

Problem Sheet - 1 BSCIT- SEM 4 Programming with Python (1020406404)

sr.no.	Practice Name	Date	Sign
Practical 1.	Write a python program to take input from the user.		
Practical 2	1. Write a python program to take multiple input from the user in the following manner :- Student name Student enrolment no Subject marks no1 Subject marks no2 Subject marks no3 Output should be printed in like :- eg RIYA 101 20 30 40		
Practical 3	Write a python program print input with its data type .		
Practical 4	Write a python program to perform all arithmetic operators in the following manner:-		
Practical 5	Write a python program to perform all comparison operators in the following manner		
Practical 6	3. Write a python program to perform all Logical operators.		
Practical 7	3. Write a python program to perform all Bitwise operators.		
Practical 8	3. Write a python program to perform all Assignment operators.		
Practical 9	3. Write a python program to perform all Assignment operators.		
Practical 10	Write a python program to take user input as dataset as your choice and perform identity and membership operator		

Problem Sheet – 2 BSCIT- SEM 4 Programming with Python (1020406404)

sr.no.	Practical Name	Date	Sign
Practical 1	1. Write a Python Programming to print data type of different variables .		
Practical 2	Write a Python Programming to create List and accessing each item form index .		
Practical 3	Write a Python Programming to create Tuple and accessing each item form index .		
Practical 4	1. Write a Python Programming to create Set and accessing each item form index .		
Practical 5	Write a Python Programming to create Dictionary and accessing each item form index		
Practical 6	1. Write a Python Programming to implement global and local variable .		
Practical 7	1. Write a Python Programming to perform assignment operator using different variable declaration.		
Practical 8	Write a Python Programming to invoke string data type in python and also create multi line string ,than access first and last characters of the string		
Practical 9	1. Write a Python Programming to create multi-dimensional List and access negative index from the List		
Practical 10	1. Write a Python Programming to use of tuple function and declare distinct nested tuples of your choice(eg tuple1,tuple2) and print all nested tuple into one common tuple(eg tuple 3) .		
Practical 11	1. Write a Python Programming to access elements from dictionary using get and key.		
Practical 12	1. Write a Python Programming to create set of mixed values and access each elements form the set.		

Practical 13	13. Write a Python Programming to use list with set ,tuple,dictionary and print the output .		
Practical 14	1. Write a Python Programming to create dictionary and perform the following: a) Empty Dictionary b) Dictionary with the use of Integer Keys c) Dictionary with the use of Mixed Keys d) Dictionary with the use of dict() e) Dictionary with each item as a pair f) Accessing a element using key g) Accessing a element using get		
Practical 15	1. Write a python programme numeric and imaginary data type in python using variables.		
Practical 16	1. Write a Python program to perform sum of first 10 numbers using range function.		
Practical 17	1. Write a Python program to illustrate if statement .		
Practical 18	1. Write a Python program to illustrate if -else statement .		
Practical 19	Write a Python program to illustrate if-elif-else statement		
Practical 20	1. Write a Python program to illustrate for loop using break,pass,continue.		
Practical 21	1. Write a Python program to demonstrate for-else loop .		

Problem Sheet – 3 BSCIT- SEM 4 Programming with Python (1020406404)

sr.no	Practical Name	Date	Sign
Practical 1	1. Write a Python program to perform sum of first 10 numbers using range function.		
Practical 2	1. Write a Python program to illustrate if statement .		

Practical 3	1. Write a Python program to illustrate if -else statement .		
Practical 4	Write a Python program to illustrate if-elif-else statement		
Practical 5	1. Write a Python program to illustrate for loop using break,pass,continue.		
Practical 6	1. Write a Python program to demonstrate for-else loop .		

Problem Sheet – 4 BSCIT- SEM 4 Programming with Python (1020406404)

Sr.no	Practical Name	Date	Sign
Practical 1	1. Write a Python program to take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero. Find total number of numbers entered and their sum. Display count and sum with appropriate titles.		
Practical 2	Write a Python program to take input of positive numbers, with an appropriate prompt, from the user until the user enters a zero. Find total number of odd & even numbers entered and sum of odd and even numbers. Display total count of odd & even numbers and sum of odd & even numbers with appropriate titles.		
Practical 3	1. Write a Python program to take input of a positive number, with an appropriate prompt, from the user. The user should be prompted again to enter the number until the user enters a positive number. Check whether the number is a prime number or not and accordingly display appropriate message		
Practical 4	1. Write a Python program to take input of a positive number, say N, with an appropriate prompt, from the user. The user should be prompted again to enter the number until the user enters a positive number. Find the sum of first N odd numbers and first N even numbers. Display both the sums with appropriate titles.		
Practical 5	1. Consider a list of numbers. Write a Python program to do the following: 1) Count total number of numbers in the list 2) Sum and Average of all the numbers in the list 3) Count and sum of all the odd numbers in the list		

	4) Count and sum of all the even numbers in the list 5) Find the largest number in the list 6) Find the smallest number in the list Display all the values with appropriate titles.		
Practical 6	1. Consider a list of characters (characters may be alphabets, special characters, digits). Write a Python program to do the following: 1) Count total number of elements in the list 2) Count total number of vowels in the list (vowels are 'a', 'e', 'i', 'o', 'u') 3) Count total number of consonants in the list (a consonant is an alphabet other than vowel) 4) Count total number of characters other than vowels and consonants Display all the values with appropriate titles.		
Practical 7	1. Consider a single list consisting of integer values, float values, character values, string values and lists. Write a Python program to do the following: 1) Count total number of elements in the list 2) Count total number of integer values, float values, character values, string values and lists Display all the values with appropriate titles		

Problem Sheet - 5 BSCIT- SEM 4 Programming with Python (1020406404)

Sr.no.	Practical Name	date	sign
Practical 1	1. Write a Python program to create a list of numbers by taking input from the user and then remove the duplicates from the list. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero. Sample Input: [10, 34, 18, 10, 12, 34, 18, 20, 25, 20] Output: [10, 34, 18, 12, 20, 25]		
Practical 2	1. Write a Python program to create a list of lists of numbers (i.e. each element of the list is a list of numbers e.g. [[1, 2], [3, 2, 5], [1], [5, 3, 6, 7]]. Then generate a list from the given list where each element of the list is the length of each list in the given list. i.e. for the given example the output should be [2, 3, 1, 4]. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero. Sample Input: [[1, 2], [3, 2, 5], [1], [5, 3, 6, 7]]		

Practical 3	1. Write a Python program to create a list of numbers by taking input from the user. Split this list into two tuples where one tuple contains odd numbers and the other tuple contains even numbers from the list. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero. Sample Input: [10, 12, 13, 90, 43, 32, 30, 11]												
Practical 4	1. Write a Python program to create a list of elements of any data type (mixed data type, i.e. some elements maybe of type int, some elements of type float and some elements of type string). Split this list into three tuples containing elements of same data type (i.e. 1st tuple of integers only, 2 nd tuple of float only and 3rd tuple of strings only). Take input from the user to create the list. Sample Input: [25, 4.5, 'Hello', 90, 20, 7.5, 9.25, 'World']												
Practical 5	1. Write a Python program to create a dictionary of student data by taking input from the user, where each student data contains Rollno (to be considered as key), and marks in 3 subjects (to be considered as list of values). e.g. {1 : [45, 40, 35], 2 : [41, 38, 39], 3 : [35, 30, 37]}. Prepare mark list in the following format: <table><tr><td>Roll No.</td><td>Mark-1</td><td>Mark-2</td><td>Mark-3</td><td>Total</td></tr><tr><td>1</td><td>45</td><td>40</td><td>35</td><td>120</td></tr></table>	Roll No.	Mark-1	Mark-2	Mark-3	Total	1	45	40	35	120		
Roll No.	Mark-1	Mark-2	Mark-3	Total									
1	45	40	35	120									
Practical 6	1. Write a Python program to take input of a string from the user and then create a dictionary containing each character of the string along with their frequencies. (e.g. if the string is 'banana' then output should be {'b': 1, 'a': 3, 'n': 2}).												
Practical 7	1. Write a Python program to create a list of strings by taking input from the user and then create a dictionary containing each string along with their frequencies. (e.g. if the list is ['apple', 'banana', 'fig', 'apple', 'fig', 'banana', 'grapes', 'fig', 'grapes', 'apple'] then output should be {'apple': 3, 'banana': 2, 'fig': 3, 'grapes': 2}).												
Practical 8	1. Write a Python program to input a string that is a list of words separated by commas. Construct a dictionary that contains all these words as keys and their frequencies as values. Then display the words with their quantities.												
Practical 9	Consider a very small dictionary that contains the translations of English words to Dutch as shown below												

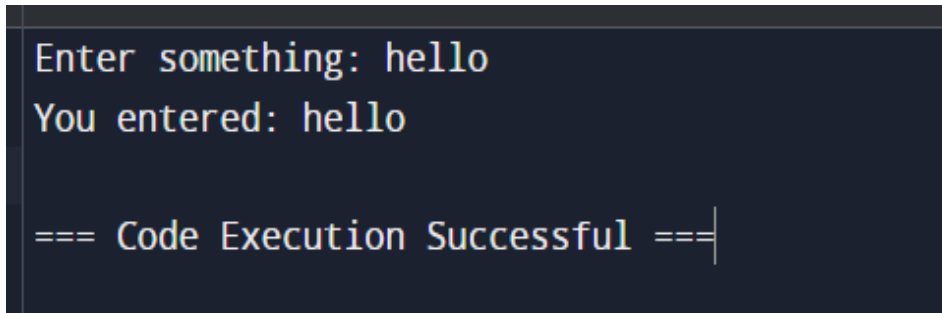
Practical 10	<p>1. Consider the following list of movies. For each movie it also shows list of ratings. Write a Python program to convert it in such a way that it stores all this data in one dictionary, then use the dictionary to print the average rating for each movie, rounded to one decimal.</p> <p>movies = ["Where Eagle's Dare", "Enter the Dragon", "Iron Fist", "Fist of Fury"]</p> <p>dare_ratings = [9, 10, 9.5, 8.5, 3, 7.5, 8] for the movie "Where Eagle's Dare"</p> <p>dragon_ratings = [10, 10, 0, 9, 1, 8, 7.5, 8, 6, 9] for the movie "Enter the Dragon"</p> <p>fist_ratings = [7, 6, 5] for the movie "Iron Fist"</p> <p>fury_ratings = [6, 5, 6, 6] for the movie "Fist of Fury"</p>		
--------------	--	--	--

