**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” | **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. | **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:***

* 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:

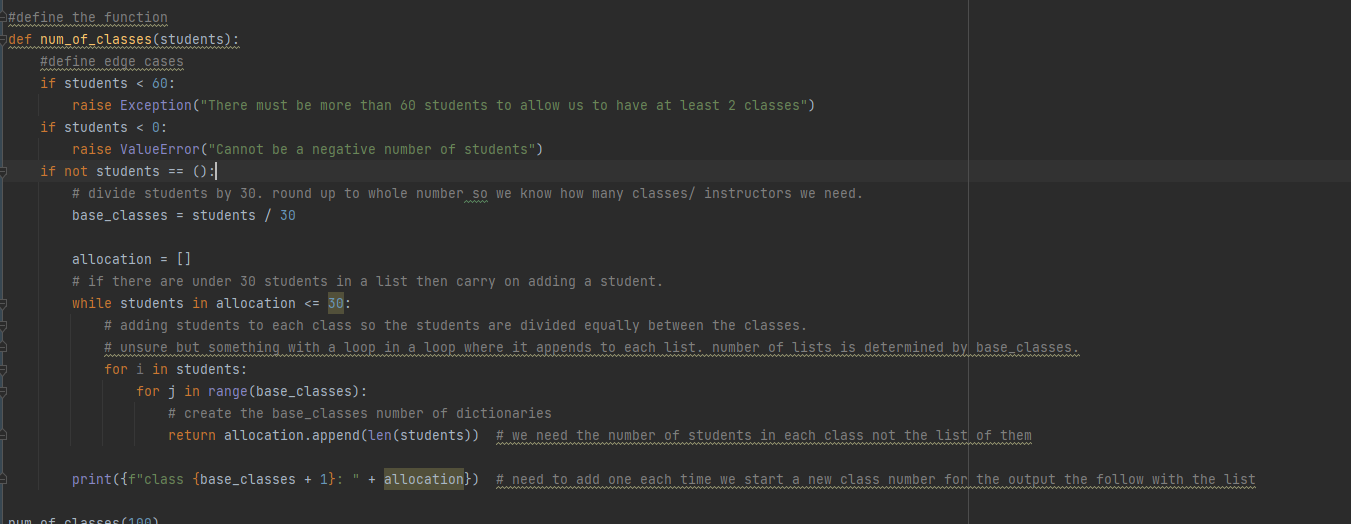
| Proposed Allocation: 2 classes  {'Class 1': 16, 'Class 2': 15} |
| --- |

* 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} |
| --- |



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

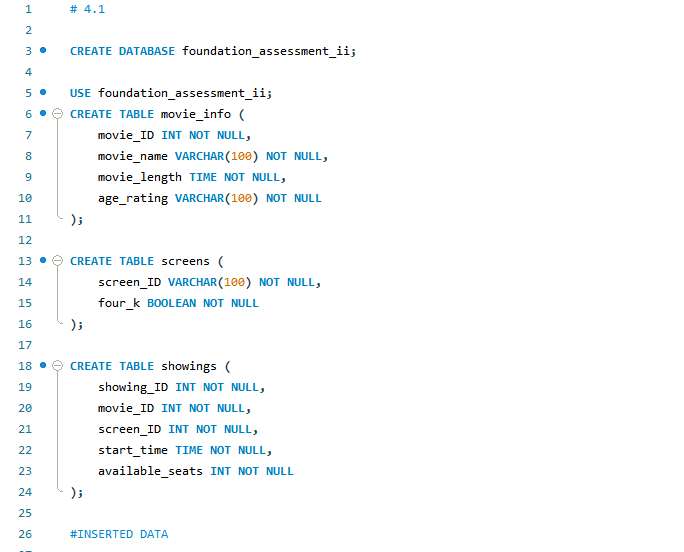
In this section you will be fleshing out a database and performing queries.

**Starter Code:**

CREATE DATABASE foundation\_assessment\_ii;

USE foundation\_assessment\_ii;

| **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\_assessment\_ii.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** |
| --- | --- |