1.1 Python

Question 1

Explain the difference between the list, tuple , set and dictionary ?

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# List
my_list = [1, 2, 3, 4, 5]

print(my_list)
# Lists maintain order
for item in my_list:
    print(item)

# Tuple
my_tuple = (1, 2, 3, 4, 5)

# Tuples are immutable - attempting to modify will raise an error

for item in my_tuple:
    print(item)

# Set
my_set = {1, 2, 3, 4, 5}

# Sets are mutable
my_set.add(6)
print(my_set)  # Output: {1, 2, 3, 4, 5, 6}

# Sets do not maintain order
for item in my_set:
   print(item)

# Dictionary
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Dictionaries are mutable
my_dict['age'] = 35
print(my_dict)  # Output: {'name': 'John', 'age': 35, 'city': 'New York'}

# Dictionaries do not maintain order (prior to Python 3.7)
for key, value in my_dict.items():
    print(key, ':', value)
```

```
Output:

Rishabh Rathore
0827CI211155


[10, 2, 3, 4, 5]
```

```
10
2
3
4
5

1
2
3
4
5

{1,2,3,4,5,6}

1
2
3
4
5
6

{'name': 'John', 'age': 35, 'city': 'New York'}

name : John
age : 35
city : New York
```

Question 2

Write a program to Counts the number of occurrences of item 50 from a tuple.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def count_occurrences(tuple, item):
count = 0
for element in tuple:
    if element == item:
        count += 1
return count

# Example tuple
example_tuple = (10, 20, 30, 40, 50, 50, 50, 60, 70)

# Count occurrences of item 50
occurrences = count_occurrences(example_tuple, 50)
print("Number of occurrences of item 50:", occurrences)
```

```
Output:

Rishabh Rathore
0827CI211155
Number of occurrences of item 50: 3
```

Question 3

Write a program to Check if all items in the tuple are the same

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def all_same(tuple):
# Check if all elements are the same
return all(element == tuple[0] for element in tuple)

# Example tuples
tuple1 = (10, 10, 10, 10)
tuple2 = (10, 20, 10, 10)

# Check if all items in tuple1 are the same
if all_same(tuple1):
    print("All items in tuple1 are the same")
else:
    print("Not all items in tuple1 are the same")

# Check if all items in tuple2 are the same
if all_same(tuple2):
    print("All items in tuple2 are the same")
else:
    print("Not all items in tuple2 are the same")
```

```
Output:

Rishabh Rathore
0827CI211155
All items in tuple1 are the same
Not all items in tuple2 are the same
```

Question 4

Write a Program to create a tuple which is having integer,float,list and tuple as its elements and print an element present in list of these tuple

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define the tuple with different types of elements
mixed_tuple = (10, 3.14, [1, 2, 3], ('a', 'b', 'c'))

# Function to print an element from a list within the tuple
def print_list_element(tuple):
    # Get the list from the tuple
    list_element = tuple[2]
    # Print an element from the list
    print("Element from the list within the tuple:", list_element[1])

# Call the function to print an element from the list within the tuple
print_list_element(mixed_tuple)
```

```
Output:

Rishabh Rathore
0827CI211155
Element from the list within the tuple: 2
```

Question 5

Write a program to create a tuple with the name of 10 cities of India Check whether a City is present in the tuple or not

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def city_present(city, cities_tuple):
# Check if the city is present in the tuple
return city in cities_tuple

# Create a tuple with the names of 10 cities of India
indian_cities = ('Delhi', 'Mumbai', 'Bangalore', 'Kolkata', 'Chennai', 'Hyderabad',
                 'Pune', 'Ahmedabad', 'Jaipur', 'Lucknow')

# Input city to check
city_to_check = input("Enter the name of the city to check: ")

# Check if the city is present in the tuple
if city_present(city_to_check, indian_cities):
    print(city_to_check, "is present in the tuple of Indian cities.")
else:
    print(city_to_check, "is not present in the tuple of Indian cities.")
```

```
Output:

