Slip1:

```
# Write a python program to show use of multiple exception handing
def divide_no(x,y):
    try:
        result=x/y;
        print(f"The {x} and {y} division is {result} ");
    except ZeroDivisionError:
        print(f"Zero is not allowed")
    except TypeError:
        print("Numeric error")
    finally:
        print("Block")

try:
    n1=float(input("Enter a first no:"))
    n2=float(input("Enter a second no:"))
    divide_no(n1,n2);
except ValueError:
    print("Please enter valid numeric")
finally:
    print("Program execution completed.")
```

```
#Write a python function to check whether a number is perfect or not
def is_perfect_no(num):
    return num==sum([i for i in range(1,num) if num%i==0])
no=int(input("Enter a no to check for perfection::"))
if is_perfect_no(no):
    print("no is perfect")
else:
    print("no is not perfect")
```

```
#Write a python program to display only those words from the text file which
contains three characters in it.
def display_three(file_path):
    try:
        with open(file_path,'r') as file:
            words =file.read().split()
            three_char_word=[word for word in words if len(word)==3]
            if three_char_word:
                print(three_char_word)
            else:
                print("No words with three characters found")
    except FileNotFoundError:
        print("File not found")

#Call the function
```

```
if __name__=="__main__":
    file_path=input("Enter path")
    display_three(file_path)
```

slip2

```
#Q1. Write a python program to show how to use else clause with try and except clauses
# Write a python program to show use of multiple exception handing
def divide_no(x,y):
    try:
        result=x/y;
        print(f"The {x} and {y} division is {result} ");
    except ZeroDivisionError:
        print(f"Zero is not allowed")
    except TypeError:
        print("Numeric error")
    finally:
        print("Block")

try:
    n1=float(input("Enter a first no:"))
    n2=float(input("Enter a second no:"))
    divide_no(n1,n2);
except ValueError:
    print("Please enter valid numeric")
finally:
    print("Program execution completed.")
```

```
#2. Write a python program to count and display even and odd numbers of a List.
def count(no):
    evencnt=0
    oddcnt=0
    for i in no:
        if i%2==0:
            evencnt+=1
        else:
            oddcnt+=1
    print("Even numbers:",evencnt)
    print("odd numbers:",oddcnt)
if __name__=="__main__":
    try:
        no=list(map(int,input("Enter a list of no seperated by space::").split()))
        count(no)
    except:
        print("Invalid")
```

```
#Write a python program to find sum of items of a Dictionary
def sumofdict(dictt):
    total_sum=sum(dictt.values())
    return total_sum
if __name__=="__main__":
    try:
        dictt=eval(input("Enter a dictionires::"))
        if not isinstance(dictt,dict):
            raise ValueError("Error::")
        result=sumofdict(dictt)
        print("Sum of values",result)
    except:
        print("Error::")
```

slip 3

```python
#. Write a python program to show use of finally clause of exception handling.
def divide_no(x,y):
    try:
        result=x/y;
        print(f"The {x} and {y} division is {result} ");
    except ZeroDivisionError:
        print(f"Zero is not allowed")
    except TypeError:
        print("Numeric error")
    finally:
        print("Block")

try:
    n1=float(input("Enter a first no:"))
    n2=float(input("Enter a second no:"))
    divide_no(n1,n2);
except ValueError:
    print("Please enter valid numeric")
finally:
    print("Program execution completed.")
```

```python
# Write a program that returns a list that contains only the elements that are common
#between the lists (without duplicates). Make sure your program works on two lists of different
#sizes.
def findcommon(list1,list2):
    return list(set(list1)&set(list2))
if __name__=="__main__":
    try:
        list1=[34,56,78,90]
        list2=[36,56,85,90]
        ele=findcommon(list1,list2)
        if ele:
            print("common ele",ele)
        else:
            print("No common")
    except:
        print("Error")
```

```python
#Write a python program which accepts file name and word to be searched in file.
#Display line numbers which contain given word and total occurrences of it
def searchwordinfile(file_name,search_word):
    try:
        with open(file_name,'r')as file:
            lines=file.readlines()
            occurrences=0
            line_numbers=[]
            #iterate
            for i,line in enumerate(lines,start=1):
                if search_word in line:
                    occurrences+=1
                    line_numbers.append(i)
            if occurrences>0:
               print(f"'{search_word}' found in lines:")
               print("Line No:",line_numbers)
               print("Occur:",occurrences)
            else:
                print("Not found")
    except FileNotFoundError:
        print(f"Error: File '{file_name}' not found.")

if __name__ == "__main__":
    file_name = input("Enter the file name: ")
    search_word = input("Enter the word to search: ")
    searchwordinfile(file_name, search_word)
```

