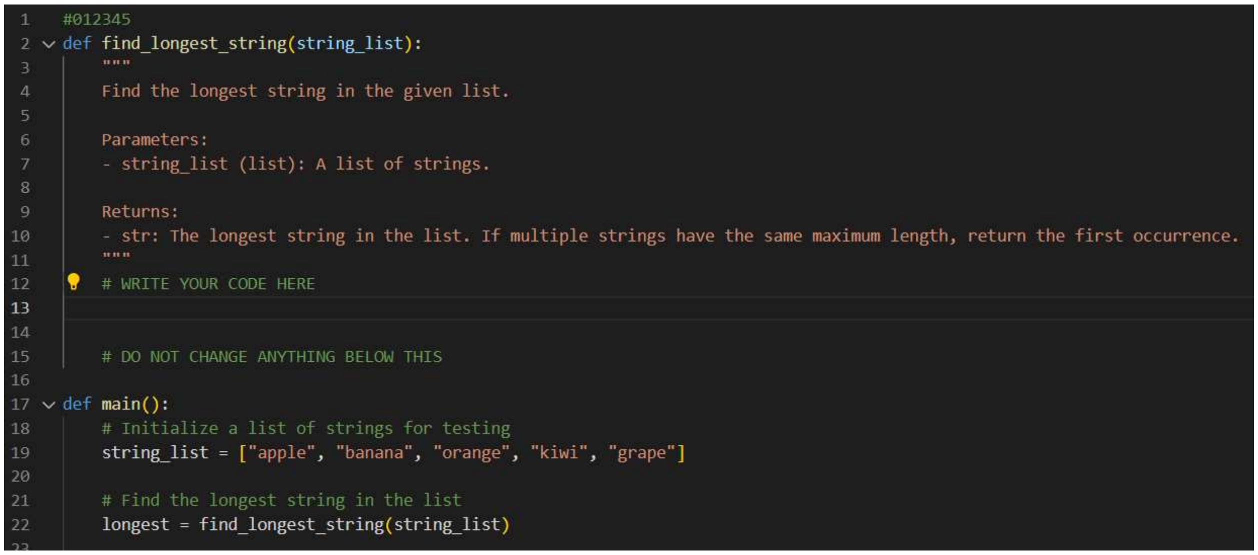
Final Exam

Deadline: As in Moodle

Problem 1

You are tasked with implementing a Python function that takes a list of strings as input and returns the longest string in the list. If multiple strings have the same maximum length, return the first occurrence.

You are provided with a skeleton file, you are not allowed to modify any other part of code except find_longest_string() function from line 12 in **main1.py**.



```
1   #012345
2   def find_longest_string(string_list):
3       """
4       Find the longest string in the given list.
5
6       Parameters:
7       - string_list (list): A list of strings.
8
9       Returns:
10      - str: The longest string in the list. If multiple strings have the same maximum length, return the first occurrence.
11      """
12      # WRITE YOUR CODE HERE
13
14
15      # DO NOT CHANGE ANYTHING BELOW THIS
16
17  def main():
18      # Initialize a list of strings for testing
19      string_list = ["apple", "banana", "orange", "kiwi", "grape"]
20
21      # Find the longest string in the list
22      longest = find_longest_string(string_list)
```

Requirements

1. **Input**
   - ◆ The skeleton file contains an initialized list of string.
2. **Output**
   - ◆ The function should return the longest string in the list as a string.
   - ◆ If multiple strings have the same maximum length, return the first occurrence.

5. **Save the File Offline**
   - ◆ Click the “Download Code” button/icon to save the file offline. Rename the file to main4.py.

6. **Submission**
   - ◆ Include your Student ID as comment at the top of your code i.e #012345. Submission instructions are at the end of this document.