Rishabh Rathore
0827CI211155
Enter the name of the city to check: Kolkata
Kolkata is present in the tuple of Indian cities.
```

Question 6

Write a program to get the number of occurrence of a word in tuple.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def count_word_occurrences(word, tuple_of_strings):
count = 0
for string in tuple_of_strings:
    # Split the string into words and count occurrences of the target word
    count += string.count(word)
return count

# Example tuple of strings
tuple_of_strings = ("apple banana", "banana orange",
                    "banana apple", "orange mango banana")

# Word to count occurrences of
word_to_count = input("Enter the word to count occurrences: ")

# Count occurrences of the word in the tuple
occurrences = count_word_occurrences(word_to_count, tuple_of_strings)
print("Number of occurrences of",
     word_to_count, "in the tuple:",
      occurrences)
```

```
Output:

Rishabh Rathore
0827CI211155
Enter the word to count occurrences: apple
Number of occurrences of apple in the tuple: 2
```

Question 7

Write a program to get the index of a word in tuple

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def get_word_index(word, tuple_of_strings):
indexes = []
for i, string in enumerate(tuple_of_strings):
    # Split the string into words and check if the word is present
    if word in string:
        indexes.append(i)
return indexes

# Example tuple of strings
tuple_of_strings = ("apple banana", "banana orange",
            "banana apple", "orange mango banana")

# Word to get the index of
word_to_find = input("Enter the word to find its index: ")

# Get the index of the word in the tuple
word_indexes = get_word_index(word_to_find, tuple_of_strings)

if word_indexes:
    print("Indexes of", word_to_find, "in the tuple:", word_indexes)
else:
    print(word_to_find, "not found in the tuple.")
```

```
Output:

Rishabh Rathore
0827CI211155
Enter the word to find its index: the
the not found in the tuple.
```

Question 8

Write a program to create four tuples viz roll no. name. CGPA and SGPA of students. Print individual student's details using these 4 tuples.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")


# Function to print individual student's details
def print_student_details(roll_no, name, cgpa, sgpa):
    print("Roll No:", roll_no)
    print("Name:", name)
    print("CGPA:", cgpa)
    print("SGPA:", sgpa)
    print()

# Create tuples for student details
roll_numbers = (101, 102, 103, 104)
names = ("John Doe", "Jane Smith", "Alice Johnson", "Bob Brown")
cgpa = (3.8, 3.9, 3.7, 4.0)
sgpa = (4.0, 3.9, 3.8, 4.0)

# Print individual student's details
for i in range(len(roll_numbers)):
    print("Student", i+1, "details:")
    print_student_details(roll_numbers[i], names[i], cgpa[i], sgpa[i])
print("Ritesh Telkar")
print("0827CI211158")
```

```
Output:

Rishabh Rathore
0827CI211155
Student 1 details:
Roll No: 101
Name: John Doe
CGPA: 3.8
SGPA: 4.0

Student 2 details:
Roll No: 102
Name: Jane Smith
CGPA: 3.9
SGPA: 3.9

Student 3 details:
Roll No: 103
Name: Alice Johnson
CGPA: 3.7
SGPA: 3.8

Student 4 details:
Roll No: 104
Name: Bob Brown
```

CGPA: 4.0
SGPA: 4.0

Question 9

write a Python program to create a set.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Create a set
my_set = {1, 2, 3, 4, 5}

# Print the set
print("Set:", my_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Set: {1, 2, 3, 4, 5}
```

Question 10

Write a Python program to iterate over sets.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define a set
my_set = {1, 2, 3, 4, 5}

# Iterate over the set and print each element
print("Elements of the set:")
for element in my_set:
    print(element)
```

```
Output:

Rishabh Rathore
0827CI211155
Elements of the set:
1
2
3
4
5
```

Question 11

Write a Python program to add member(s) to a set.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define a set
my_set = {1, 2, 3, 4, 5}

# Print the initial set
print("Initial set:", my_set)

# Add a single member to the set
my_set.add(6)
print("Set after adding a single member:", my_set)

# Add multiple members to the set using update() method
my_set.update([7, 8, 9])
print("Set after adding multiple members:", my_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Initial set: {1, 2, 3, 4, 5}
Set after adding a single member: {1, 2, 3, 4, 5, 6}
Set after adding multiple members: {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Question 12

Write a Python program to remove Item(s) from a given set.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

my_set = {1, 2, 3, 4, 5}

# Print the initial set
print("Initial set:", my_set)

# Remove a single item from the set using discard() method
my_set.discard(3)
print("Set after removing a single item:", my_set)

