



**FOUNDATION ASSESSMENT II MATERIAL RELEASE**

THEORY QUESTIONS

|  |  |
| --- | --- |
| **SECTION** | **MARK** |
| **Theory Questions** | 31 |
| **Concept Questions** | 19 |
| **Python Challenge** | 25 |
| **SQL Challenge** | 25 |
| **TOTAL** | **100** |

**Important notes:**

* This document shares the first section of the Foundation Assessment II which is composed of 9 Theory Questions
* It is worth just under a third of your assessment mark
* You have 24 hours before the assessment to prepare.
* If any plagiarism is found in how you choose to answer a question you will receive a 0 and the instance will be recorded. Consequences will occur if this is a repeated offence. You can remind yourself of the plagiarism policy [here](https://drive.google.com/file/d/1k9UaGOR7hx54QRZ8jvp2jtC4P-8_Rs4F/view?usp=sharing).

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

|  |  |
| --- | --- |
| **1.1 What does SDLC stand for?** | **1 mark** |

- Software Development Lifecycle

|  |  |
| --- | --- |
| **1.2 What exception is thrown when you divide a number by 0?** | **1 mark** |

- ZeroDivisionError

|  |  |
| --- | --- |
| **1.3 What is the git command that moves code from the local repository**  **to the remote repository?** | **1 mark** |

- Git push

|  |  |
| --- | --- |
| **1.4 What does NULL represent in a database?** | **1 mark** |

- NULL means that nothing was entered. The data doesn’t exist.

|  |  |
| --- | --- |
| **1.5 Name 2 responsibilities of the Scrum Master** | **2 marks** |

* Lead scrum meetings, e.g. sprint planning meetings or daily standups
* Helps aid in the communication between the development team, product owners, and stakeholders

|  |  |
| --- | --- |
| **1.6 Name 2 debugging methods, and when you would use them.** | **4 marks** |

- in Pycharm you can use the built in debugging functionality or you can import ‘pdb’ to run programs in debugging mode.

- You can also use breakpoints in your code to run through the code line-by-line to identify the source of the error.

-Debugging is used when you don’t know the location and reason for your code crashing.

|  |  |  |
| --- | --- | --- |
| **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** | | **5 marks** |

- An error would occur if the type of price or cash\_given wasn’t a float or integer. For example if price was ‘3.50’ it would be a string and therefore a TypeError would be raised.

- What I would do is use a try statement that will run the code first. If numeric values have been used the code will run fine, however if not, a TypeError will occur and a message will be printed.

Text

Description automatically generated

If the below function is executed:



Then this will be printed:

Text

Description automatically generated

|  |  |
| --- | --- |
| **1.8 What is git branching?** Explain how it is used in Git. | **6 marks** |

- git branching creates a branch (or ‘folder’). If you make a new ‘git commit’ the branch doesn’t move with the master (see first screenshot). You can then use ‘git checkout’ to switch branches (see second screenshot). This means the user can safely experiment and make changes without disrupting/interfering with other peoples codes or causing an error within the master branch.

- Once code has been written and is working in a separate branch, you can then merge it into the master branch.

Text

Description automatically generated with medium confidence

Timeline

Description automatically generated

|  |  |
| --- | --- |
| **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? | **10 marks** |

Key requirements

* Customer can view menu
* Customer can select drinks and food
* Customer can place order
* Customer can choose delivery or takeaway
* Customer can state any allergies
* Customer can remove items from order
* Customer can see total price

Main considerations and problems

* If there is a long wait time, alert customer before they can place an order.
* Each item has a list of all the ingredients for allergies.
* If the restaurant is out of stock of an item, the restaurant is able to update the app so customers can’t order the item.
* App is secure owing to customers disclosing private information e.g. address, bank account details.
* If payment fails, ask to try a different payment method
* If restaurant is closed, update the app to not take any orders.

Components and tools

* Use SQL to store the menu items
* Use python to update the SQL database if the restaurant runs out of items.
* An API for the user to communicate through
* Diagram

  Description automatically generated