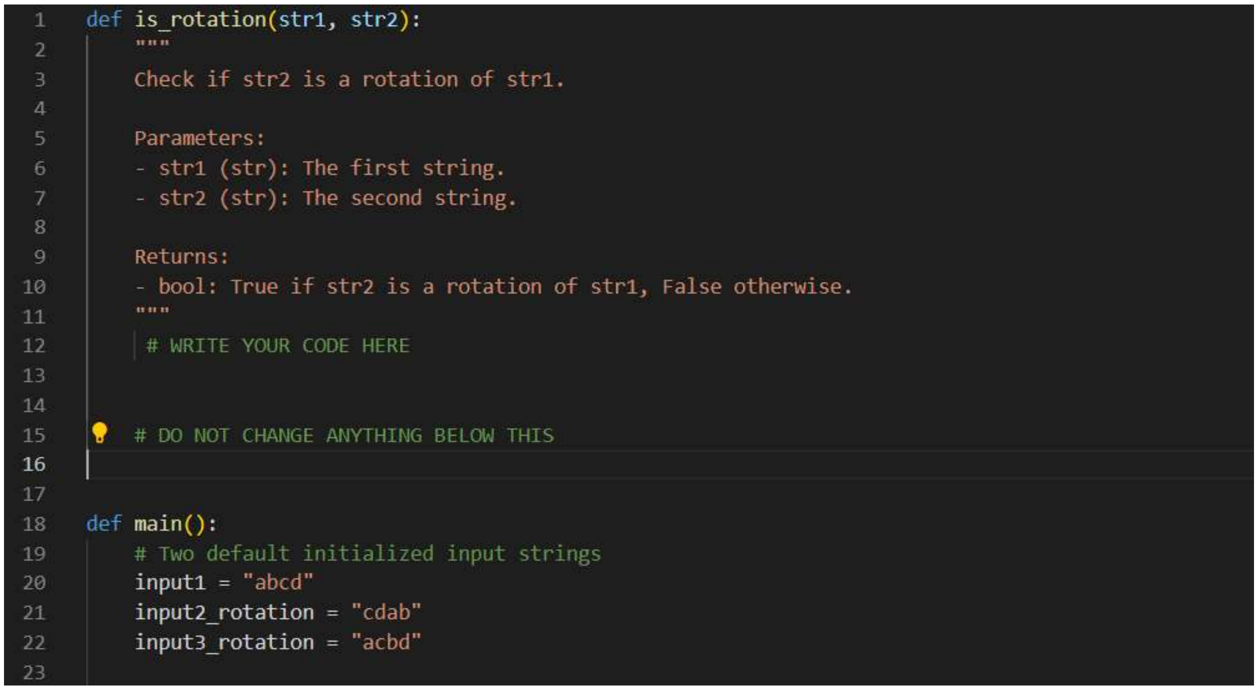
7. **Grading**

- (1 point) Correct submission of working code with no errors and loop implementation.
- (5 point) Correct solution to the problem.

Problem 2

You are tasked with implementing a program that takes two strings as input and checks if the second string is a rotation of the first string without using any built-in string functions.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **is_rotation()** function from line 12 in **main2.py**.



```
1   def is_rotation(str1, str2):
2       """
3       Check if str2 is a rotation of str1.
4
5       Parameters:
6       - str1 (str): The first string.
7       - str2 (str): The second string.
8
9       Returns:
10      - bool: True if str2 is a rotation of str1, False otherwise.
11      """
12       # WRITE YOUR CODE HERE
13
14
15      # DO NOT CHANGE ANYTHING BELOW THIS
16
17
18  def main():
19      # Two default initialized input strings
20      input1 = "abcd"
21      input2_rotation = "cdab"
22      input3_rotation = "acbd"
23
```

Requirements

1. **Input**
   - The skeleton file contains an initialized input string and 2 string to check if it is rotation or not.

2. **Rotation Check**
   - Implement the function is_rotation to check if str2 is a rotation of str1 without using any built-in string reversal functions.
   - A string is considered a rotation of another string if it can be obtained by rotating the characters of the first string cyclically.