# Remove multiple items from the set using discard() method
items_to_remove = {1, 4}
my_set.difference_update(items_to_remove)
print("Set after removing multiple items:", my_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Initial set: {1, 2, 3, 4, 5}
Set after removing a single item: {1, 2, 4, 5}
Set after removing multiple items: {2, 5}
```

Question 13

Write a Python program to remove an item from a set if it is

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def remove_item(set_name, item):
if item in set_name:
    set_name.remove(item)
    print(f"Item '{item}' removed from the set")
else:
    print(f"Item '{item}' is not present in the set")

# Define a set
my_set = {1, 2, 3, 4, 5}

# Print the initial set
print("Initial set:", my_set)

# Remove an item from the set
item_to_remove = int(input("Enter the item to remove: "))
remove_item(my_set, item_to_remove)

# Print the set after removal
print("Set after removal:", my_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Initial set: {1, 2, 3, 4, 5}
Enter the item to remove: 3
Item '3' removed from the set
Set after removal: {1, 2, 4, 5}
```

Question 14

Write a Python program to create an intersection of sets.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Find the intersection of the sets using the intersection() method
intersection_set = set1.intersection(set2)

# Print the intersection set
print("Intersection of set1 and set2:", intersection_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Intersection of set1 and set2: {4, 5}
```

Question 15

Write a Python program to create a union of sets.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Find the union of the sets using the union() method
union_set = set1.union(set2)

# Print the union set
print("Union of set1 and set2:", union_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Union of set1 and set2: {1, 2, 3, 4, 5, 6, 7, 8}
```

Question 16

Write a Python program to create set difference

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Find the set difference using the difference() method
difference_set = set1.difference(set2)

# Print the set difference
print("Set difference (set1 - set2):", difference_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Set difference (set1 - set2): {1, 2, 3}
```

Question 17

Write a Python program to create a symmetric difference.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Find the symmetric difference using the symmetric_difference() method
symmetric_difference_set = set1.symmetric_difference(set2)

# Print the symmetric difference
print("Symmetric difference of set1 and set2:", symmetric_difference_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Symmetric difference of set1 and set2: {1, 2, 3, 6, 7, 8}
```

Question 18

write a Python program to check if a set is a subset of another set.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def is_subset(set1, set2):
# Check if set1 is a subset of set2
return set1.issubset(set2)

# Example usage:
set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}

if is_subset(set1, set2):
    print("set1 is a subset of set2")
else:
    print("set1 is not a subset of set2")
```

```
Output:

Rishabh Rathore
0827CI211155
set1 is a subset of set2
```

Question 19

Write a Python program to remove all elements from a given set

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def remove_all_elements(some_set):
some_set.clear()  # Clears all elements from the set

# Example usage:
my_set = {1, 2, 3, 4, 5}
print("Before removing elements:", my_set)

remove_all_elements(my_set)
print("After removing elements:", my_set)
```

```
Output:

Rishabh Rathore
0827CI211155
Before removing elements: {1, 2, 3, 4, 5}
After removing elements: set()
```

Question 20

Write a Python program to find the maximum and minimum values in a set

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def find_max_min(some_set):
if len(some_set) == 0:
    print("The set is empty.")
else:
    max_value = max(some_set)
    min_value = min(some_set)
    return max_value, min_value

# Example usage:
my_set = {5, 3, 9, 2, 8, 1}

result = find_max_min(my_set)
if result:
    max_val, min_val = result
    print("Maximum value:", max_val)
    print("Minimum value:", min_val)
```

```
Output:

Rishabh Rathore
0827CI211155
Maximum value: 9
Minimum value: 1
```

Question 21

write a Python program to find length of a set

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def find_set_length(some_set):
return len(some_set)

# Example usage:
my_set = {1, 2, 3, 4, 5}
set_length = find_set_length(my_set)
print("Length of the set:", set_length)
```

```
Output:

Rishabh Rathore
0827CI211155
Length of the set: 5
```

Question 22

Write a Python program to check if a given value is present in a set or not.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def check_value_in_set(some_set, value):
return value in some_set

# Example usage:
my_set = {1, 2, 3, 4, 5}
value_to_check = 3

if check_value_in_set(my_set, value_to_check):
    print("The value", value_to_check, "is present in the set.")
else:
    print("The value", value_to_check, "is not present in the set.")
```

```
Output:

Rishabh Rathore
0827CI211155
The value 3 is present in the set
```

Question 23

Write a Python program to check if two given sets have no elements in common.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def no_common_elements(set1, set2):
return set1.isdisjoint(set2)

# Example usage:
set1 = {1, 2, 3}
set2 = {4, 5, 6}

if no_common_elements(set1, set2):
    print("The sets have no common elements.")
else:
    print("The sets have common elements.")
```

```
Output:

Rishabh Rathore
0827CI211155
The sets have no common elements.
```