Problem Sheet - 1 BSCIT- SEM 4 Programming with Python (1020406404)

1. Write a python program to take input from the user.

Input: user_input = input("Enter something: ")
print("You entered:", user_input)

Output:



```
Enter something: hello
You entered: hello

=== Code Execution Successful ===
```

2. Write a python program to take multiple input from the user in the following manner :-

Student name
Student enrolment no
Subject marks no1
Subject marks no2
Subject marks no3
Output should be printed in like :- eg RIYA 101 20 30 40

Input:

```
student_name = input("Enter student name: ")
enrolment_no = int(input("Enter student enrolment number: "))
subject_marks_1 = float(input("Enter marks for subject 1: "))
subject_marks_2 = float(input("Enter marks for subject 2: "))
subject_marks_3 = float(input("Enter marks for subject 3: "))

print("\nStudent Information:")
print("Name:", student_name)
print("Enrolment Number:", enrolment_no)
print("Subject 1 Marks:", subject_marks_1)
print("Subject 2 Marks:", subject_marks_2)
print("Subject 3 Marks:", subject_marks_3)
```



```

Enter student name: Radha
Enter student enrolment number: 101
Enter marks for subject 1: 50
Enter marks for subject 2: 48
Enter marks for subject 3: 46

```

```

Student Information:
Name: Radha
Enrolment Number: 101
Subject 1 Marks: 50.0
Subject 2 Marks: 48.0
Subject 3 Marks: 46.0

```

```

=== Code Execution Successful ===

```

Output :

3. Write a python program print input with its data type .

Input:

```

user_input = input("Enter something: ")
print("Input:", user_input)
print("Data type:", type(user_input))

```

```

Enter something: Hello Good Morning
Input: Hello Good Morning
Data type: <class 'str'>

=== Code Execution Successful ===

```

Output :

4. Write a python program to perform all arithmetic operators in the following manner:-

Student name	
Enrolment	
Course/Semester	
1) Subject name	Marks 1
2) Subject name	Marks 2
3) Subject name	Marks 3
4) Subject name	Marks 4
5) Subject name	Marks 5
Total Marks Obtained	
SCGPA	
Average Marks in 2 semesters	

Input:

```

student_name = input("Enter student name: ")
enrolment =int( input("Enter enrolment number: "))

```

```

course_semester = input("Enter course/semester: ")
subject1 = int(input("Enter the marks for subject1:"))
subject2= int(input("Enter the marks for subject2:"))
subject3 = int(input("Enter the marks for subject3:"))
subject4 = int(input("Enter the marks for subject4:"))
subject5 = int(input("Enter the marks for subject5:"))
total = subject1 + subject2 + subject3 + subject4 + subject5
average = (total / 5)
scgpa = average
print("Total Marks:", total)
print("SCGPA of semester: ", scgpa)
print("Average Marks:", average)

```

```

Enter student name: Mansi
Enter enrolment number: 1001
Enter course/semester: Bsc.It/4
Enter the marks for subject1: 47
Enter the marks for subject2: 48
Enter the marks for subject3: 50
Enter the marks for subject4: 38
Enter the marks for subject5: 49
Total Marks: 232
SCGPA of semester: 46.4
Average Marks: 46.4

=== Code Execution Successful ===

```

Output:

5. Write a python program to perform all comparison operators in the following manner:-

Student name	
Enrolment	
Course/Semester	
1) Subject name	Marks 1
2) Subject name	Marks 2
3) Subject name	Marks 3
4) Subject name	Marks 4
5) Subject name	Marks 5
Total Marks Obtained	
SCGPA	
Average Marks in 2 semesters	

Compare marks of all subjects and
print the highest,
lowest among them.

Also compare the SCGPA of any 2 semesters.

Input: student_name = input("Enter student name: ")
enrolment =int(input("Enter enrolment number: "))
course_semester = input("Enter course/semester: ")

```

subject1 = int(input("Enter the marks for subject1:"))
subject2= int(input("Enter the marks for subject2:"))
subject3 = int(input("Enter the marks for subject3:"))
subject4 = int(input("Enter the marks for subject4:"))
subject5 = int(input("Enter the marks for subject5:"))
total = subject1 + subject2 + subject3 + subject4 + subject5
average = (total / 5)
scgpa = average
print("Total Marks:", total)
print("SCGPA of semester: ", scgpa)
print("To check Smaller Marks")
if(subject1<subject2 & subject1<subject3 & subject1<subject4 & subject1<subject5):
    print("Subject 1 is Smaller.")
elif (subject2<subject1 & subject2<subject3 & subject2<subject4 & subject2<subject5):
    print("Subject 2 is Smaller.")
elif (subject3<subject1 & subject3<subject2&subject3<subject4 &subject3<subject5):
    print("Subject 3 is Smaller.")
elif (subject4<subject1 &subject4<subject2&subject4<subject3&subject4<subject5):
    print("Subject 4 is Smaller.")
else:
    print("Subject 5 is Smaller.")
print("To check Greater Marks")
if(subject1>subject2 & subject1>subject3 & subject1>subject4 & subject1>subject5):
    print("Subject 1 is Greater.")
elif (subject2>subject1 & subject2>subject3 & subject2>subject4 & subject2>subject5):
    print("Subject 2 is Greater.")
elif (subject3>subject1 & subject3>subject2 & subject3>subject4 & subject3>subject5):
    print("Subject 3 is Greater.")
elif (subject4>subject1 & subject4>subject2 & subject4>subject3 & subject4>subject5):
    print("Subject 4 is Greater.")
else:
    print("Subject 5 is Greater.")

```

```
Enter student name: Misti
Enter enrolment number: 1001
Enter course/semester: Bsc.It/4
Enter the marks for subject1: 48
Enter the marks for subject2: 43
Enter the marks for subject3: 50
Enter the marks for subject4: 47
Enter the marks for subject5: 37
Total Marks: 225
SCGPA of semester: 45.0
To check Smaller Marks
Subject 5 is Smaller.
To check Greater Marks
Subject 3 is Greater.

=== Code Execution Successful ===
```

Output:

6. Write a python program to perform all Logical operators.

Input:

```
a = True
b = False
print("Logical AND:")
print("a and b:", a and b) # False
print("\nLogical OR:")
print("a or b:", a or b) # True
print("\nLogical NOT:")
print("not a:", not a) # False
print("not b:", not b) # True
print("\nLogical XOR (Exclusive OR):")
print("a xor b:", (a and not b) or (not a and b)) # True
print("\nLogical NAND:")
print("not (a and b):", not (a and b)) # True
print("\nLogical NOR:")
print("not (a or b):", not (a or b)) # False
```

```
Logical AND:
a and b: False

Logical OR:
a or b: True

Logical NOT:
not a: False
not b: True

Logical XOR (Exclusive OR):
a xor b: True

Logical NAND:
not (a and b): True

Logical NOR:
not (a or b): False

=== Code Execution Successful ===
```

Output:

7. Write a python program to perform all Bitwise operators.

Input:

```
a = 60      # 60 in binary: 0011 1100
b = 13      # 13 in binary: 0000 1101
print("Bitwise AND:")
print("a & b:", a & b) # 0000 1100 -> 12
print("\nBitwise OR:")
print("a | b:", a | b) # 0011 1101 -> 61
print("\nBitwise XOR (Exclusive OR):")
print("a ^ b:", a ^ b) # 0011 0001 -> 49
print("\nBitwise NOT:")
print("~a:", ~a)      # -61 (2's complement of 0011 1100)
print("\nBitwise LEFT SHIFT:")
print("a << 2:", a << 2) # 1111 0000 -> 240
print("\nBitwise RIGHT SHIFT:")
print("a >> 2:", a >> 2) # 0000 1111 -> 15
```

```

Bitwise AND:
a & b: 12

Bitwise OR:
a | b: 61

Bitwise XOR (Exclusive OR):
a ^ b: 49

Bitwise NOT:
~a: -61

Bitwise LEFT SHIFT:
a << 2: 240

Bitwise RIGHT SHIFT:
a >> 2: 15

=== Code Execution Successful ===

```

Output:

8. Write a python program to perform all Assignment operators.

Input:

```

# Define two variables
a = 10
b = 5
c = a + b
print("Assignment operator (=):")
print("c =", c) # c = 15
c += a
print("\nAddition assignment (+=):")
print("c =", c) # c = 25
c -= b
print("\nSubtraction assignment (-=):")
print("c =", c) # c = 20
c *= a
print("\nMultiplication assignment (*=):")
print("c =", c) # c = 200
c /= b
print("\nDivision assignment (/=):")
print("c =", c) # c = 40.0
c %= a
print("\nModulus assignment (%=):")

```

```

print("c =", c) # c = 0.0
c **= b
print("\nExponentiation assignment (**=):")
print("c =", c) # c = 0.0
c //= a
print("\nFloor division assignment (//=):")
print("c =", c) # c = 0.0