slip4

```
#Write a python program to show how to raise an exception in python.
class CustomException(Exception):
    """Custom exception class"""
    pass
def example_function(value):
    if value < 0:
        raise CustomException("Negative values are not alloed")
    return value*2
try:
    intt=int(input("Enter a positive no"));
    result=example_function(intt)
    print("Result",result)
except CustomException as ce:
    print(f"Custome Exception:{ce}")
except ValueError:
    print("Invalid input, please enter a number")
```

```
# Write a python program to display ‘n’ terms of Fibonacci series using recursion
def fibonacci(n, series=[0, 1]):
    return series if n <= len(series) else fibonacci(n, series + [series[-1] + series[-2]])
n = int(input("Enter the number of terms for the Fibonacci series: "))
result = fibonacci(n)

print(f"The first {n} terms of the Fibonacci series are: {result}")
```

slip5

```
# Write a python program to find the repeated items of a tuple
def findreqitem(input_tuple):
    return set(item for item in input_tuple if input_tuple.count(item)>1)
input_tuple=(1,2,3,2,9,5,5,2,2)
res=findreqitem(input_tuple)
print(res)
```

2. Write a python program to perform following task. [20 marks] a. Calculate the factorial of given number. b. Reverse the given number

```
def fact(n):
    if n==0 or n==1:
        return 1
    else:
        return n*fact(n-1)

def rev(n2):
    return int(str(n2)[::-1])

no=int(input("Enter no:"))
print("factorial no is ::",fact(no))
print("Reversed number is :",rev(no))
```

slip6

```
# Write a python program to create a lambda function that adds 15 to a given number
# passed in as an argument.
add15=lambda x:x+15
no=int(input("Enter no:"))
res=add15(no)
print("after adding 15:",res)
```

```
# Q2. Write a python program to print following pattern using loop. [20 marks]
# A
# B C
# D E F
# G H I J
current_char=ord('A')
for i in range(4):
    print("\n")
    for j in range(i+1):
        print(chr(current_char),end=" ")
        current_char+=1
print()
```

```
# Write a Python program to reverse the content of a file and store it in
another file.
def copycontent(file1,file2):
    try:
        with open(file1,'r') as file:
            content=file.read()
        reversed=content[::-1]
        with open(file2,'w') as file:
            file.write(reversed)
    except FileNotFoundError:
        print("Error:")
copycontent('abc.txt','xyz.txt')
```

slip7

```
#  Write a python program to show how to handle multiple exceptions
def div():
    try:
        x=int(input("Enter the no"))
        t=int(input("Enter the no"))
        result=x/t
        print("The division is",result)
    except(ValueError,TypeError):
        print("Please enter valid numbers")
    except ZeroDivisionError:
        print("Cannot divide by zero")
    except Exception as e:
        print("Unexpected")
div()
```

```
#  Write a python program to display tables from m to n. [20 marks]
#  Example Input: m=3, n=7
#  Output: 3*1=3 4*1=4 …… 7*1=7
#  3*2=6 4*2=8 …… 7*2=14
```

```
# .
# .
# .
#  3*10=30 4*10=40 …… 7*10=7
def table(m,n):
    for i in range(1,11):
        for j in range(m,n+1):
            print(f"{j}*{i}={j*i}",end="\t")
        print()
m=3
n=7
table(m,n)
```

```
#Write a python program to accept directory name and print names of all files
whose
# extension is ‘.txt’ in the given directory.
import os
def listtxtfiles(dir):
    try:
        files=os.listdir()
        txt_files=[file for file in files if file.endswith('txt')]
        if txt_files:
            print(txt_files)
        else:
            print("not found")
    except FileExistsError:
        print("Errr")
listtxtfiles('TRYHVD')
```