Question 24

Write a Python program to create a Dictionary of student information. take roll no as key

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def create_student_dictionary():
num_students = int(input("Enter the number of students: "))
student_dict = {}

for _ in range(num_students):
    roll_no = input("Enter roll number: ")
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    grade = input("Enter grade: ")
    student_dict[roll_no] = {"Name": name, "Age": age, "Grade": grade}

return student_dict

def main():
    student_info = create_student_dictionary()
    print("\nStudent Information:")
    for roll_no, info in student_info.items():
        print("for student ")
        print(f"Roll No: {roll_no}")
        print(f"Name: {info['Name']}")
        print(f"Age: {info['Age']}")
        print(f"Grade: {info['Grade']}")
        print()

if __name__ == "__main__":
    main()
```

```
Output:

Rishabh Rathore
0827CI211155
Enter the number of students: 3
Enter roll number: 101
Enter name: abhay
Enter age: 23
Enter grade: A
Enter roll number: 102
Enter name: akaay
Enter age: 22
Enter grade: C
Enter roll number: 103
Enter name: asif
Enter age: 21
Enter grade: B

Student Information:
Roll No: 101
Name: abhay
Age: 23
```

```
Grade: A

Roll No: 102
Name: akaay
Age: 22
Grade: C

Roll No: 103
Name: asif
Age: 21
Grade: B
```

Question 25

Write a Python progarm to delete a key from the dictionary

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def delete_key_from_dict(student_dict, roll_no):
if roll_no in student_dict:
    del student_dict[roll_no]
    print(f"Key '{roll_no}' deleted successfully.")
else:
    print(f"Key '{roll_no}' not found in the dictionary.")

def main():
    # Example student dictionary
    student_dict = {
        "001": {"Name": "Alice", "Age": 18, "Grade": "A"},
        "002": {"Name": "Bob", "Age": 19, "Grade": "B"},
        "003": {"Name": "Charlie", "Age": 20, "Grade": "C"}
    }

    print("Original Dictionary:")
    print(student_dict)

    roll_no_to_delete = input("\nEnter the roll number to delete: ")
    delete_key_from_dict(student_dict, roll_no_to_delete)

    print("\nUpdated Dictionary:")
    print(student_dict)

if __name__ == "__main__":
    main()
```

Output:

```
Rishabh Rathore
0827CI211155
Original Dictionary:
{'001': {'Name': 'Alice', 'Age': 18, 'Grade': 'A'},
 '002': {'Name': 'Bob', 'Age': 19, 'Grade': 'B'},
 '003': {'Name': 'Charlie', 'Age': 20, 'Grade': 'C'}}

Enter the roll number to delete: 003
Key '003' deleted successfully.

Updated Dictionary:
{'001': {'Name': 'Alice', 'Age': 18, 'Grade': 'A'},
'002': {'Name': 'Bob', 'Age': 19, 'Grade': 'B'}}
```

Question 26

Write a Python program to add or update the data.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def add_or_update_data(dictionary, key, value):
"""
Function to add or update data in a dictionary.

Parameters:
dictionary (dict): The dictionary to modify.
key: The key to add/update in the dictionary.
value: The value corresponding to the key.

Returns:
None
"""
dictionary[key] = value


my_dictionary = {'a': 1, 'b': 2, 'c': 3}

print("Original Dictionary:", my_dictionary)


add_or_update_data(my_dictionary, 'd', 4)
print("Dictionary after adding 'd':", my_dictionary)


add_or_update_data(my_dictionary, 'b', 5)
print("Dictionary after updating 'b':", my_dictionary)
```

```
Output:

Rishabh Rathore
0827CI211155
Original Dictionary: {'a': 1, 'b': 2, 'c': 3}
Dictionary after adding 'd': {'a': 1, 'b': 2, 'c': 3, 'd': 4}
Dictionary after updating 'b': {'a': 1, 'b': 5, 'c': 3, 'd': 4}
```

Question 27

Write a Python script to concatenate the following dictionaries to create a new one sample Dictionary:dic1=1:10,2:20, dic2=3:30,4:40

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def concatenate_dicts(*dicts):
"""
Concatenate multiple dictionaries into a new one.

Parameters:
*dicts: Variable number of dictionaries to concatenate.

Returns:
dict: The concatenated dictionary.
"""
concatenated_dict = {}
for d in dicts:
    concatenated_dict.update(d)
return concatenated_dict

# Sample dictionaries
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}

# Concatenate dictionaries
concatenated_dict = concatenate_dicts(dic1, dic2)

# Output the concatenated dictionary
print("Concatenated Dictionary:", concatenated_dict)
```

```
Output:

Rishabh Rathore
0827CI211155
Concatenated Dictionary: {1: 10, 2: 20, 3: 30, 4: 40}
```

Question 28

Write a Python script to check whether a given key already exists in a dictionary

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def check_key_existence(dictionary, key):
"""
Check whether a given key exists in a dictionary.