   - Example, For the string "wxyz", it has a total of four rotations. These rotations are obtained by shifting the characters of the string cyclically to the right. The rotations are: "wxyz", "xyzw", "yzwx", and "zwxy".

3. **Output**
   - The function should return a boolean value:
     - True if str2 is a rotation of str1
     - False otherwise.
4. **No Built-in Reversal**
   - Do not use any built-in string reversal functions such as **[::-1] or reverse()**.

5. **Save the File Offline**
   - Click the “Download Code” button/icon to save the file offline. Rename the file to main4.py.

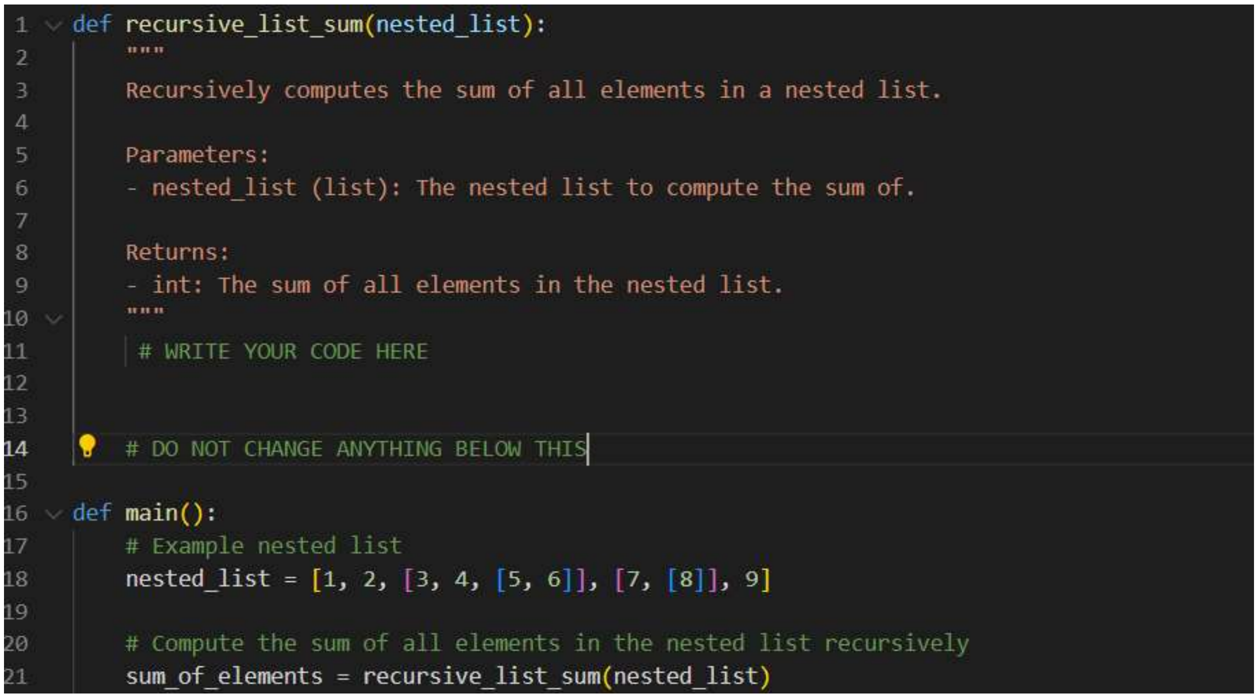
6. **Submission**
   - Include your Student ID as comment at the top of your code i.e #012345. Submission instructions are at the end of this document.

7. **Grading**
   - (1 point) Correct submission of working code with no errors and loop implementation.
   - (5 point) Correct solution to the problem.