slip8

```
#  Write a python program to calculate the cube of all numbers from 1 to n
def cal_cubes(n):
    return [i**3 for i in range(1,n+1)]
n=int(input("Enter a no:"))
cubes=cal_cubes(n)
print(f"The cubes from 1 to {n} are :{cubes}")
```

```
# Write a python program to display all prime numbers within given range
def displayprimerange(start,end):
    primes=[no for no in range(start,end+1)if all(no%i!=0 for i in
range(2,int(no**0.5)+1))and no>1]
    print(f"Prime numbers in the range {start} to {end}: {primes}")
displayprimerange(20,50)
```

slip9

```
# Write a python program to find square of given number using list
comprehension
number = int(input("Enter a number: "))
squares = [x ** 2 for x in [number]]
print(f"The square of {number} is: {squares[0]}")
```

```
# Write a python program which will find all such numbers which are divisible
by 3 and
# not by 7 within given range m to n.
# Get input for the range (m and n)
m = int(input("Enter the lower bound (m): "))
n = int(input("Enter the upper bound (n): "))
# Find numbers divisible by 3 and not by 7 within the given range
result_numbers = [num for num in range(m, n + 1) if num % 3 == 0 and num % 7
!= 0]
# Print the result
print(f"Numbers divisible by 3 and not by 7 within the range {m} to {n}:
{result_numbers}")
```

```
# Write a python program to reverse each word of file and also count total
lines
def reverse_words_in_file(input_filename, output_filename):
    try:
        # Open the input file in read mode
        with open(input_filename, 'r') as file:
            lines = file.readlines()

        # Reverse each word in each line
        reversed_lines = [' '.join(word[::-1] for word in line.split()) for
line in lines]

        # Count the total number of lines
        total_lines = len(reversed_lines)

        # Open the output file in write mode
        with open(output_filename, 'w') as file:
            # Write the reversed lines to the output file
            file.writelines(reversed_lines)

        print(f"Reversed words written to {output_filename}")
        print(f"Total lines in the file: {total_lines}")

    except FileNotFoundError:
        print(f"Error: File '{input_filename}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
```

```
# Example usage
input_filename = 'input.txt'  # Replace with your input file name
output_filename = 'output.txt'  # Replace with your output file name

reverse_words_in_file(input_filename, output_filename)
```

slip10

```
# #Q1. Write a python program which create a lambda function that multiplies
argument x with
# argument y and print the result. [10 marks]
multiply = lambda x, y: x * y
# Example usage
x = float(input("Enter the value for x: "))
y = float(input("Enter the value for y: "))
result = multiply(x, y)
print(f"The result of multiplying {x} and {y} is: {result}")
```

```
# Q2. Write a python program to display all files in directory and
subdirectories.
# [20 marks]
import os

# Q2: Display all files in directory and subdirectories
def display_files(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            print(os.path.join(root, file))

# Example usage
directory_path = input("Enter the directory path: ")
display_files(directory_path)
```

```
# Q2. Write a python program to delete repeated lines from a file.
# Q2: Delete repeated lines from a file
def delete_repeated_lines(input_filename, output_filename):
    try:
        with open(input_filename, 'r') as file:
            lines = file.readlines()
        unique_lines = set(lines)
        with open(output_filename, 'w') as file:
            file.writelines(unique_lines)
        print(f"Repeated lines removed. Result written to {output_filename}")
```

```
    except FileNotFoundError:
        print(f"Error: File '{input_filename}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
# Example usage
input_filename = 'input.txt'  # Replace with your input file name
output_filename = 'output.txt'  # Replace with your output file name

delete_repeated_lines(input_filename, output_filename)
```