Parameters:
dictionary (dict): The dictionary to check.
key: The key to check for existence.

Returns:
bool: True if the key exists, False otherwise.
"""
return key in dictionary

# Sample dictionary
my_dict = {'a': 1, 'b': 2, 'c': 3}

# Key to check
key_to_check = 'b'

# Check if the key exists in the dictionary
if check_key_existence(my_dict, key_to_check):
    print(f"The key '{key_to_check}' exists in the dictionary.")
else:
    print(f"The key '{key_to_check}' does not exist in the dictionary.")
```

```
Output:

Rishabh Rathore
0827CI211155
The key 'b' exists in the dictionary.
```

Question 29

Write a Python program to iterate over dictionaries using for loops

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Sample dictionary
my_dict = {'a': 1, 'b': 2, 'c': 3}

# Iterate over keys
print("Iterating over keys:")
for key in my_dict:
    print(key)

# Iterate over values
print("\nIterating over values:")
for value in my_dict.values():
    print(value)

# Iterate over key-value pairs
print("\nIterating over key-value pairs:")
for key, value in my_dict.items():
    print(key, "->", value)
```

```
Output:

Rishabh Rathore
0827CI211155
Iterating over keys:
a
b
c

Iterating over values:
1
2
3

Iterating over key-value pairs:
a -> 1
b -> 2
c -> 3
```

Question 30

Write a Python script to generate and print a dictionary that contains a number (between I and n) in the form (x, x*x).Sample dictionary (n=5): Expected output: 1:1,2:4,3:9,4:16,5:25

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

def generate_squared_dict(n):
"""
Generate a dictionary containing numbers and their squares from 1 to n.

Parameters:
n (int): The maximum number to include in the dictionary.

Returns:
dict: The generated dictionary.
"""
squared_dict = {}
for x in range(1, n+1):
    squared_dict[x] = x * x
return squared_dict

# Sample value of n
n = 8

# Generate the dictionary
result_dict = generate_squared_dict(n)

# Print the generated dictionary
print("Generated Dictionary:", result_dict)
```

```
Output:

Rishabh Rathore
0827CI211155
Generated Dictionary: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
```

Question 31

Write a python function to calculate Sum of two variables.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def sum_of_two_variables(a, b):
"""
Calculate the sum of two variables.

Parameters:
a: The first variable.
b: The second variable.

Returns:
The sum of a and b.
"""
return a + b

# Example usage:
result = sum_of_two_variables(5, 7)
print("Sum of the two variables:", result)
```

```
Output:

Rishabh Rathore
0827CI211155
Sum of the two variables: 12
```

Question 32

Write a Python function to show the use of Default Parameter

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def greet(name, greeting="Hello"):
"""
Function to greet a person with a specified greeting.

Parameters:
name: The name of the person to greet.
greeting \(optional\): The greeting to use. Defaults to "Hello" if not specified.

Returns:
str: The greeting message.
"""
return f"{greeting}, {name}!"

# Example usage:
print(greet("Ram"))  # Uses the default greeting "Hello"
```

```
Output:

Rishabh Rathore
0827CI211155
Hello, Ram!
Good morning, Siya!
```

Question 33

Write a function to find the maximum of three numbers

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def find_maximum(a, b, c):
"""
Function to find the maximum of three numbers.

Parameters:
a: The first number.
b: The second number.
c: The third number.

Returns:
The maximum of the three numbers.
"""
if a >= b:
    if a >= c:
        return a
    else:
        return c
else:
    if b >= c:
        return b
    else:
        return c

# Example usage:
result = find_maximum(10, 5, 8)
print("Maximum of the three numbers:", result)
```

```
Output:

Rishabh Rathore
0827CI211155
Maximum of the three numbers: 10
```

Question 34

Write a python function to use arbitrary argument

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def print_arguments(*args):
"""
Function to print arbitrary arguments passed to it.

