Q1. Write a python program to show use of multiple exception handing. [10 marks]

🡪

try:

    num = int(input("Enter a number: "))

    result = 10 / num

    my\_list = [1, 2, 3]

    index = int(input("Enter an index for the list: "))

    value = my\_list[index]

    file = open("nonexistent.txt", "r")

    content = file.read()

except ZeroDivisionError:

    print("Division by zero is not allowed.")

except ValueError:

    print("Invalid input. Please enter a valid number.")

except IndexError:

    print("Index out of range.")

except FileNotFoundError:

    print("The file does not exist.")

except Exception as e:

    print(f"An unexpected error occurred: {e}")

else:

    print(f"Result is: {result}")

    print(f"Value at the specified index: {value}")

    file.close()

    print("File closed successfully.")

finally:

    print("Execution completed.")

Q2. Write a python function to check whether a number is perfect or not. [20 marks]

-->

num = int(input("Enter a number: "))

sum\_divisors = 0

for i in range(1, num):

    if num % i == 0:

        sum\_divisors += i

if sum\_divisors == num:

    print(num, "is a perfect number.")

else:

    print(num, "is not a perfect number.")

OR

Q2. Write a python program to display only those words from the text file which contains three

characters in it

🡪

with open('text\_file.txt', 'r') as file:

    content = file.read()

words = content.split()

three\_letter\_words = []

for word in words:

    if len(word) == 3:

        three\_letter\_words.append(word)

for word in three\_letter\_words:

    print(word)

Slip 2:

Q1. Write a python program to show how to use else clause with try and except clauses.

[10 marks]

* try:
* num = int(input("Enter a number: "))
* result = 10 / num
* except ZeroDivisionError:
* print("Division by zero is not allowed.")
* except ValueError:
* print("Invalid input. Please enter a valid number.")
* else:
* print(f"Result is: {result}")

Q2. Write a python program to count and display even and odd numbers of a List. [20 marks]

num\_list = [int(x) for x in input("Enter a list of numbers separated by spaces: ").split()]

even\_count = 0

odd\_count = 0

for num in num\_list:

    if num % 2 == 0:

        even\_count += 1

    else:

        odd\_count += 1

print("Even numbers:", even\_count)

print("Odd numbers:", odd\_count)

OR

Q2. Write a python program to find sum of items of a Dictionary

my\_dict = {'a': 10, 'b': 20, 'c': 30, 'd': 40}

sum\_values = 0

for value in my\_dict.values():

    sum\_values += value

print("Sum of values:", sum\_values)

Slip 3:

Q1. Write a python program to show use of finally clause of exception handling. [10 marks]

try:

    num = int(input("Enter a number: "))

    result = 10 / num

except ZeroDivisionError:

    print("Division by zero is not allowed.")

except ValueError:

    print("Invalid input. Please enter a valid number.")

else:

    print(f"Result is: {result}")

finally:

    print("Execution completed.")

Q2. 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. [20 marks]

def common\_elements(list1, list2):

    common\_elements = []

    for element in list1:

        if element in list2 and element not in common\_elements:

            common\_elements.append(element)

    return common\_elements

user\_input1 = input("Enter the first list of elements separated by spaces : ")

user\_input2 = input("Enter the second list of elements separated by spaces : ")

list1 = list(map(int, user\_input1.split()))

list2 = list(map(int, user\_input2.split()))

common\_elements1 = common\_elements(list1, list2)

print("Common elements between the two lists:", common\_elements1)

OR

Q2. 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 search\_word\_in\_file(file\_name, search\_word):

    try:

        with open(file\_name, 'r') as file:

            lines = file.readlines()

            total\_occurrences = 0

            for line\_number, line in enumerate(lines, start=1):

                line = line.strip()

                words = line.split()

                if search\_word in words:

                    occurrences = words.count(search\_word)

                    total\_occurrences += occurrences

                    print(f"Line {line\_number}: {line} (Occurrences: {occurrences})")

            print(f"Total Occurrences of '{search\_word}': {total\_occurrences}")

    except FileNotFoundError:

        print(f"File '{file\_name}' not found.")