slip11

```
#Write a python program to find the repeated items of a tuple
def find(input):
    return set(item for item in input if input.count(item)>1)
input=(1,2,3,4,5,6,7,1,2,2,2,2)
res=find(input)
print(res)
```

```
#  Write a python program with user defined function which accept long string
containing
# multiple words and it return same string with the words in backwards order.
# Example: Input= “I am Msc student” then output = ”student Msc am I
# Q2: Reverse words in a string
def reverse_words(input_string):
    words = input_string.split()
    reversed_words = ' '.join(words[::-1])
    return reversed_words
input_string = input("Enter a string: ")
result = reverse_words(input_string)
print(f"The string with reversed words: {result}")
```

```
# Q2. Define a class Employee having members – id, name, department, salary.
Create a
# subclass called –Manager with member bonus. Define methods accept and
display in both
# the classes. Create n objects of the Manager class and display the details
of the manager
# having the maximum total salary (salary + bonus)
class Employee:
    def __init__(self, emp_id, name, department, salary):
        self.emp_id = emp_id
        self.name = name
        self.department = department
        self.salary = salary
    def accept(self):
```

```
        self.emp_id = input("Enter employee ID: ")
        self.name = input("Enter employee name: ")
        self.department = input("Enter employee department: ")
        self.salary = float(input("Enter employee salary: "))
    def display(self):
        print(f"ID: {self.emp_id}, Name: {self.name}, Department:
{self.department}, Salary: {self.salary}")
class Manager(Employee):
    def __init__(self, emp_id, name, department, salary, bonus):
        super().__init__(emp_id, name, department, salary)
        self.bonus = bonus
    def accept(self):
        super().accept()
        self.bonus = float(input("Enter manager bonus: "))
    def display(self):
        super().display()
        print(f"Bonus: {self.bonus}")
# Function to find the manager with the maximum total salary
def find_max_salary_manager(managers):
    if not managers:
        return None
    max_salary_manager = managers[0]
    max_total_salary = max_salary_manager.salary + max_salary_manager.bonus
    for manager in managers[1:]:
        total_salary = manager.salary + manager.bonus
        if total_salary > max_total_salary:
            max_salary_manager = manager
            max_total_salary = total_salary
    return max_salary_manager
# Example usage
n = int(input("Enter the number of managers: "))
managers_list = []
for i in range(n):
    manager = Manager("", "", "", 0.0, 0.0)
    manager.accept()
    managers_list.append(manager)
max_salary_manager = find_max_salary_manager(managers_list)
if max_salary_manager:
    print("\nDetails of the manager with the maximum total salary:")
    max_salary_manager.display()
else:
    print("No managers to display.")
```

slip12

```python
# . Write a python program to find the length of a set. (Don’t use built in function len)
# Q1: Find the length of a set without using built-in len function
def find_set_length(input_set):
    count = 0
    for _ in input_set:
        count += 1
    return count

# Example usage
input_set = {1, 2, 3, 4, 5}
length = find_set_length(input_set)
print(f"The length of the set is: {length}")
```

```python
#  Write a python program that accepts a sentence and calculate the number of uppercase
# letters and lowercase letters
# Q2: Count uppercase and lowercase letters in a sentence
def count_letters(sentence):
    uppercase_count = 0
    lowercase_count = 0

    for char in sentence:
        if char.isupper():
            uppercase_count += 1
        elif char.islower():
            lowercase_count += 1

    return uppercase_count, lowercase_count

# Example usage
sentence = input("Enter a sentence: ")
uppercase, lowercase = count_letters(sentence)

print(f"Number of uppercase letters: {uppercase}")
print(f"Number of lowercase letters: {lowercase}")
```

```python
#3 Define a class named Rectangle which can be constructed by a length and
width. The
# Rectangle class has amethod which can compute the area and perimeter.
Display area and
# perimeter. Also delete the object.
# Q2: Rectangle class with area and perimeter methods
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def compute_area(self):
        return self.length * self.width

    def compute_perimeter(self):
        return 2 * (self.length + self.width)

# Example usage
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

rectangle = Rectangle(length, width)

area = rectangle.compute_area()
perimeter = rectangle.compute_perimeter()

print(f"Area of the rectangle: {area}")
print(f"Perimeter of the rectangle: {perimeter}")

# Deleting the object
del rectangle
```