Parameters:
*args: Arbitrary number of arguments.

Returns:
None
"""
for arg in args:
    print(arg)

# Example usage:
print_arguments(1, 2, 3)
print_arguments('Hello', 'world', '!')
```

```
Output:

Rishabh Rathore
0827CI211155
1
2
3
Hello
world
!
```

Question 35

Write a Python function to reverse a string

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def reverse_string(input_string):
"""
Function to reverse a given string.

Parameters:
input_string (str): The string to be reversed.

Returns:
str: The reversed string.
"""
return input_string[::-1]

# Example usage:
original_string = "hello"
reversed_string = reverse_string(original_string)
print("Original string:", original_string)
print("Reversed string:", reversed_string)
```

```
Output:

Rishabh Rathore
0827CI211155
Original string: hello
Reversed string: olleh
```

Question 36

Write a Python function to print the document string

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def print_docstring(func):
"""
Function to print the docstring of a given function.

Parameters:
func: The function whose docstring is to be printed.

Returns:
None
"""
print(func.__doc__)

# Example usage:
def example_function():
    """
    This is an example function.
    It does nothing.
    """
    pass

print_docstring(example_function)
```

Output:

```
Rishabh Rathore
0827CI211155

   This is an example function.
   It does nothing.
```

Question 37

Write a Python function to demonstrate the scope of local and global variable.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
global_variable = "I am global"

def demonstrate_scope():
    """
    Function to demonstrate the scope of local and global variables.
    """
    local_variable = "I am local"
    print("Inside the function:")
    print("Local variable:", local_variable)
    print("Global variable:", global_variable)

# Call the function
demonstrate_scope()

# Attempt to access local_variable outside the function - this will raise an error
# print("Outside the function:")
# print("Local variable:", local_variable)

# Access global_variable outside the function
print("Accessing global variable outside the function:", global_variable)
```

```
Output:

Rishabh Rathore
0827CI211155
Inside the function:
Local variable: I am local
Global variable: I am global
Accessing global variable outside the function: I am global
```

Question 38

Write a Python function to calculate the factorial of a number (a nonnegative integer). The function accepts the number as an argument

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def factorial(n):
"""
Function to calculate the factorial of a nonnegative integer.

Parameters:
n (int): The number whose factorial is to be calculated.

Returns:
int: The factorial of the input number.
"""
if n < 0:
    return "Factorial is not defined for negative numbers"
elif n == 0 or n == 1:
    return 1
else:
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Example usage:
number = 5
print(f"The factorial of {number} is:", factorial(number))
print("\n")
```

```
Output:

Rishabh Rathore
0827CI211155
The factorial of 5 is: 120
```

Question 39

Write a Python function to check whether a number falls within a given range.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def check_range(number, start, end):
"""
Function to check whether a number falls within a given range.

Parameters:
number: The number to check.
start: The start of the range (inclusive).
end: The end of the range (inclusive).

Returns:
bool: True if the number falls within the range, False otherwise.
"""
return start <= number <= end

# Example usage:
num = 7
start_range = 5
end_range = 10

if check_range(num, start_range, end_range):
    print(f"{num} falls within the range [{start_range}, {end_range}]")
else:
    print(f"{num} does not fall within the range [{start_range}, {end_range}]")
```

```
Output:

Rishabh Rathore
0827CI211155
7 falls within the range [5, 10]
```

Question 40

Write a Python function that accepts a string and counts the number of upper and lower case letters.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def count_upper_lower(string):
"""
Function to count the number of upper and lower case letters in a string.

Parameters:
string (str): The input string.

Returns:
tuple: A tuple containing the count of upper case letters and lower case letters, respec
"""
upper_count = 0
lower_count = 0
for char in string:
    if char.isupper():
        upper_count += 1
    elif char.islower():
        lower_count += 1
return upper_count, lower_count

# Example usage:
input_string = "Hello World"
upper, lower = count_upper_lower(input_string)
print("Number of upper case letters:", upper)
print("Number of lower case letters:", lower)
```

```
Output:

Rishabh Rathore
0827CI211155
Number of upper case letters: 2
Number of lower case letters: 8
```

Question 41

Write a Python function that takes a list and returns a new list with distinct elements from the first list. sample list:[1,2,3,3,3,3,4,5],Unique list:[1,2,3,4,5]

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def distinct_elements(input_list):
"""
Function to return a new list with distinct elements from the input list.

Parameters:
input_list (list): The input list.

Returns:
list: A new list with distinct elements.
"""
return list(set(input_list))

# Example usage:
original_list = [1, 2, 2, 3, 3, 4, 5, 5]
distinct_list = distinct_elements(original_list)
print("Original List:", original_list)
print("Distinct List:", distinct_list)
```

```
Output:

Rishabh Rathore
0827CI211155
Original List: [1, 2, 2, 3, 3, 4, 5, 5]
Distinct List: [1, 2, 3, 4, 5]
```

Question 42

Write a Python function that takes a number as a parameter and checks whether the numberis prime or not

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def is_prime(number):
"""
Function to check whether a number is prime or not.