```

```

Assignment operator (=):
c = 15

Addition assignment (+=):
c = 25

Subtraction assignment (-=):
c = 20

Multiplication assignment (*=):
c = 200

Division assignment (/=):
c = 40.0

Modulus assignment (%=):
c = 0.0

Exponentiation assignment (**=):
c = 0.0

Floor division assignment (//=):
c = 0.0

=== Code Execution Successful ===

```

Output:

9. Write a python program to perform all Assignment operators.

Input:

```

# Define two variables
a = 10
b = 5
c = a + b
print("Assignment operator (=:)")

```

```
print("c =", c) # c = 15
c += a
print("\nAddition assignment (+=):")
print("c =", c) # c = 25
c -= b
print("\nSubtraction assignment (-=):")
print("c =", c) # c = 20
c *= a
print("\nMultiplication assignment (*=):")
print("c =", c) # c = 200
c /= b
print("\nDivision assignment (/=):")
print("c =", c) # c = 40.0
c %= a
print("\nModulus assignment (%=):")
print("c =", c) # c = 0.0
c **= b
print("\nExponentiation assignment (**=):")
print("c =", c) # c = 0.0
c //= a
print("\nFloor division assignment (//=):")
print("c =", c) # c = 0.0
```



```
Assignment operator (=):
c = 15

Addition assignment (+=):
c = 25

Subtraction assignment (-=):
c = 20

Multiplication assignment (*=):
c = 200

Division assignment (/=):
c = 40.0

Modulus assignment (%=):
c = 0.0

Exponentiation assignment (**=):
c = 0.0

Floor division assignment (//=):
c = 0.0

=== Code Execution Successful ===
```

Output:

10. Write a python program to take user input as dataset as your choice and perform identity and membership operator.

Input:

```
dataset = input("Enter dataset (comma-separated values): ").split(',')
print("\nIdentity Operator (is):")
if 'hello' is dataset:
    print("'hello' is in dataset")
else:
    print("'hello' is not in dataset")

print("\nMembership Operator (in):")
if 'hello' in dataset:
    print("'hello' is in dataset")
else:
    print("'hello' is not in dataset")
```

Output:

```
<main.py>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?  
Enter dataset (comma-separated values): Hello,Good Morning  
  
Identity Operator (is):  
'hello' is not in dataset  
  
Membership Operator (in):  
'hello' is not in dataset  
  
=== Code Execution Successful ===
```

Problem Sheet - 2 BSCIT- SEM 4 Programming with Python (1020406404)

1. Write a Python Programming to print data type of different variables .

Input:

```
var_int = 10
```

```

var_float = 3.14
var_str = "Hello, World!"
var_list = [1, 2, 3, 4, 5]
var_tuple = (1, 2, 3, 4, 5)
var_dict = {"a": 1, "b": 2, "c": 3}
var_set = {1, 2, 3, 4, 5}
var_bool = True
print("Data types of different variables:")
print("var_int:", type(var_int))
print("var_float:", type(var_float))
print("var_str:", type(var_str))
print("var_list:", type(var_list))
print("var_tuple:", type(var_tuple))
print("var_dict:", type(var_dict))
print("var_set:", type(var_set))
print("var_bool:", type(var_bool))

```

```

Data types of different variables:
var_int: <class 'int'>
var_float: <class 'float'>
var_str: <class 'str'>
var_list: <class 'list'>
var_tuple: <class 'tuple'>
var_dict: <class 'dict'>
var_set: <class 'set'>
var_bool: <class 'bool'>

=== Code Execution Successful ===

```

Output:

2. Write a Python Programming to create List and accessing each item form index .

Input: my_list = ['apple', 'banana', 'cherry', 'date', 'elderberry']
print("Accessing each item from index:")
for i in range(len(my_list)):
 print("Index", i, ":", my_list[i])

```
Accessing each item from index:
Index 0 : apple
Index 1 : banana
Index 2 : cherry
Index 3 : date
Index 4 : elderberry

=== Code Execution Successful ===
```

Output:

3. Write a Python Programming to create Tuple and accessing each item form index .

Input: my_tuple = ('apple', 'banana', 'cherry', 'date', 'elderberry')
print("Accessing each item from index:")
for i in range(len(my_tuple)):
 print("Index", i, ":", my_tuple[i])

```
Accessing each item from index:
Index 0 : apple
Index 1 : banana
Index 2 : cherry
Index 3 : date
Index 4 : elderberry

=== Code Execution Successful ===
```

Output:

4. Write a Python Programming to create Set and accessing each item form index .

Input: my_set = {'apple', 'banana', 'cherry', 'date', 'elderberry'}
print("Accessing each item from the set:")
for item in my_set:
 print(item)

```
Accessing each item from the set:
cherry
elderberry
banana
date
apple
```

Output:

```
=== Code Execution Successful ===
```

5. Write a Python Programming to create Dictionary and accessing each item form index .

Input: my_dict = {'name': 'John', 'age': 30, 'city': 'New York', 'country': 'USA'}
print("Accessing each item from the dictionary:")
for key in my_dict:
 print("Key:", key, "\tValue:", my_dict[key])

```
Accessing each item from the dictionary:
Key: name      Value: John
Key: age       Value: 30
Key: city      Value: New York
Key: country   Value: USA

=== Code Execution Successful ===
```

Output:

6. Write a Python Programming to implement global and local variable .

Input:
global_var = 10
def my_function():
 local_var = 20
 print("Inside the function:")
 print("Global variable:", global_var)
 print("Local variable:", local_var)
print("Outside the function:")
print("Global variable:", global_var)
print("Local variable:", local_var)
my_function()

```
Outside the function:
Global variable: 10
Inside the function:
Global variable: 10
Local variable: 20

=== Code Execution Successful ===
```

Output:

7. Write a Python Programming to perform assignment operator using different variable declaration.

Input: # Integer assignment

a = 10

b = 5

a += b

print("Addition assignment (+=):", a) # Output: 15

a -= b

print("Subtraction assignment (-=):", a) # Output: 10

a *= b

print("Multiplication assignment (*=):", a) # Output: 50

a /= b

print("Division assignment (/=):", a) # Output: 10.0

a %= b

print("Modulus assignment (%=):", a) # Output: 0.0

a **= b

print("Exponentiation assignment (**=):", a) # Output: 0.0

a //= b

print("Floor division assignment (//=):", a) # Output: 0.0

```
Addition assignment (+=): 15
Subtraction assignment (-=): 10
Multiplication assignment (*=): 50
Division assignment (/=): 10.0
Modulus assignment (%=): 0.0
Exponentiation assignment (**=): 0.0
Floor division assignment (//=): 0.0

=== Code Execution Successful ===
```

Output:

8. Write a Python Programming to invoke string data type in python and also create multi line string ,than access first and last characters of the string .

Input: my_string = "Hello, World!"

```
print("String:", my_string)
```

```
multi_line_string = """This is a
multi-line
string."""
```

```
print("\nMulti-line String:")
```

```
print(multi_line_string)
```

```
first_character = my_string[0]
```

```
last_character = my_string[-1]
```

```
print("\nFirst character of the string:", first_character)
```

```
print("Last character of the string:", last_character)
```

```
String: Hello, World!

Multi-line String:
This is a
multi-line
string.

First character of the string: H
Last character of the string: !

=== Code Execution Successful ===
```

Output:

9. Write a Python Programming to create multi-dimensional List and access negative index from the List

Input:

```
multi_dimensional_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
print("Accessing elements using negative indices:")
print("Last element of the first sublist:", multi_dimensional_list[0][-1])
print("Second-to-last element of the second sublist:", multi_dimensional_list[1][-2])
print("First element of the last sublist:", multi_dimensional_list[-1][0])
```

```
Accessing elements using negative indices:
Last element of the first sublist: 3
Second-to-last element of the second sublist: 5
First element of the last sublist: 7

=== Code Execution Successful ===
```

Output:

10. Write a Python Programming to use of tuple function and declare distinct nested tuples of your choice(eg tuple1,tuple2) and print all nested tuple into one common tuple(eg tuple 3) .

Input:

```
tuple1 = (1, 2, 3)
tuple2 = ('a', 'b', 'c')
tuple3 = (True, False)
```



```
combined_tuple = tuple(tuple1 + tuple2 + tuple3)
print("Combined Tuple:", combined_tuple)\
```

Output:

```
Combined Tuple: (1, 2, 3, 'a', 'b', 'c', True, False)

=== Code Execution Successful ===
```

11. Write a Python Programming to access elements from dictionary using get and key.

Input:

```
# Define a dictionary
my_dict = {'name': 'John', 'age': 30, 'city': 'New York', 'country': 'USA'}

# Access elements using the get() method
print("Accessing elements using the get() method:")
print("Name:", my_dict.get('name'))
print("Age:", my_dict.get('age'))
print("Gender:", my_dict.get('gender')) # Returns None if key not found

# Access elements directly using keys
print("\nAccessing elements directly using keys:")
print("City:", my_dict['city'])
print("Country:", my_dict['country'])
# Attempting to access a key that doesn't exist will raise a KeyError
# print("Occupation:", my_dict['occupation'])
```

```
Accessing elements using the get() method:
Name: John
Age: 30
Gender: None

Accessing elements directly using keys:
City: New York
Country: USA

=== Code Execution Successful ===
```

Output:

12. Write a Python Programming to create set of mixed values and access each elements form the set.

Input: mixed_set = {1, 'apple', 3.14, True, (4, 5)}

```
print("Accessing each element from the set:")
for element in mixed_set:
    print(element)
```

```
Accessing each element from the set:
1
(4, 5)
3.14
apple

=== Code Execution Successful ===
```

