



**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?**  Software development life cycle | **1 mark** |
| --- | --- |

| **1.2 What exception is thrown when you divide a number by 0?**  ‘ZeroDivisionError: division by zero’ | **1 mark** |
| --- | --- |

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

| **1.4 What does NULL represent in a database?**  An unknown or missing value | **1 mark** |
| --- | --- |

| **1.5 Name 2 responsibilities of the Scrum Master**   1. Facilitate team meetings such as the daily scrum or sprint sessions. 2. Remove any distractions or impediments so the team can remain focused. | **2 marks** |
| --- | --- |

| **1.6 Name 2 debugging methods, and when you would use them.**   1. Exceptions handling – this allows the user to identify, detect and handle errors but keep the program running. For example, if an integer was expected in a function, but a string was entered this would be a TypeError.     This helps identify what the problem is to the developer and can be raised as an exception to catch it for the user, giving them a chance to retry.     1. Asset – is a statement in python that can check if certain conditions are true in the code. For example, this can be used in testing to check if the expected output is correct. Below is an example of this that test the ‘is\_pailindrome’ function. We know that the word racecar is a palindrome so we are expecting it to be true. This tests that the function is running correctly. Multiple cases can be included. | **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** | | --- |   The function is expecting an integer for the arguments price and cash given. If this is not an integer a type error will be raised:    To ensure the program does not end unexpectedly, we can include exception handling. We want to ensure that the user is arguments are integers:    We would also want to add an exception to catch any negative integers: | **5 marks** |
| --- | --- | --- |

| **1.8 What is git branching?** Explain how it is used in Git.  Braches can be created in git which copy the main branch and allow a user to start working on the code without effecting the original. Multiple branches can be created and eventually merged back to the main branch. The code can be compared across branches, pulled by others and commented on before merging.  One of the main advantages is that it allows users to work on different parts of the same code without preventing others from also working on it. It can also be good for trying things out in case they go wrong- you can always revert back to the main branch.  To create a new git branch from PyCharm, you can use the git option in the top menu or enter the following:  -Go to the relevant repository and use ‘git branch <branch\_name>’  -Switch to using this new branch rather than the main branch ‘git checkout <branch\_name>  -Once any alterations have been made this can be added and committed to the new branch ‘git add <file\_name>’, ‘git commit –m “message text”. A message must be included.  -Then push to the branch ‘git push –u origin <branch\_name>’  -Once everyone is happy it can then be merged back to the main branch by first checking it out instead of the new branch then merging the two. ‘git merge <branch-name> | **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?   Assumptions: Orders are placed via a device directly by the customer and sent to the kitchen.  Key requirements:   * User interface: including menu, prices, availability, choice selection, ordering and taking payment. * Stock database: availability of all items, descriptions and pricing. * Payment processing: a system that can process the payments via card or cash. * Orders to kitchen: a system that can process the orders by order and send what is required to the kitchen. This can also say when the order is ready to be collected or what their order number is for it to be delivered to their table.   Main considerations and problems:   * Might be times with large volumes of orders at the same time. Needs to be able to handle that. * Exception handling for incorrect inputs or out of stock items. * User interface and information to the kitchen needs to be clear and easy to read and understand. Pictures might be useful. * Changes in pricing or the menu need to be easy to implement. * Security of the system (mostly people contact info and payment details) * Somewhere to contact if anything does go wrong with the system. * Administration access, access for managers and servers if needed. * Data collection for improvements: how long are orders taking?, what happens if something goes wrong?.   Components or tools:   * User and Kitchen interfaces: in house screens or a mobile app. Made with java/ javascript/ css/ html * Database: to store information about the menu, pricing, orders, ingredients and availability. Use MySQL * Payments: a payment gateway * Point of sale system: processing payments. * Marketing information: send emails to customers. * Mobile app: for ordering, maybe a rewards scheme. Made with java/ javascript/ css/ html. | **10 marks** |
| --- | --- |