    except Exception as e:

        print(f"An error occurred: {e}")

if \_\_name\_\_ == '\_\_main\_\_':

    file\_name = input("Enter the file name: ")

    search\_word = input("Enter the word to search: ")

    search\_word\_in\_file(file\_name, search\_word)

Slip 4 :

Q1. Write a python program to show how to raise an exception in python. [10 marks]

🡪

def divide(a, b):

    if b == 0:

        raise ZeroDivisionError("Division by zero is not allowed.")

    return a / b

try:

    result = divide(10, 0)

except ZeroDivisionError as e:

    print(f"Exception: {e}")

else:

    print(f"Result is: {result}")

Q2. Write a python program to display ‘n’ terms of Fibonacci series using recursion.

[20 marks]

🡪

def fibonacci\_recursive(n):

    if n <= 0:

        return []

    elif n == 1:

        return [0]

    elif n == 2:

        return [0, 1]

    else:

        fib\_list = fibonacci\_recursive(n - 1)

        fib\_list.append(fib\_list[-1] + fib\_list[-2])

        return fib\_list

n = int(input("Enter the number of terms in Fibonacci series: "))

fib\_series = fibonacci\_recursive(n)

print("Fibonacci series:", fib\_series)

OR

Q2. Write a python program to reverse each word of sentence of a file and also count total

lines.

🡪

def reverse\_words\_in\_sentence(sentence):

    words = sentence.split()

    reversed\_sentence = ' '.join(word[::-1] for word in words)

    return reversed\_sentence

file\_name = input("Enter the file name: ")

try:

    with open(file\_name, 'r') as file:

        lines = file.readlines()

        total\_lines = len(lines)

        for line\_number, line in enumerate(lines, start=1):

            reversed\_line = reverse\_words\_in\_sentence(line)

            print(f"Line {line\_number}: {reversed\_line}")

        print(f"Total Lines: {total\_lines}")

except FileNotFoundError:

    print(f"File '{file\_name}' not found.")

except Exception as e:

    print(f"An error occurred: {e}")

Slip 5 :

Q1. Write a python program to show use of assert keyword. [10 marks]

🡪

def divide(a, b):

    assert b != 0, "Division by zero is not allowed."

    return a / b

try:

    result = divide(10, 0)

except AssertionError as e:

    print(f"AssertionError: {e}")

else:

    print(f"Result is: {result}")

Q2. Write a python program to perform following task. [20 marks]

a. Calculate the factorial of given number.

b. Reverse the given number.

🡪

def calculate\_factorial(n):

    if n == 0:

        return 1

    else:

        return n \* calculate\_factorial(n - 1)

def reverse\_number(num):

    reversed\_num = 0

    while num > 0:

        digit = num % 10

        reversed\_num = reversed\_num \* 10 + digit

        num //= 10

    return reversed\_num

num = int(input("Enter a number: "))

factorial = calculate\_factorial(num)

print(f"Factorial of {num} is {factorial}")

reversed\_num = reverse\_number(num)

print(f"Reversed number of {num} is {reversed\_num}")

OR

Q2. Write a python program which takes file name as input and print the lines after making

only first character of each word in the sentence capitalized.

🡪

def capitalize\_first\_character(sentence):

    words = sentence.split()

    capitalized\_words = [word.capitalize() for word in words]

    capitalized\_sentence = ' '.join(capitalized\_words)

    return capitalized\_sentence

file\_name = input("Enter the file name: ")

try:

    with open(file\_name, 'r') as file:

        lines = file.readlines()

        for line in lines:

            capitalized\_line = capitalize\_first\_character(line)

            print(capitalized\_line, end='')

except FileNotFoundError:

    print(f"File '{file\_name}' not found.")

except Exception as e:

    print(f"An error occurred: {e}")

Slip 6 :

Q1. Write a python program to create a lambda function that adds 15 to a given number

passed in as an argument. [10 marks]