Output:

13. Write a Python Programming to use list with set ,tuple,dictionary and print the output .

Input:

```
mixed_list = [
    {'name': 'John', 'age': 30},
    (1, 2, 3),
    {4, 5, 6}
]
print("List containing a set, tuple, and dictionary:")
for item in mixed_list:
    print(item)
```

Output:

```
List containing a set, tuple, and dictionary:
{'name': 'John', 'age': 30}
(1, 2, 3)
{4, 5, 6}

=== Code Execution Successful ===
```

14. Write a Python Programming to create dictionary and perform the following:

- a) Empty Dictionary
- b) Dictionary with the use of Integer Keys
- c) Dictionary with the use of Mixed Keys
- d) Dictionary with the use of dict()
- e) Dictionary with each item as a pair

f) Accessing a element using key

g) Accessing a element using get

Input:

```
empty_dict = { }
```

```
print("Empty Dictionary:", empty_dict)
```

```
integer_dict = { 1: 'one', 2: 'two', 3: 'three' }
```

```
print("\nDictionary with Integer Keys:", integer_dict)
```

```
mixed_dict = { 'name': 'John', 1: 'one', (2, 3): 'tuple_key' }
```

```
print("\nDictionary with Mixed Keys:", mixed_dict)
```

```
dict_constructor = dict([('a', 1), ('b', 2), ('c', 3)])
```

```
print("\nDictionary with the use of dict():", dict_constructor)
```

```
pair_dict = dict(a=1, b=2, c=3)
```

```
print("\nDictionary with each item as a pair:", pair_dict)
```

```
print("\nAccessing an element using key 'name':", mixed_dict['name'])
```

```
print("Accessing an element using get('name'):", mixed_dict.get('name'))
```

Output:

```
Empty Dictionary: { }
```

```
Dictionary with Integer Keys: {1: 'one', 2: 'two', 3: 'three' }
```

```
Dictionary with Mixed Keys: { 'name': 'John', 1: 'one', (2, 3): 'tuple_key' }
```

```
Dictionary with the use of dict(): { 'a': 1, 'b': 2, 'c': 3 }
```

```
Dictionary with each item as a pair: { 'a': 1, 'b': 2, 'c': 3 }
```

```
Accessing an element using key 'name': John
```

```
Accessing an element using get('name'): John
```

```
=== Code Execution Successful ===
```

15. Write a python programme numeric and imaginary data type in python using variables.

Input:

```
integer_var = 10
```

```
float_var = 3.14
```

```
complex_var = 2 + 3j
```

```
# Print numeric variables
```

```
print("Integer Variable:", integer_var)
```

```
print("Float Variable:", float_var)
```

```
print("Complex Variable:", complex_var)
```

```
# Accessing real and imaginary parts of a complex number
```

```
print("\nReal Part of Complex Variable:", complex_var.real)
```

```
print("Imaginary Part of Complex Variable:", complex_var.imag)\
```

Output:

```
Integer Variable: 10
Float Variable: 3.14
Complex Variable: (2+3j)

Real Part of Complex Variable: 2.0
Imaginary Part of Complex Variable: 3.0

=== Code Execution Successful ===
```

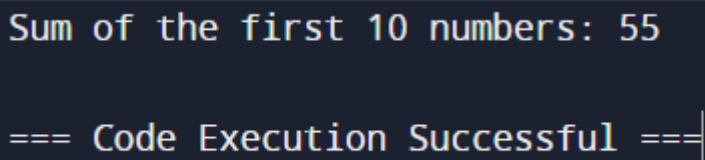
Problem Sheet – 3 BSCIT- SEM 4 Programming with Python (1020406404)

1. Write a Python program to perform sum of first 10 numbers using range function.

Input:

```
numbers = range(1, 11)
sum_of_numbers = sum(numbers)
print("Sum of the first 10 numbers:", sum_of_numbers)
```

Output:



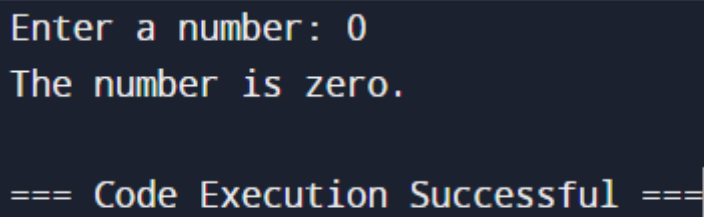
```
Sum of the first 10 numbers: 55
=== Code Execution Successful ===
```

2. Write a Python program to illustrate if statement .

Input:

```
number = float(input("Enter a number: "))
if number == 0:
    print("The number is zero.")
```

Output:



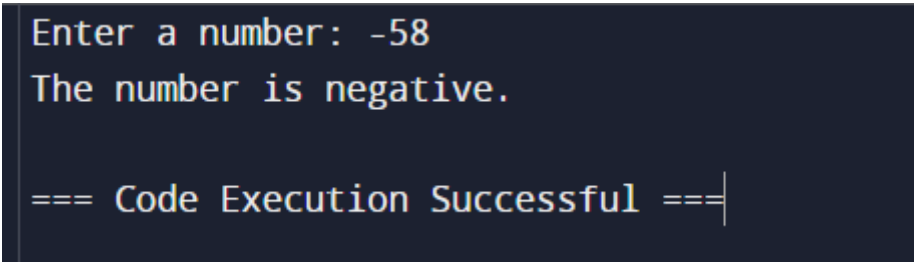
```
Enter a number: 0
The number is zero.
=== Code Execution Successful ===
```

3. Write a Python program to illustrate if -else statement .

Input:

```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

Output:



```
Enter a number: -58
The number is negative.

=== Code Execution Successful ===
```

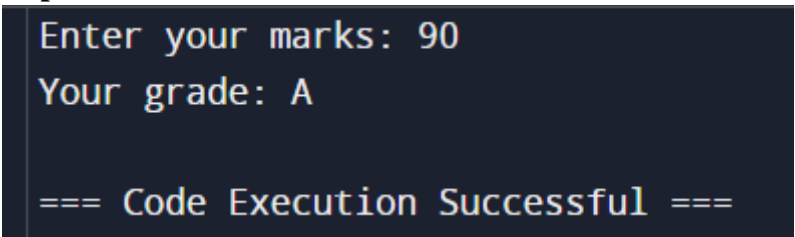
4. Write a Python program to illustrate if-elif-else statement .

Input:

```
marks = float(input("Enter your marks: "))
if marks >= 90:
    grade = 'A'
elif marks >= 80:
    grade = 'B'
elif marks >= 70:
    grade = 'C'
elif marks >= 60:
    grade = 'D'
else:
    grade = 'F'

print("Your grade:", grade)
```

Output:



```
Enter your marks: 90
Your grade: A

=== Code Execution Successful ===
```

5. Write a Python program to illustrate for loop using break,pass,continue.

Input:

```
print("Example 1: Using break")
```

```
for i in range(1, 6):
```

```
    if i == 3:
```

```
        break # exit the loop when i equals 3
```

```
    print(i)
```

```
print("\nExample 2: Using pass")
```

```
for i in range(1, 6):
```

```
    if i == 3:
```

```
        pass # do nothing when i equals 3
```

```
    else:
```

```
        print(i)
```

```
print("\nExample 3: Using continue")
```

```
for i in range(1, 6):
```

```
    if i == 3:
```

```
        continue # skip the rest of the loop and continue with the next iteration when i equals 3
```

```
    print(i)
```

Output:

```
Example 1: Using break
```

```
1
```

```
2
```

```
Example 2: Using pass
```

```
1
```

```
2
```

```
4
```

```
5
```

```
Example 3: Using continue
```

```
1
```

```
2
```

```
4
```

```
5
```

```
=== Code Execution Successful ===
```

6. Write a Python program to demonstrate for-else loop .

Input:

start = 10

end = 20

```
for num in range(start, end + 1):
    if num > 1:
        for i in range(2, num):
            if (num % i) == 0:
                break
        else:
            print(num, "is a prime number.")
else:
    print("No prime numbers found in the range from", start, "to", end)
```

Output:

```
11 is a prime number.
13 is a prime number.
17 is a prime number.
19 is a prime number.
No prime numbers found in the range from 10 to 20

=== Code Execution Successful ===
```


Problem Sheet – 4 BSCIT- SEM 4 Programming with Python (1020406404)

1. Write a Python program to take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero. Find total number of numbers entered and their sum. Display count and sum with appropriate titles.

Input:

```
count = 0
```

```
total_sum = 0
```

```
while True:
```

```
    num = float(input("Enter a non-zero number (enter 0 to stop): "))
```

```
    if num == 0:
```

```
        break
```

```
    count += 1
```

```
    total_sum += num
```

Output:

```
Enter a non-zero number (enter 0 to stop): 45
Enter a non-zero number (enter 0 to stop): 21
Enter a non-zero number (enter 0 to stop): 0

=== Code Execution Successful ===
```

```
print("\nTotal number of numbers entered:", count)
print("Sum of the numbers entered:", total_sum)
```

2. Write a Python program to take input of positive numbers, with an appropriate prompt, from the user until the user enters a zero. Find total number of odd & even numbers entered and sum of odd and even numbers. Display total count of odd & even numbers and sum of odd & even numbers with appropriate titles.

Input:

```
even_count = 0
odd_count = 0
even_sum = 0
odd_sum = 0
```

```
while True:
```

```
    num = float(input("Enter a positive number (enter 0 to stop): "))
```

```
    if num == 0:
        break
```

```
    if num > 0:
        if num % 2 == 0:
            even_count += 1
            even_sum += num
        else:
            odd_count += 1
            odd_sum += num
```