slip13

```python
# Q1. Write a python program to accept n elements in a set and find the length
of a set,
# maximum, minimum value and the sum of values in a set.
def analyze_set():
    # Accept n elements in a set
    n = int(input("Enter the number of elements in the set: "))
    my_set = set()

    for i in range(n):
        element = int(input(f"Enter element {i+1}: "))
        my_set.add(element)

    # Find the length of the set
```

```
    set_length = len(my_set)

    # Find the maximum value
    max_value = max(my_set)

    # Find the minimum value
    min_value = min(my_set)

    # Find the sum of values
    sum_values = sum(my_set)

    # Display the results
    print(f"Set: {my_set}")
    print(f"Length of set: {set_length}")
    print(f"Maximum value: {max_value}")
    print(f"Minimum value: {min_value}")
    print(f"Sum of values: {sum_values}")

# Call the function
analyze_set()
```

```
# Q2. Write a python program that accepts a sentence and calculate the number
of letters and
# digits in it
def count_letters_digits(sentence):
    # Initialize counters
    letter_count = 0
    digit_count = 0

    # Iterate through each character in the sentence
    for char in sentence:
        if char.isalpha():
            letter_count += 1
        elif char.isdigit():
            digit_count += 1

    # Display the results
    print(f"Number of letters: {letter_count}")
    print(f"Number of digits: {digit_count}")

# Accept a sentence from the user
sentence = input("Enter a sentence: ")

# Call the function
count_letters_digits(sentence)
```

```python
# Write a python program to create a class Circle and compute the area and the
# circumference of the Circle. (Use parameterized constructor).
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        area = math.pi * (self.radius ** 2)
        return area

    def calculate_circumference(self):
        circumference = 2 * math.pi * self.radius
        return circumference

# Accept the radius from the user
radius = float(input("Enter the radius of the circle: "))

# Create an instance of the Circle class with the provided radius
circle_instance = Circle(radius)

# Calculate and display the area and circumference
area = circle_instance.calculate_area()
circumference = circle_instance.calculate_circumference()

print(f"Radius of the circle: {radius}")
print(f"Area of the circle: {area:.2f}")
print(f"Circumference of the circle: {circumference:.2f}")
```

slip14

```python
# 1. Write a program which checks whether given element exists within a tuple.
[10 marks]
def element_exists_in_tuple(my_tuple, element):
    return element in my_tuple

# Example usage
my_tuple = (1, 2, 3, 4, 5)
element_to_check = int(input("Enter the element to check: "))

if element_exists_in_tuple(my_tuple, element_to_check):
    print(f"{element_to_check} exists in the tuple.")
else:
    print(f"{element_to_check} does not exist in the tuple.")
```

```python
# Q2. Write a Python program to find the greatest common divisor (gcd) of two integers.
# [20 marks]
def calculate_gcd(a, b):
    while b:
        a, b = b, a % b
    return abs(a)

# Example usage
num1 = int(input("Enter the first integer: "))
num2 = int(input("Enter the second integer: "))

gcd_result = calculate_gcd(num1, num2)

print(f"The GCD of {num1} and {num2} is: {gcd_result}")
```

```python
# Q2. Define a class Student having members – rollno, name, age, gender. Create a subclass
# called –Test with member marks of 3 subjects. Create three objects of the Test class and
# display all the details of the student with percentage.
class Student:
    def __init__(self, rollno, name, age, gender):
        self.rollno = rollno
        self.name = name
        self.age = age
        self.gender = gender

class Test(Student):
    def __init__(self, rollno, name, age, gender, marks):
        super().__init__(rollno, name, age, gender)
        self.marks = marks

    def calculate_percentage(self):
        total_marks = sum(self.marks)
        percentage = (total_marks / (len(self.marks) * 100)) * 100
        return percentage

# Create three objects of the Test class
student1 = Test(1, "John", 20, "Male", [85, 90, 75])
student2 = Test(2, "Alice", 22, "Female", [78, 88, 92])
student3 = Test(3, "Bob", 21, "Male", [92, 85, 88])

# Display details of the students with percentage
students = [student1, student2, student3]
```

```
for student in students:
    print(f"Roll No: {student.rollno}")
    print(f"Name: {student.name}")
    print(f"Age: {student.age}")
    print(f"Gender: {student.gender}")
    print(f"Marks: {student.marks}")
    percentage = student.calculate_percentage()
    print(f"Percentage: {percentage:.2f}%")
    print("--------------------")
```