Problem 3

You are tasked with implementing a program that recursively computes the sum of all elements in a nested list. The nested list can contain integers and other nested lists.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **recursive_list_sum()** function from line 11 in **main3.py**.



```
1  def recursive_list_sum(nested_list):
2      """
3      Recursively computes the sum of all elements in a nested list.
4
5      Parameters:
6      - nested_list (list): The nested list to compute the sum of.
7
8      Returns:
9      - int: The sum of all elements in the nested list.
10     """
11     # WRITE YOUR CODE HERE
12
13
14     # DO NOT CHANGE ANYTHING BELOW THIS
15
16 def main():
17     # Example nested list
18     nested_list = [1, 2, [3, 4, [5, 6]], [7, [8]], 9]
19
20     # Compute the sum of all elements in the nested list recursively
21     sum_of_elements = recursive_list_sum(nested_list)
```

Requirements

1. **Input**
   - ◆ The skeleton file contains an initialized list which consist of numbers and other list consisting of numbers.

2. **Recursive Sum Calculation**
   - ◆ Implement a function named recursive_list_sum to recursively compute the sum of all elements in the nested list.
   - ◆ If an element in the nested list is an integer, add it to the sum.
   - ◆ If an element in the nested list is another nested list, recursively compute its sum.

3. **Output**
   - ◆ The function should return the sum of all elements in the nested list.. Do not include any print statements in the code.

4. **Save the File Offline**
   - ◆ Click the “Download Code” button/icon to save the file offline. Rename the file to main3.py.

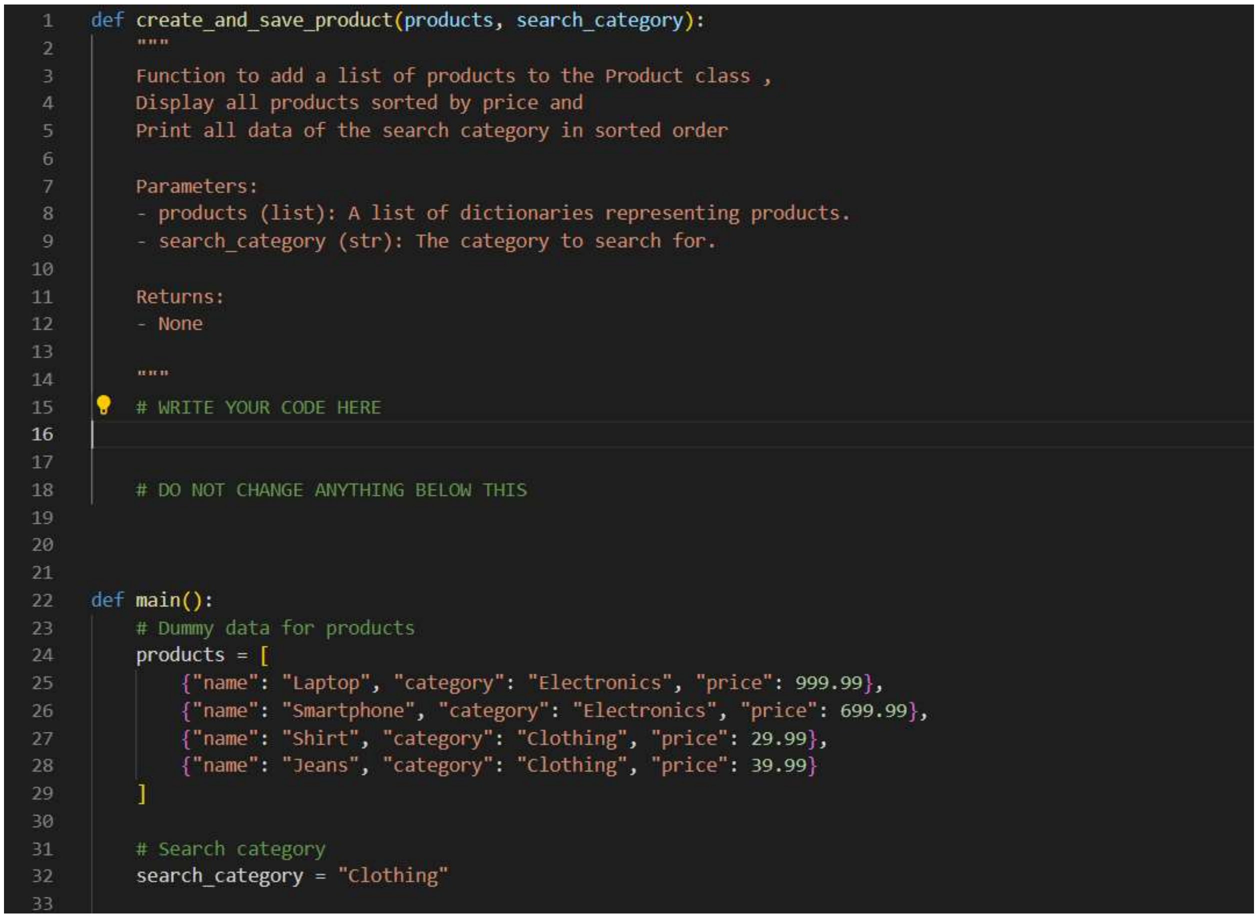
5. **Submission**
   - ◆ Include your Student ID as comment at the top of your code i.e #012345. Submission instructions are at the end of this document.