```
print("\nTotal number of even numbers entered:", even_count)
print("Sum of even numbers entered:", even_sum)
print("\nTotal number of odd numbers entered:", odd_count)
print("Sum of odd numbers entered:", odd_sum)
```

Output:

```
Enter a positive number (enter 0 to stop): 25
Enter a positive number (enter 0 to stop): 21
Enter a positive number (enter 0 to stop): 58
Enter a positive number (enter 0 to stop): 68
Enter a positive number (enter 0 to stop): 0
```

```
Total number of even numbers entered: 2
```

```
Sum of even numbers entered: 126.0
```

```
Total number of odd numbers entered: 2
```

```
Sum of odd numbers entered: 46.0
```

```
=== Code Execution Successful ===
```

3. Write a Python program to take input of a positive number, with an appropriate prompt, from the user. The user should be prompted again to enter the number until the user enters a positive number. Check whether the number is a prime number or not and accordingly display appropriate message

Input:

```
def is_prime(n):
    """Function to check if a number is prime"""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

while True:
    num = int(input("Enter a positive number: "))

    if num > 0:
        break
    else:
        print("Please enter a positive number.")

if is_prime(num):
    print(num, "is a prime number.")
else:
    print(num, "is not a prime number.")
```

Output:

```
Enter a positive number: 10
10 is not a prime number.
```

```
=== Code Execution Successful ===
```

4. Write a Python program to take input of a positive number, say N, with an appropriate prompt, from the user. The user should be prompted again to enter the number until the user enters a positive number. Find the sum of first N odd numbers and first N even numbers. Display both the sums with appropriate titles.

Input:

```
def sum_of_first_N_odd(N):
    return N**2
```

```
def sum_of_first_N_even(N):
    return N * (N + 1)
```

```
while True:
```

```
    N = int(input("Enter a positive number (N): "))
```

```
    if N > 0:
```

```
        break
```

```
    else:
```

```
        print("Please enter a positive number.")
```

```
sum_of_odd = sum_of_first_N_odd(N)
```

```
sum_of_even = sum_of_first_N_even(N)
```

```
print("\nSum of first", N, "odd numbers:", sum_of_odd)
```

```
print("Sum of first", N, "even numbers:", sum_of_even)
```

```
Enter a positive number (N): 35

Sum of first 35 odd numbers: 1225
Sum of first 35 even numbers: 1260

=== Code Execution Successful ===
```

Output:

5. Consider a list of numbers. Write a Python program to do the following:

- 1) Count total number of numbers in the list
- 2) Sum and Average of all the numbers in the list
- 3) Count and sum of all the odd numbers in the list
- 4) Count and sum of all the even numbers in the list
- 5) Find the largest number in the list
- 6) Find the smallest number in the list Display all the values with appropriate titles.

Input:

numbers = [23, 45, 12, 67, 89, 34, 56, 78, 91, 10]

```
total_numbers = len(numbers)
total_sum = sum(numbers)
average = total_sum / total_numbers
odd_numbers = [num for num in numbers if num % 2 != 0]
count_odd = len(odd_numbers)
sum_odd = sum(odd_numbers)
even_numbers = [num for num in numbers if num % 2 == 0]
count_even = len(even_numbers)
sum_even = sum(even_numbers)
largest = max(numbers)
smallest = min(numbers)
print("1) Total number of numbers in the list:", total_numbers)
print("2) Sum of all the numbers in the list:", total_sum)
print("   Average of all the numbers in the list:", average)
print("3) Count of all the odd numbers in the list:", count_odd)
print("   Sum of all the odd numbers in the list:", sum_odd)
print("4) Count of all the even numbers in the list:", count_even)
print("   Sum of all the even numbers in the list:", sum_even)
print("5) Largest number in the list:", largest)
print("6) Smallest number in the list:", smallest)
```

Output:

```
1) Total number of numbers in the list: 10
2) Sum of all the numbers in the list: 505
   Average of all the numbers in the list: 50.5
3) Count of all the odd numbers in the list: 5
   Sum of all the odd numbers in the list: 315
4) Count of all the even numbers in the list: 5
   Sum of all the even numbers in the list: 190
5) Largest number in the list: 91
6) Smallest number in the list: 10

=== Code Execution Successful ===
```

6. Consider a list of characters (characters may be alphabets, special characters, digits). Write a Python program to do the following:
- 1) Count total number of elements in the list
 - 2) Count total number of vowels in the list (vowels are 'a', 'e', 'i', 'o', 'u')
 - 3) Count total number of consonants in the list (a consonant is an alphabet other than vowel)
 - 4) Count total number of characters other than vowels and consonants Display all the values with appropriate titles.

Input:

```
characters = ['a', 'b', '5', '!', 'e', 'i', '9', 'o', 'u', '$', 'z', 'x', '1', '2', '&', '%']
```

```
vowels = ['a', 'e', 'i', 'o', 'u']
```

```
total_elements = len(characters)
```

```
count_vowels = sum(1 for char in characters if char.lower() in vowels)
```

```
count_consonants = sum(1 for char in characters if char.isalpha() and char.lower() not in vowels)
```

```
count_other = total_elements - count_vowels - count_consonants
```

```
print("1) Total number of elements in the list:", total_elements)
```

```
print("2) Total number of vowels in the list:", count_vowels)
```

```
print("3) Total number of consonants in the list:", count_consonants)
```

```
print("4) Total number of characters other than vowels and consonants:", count_other)
```

Output:

```
1) Total number of elements in the list: 16
2) Total number of vowels in the list: 5
3) Total number of consonants in the list: 3
4) Total number of characters other than vowels and consonants: 8

=== Code Execution Successful ===
```

7. Consider a single list consisting of integer values, float values, character values, string values and lists. Write a Python program to do the following:
- 1) Count total number of elements in the list
 - 2) Count total number of integer values, float values, character values, string values and lists Display all the values with appropriate titles.

Input:

```
# Single list consisting of various types of elements
```

```
mixed_list = [1, 3.14, 'a', "hello", [5, 6, 7], 9, 'b', 2.718, ["apple", "banana"], 'c']
```

```
# Initialize counters
```

```
count_total = 0
```

```
count_integers = 0
```

```
count_floats = 0
count_characters = 0
count_strings = 0
count_lists = 0

# Iterate over the elements in the list
for element in mixed_list:
    # Increment total count
    count_total += 1

    # Check type of element and increment respective counters
    if isinstance(element, int):
        count_integers += 1
    elif isinstance(element, float):
        count_floats += 1
    elif isinstance(element, str):
        if len(element) == 1 and element.isalpha():
            count_characters += 1
        else:
            count_strings += 1
    elif isinstance(element, list):
        count_lists += 1

# Displaying all the values with appropriate titles
print("1) Total number of elements in the list:", count_total)
print("2) Total number of integer values:", count_integers)
print("3) Total number of float values:", count_floats)
print("4) Total number of character values:", count_characters)
print("5) Total number of string values:", count_strings)
print("6) Total number of list values:", count_lists)
```

Output:

```
1) Total number of elements in the list: 10
2) Total number of integer values: 2
3) Total number of float values: 2
4) Total number of character values: 3
5) Total number of string values: 1
6) Total number of list values: 2

=== Code Execution Successful ===
```

Problem Sheet - 5 BSCIT- SEM 4 Programming with Python (1020406404)

1. Write a Python program to create a list of numbers by taking input from the user and then remove the duplicates from the list. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero. Sample Input: [10, 34, 18, 10, 12, 34, 18, 20, 25, 20] Output: [10, 34, 18, 12, 20, 25]

Input:

```
numbers = []
while True:
    num = int(input("Enter a non-zero number (enter 0 to stop): "))

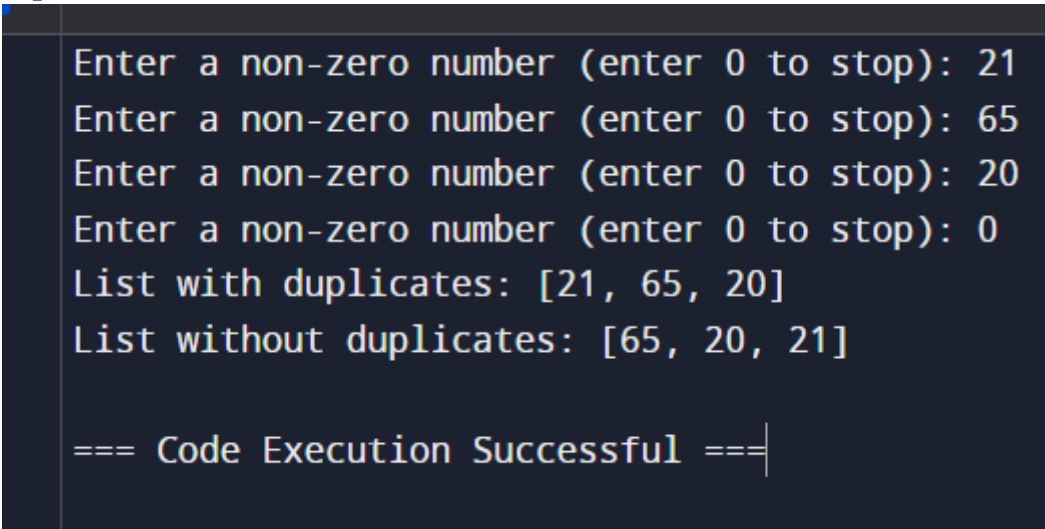
    if num == 0:
        break

    numbers.append(num)

numbers_without_duplicates = list(set(numbers))

print("List with duplicates:", numbers)
print("List without duplicates:", numbers_without_duplicates)
```

Output:



```
Enter a non-zero number (enter 0 to stop): 21
Enter a non-zero number (enter 0 to stop): 65
Enter a non-zero number (enter 0 to stop): 20
Enter a non-zero number (enter 0 to stop): 0
List with duplicates: [21, 65, 20]
List without duplicates: [65, 20, 21]

=== Code Execution Successful ===
```

2. Write a Python program to create a list of lists of numbers (i.e. each element of the list is a list of numbers e.g. [[1, 2], [3, 2, 5], [1], [5, 3, 6, 7]]). Then generate a list from the given list where each element of the list is the
- RNWIET/Bsc.IT/SEM-4/PYTHON-9761

length of each list in the given list. i.e. for the given example the output should be [2, 3, 1, 4]. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero.

Sample Input: [[1, 2], [3, 2, 5], [1], [5, 3, 6, 7]]

Output: [2, 3, 1, 4]

Input:

```
list_of_lists = []
```

```
while True:
```

```
    nums = input("Enter a list of numbers separated by spaces (enter 0 to stop): ")
```

```
    if nums == '0':
```

```
        break
```

```
    nums_list = [int(num) for num in nums.split()]
```

```
    list_of_lists.append(nums_list)
```

```
lengths_list = [len(lst) for lst in list_of_lists]
```

```
print("List of lists of numbers:", list_of_lists)
```

```
print("Lengths list:", lengths_list)
```

Output:

```
Enter a list of numbers separated by spaces (enter 0 to stop): 40
Enter a list of numbers separated by spaces (enter 0 to stop): 50
Enter a list of numbers separated by spaces (enter 0 to stop): 0
List of lists of numbers: [[40], [50]]
Lengths list: [1, 1]
```

```
=== Code Execution Successful ===
```

3. Write a Python program to create a list of numbers by taking input from the user. Split this list into two tuples where one tuple contains odd numbers and the other tuple contains even numbers from the list. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the list assuming that the numbers are non-zero.

Sample Input: [10, 12, 13, 90, 43, 32, 30, 11]

Output: Tuple of Odd Numbers: (13, 43, 11)

 Tuple of Even Numbers: (10, 12, 90, 32, 30)

Input:

```
odd_numbers = []
```

```
even_numbers = []
```

```
while True:
```

```
    num = int(input("Enter a non-zero number (enter 0 to stop): "))
```

```
    if num == 0:
```

```
        break
```

```
    if num % 2 == 0:
```

```
        even_numbers.append(num)
```

```
    else:
```

```
        odd_numbers.append(num)
```

```
odd_tuple = tuple(odd_numbers)
```

```
even_tuple = tuple(even_numbers)
```

```
print("Tuple of Odd Numbers:", odd_tuple)  
print("Tuple of Even Numbers:", even_tuple)
```

Output:

```
Enter a non-zero number (enter 0 to stop): 20  
Enter a non-zero number (enter 0 to stop): 0  
Tuple of Odd Numbers: ()  
Tuple of Even Numbers: (20,)  
  
=== Code Execution Successful ===
```

4. Write a Python program to create a list of elements of any data type (mixed data type, i.e. some elements maybe of type int, some elements of type float and some elements of type string). Split this list into three tuples containing elements of same data type (i.e. 1st tuple of integers only, 2nd tuple of float only and 3rd tuple of strings only). Take input from the user to create the list.

Sample Input: [25, 4.5, 'Hello', 90, 20, 7.5, 9.25, 'World']

Output: List of Integers: [25, 90, 20]

List of Float Values: [4.5, 7.5, 9.25]

List of Strings: ['Hello', 'World']

Input:

```
integers = []
```

```
floats = []
```

```
strings = []
```

```
while True:
```

```
    value = input("Enter a value (enter 0 to stop): ")
```

```
    if value == '0':
```

```
        break
```

```
    try:
```

```
        value = eval(value)
```

```
        if isinstance(value, int):
```

```
            integers.append(value)
```

```
        elif isinstance(value, float):
```

```
            floats.append(value)
```

```
        elif isinstance(value, str):
```

```
            strings.append(value)
```

```
    else:
```

```
        print("Invalid input! Please enter an integer, float, or string.")
```

```
except:
    print("Invalid input! Please enter an integer, float, or string.")
```

```
integers_tuple = tuple(integers)
floats_tuple = tuple(floats)
strings_tuple = tuple(strings)
```

```
print("List of Integers:", integers_tuple)
print("List of Float Values:", floats_tuple)
print("List of Strings:", strings_tuple)
```

Output:

```
Enter a value (enter 0 to stop): 20
Enter a value (enter 0 to stop): 49
Enter a value (enter 0 to stop): 0
List of Integers: (20, 49)
List of Float Values: ()
List of Strings: ()

=== Code Execution Successful ===
```

5. Write a Python program to create a dictionary of student data by taking input from the user, where each student data contains Rollno (to be considered as key), and marks in 3 subjects (to be considered as list of values). e.g. { 1 : [45, 40, 35], 2 : [41, 38, 39], 3 : [35, 30, 37]}. Prepare mark list in the following format:

Roll No.	Mark-1	Mark-2	Mark-3	Total
1	45	40	35	120

Input:

```
def calculate_total(marks):
    return sum(marks)

def format_mark_list(student_data):
    print("Roll No.\tMark-1\tMark-2\tMark-3\tTotal")
    for rollno, marks in student_data.items():
        total_marks = calculate_total(marks)
        print(f"{rollno}\t{marks[0]}\t{marks[1]}\t{marks[2]}\t{total_marks}")
    student_data = { }

while True:
    rollno = int(input("Enter Roll No. (enter 0 to stop): "))
    if rollno == 0:
```

```

        break
marks = [int(x) for x in input("Enter marks in 3 subjects separated by space: ").split()]
student_data[rollno] = marks

format_mark_list(student_data)

```

Output:

```

Enter Roll No. (enter 0 to stop): 1
Enter marks in 3 subjects separated by space: 39 45 50
Enter Roll No. (enter 0 to stop): 2
Enter marks in 3 subjects separated by space: 34 46 47
Enter Roll No. (enter 0 to stop): 3
Enter marks in 3 subjects separated by space: 47 50 49
Enter Roll No. (enter 0 to stop): 0
Roll No.      Mark-1  Mark-2  Mark-3  Total
1             39      45      50      134
2             34      46      47      127
3             47      50      49      146

=== Code Execution Successful ===

```

6. Write a Python program to take input of a string from the user and then create a dictionary containing each character of the string along with their frequencies. (e.g. if the string is 'banana' then output should be {'b': 1, 'a': 3, 'n': 2}).

Input:

```
input_string = input("Enter a string: ")
```

```
char_freq = {}
```

```

for char in input_string:
    if char in char_freq:

```

```

        char_freq[char] += 1
    else:
        char_freq[char] = 1
print("Dictionary containing character frequencies:", char_freq)

```

Output:

```

Enter a string: "Hello World"
Dictionary containing character frequencies: {' ': 2, 'H': 1, 'e': 1, 'l': 3, 'o': 2,
' ': 1, 'W': 1, 'r': 1, 'd': 1}

=== Code Execution Successful ===

```

7. Write a Python program to create a list of strings by taking input from the user and then create a dictionary containing each string along with their frequencies. (e.g. if the list is ['apple', 'banana', 'fig', 'apple', 'fig', 'banana', 'grapes', 'fig', 'grapes', 'apple'] then output should be {'apple': 3, 'banana': 2, 'fig': 3, 'grapes': 2}).

Input:

```

input_string = input("Enter a string: ")
char_freq = {}

for char in input_string:
    if char in char_freq:
        char_freq[char] += 1
    else:
        char_freq[char] = 1
print("Dictionary containing character frequencies:", char_freq)

```

Output:

```

Enter a string: Good Morning
Dictionary containing character frequencies: {'G': 1, 'o': 3, 'd': 1, ' ': 1, 'M': 1,
'r': 1, 'n': 2, 'i': 1, 'g': 1}

=== Code Execution Successful ===

```

8. Write a Python program to input a string that is a list of words separated by commas. Construct a dictionary that contains all these words as keys and their frequencies as values. Then display the words with their quantities.

Input:

```

input_string = input("Enter a list of words separated by commas: ")

words_list = input_string.split(',')

word_freq = {}

```

```

for word in words_list:
    word = word.strip()
    if word in word_freq:
        word_freq[word] += 1
    else:
        word_freq[word] = 1

print("Words with their frequencies:")
for word, freq in word_freq.items():
    print(f'{word}: {freq}')

```

Output:

```

Enter a list of words separated by commas: Apple,Banana,Apple,Mango
Words with their frequencies:
Apple: 2
Banana: 1
Mango: 1

=== Code Execution Successful ===

```

9. Consider a very small dictionary that contains the translations of English words to Dutch as shown below:
- ```

english_dutch = { "last" : "laatst", "week" : "week", "the" : "de", "royal" : "koninklijk", "festival" : "feest",
"hall" : "hal", "saw" : "zaag", "first" : "eerst", "performance" : "optreden", "of" : "van", "a" : "een", "new" :
"nieuw", "symphony" : "symphonie", "by" : "bij", "one" : "een", "world" : "wereld", "leading" : "leidend",
"modern" : "modern", "composer" : "componist", "composers" : "componisten", "two" : "twee", "shed" :
"schuur", "sheds" : "schuren" }

```

Write a program that uses this dictionary to create a word-for-word translation of a sentence to be taken as an input from the user. A word for which you cannot find a translation, you can leave “as is.” The dictionary is supposed to be used case-insensitively, but your translation may consist of all lower-case words. It is nice if you leave punctuation in the translation, but if you take it out, that is acceptable (as leaving punctuation in is

quite a bit of work, and does not really have anything to do with dictionaries – besides, leaving punctuation in is much easier to do once you have learned about regular expressions).

### Input:

```
english_dutch = {
 "last": "laatst", "week": "week", "the": "de", "royal": "koninklijk", "festival": "feest",
 "hall": "hal", "saw": "zaag", "first": "eerst", "performance": "optreden", "of": "van",
 "a": "een", "new": "nieuw", "symphony": "symphonie", "by": "bij", "one": "een", "world": "wereld",
 "leading": "leidend", "modern": "modern", "composer": "componist", "composers": "componisten", "two":
 "twee", "shed": "schuur", "sheds": "schuren" }

sentence = input("Enter a sentence to translate to Dutch: ")
words = sentence.split()
translated_sentence = []
for word in words:
 translated_word = english_dutch.get(word.lower(), word) # Use get() to handle missing translations
 translated_sentence.append(translated_word)
translated_sentence = ' '.join(translated_sentence)
print("Translated sentence:", translated_sentence)
```

### Output:

```
Enter a sentence to translate to Dutch: "the last week was the royal festival"

Translated sentence: "the laatst week was de koninklijk festival"

=== Code Execution Successful ===
```

10. Consider the following list of movies. For each movie it also shows list of ratings. Write a Python program to convert it in such a way that it stores all this data in one dictionary, then use the dictionary to print the average rating for each movie, rounded to one decimal.

```
movies = ["Where Eagle's Dare", "Enter the Dragon", "Iron Fist", "Fist of Fury"]
dare_ratings = [9, 10, 9.5, 8.5, 3, 7.5, 8] for the movie "Where Eagle's Dare"
dragon_ratings = [10, 10, 0, 9, 1, 8, 7.5, 8, 6, 9] for the movie "Enter the Dragon"
fist_ratings = [7, 6, 5] for the movie "Iron Fist"
fury_ratings = [6, 5, 6, 6] for the movie "Fist of Fury"
```

### Input:

```
movies = ["Where Eagle's Dare", "Enter the Dragon", "Iron Fist", "Fist of Fury"]
dare_ratings = [9, 10, 9.5, 8.5, 3, 7.5, 8]
dragon_ratings = [10, 10, 0, 9, 1, 8, 7.5, 8, 6, 9]
fist_ratings = [7, 6, 5]
fury_ratings = [6, 5, 6, 6]
movie_ratings = {
 movies[0]: dare_ratings,
```

```
movies[1]: dragon_ratings,
movies[2]: fist_ratings,
movies[3]: fury_ratings
}
for movie, ratings in movie_ratings.items():
 average_rating = round(sum(ratings) / len(ratings), 1)
 print(f"Average ratings of the movie \"{movie}\" is {average_rating}")
```

### Output:

```
Average ratings of the movie "Where Eagle's Dare" is 7.9
Average ratings of the movie "Enter the Dragon" is 6.8
Average ratings of the movie "Iron Fist" is 6.0
Average ratings of the movie "Fist of Fury" is 5.8

=== Code Execution Successful ===
```

S



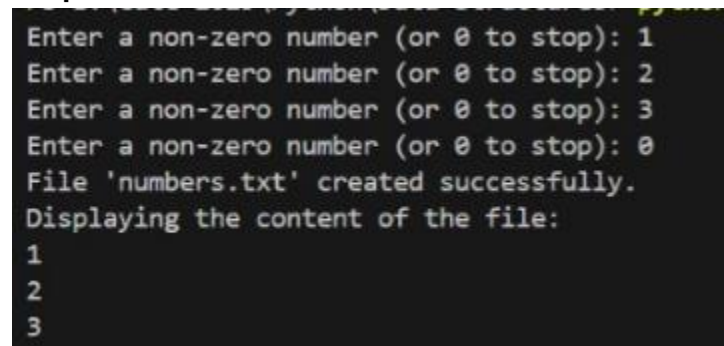
## Problem Sheet -6 BSCIT- SEM 4 Programming with Python (1020406404)

**Q.1.** Write a Python program to create a file of numbers by taking input from the user and then display the content of the file. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the file assuming that the numbers are non-zero.

```
def create_and_display_file():
 file_name = "numbers.txt"
 with open(file_name, "w") as file:
 while True:
 user_input = input("Enter a non-zero number (or 0 to stop): ")
 if user_input == "0":
 break
 file.write(user_input + "\n")
 print(f"File '{file_name}' created successfully.")
 print("Displaying the content of the file:")
 with open(file_name, "r") as file:
 print(file.read())
```

create\_and\_display\_file()

**Output:**



```
Enter a non-zero number (or 0 to stop): 1
Enter a non-zero number (or 0 to stop): 2
Enter a non-zero number (or 0 to stop): 3
Enter a non-zero number (or 0 to stop): 0
File 'numbers.txt' created successfully.
Displaying the content of the file:
1
2
3
```

**Q.2.** Write a Python program to create a text file of multiple lines. Display the following: 1. Entire text file 2. 1 st n lines of the text tile. 3. m lines starting from the n th line 4. number of words in each line

```
def create_and_display_file():
 # 1
 file_name = "example.txt"
 with open(file_name, "w") as file:
 for i in range(10):
 file.write(f"Line {i+1} - This is line number {i+1}.\n")
```

```
print(f"File '{file_name}' created successfully.")
print("Displaying the content of the file:")
with open(file_name, "r") as file:
 print(file.read())
```

**# 2**

```
print("\nDisplaying the 1st line of the text file:")
with open(file_name, "r") as file:
 print(file.readline().strip())
```

```
print("\nDisplaying the 5th line of the text file:")
with open(file_name, "r") as file:
 for i in range(4):
 file.readline()
 print(file.readline().strip())
```

**# 3**

```
n=int(input("Enter nth line: "))
m=int(input("Enter m lines from the nth lines: "))
```

```
print("\nDisplaying the nth to mth lines of the text file:")
with open(file_name, "r") as file:
 for i in range(n-1):
 file.readline()
 for line in range(m-n+1):
 print(file.readline().strip())
```

**# 4**

```
print("\nDisplaying the number of words in each line:")
with open(file_name, "r") as file:
 for line in file:
 words = line.strip().split()
 print(f"Line: {line.strip()}, Number of words: {len(words)}")
```

```
create_and_display_file()
```

### Output:

```
Displaying the content of the file:
Line 1 - This is line number 1.
Line 2 - This is line number 2.
Line 3 - This is line number 3.
Line 4 - This is line number 4.
Line 5 - This is line number 5.
Line 6 - This is line number 6.
Line 7 - This is line number 7.
Line 8 - This is line number 8.
Line 9 - This is line number 9.
Line 10 - This is line number 10.

Displaying the 1st line of the text file:
Line 1 - This is line number 1.

Displaying the 5th line of the text file:
Line 5 - This is line number 5.
Enter nth line: 4
Enter m lines from the nth lines: 8

Displaying the nth to mth lines of the text file:
Line 4 - This is line number 4.
Line 5 - This is line number 5.
Line 6 - This is line number 6.
Line 7 - This is line number 7.
Line 8 - This is line number 8.

Displaying the number of words in each line:
Line: Line 1 - This is line number 1., Number of words: 8
Line: Line 2 - This is line number 2., Number of words: 8
Line: Line 3 - This is line number 3., Number of words: 8
Line: Line 4 - This is line number 4., Number of words: 8
Line: Line 5 - This is line number 5., Number of words: 8
Line: Line 6 - This is line number 6., Number of words: 8
Line: Line 7 - This is line number 7., Number of words: 8
Line: Line 8 - This is line number 8., Number of words: 8
Line: Line 9 - This is line number 9., Number of words: 8
Line: Line 10 - This is line number 10., Number of words: 8
PS E:\Gate 2025\Python\Data Structures> █
```

**Q.3. Write a Python program to create a file of numbers by taking input from the user. Split this file into two files where one file contains odd numbers and the other file contains even numbers from the file. You can take input of non-zero numbers, with an appropriate prompt, from the user until the user enters a zero to create the file assuming that the numbers are non-zero.**

```
def create_and_split_file():
 main_file_name = input("Please provide a name for the main file that will contain the numbers: ")
 odd_file_name = main_file_name.replace(".txt", "_odd.txt")
 even_file_name = main_file_name.replace(".txt", "_even.txt")

 with open(main_file_name, "w") as main_file:
 while True:
```

```

user_input = input("Enter a non-zero number (or 0 to stop): ")
if user_input == "0":
 break
main_file.write(user_input + "\n")

print(f"File '{main_file_name}' created successfully.")

with open(main_file_name, "r") as main_file:
 with open(odd_file_name, "w") as odd_file:
 with open(even_file_name, "w") as even_file:
 for line in main_file:
 num = int(line.strip())
 if num % 2 == 0:
 even_file.write(line)
 else:
 odd_file.write(line)

print(f"File '{odd_file_name}' created successfully.")
print(f"File '{even_file_name}' created successfully.")

print("\nContent of the Odd File:")
with open(odd_file_name, "r") as odd_file:
 print(odd_file.read())

print("\nContent of the Even File:")
with open(even_file_name, "r") as even_file:
 print(even_file.read())

create_and_split_file()

```

Output:

```
Please provide a name for the main file that will contain the numbers: ta.txt
Enter a non-zero number (or 0 to stop): 1
Enter a non-zero number (or 0 to stop): 2
Enter a non-zero number (or 0 to stop): 3
Enter a non-zero number (or 0 to stop): 4
Enter a non-zero number (or 0 to stop): 5
Enter a non-zero number (or 0 to stop): 0
File 'ta.txt' created successfully.
File 'ta_odd.txt' created successfully.
File 'ta_even.txt' created successfully.
```