slip15

```
# Q1. Write a python program to find the repeated items of a tuple. [10 marks]
def find_repeated_items(my_tuple):
    repeated_items = set()
    unique_items = set()

    for item in my_tuple:
        if item in unique_items:
            repeated_items.add(item)
        else:
            unique_items.add(item)

    return repeated_items

# Example usage
my_tuple = (1, 2, 3, 4, 2, 5, 3, 6)
repeated_items = find_repeated_items(my_tuple)

if repeated_items:
    print(f"Repeated items in the tuple: {repeated_items}")
else:
    print("No repeated items found in the tuple.")
```

```
# Q2. Write a python program to accept string and remove the characters which
have odd index
# values of a given string using user defined function. [20 marks]
def remove_odd_index_chars(input_string):
    result = ""
    for index in range(len(input_string)):
        if index % 2 == 0:
            result += input_string[index]
    return result

# Example usage
input_string = input("Enter a string: ")
```

```
result_string = remove_odd_index_chars(input_string)

print(f"Original string: {input_string}")
print(f"String after removing characters with odd index values:
{result_string}")
```

```
# Q2. Define a class Person having members – name, address. Create a subclass
called
# –Employee with member staffed, salary. Create ‘n’ objects of the Employee
class and
# display all the details of the Employee
class Person:
    def __init__(self, name, address):
        self.name = name
        self.address = address

class Employee(Person):
    def __init__(self, name, address, staffed, salary):
        super().__init__(name, address)
        self.staffed = staffed
        self.salary = salary

# Accept the number of employees (n) from the user
n = int(input("Enter the number of employees: "))

# Create 'n' objects of the Employee class
employees = []
for i in range(n):
    name = input(f"Enter name of Employee {i + 1}: ")
    address = input(f"Enter address of Employee {i + 1}: ")
    staffed = input(f"Enter staffed status of Employee {i + 1}: ")
    salary = float(input(f"Enter salary of Employee {i + 1}: "))

    employee = Employee(name, address, staffed, salary)
    employees.append(employee)

# Display details of the employees
for i, employee in enumerate(employees, 1):
    print(f"\nEmployee {i} Details:")
    print(f"Name: {employee.name}")
    print(f"Address: {employee.address}")
    print(f"Staffed: {employee.staffed}")
    print(f"Salary: {employee.salary}")
```

slip16

```python
# Q.1 Write a python program that prints out all the elements of the list that are less than 25.
# [10 marks]
def print_elements_less_than_25(my_list):
    result_list = [element for element in my_list if element < 25]
    return result_list

# Example usage
my_list = [10, 30, 15, 5, 40, 20, 25, 18]
result = print_elements_less_than_25(my_list)

print(f"Original List: {my_list}")
print(f"Elements less than 25: {result}")
```

```python
# Q2. Create a class named Circle constructed by a radius and two methods which will compute
# the area and the perimeter of a circle. [20 marks]
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        area = math.pi * (self.radius ** 2)
        return area

    def calculate_perimeter(self):
        perimeter = 2 * math.pi * self.radius
        return perimeter

# Example usage
radius = float(input("Enter the radius of the circle: "))
circle_instance = Circle(radius)

area = circle_instance.calculate_area()
perimeter = circle_instance.calculate_perimeter()

print(f"Radius of the circle: {radius}")
print(f"Area of the circle: {area:.2f}")
print(f"Perimeter of the circle: {perimeter:.2f}")
```

```python
# Q2. For given a .txt file that has a list of a bunch of names, count how
many of each name
# there are in the file and print count
def count_names_in_file(file_path):
    name_counts = {}

    with open(file_path, 'r') as file:
        for line in file:
            names = line.strip().split()
            for name in names:
                name_counts[name] = name_counts.get(name, 0) + 1

    return name_counts

# Example usage
file_path = 'names.txt'  # Replace with the actual path to your file
name_counts = count_names_in_file(file_path)

print("Name Counts:")
for name, count in name_counts.items():
    print(f"{name}: {count}")
```

slip17

```python
# Q.1 Write a python program which reverse given string and displays both
original and
# reversed string. (Don’t use built-in function) [10 marks]
def reverse_string(input_string):
    reversed_string = ""
    for char in input_string:
        reversed_string = char + reversed_string
    return reversed_string

# Example usage
original_string = input("Enter a string: ")
reversed_string = reverse_string(original_string)

print(f"Original String: {original_string}")
print(f"Reversed String: {reversed_string}")
```

```python
# Q2. Write a python program to implement binary search to search the given
element using
# function. [20 marks]
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
```

```
    while low <= high:
        mid = (low + high) // 2
        mid_element = arr[mid]

        if mid_element == target:
            return mid  # Element found, return its index
        elif mid_element < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1  # Element not found

# Example usage
sorted_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
element_to_search = int(input("Enter the element to search: "))

result = binary_search(sorted_list, element_to_search)

if result != -1:
    print(f"Element {element_to_search} found at index {result}.")
else:
    print(f"Element {element_to_search} not found in the list.")
```

```
# Q2. Write a python program that returns a list that contains only the
elements that are
# common between the lists (without duplicates). Make sure your program works
on two lists
# of different sizes
def find_common_elements(list1, list2):
    common_elements = set(list1) & set(list2)
    return list(common_elements)

# Example usage
list1 = [1, 2, 3, 4, 5, 6]
list2 = [3, 5, 7, 8, 9]

result = find_common_elements(list1, list2)

print(f"List 1: {list1}")
print(f"List 2: {list2}")
print(f"Common Elements: {result}")
```