🡪

add\_fifteen = lambda x: x + 15

number = int(input("Enter a number: "))

result = add\_fifteen(number)

print(f"The result of adding 15 to {number} is {result}")

Q2. Write a python program to print following pattern using loop. [20 marks]

A

B C

D E F

G H I J

🡪

n = 4

current\_char = ord('A')

for i in range(n):

    for j in range(i + 1):

        print(chr(current\_char), end=' ')

        current\_char += 1

    print()

OR

Q2. Write a Python program to reverse the content of a file and store it in another file

-->

def reverse\_file\_content(input\_file, output\_file):

    try:

        with open(input\_file, 'r') as input\_file:

            content = input\_file.read()

        reversed\_content = content[::-1]

        with open(output\_file, 'w') as output\_file:

            output\_file.write(reversed\_content)

        print(f"Content reversed and saved to '{output\_file}'.")

    except FileNotFoundError:

        print(f"Input file '{input\_file}' not found.")

    except Exception as e:

        print(f"An error occurred: {e}")

input\_file\_name = input("Enter the input file name: ")

output\_file\_name = input("Enter the output file name: ")

reverse\_file\_content(input\_file\_name, output\_file\_name)

Slip 7:

Q1. Write a python program to show how to handle multiple exceptions. [10 marks]

-->

try:

    num = int(input("Enter a number: "))

    result = 10 / num

    my\_list = [1, 2, 3]

    index = int(input("Enter an index for the list: "))

    value = my\_list[index]

    file = open("nonexistent.txt", "r")

    content = file.read()

except ZeroDivisionError:

    print("Division by zero is not allowed.")

except ValueError:

    print("Invalid input. Please enter a valid number.")

except IndexError:

    print("Index out of range.")

except FileNotFoundError:

    print("The file does not exist.")

except Exception as e:

    print(f"An unexpected error occurred: {e}")

else:

    print(f"Result is: {result}")

    print(f"Value at the specified index: {value}")

    file.close()

    print("File closed successfully.")

finally:

    print("Execution completed.")

Q2. 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=70

🡪

m = int(input("Enter the starting number (m): "))

n = int(input("Enter the ending number (n): "))

for i in range(1, 11):

    for num in range(m, n + 1):

        print(f"{num}\*{i}={num\*i}", end="\t")

    print()

OR

Q2. 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 list\_txt\_files\_in\_directory(directory):

    try:

        files = os.listdir(directory)

        for file in files:

            if file.endswith('.txt') and os.path.isfile(os.path.join(directory, file)):

                print(file)

    except FileNotFoundError:

        print(f"Directory '{directory}' not found.")

    except Exception as e:

        print(f"An error occurred: {e}")

directory\_name = input("Enter the directory name: ")

list\_txt\_files\_in\_directory(directory\_name)

Slip 8 :

1. Write a python program to calculate the cube of all numbers from 1 to n.

[10 marks]

🡪

def calculate\_cubes(n):

    if n < 1:

        print("Please enter a positive integer.")

    else:

        for i in range(1, n + 1):

            cube = i \*\* 3

            print(f"The cube of {i} is {cube}")

n = int(input("Enter a positive integer (n): "))

calculate\_cubes(n)

Q2. Write a python program to display all prime numbers within given range. [20 marks]

OR

def is\_prime(num):

    if num <= 1:

        return False

    if num <= 3:

        return True

    if num % 2 == 0 or num % 3 == 0:

        return False

    i = 5

    while i \* i <= num:

        if num % i == 0 or num % (i + 2) == 0:

            return False

        i += 6

    return True

def list\_primes\_in\_range(start, end):

    for number in range(start, end + 1):

        if is\_prime(number):

            print(number)

try:

    start = int(input("Enter the start of the range: "))

    end = int(input("Enter the end of the range: "))

    if start > end:

        print("Invalid range. Start should be less than or equal to end.")

    else:

        print(f"Prime numbers in the range {start} to {end}:")

        list\_primes\_in\_range(start, end)

except ValueError:

    print("Invalid input. Please enter valid integers.")