6. **Grading**
   - ◆ (1 point) Correct submission of working code with no errors and loop implementation.
   - ◆ (5 point) Correct solution to the problem.

Problem 4

You are tasked with implementing a function that accepts a list of products as its argument and a search category value. The function should add all elements in the list to a class named Product, display all elements sorted according to price, and print all data of the search category in sorted order.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **create_and_save_product ()** function from line 17 in **main4.py**.



```
1   def create_and_save_product(products, search_category):
2       """
3       Function to add a list of products to the Product class ,
4       Display all products sorted by price and
5       Print all data of the search category in sorted order
6
7       Parameters:
8       - products (list): A list of dictionaries representing products.
9       - search_category (str): The category to search for.
10
11      Returns:
12      - None
13
14      """
15      # WRITE YOUR CODE HERE
16
17
18      # DO NOT CHANGE ANYTHING BELOW THIS
19
20
21
22  def main():
23      # Dummy data for products
24      products = [
25          {"name": "Laptop", "category": "Electronics", "price": 999.99},
26          {"name": "Smartphone", "category": "Electronics", "price": 699.99},
27          {"name": "Shirt", "category": "Clothing", "price": 29.99},
28          {"name": "Jeans", "category": "Clothing", "price": 39.99}
29      ]
30
31      # Search category
32      search_category = "Clothing"
33
```

Requirements

1. **Input**
   - ◆ The skeleton file contains an initialized list which consist of products and a string named search_category.Each product is represted inside the list as a dictionary with attributes name,category and price.

2. **Product Class**

   Inside the function, it should implement the following:

   ◆ Define a class named Product with the following attributes and methods:
   - Attributes:
     1. name (str): The name of the product.
     2. category (str): The category of the product.
     3. price (float): The price of the product
   - Methods:
     1. display: Display the details of the product (name, category, price).

   ◆ Adding Products
   - Add each product from the products list to the Product class as objects.

   ◆ Displaying Sorted Elements
   - Display all products sorted by price in ascending order.

   ◆ Printing Data of Search Category
   - Print all data of the search category in sorted order, including the name, category, and price of each product.

3. **Output**

   ◆ The function should create the class product,add all products to it , then display all products sorted by price and print all data of the search category in sorted order.

4. **Save the File Offline**

   ◆ Click the "Download Code" button/icon to save the file offline. Rename the file to main4.py.

5. **Submission**

   ◆ Include your Student ID as comment at the top of your code i.e #012345. Submission instructions are at the end of this document.

6. **Grading**

   ◆ (1 point) Correct submission of working code with no errors and loop implementation.

   ◆ (5 point) Correct solution to the problem.

Submission Instructions

Place the main1.py-main4.py in a folder and compress to a zip file. Submit it to moodle.