slip18

```
# Q1. Write a python program to count the number of characters in a string
without using any
# built-in function. [10 marks]
def count_characters(input_string):
    count = 0
    for char in input_string:
        count += 1
    return count

# Example usage
user_input = input("Enter a string: ")
character_count = count_characters(user_input)

print(f"Number of characters in the string: {character_count}")
```

```
# . Define a class Person having members – name, address. Create a subclass
called
# –Employee with member staffid, salary. Create ‘n’ objects of the Employee
class and
# display all the details of highest salaried employee.
class Person:
    def __int__(self,name,address):
        self.name=name
        self.address=address
class Employee(Person):
    def __init__(self,name,address,staff_id,salary):
        super().__init__(name,address)
        self.staff_id=staff_id
        self.salary=salary
n=int(input("Enter a no of employees"))
employees=[]
for _ in range(n):
    name=input("Enter name:")
    address = input("Enter address: ")
    staff_id = input("Enter staff ID: ")
    salary = float(input("Enter salary: "))
    employee=Employee(name,address,staff_id,salary)
    employees.append(employee)

if employee:
    highsal=mac(employee,key=lambda x:x.salary)
    print("\nDetails of the highest salaried employee:")
    print(f"Name: {highsal.name}")
    print(f"Address: {highsal.address}")
    print(f"Staff ID: {highsal.staff_id}")
```

```
    print(f"Salary: {highsal.salary}")
else:
    print("na")
```

```
# 2. Write a python program to check if a given key already exists in a
dictionary. If key
# exists replace with another key/value pair.
def replace_key(dictionary, old_key, new_key, new_value):
    if old_key in dictionary:
        dictionary[new_key] = new_value
        del dictionary[old_key]
        print(f"Key '{old_key}' replaced with '{new_key}': {dictionary}")
    else:
        print(f"Key '{old_key}' does not exist in the dictionary.")

# Example usage
my_dict = {'a': 1, 'b': 2, 'c': 3}

print(f"Original Dictionary: {my_dict}")

key_to_replace = input("Enter the key to check and replace: ")
new_key = input("Enter the new key: ")
new_value = input("Enter the new value: ")

replace_key(my_dict, key_to_replace, new_key, new_value)
```

