getSearch()">Add Task</button>

<ul>

</ul>

</body>

<script>

let search = document.getElementById('search');

let ul = document.getElementById('list');

let li = document.createElement('li');

arrays = list = []

function getSearch() {

li.appendChild(document.createTextNode(search.value));

list.appendChild(li);

}

</script>

</html>

.navbar{

background-color: rgb(252, 171, 64);

height: 100px;

width: 100%;

}

.main{

display: flex;

flex-direction: column;

align-items: center;

background-color: rgb(255, 212, 156);

margin: 0;

padding: 0;

gap: 20px;

}

.search{

width: 500px;

height: 30px;

border-radius: 15px;

}

.button{

width: 400px;

height: 40px;

background-color: rgb(255, 156, 27);

box-shadow: 15px, 15px, 15px, 5px rgba(0, 0, 0, 0.453);

border-radius: 15px;

font-weight: 600;

}

.item{

background-color: rgb(202, 202, 202);

border-color: black;

border-width: 5px;

width: 500px;

padding: 15px;

list-style-type: none;

}