Q2. Write a python program to list only files from a directory and print files count

🡪

import os

def list\_files\_in\_directory(directory):

    try:

        if not os.path.exists(directory):

            print(f"Directory '{directory}' not found.")

            return

        files = [f for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f)]

        if not files:

            print(f"No files found in the directory '{directory}'.")

        else:

            print(f"List of files in '{directory}':")

            for file in files:

                print(file)

            print(f"Total number of files: {len(files)}")

    except Exception as e:

        print(f"An error occurred: {e}")

directory\_name = input("Enter the directory name: ")

list\_files\_in\_directory(directory\_name)

Slip 9:

Q1. Write a python program to find square of given number using list comprehension.

[10 marks]

🡪

number = int(input("Enter a number: "))

squared = [x \*\* 2 for x in [number]]

print(f"The square of {number} is {squared[0]}")

Q2. 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. [20 marks]

def find\_numbers\_divisible\_by\_3\_not\_by\_7(m, n):

    for num in range(m, n + 1):

        if num % 3 == 0 and num % 7 != 0:

            print(num)

try:

    m = int(input("Enter the lower bound (m): "))

    n = int(input("Enter the upper bound (n): "))

    if m > n:

        print("Invalid range. Lower bound should be less than or equal to the upper bound.")

    else:

        print(f"Numbers divisible by 3 but not by 7 in the range {m} to {n}:")

        find\_numbers\_divisible\_by\_3\_not\_by\_7(m, n)

except ValueError:

    print("Invalid input. Please enter valid integers.")

OR

Q2. Write a python program to reverse each word of file and also count total lines.

🡪

def reverse\_words\_in\_file(input\_file, output\_file):

    try:

        with open(input\_file, 'r') as file:

            lines = file.readlines()

            total\_lines = len(lines)

        with open(output\_file, 'w') as output:

            for line in lines:

                words = line.split()

                reversed\_words = [word[::-1] for word in words]

                reversed\_line = ' '.join(reversed\_words)

                output.write(reversed\_line + '\n')

        print(f"Content reversed and saved to '{output\_file}'.")

        print(f"Total Lines: {total\_lines}")

    except FileNotFoundError:

        print(f"Input file '{input\_file}' not found.")

    except Exception as e:

        print(f"An error occurred: {e}")

input\_file\_name = input("Enter the input file name: ")

output\_file\_name = input("Enter the output file name: ")

reverse\_words\_in\_file(input\_file\_name, output\_file\_name)

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

x = float(input("Enter the first number (x): "))

y = float(input("Enter the second number (y): "))

result = multiply(x, y)

print(f"The result of {x} \* {y} is {result}")

Q2. Write a python program to display all files in directory and subdirectories.

[20 marks]

🡪

import os

def list\_files\_in\_directory(directory):

    for root, \_, files in os.walk(directory):

        for file in files:

            print(os.path.join(root, file))

try:

    directory\_name = input("Enter the directory name: ")

    if not os.path.exists(directory\_name):

        print(f"Directory '{directory\_name}' not found.")

    else:

        print(f"List of files in '{directory\_name}' and its subdirectories:")

        list\_files\_in\_directory(directory\_name)

except Exception as e:

    print(f"An error occurred: {e}")

OR

Q2. Write a python program to delete repeated lines from a file.

🡪

def remove\_duplicates(input\_file, output\_file):

    try:

        with open(input\_file, 'r') as file:

            lines = file.readlines()

        # Remove duplicates while preserving the order

        unique\_lines = list(dict.fromkeys(lines))

        with open(output\_file, 'w') as file:

            file.writelines(unique\_lines)

        print(f"Repeated lines removed and saved to '{output\_file}'.")

    except FileNotFoundError:

        print(f"Input file '{input\_file}' not found.")

    except Exception as e:

        print(f"An error occurred: {e}")

# Input and output file names

input\_file\_name = input("Enter the input file name: ")

output\_file\_name = input("Enter the output file name: ")