Parameters:
number (int): The number to check.

Returns:
bool: True if the number is prime, False otherwise.
"""
if number <= 1:
    return False
elif number <= 3:
    return True
elif number % 2 == 0 or number % 3 == 0:
    return False
else:
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True

# Example usage:
num = 19
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
print("\n")
```

```
Output:

Rishabh Rathore
0827CI211155
19 is a prime number.
```

Question 43

Write a Python program to access a function inside a function

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def outer_function():
"""
Outer function.
"""
    print("This is the outer function.")

    def inner_function():
        """
        Inner function.
        """
        print("This is the inner function.")

# Call the inner function
inner_function()

# Call the outer function
outer_function()
```

```
Output:

Rishabh Rathore
0827CI211155
This is the outer function.
This is the inner function.
```

Question 44

Write a Python function that checks whether a passed string is a palindrome or not.

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def is_palindrome(string):
"""
Function to check whether a passed string is a palindrome or not.

Parameters:
string (str): The string to check.

Returns:
bool: True if the string is a palindrome, False otherwise.
"""
# Convert the string to lowercase and remove spaces
clean_string = string.lower().replace(" ", "")

# Compare the string with its reverse
return clean_string == clean_string[::-1]

# Example usage:
input_string = "A man a plan a canal Panama"
if is_palindrome(input_string):
    print(f"'{input_string}' is a palindrome.")
else:
    print(f"'{input_string}' is not a palindrome.")
```

```
Output:

Rishabh Rathore
0827CI211155
'A man a plan a canal Panama' is a palindrome.
```

Question 45

Write a python function to create and print a list where the values are the squares of numbers between 1 and 30

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def squares_list():
"""
Function to create and print a list where the values are the squares of numbers between

Returns:
list: A list containing the squares of numbers between 1 and 30.
"""
squares = [x ** 2 for x in range(1, 31)]
return squares

# Example usage:
squares = squares_list()
print("List of squares of numbers between 1 and 30:", squares)
```

Output:

```
Rishabh Rathore
0827CI211155
List of squares of numbers between 1 and 30: [1, 4, 9, 16, 25, 36, 49, 64, 81,
100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441,
484, 529, 576, 625, 676, 729, 784, 841, 900]
```

Question 46

Write a Python program to print the even numbers from a given list

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
def print_even_numbers(input_list):
"""
Function to print the even numbers from a given list.

Parameters:
input_list (list): The input list.

Returns:
None
"""
even_numbers = [num for num in input_list if num % 2 == 0]
print("Even numbers from the given list:", even_numbers)

# Example usage:
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print_even_numbers(numbers)
```

```
Output:

Rishabh Rathore
0827CI211155
Even numbers from the given list: [2, 4, 6, 8, 10]
```

Question 47

Write a program to create a class student with data member name. roll no. Semester. and display the data using object

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
class Student:
"""
Class representing a student.
"""
def __init__(self, name, roll_no, semester):
    """
    Constructor to initialize the data members of the Student class.

    Parameters:
    name (str): The name of the student.
    roll_no (str): The roll number of the student.
    semester (str): The semester of the student.
    """
    self.name = name
    self.roll_no = roll_no
    self.semester = semester

def display_data(self):
    """
    Method to display the data of the student.
    """
    print("Name:", self.name)
    print("Roll No:", self.roll_no)
    print("Semester:", self.semester)

# Create an object of the Student class
student1 = Student("Ramesh Kumar", "12345", "Spring 2024")

# Display the data using the object
print("Student Data:")
student1.display_data()
```

Output:

```
Rishabh Rathore
0827CI211155
Student Data:
Name: Ramesh Kumar
Roll No: 12345
Semester: Spring 2024
```

Question 48: Write a program to demonstrate the use of *init*

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
class Car:
"""
Class representing a car.
"""
def __init__(self, make, model, year):
    """
    Constructor to initialize the data members of the Car class.