```
Content of the Odd File:
```

```
1
3
5
```

```
Content of the Even File:
```

```
2
4
```

**Q.4. Write a Python program to create a file of elements of any data type (mixed data type, i.e. some elements maybe of type int, some elements of type float and some elements of type string). Split this file into three file containing elements of same data type (i.e. 1st file of integers only, 2nd file of float only and 3rd file of strings only). Take input from the user to create the file.**

```
def create_and_split_file():
 main_file_name = input("Please provide a name for the main file: ")
 int_file_name = main_file_name.replace(".txt", "_int.txt")
 float_file_name = main_file_name.replace(".txt", "_float.txt")
 str_file_name = main_file_name.replace(".txt", "_str.txt")

 with open(main_file_name, "w") as main_file:
 while True:
 user_input = input("Enter an element (or 'q' to stop): ")
 if user_input.lower() == "q":
 break
 main_file.write(user_input + "\n")

 print(f"File '{main_file_name}' created successfully.")

 with open(main_file_name, "r") as main_file:
 with open(int_file_name, "w") as int_file:
 with open(float_file_name, "w") as float_file:
```

```

with open(str_file_name, "w") as str_file:
 for line in main_file:
 element = line.strip()
 try:
 int_value = int(element)
 int_file.write(str(int_value) + "\n")
 except ValueError:
 try:
 float_value = float(element)
 float_file.write(str(float_value) + "\n")
 except ValueError:
 str_file.write(element + "\n")

print(f"File '{int_file_name}' created successfully.")
print(f"File '{float_file_name}' created successfully.")
print(f"File '{str_file_name}' created successfully.")

print("\nContent of the Integer File:")
with open(int_file_name, "r") as int_file:
 print(int_file.read())

print("\nContent of the Float File:")
with open(float_file_name, "r") as float_file:
 print(float_file.read())

print("\nContent of the String File:")
with open(str_file_name, "r") as str_file:
 print(str_file.read())

create_and_split_file()

```

### Output:

```

Please provide a name for the main file: t4.txt
Enter an element (or 'q' to stop): 10
Enter an element (or 'q' to stop): 3.23
Enter an element (or 'q' to stop): hello
Enter an element (or 'q' to stop): 5
Enter an element (or 'q' to stop): world
Enter an element (or 'q' to stop): 2.31
Enter an element (or 'q' to stop): q
File 't4.txt' created successfully.
File 't4_int.txt' created successfully.
File 't4_float.txt' created successfully.
File 't4_str.txt' created successfully.

```

Content of the Integer File:

```

10
5

```

Content of the Float File:

```

3.23
2.31

```

Content of the String File:

```

hello
world

```

**Q.5. Write a Python program to create a file containing student records where each record contain rollno and marks in 3 subjects separated by a comma (marks to be considered as list of 3 values). e.g. records of students: 1, [45, 40, 35], 2, [41, 38, 39], 3, [35, 30, 37] (each line of the file containing record of only 1 student). Prepare mark list in the following format: Roll No. Mark-1 Mark-2 Mark-3 Total**

```

def create_student_records_file():
 file_name = "student_records.txt"
 with open(file_name, "w") as file:
 records = [
 (1, [45, 40, 35]),
 (2, [41, 38, 39]),
 (3, [35, 30, 37])
]
 for record in records:
 roll_no = record[0]
 marks = record[1]
 total_marks = sum(marks)
 file.write(f"{roll_no}, {marks[0]}, {marks[1]}, {marks[2]}, {total_marks}\n")

```

```

print(f"File '{file_name}' created successfully.")

def display_mark_list():
 print("Roll No. Mark-1 Mark-2 Mark-3 Total")
 with open("student_records.txt", "r") as file:
 for line in file:
 data = line.strip().split(" ")
 roll_no = data[0]
 marks = data[1:4]
 total_marks = data[4]
 print(f"{roll_no:<10} {marks[0]:<8} {marks[1]:<8} {marks[2]:<8} {total_marks}")

create_student_records_file()
display_mark_list()

```

**Output:**

| Roll No. | Mark-1 | Mark-2 | Mark-3 | Total |
|----------|--------|--------|--------|-------|
| 1        | 45     | 40     | 35     | 120   |
| 2        | 41     | 38     | 39     | 118   |
| 3        | 35     | 30     | 37     | 102   |

PS E:\Gate 2025\Python\Data Structures> █

**Q.6. Write a Python program to create a file of strings by taking input from the user and then create a dictionary containing each string along with their frequencies. (e.g. if the file contains 'apple', 'banana', 'fig', 'apple', 'fig', 'banana', 'grapes', 'fig', 'grapes', 'apple' then the output should be {'apple': 3, 'banana': 2, 'fig': 3, 'grapes': 2}).**

```

def create_string_file():
 file_name = "string_file.txt"
 with open(file_name, "w") as file:
 while True:
 user_input = input("Enter a string (or 'q' to stop): ")
 if user_input.lower() == "q":
 break
 file.write(user_input + "\n")

 print(f"File '{file_name}' created successfully.")

def create_string_frequency_dictionary():
 string_frequency = {}
 with open("string_file.txt", "r") as file:
 for line in file:

```



```

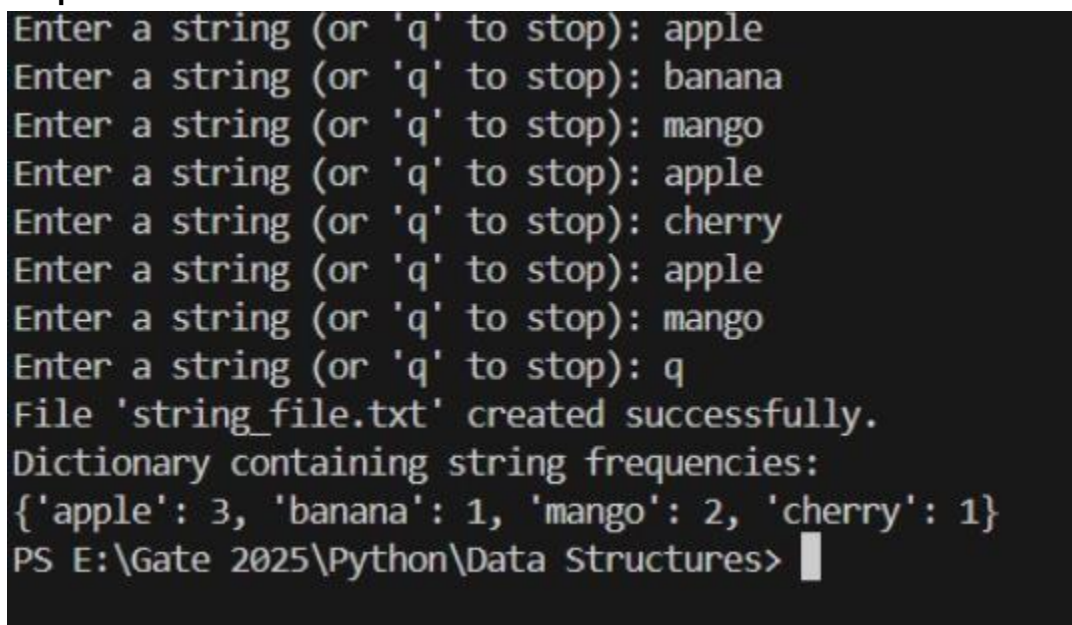
 string = line.strip()
 if string in string_frequency:
 string_frequency[string] += 1
 else:
 string_frequency[string] = 1

 print("Dictionary containing string frequencies:")
 print(string_frequency)

create_string_file()
create_string_frequency_dictionary()

```

#### Output:



```

Enter a string (or 'q' to stop): apple
Enter a string (or 'q' to stop): banana
Enter a string (or 'q' to stop): mango
Enter a string (or 'q' to stop): apple
Enter a string (or 'q' to stop): cherry
Enter a string (or 'q' to stop): apple
Enter a string (or 'q' to stop): mango
Enter a string (or 'q' to stop): q
File 'string_file.txt' created successfully.
Dictionary containing string frequencies:
{'apple': 3, 'banana': 1, 'mango': 2, 'cherry': 1}
PS E:\Gate 2025\Python\Data Structures>

```

**Q.7. Write a Python program to create a text file of multiple lines. Take input of a word from the user and then display all the lines from the file containing this word along with the frequency of the word in that line.**

```

Define the file name and sample lines
file_name = "sample_text_file.txt"
lines = [
 "This is the first line of the text file.",
 "This line contains the word text and file.",
 "Another line, but without the target word.",
 "The word text appears multiple times in this line: text text text."
]

```

# Write lines to the file

```

with open(file_name, "w") as file:
 for line in lines:
 file.write(line + "\n")

Function to search for a word in the file and display lines containing the word with its
frequency
def search_word_in_file(word):
 with open(file_name, "r") as file:
 for line in file:
 count = line.lower().count(word.lower())
 if count > 0:
 print(f"Line: {line.strip()}")
 print(f"Frequency of '{word}': {count}")
 print()

Take input of the word to search from the user
word_to_search = input("Enter the word to search for: ")

Call the function to search for the word in the file and display the results
search_word_in_file(word_to_search)

```

#### Output:

```

Enter the word to search for: text
Line: This is the first line of the text file.
Frequency of 'text': 1

Line: This line contains the word text and file.
Frequency of 'text': 1

Line: The word text appears multiple times in this line: text text text.
Frequency of 'text': 4

PS E:\Gate 2025\Python\Data Structures>

```