slip19

```
# Q.1 Write a python program to get a single string from two given strings and
swap the first
# two characters of each string.
# Sample String: 'abc', 'xyz'
# Expected Output: xycabz [10 marks]
def swap_first_two_chars(str1, str2):
    new_str1 = str2[:2] + str1[2:]
    new_str2 = str1[:2] + str2[2:]
    result_string = new_str1 + new_str2
    return result_string

# Example usage
string1 = 'abc'
string2 = 'xyz'
result = swap_first_two_chars(string1, string2)

print(f"Sample Strings: '{string1}', '{string2}'")
print(f"Expected Output: {result}")
```

```python
# Q2 Define a class Person having members – name, address. Create a subclass
called
# –Employee with members staffed, salary. Create ‘n’ objects of the Employee
class and
# display all the details of the Employee. [20 marks]
class Person:
    def __init__(self, name, address):
        self.name = name
        self.address = address

class Employee(Person):
    def __init__(self, name, address, staffed, salary):
        super().__init__(name, address)
        self.staffed = staffed
        self.salary = salary

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Address: {self.address}")
        print(f"Staffed: {self.staffed}")
        print(f"Salary: {self.salary}")
        print("")

# Creating n objects of the Employee class
n = 3  # You can change the value of n as per your requirement
employees = []

for i in range(n):
    name = input("Enter name: ")
    address = input("Enter address: ")
    staffed = input("Enter staffed: ")
    salary = input("Enter salary: ")

    employee = Employee(name, address, staffed, salary)
    employees.append(employee)

# Displaying details of all employees
print("\nDetails of Employees:")
for employee in employees:
    employee.display_details()
```

```python
# Q2. Write a python program to create a tuple of n numbers and print maximum,
minimum,
# and sum of elements in a tuple. (Don’t use built-in functions
def find_max_min_sum(tuple_of_numbers):
    # Initialize max, min, and sum with the first element of the tuple
    max_num = min_num = total = tuple_of_numbers[0]
```

```
    # Iterate through the tuple to find max, min, and sum
    for num in tuple_of_numbers[1:]:
        # Finding maximum
        if num > max_num:
            max_num = num

        # Finding minimum
        if num < min_num:
            min_num = num

        # Calculating sum
        total += num

    return max_num, min_num, total

# Input: Creating a tuple of n numbers
n = int(input("Enter the number of elements in the tuple: "))
numbers = []
for i in range(n):
    element = int(input(f"Enter element {i + 1}: "))
    numbers.append(element)

tuple_of_numbers = tuple(numbers)

# Finding maximum, minimum, and sum without using built-in functions
max_num, min_num, sum_of_elements = find_max_min_sum(tuple_of_numbers)

# Displaying the results
print(f"\nMaximum element: {max_num}")
print(f"Minimum element: {min_num}")
print(f"Sum of elements: {sum_of_elements}")
```

slip20

```
# Q.1 Write a python program to accept and convert string in uppercase or vice
versa.
# [10 marks]
def convert_case(input_string):
    converted_string = ""
    for char in input_string:
        if char.isupper():
            converted_string += char.lower()
        elif char.islower():
            converted_string += char.upper()
        else:
            converted_string += char
```

```
    return converted_string

# Input: Accepting a string from the user
user_input = input("Enter a string: ")

# Converting case and displaying the result
result = convert_case(user_input)
print(f"Converted String: {result}")
```

```
# Q2 Write a python program to create a class Calculator with basic calculator
operations
# (addition,subtraction,division,multiplication,remainder). [20 marks]
class Calculator:
    def __init__(self):
        pass
    def add(self, a, b):
        return a + b
    def subtract(self, a, b):
        return a - b
    def multiply(self, a, b):
        return a * b
    def divide(self, a, b):
        if b == 0:
            return "Cannot divide by zero"
        return a / b
    def remainder(self, a, b):
        if b == 0:
            return "Cannot calculate remainder with zero"
        return a % b
# Creating an instance of the Calculator class
calculator = Calculator()
# Performing calculator operations
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
print("\nResults:")
print(f"Addition: {calculator.add(num1, num2)}")
print(f"Subtraction: {calculator.subtract(num1, num2)}")
print(f"Multiplication: {calculator.multiply(num1, num2)}")
print(f"Division: {calculator.divide(num1, num2)}")
print(f"Remainder: {calculator.remainder(num1, num2)}")
```

```
# Q2. Write a python program to perform operations on sets which includes
union of two sets,
# an intersection of sets, set difference and a symmetric difference.
```

```python
class SetOperations:
    def __init__(self, set1, set2):
        self.set1 = set1
        self.set2 = set2

    def union_operation(self):
        return self.set1.union(self.set2)

    def intersection_operation(self):
        return self.set1.intersection(self.set2)

    def set_difference_operation(self):
        return self.set1.difference(self.set2)

    def symmetric_difference_operation(self):
        return self.set1.symmetric_difference(self.set2)

# Input: Creating two sets
set1 = set(map(int, input("Enter elements for set1 (comma-separated):
").split(',')))
set2 = set(map(int, input("Enter elements for set2 (comma-separated):
").split(',')))

# Creating an instance of SetOperations class
set_operations = SetOperations(set1, set2)

# Performing set operations
union_result = set_operations.union_operation()
intersection_result = set_operations.intersection_operation()
set_difference_result = set_operations.set_difference_operation()
symmetric_difference_result = set_operations.symmetric_difference_operation()

# Displaying results
print("\nSet Operations:")
print(f"Union: {union_result}")
print(f"Intersection: {intersection_result}")
print(f"Set Difference (set1 - set2): {set_difference_result}")
print(f"Symmetric Difference: {symmetric_difference_result}")
```