    Parameters:
    make (str): The make of the car.
    model (str): The model of the car.
    year (int): The manufacturing year of the car.
    """
    self.make = make
    self.model = model
    self.year = year
    self.odometer_reading = 0  # Additional attribute

def get_car_info(self):
    """
    Method to display information about the car.
    """
    car_info = f"{self.year} {self.make} {self.model}"
    return car_info

def read_odometer(self):
    """
    Method to read the odometer reading of the car.
    """
    print(f"This car has {self.odometer_reading} miles on it.")

# Create an object of the Car class
my_car = Car("Toyota", "Corolla", 2022)

# Display information about the car
print("Car Information:", my_car.get_car_info())

# Read the odometer reading of the car
my_car.read_odometer()
```

Output:

```
Rishabh Rathore
0827CI211155
Car Information: 2022 Toyota Corolla
This car has 0 miles on it.
```

Question 49

Write a program for Inheritence.a) single level b) Multiple c) Multilevel d) Hybrid e) Hierarchical

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")

# Single level inheritance
class Parent:
    def parent_method(self):
        print("This is the parent method.")

class Child(Parent):
    def child_method(self):
        print("This is the child method.")

# Single level inheritance
child_obj = Child()
child_obj.parent_method()
child_obj.child_method()

# Multiple inheritance
class Parent1:
    def method1(self):
        print("This is method 1 of Parent1.")

class Parent2:
    def method2(self):
        print("This is method 2 of Parent2.")

class Child1(Parent1, Parent2):
    def child_method1(self):
        print("This is the child1 method.")

# Multiple inheritance
child_obj1 = Child1()
child_obj1.method1()
child_obj1.method2()
child_obj1.child_method1()

# Multilevel inheritance
class Grandparent:
    def grandparent_method(self):
        print("This is the grandparent method.")

class Parent3(Grandparent):
    def parent_method2(self):
        print("This is the parent method.")

class Child2(Parent3):
    def child_method2(self):
        print("This is the child method.")

# Multilevel inheritance
child_obj2 = Child2()
```

```
child_obj2.grandparent_method()
child_obj2.parent_method2()
child_obj2.child_method2()

# Hybrid inheritance
class Parent4:
    def method1(self):
        print("This is method 1 of Parent4.")

class Parent5:
    def method2(self):
        print("This is method 2 of Parent5.")

class Parent6:
    def method3(self):
        print("This is method 3 of Parent6.")

class Child3(Parent4, Parent5, Parent6):
    def child_method3(self):
        print("This is the child method.")

# Hybrid inheritance
child_obj3 = Child3()
child_obj3.method1()
child_obj3.method2()
child_obj3.method3()
child_obj3.child_method3()

# Hierarchical inheritance
class Parent7:
    def parent_method4(self):
        print("This is the parent method.")

class Child4(Parent7):
    def child1_method4(self):
        print("This is the child1 method.")

class Child5(Parent7):
    def child2_method4(self):
        print("This is the child2 method.")

# Hierarchical inheritance
child1_obj = Child4()
child1_obj.parent_method4()
child1_obj.child1_method4()

child2_obj = Child5()
child2_obj.parent_method4()
child2_obj.child2_method4()
```

Output:

```
Rishabh Rathore
0827CI211155
This is the parent method.
This is the child method.
```

```
This is method 1 of Parent1.
This is method 2 of Parent2.
This is the child1 method.
This is the grandparent method.
This is the parent method.
This is the child method.
This is method 1 of Parent4.
This is method 2 of Parent5.
This is method 3 of Parent6.
This is the child method.
This is the parent method.
This is the child1 method.
This is the parent method.
This is the child2 method.
```

Question 50

Write a program to demonstrate overriding

Solution:

```
print("Rishabh Rathore")
print("0827CI211155")
class Parent:
def show_message(self):
    print("This is the parent message.")

class Child(Parent):
    def show_message(self):
        print("This is the overridden message.")

# Create objects of both Parent and Child classes
parent_obj = Parent()
child_obj = Child()

# Call the method on each object
print("Calling show_message() method of Parent class:")
parent_obj.show_message()

print("\nCalling show_message() method of Child class:")
child_obj.show_message()
```

```
Output:

Rishabh Rathore
0827CI211155
Calling show_message() method of Parent class:
This is the parent message.

Calling show_message() method of Child class:
This is the overridden message.
```