# Call the function to remove duplicates from the input file and save to the output file

remove\_duplicates(input\_file\_name, output\_file\_name)

Slip 9 :

Q1. Write a python program to find the repeated items of a tuple. [10 marks]

🡪

def find\_repeated\_items(t):

    repeated\_items = []

    seen\_items = set()

    for item in t:

        if t.count(item) > 1 and item not in seen\_items:

            repeated\_items.append(item)

            seen\_items.add(item)

    return repeated\_items

input\_tuple = (1, 2, 3, 2, 4, 5, 6, 3, 7, 8, 9, 7)

repeated\_items = find\_repeated\_items(input\_tuple)

print("Repeated items in the tuple:", repeated\_items)

Q2. 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” [20 marks]

-->

def reverse\_words(string):

    words = string.split()

    reversed\_words = words[::-1]

    reversed\_string = ' '.join(reversed\_words)

    return reversed\_string

input\_string = input("Enter a long string: ")

result = reverse\_words(input\_string)

print("Reversed string:", result)

OR

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, id, name, department, salary):

        self.id = id

        self.name = name

        self.department = department

        self.salary = salary

    def accept(self):

        self.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.id}")

        print(f"Name: {self.name}")

        print(f"Department: {self.department}")

        print(f"Salary: {self.salary}")

class Manager(Employee):

    def \_\_init\_\_(self, id, name, department, salary, bonus):

        super().\_\_init\_\_(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}")

n = int(input("Enter the number of managers: "))

managers = []

for i in range(n):

    print(f"Enter details for Manager {i + 1}:")

    manager = Manager("", "", "", 0, 0)

    manager.accept()

    managers.append(manager)

max\_total\_salary = 0

max\_total\_salary\_manager = None

for manager in managers:

    total\_salary = manager.salary + manager.bonus

    if total\_salary > max\_total\_salary:

        max\_total\_salary = total\_salary

        max\_total\_salary\_manager = manager

if max\_total\_salary\_manager is not None:

    print("\nManager with the Maximum Total Salary:")

    max\_total\_salary\_manager.display()

else:

    print("No managers provided.")

Slip 10 :

Q1. Write a python program to find the length of a set. (Don’t use built in function len)

[10 marks]

-->

def find\_set\_length(input\_set):

    length = 0

    for \_ in input\_set:

        length += 1

    return length

my\_set = {1, 2, 3, 4, 5}

length\_of\_set = find\_set\_length(my\_set)

print(f"Length of the set: {length\_of\_set}")

Q2. Write a python program that accepts a sentence and calculate the number of uppercase

letters and lowercase letters [20 marks]

🡪

sentence = input("Enter a sentence: ")

uppercase\_count = 0

lowercase\_count = 0

for char in sentence:

    if char.isupper():

        uppercase\_count += 1

    elif char.islower():

        lowercase\_count += 1

print("Uppercase letters:", uppercase\_count)

print("Lowercase letters:", lowercase\_count)

OR

Q2. Define a class named Rectangle which can be constructed by a length and width. The

Rectangle class has a method which can compute the area and perimeter. Display area and

perimeter. Also delete the object

🡪

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)

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}")

del rectangle

Slip 11:

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. [10 marks]

🡪

def input\_set():

    n = int(input("Enter the number of elements in the set: "))

    user\_set = set()

    for i in range(n):

        element = int(input(f"Enter element {i + 1}: "))

        user\_set.add(element)

    return user\_set

def analyze\_set(user\_set):

    length = len(user\_set)

    maximum = max(user\_set)

    minimum = min(user\_set)

    total = sum(user\_set)

    return length, maximum, minimum, total

user\_set = input\_set()

length, maximum, minimum, total = analyze\_set(user\_set)

print(f"Length of the set: {length}")

print(f"Maximum value in the set: {maximum}")

print(f"Minimum value in the set: {minimum}")

print(f"Sum of values in the set: {total}")

Q2. Write a python program that accepts a sentence and calculate the number of letters and

digits in it. [20 marks]

sentence = input("Enter a sentence: ")

letter\_count = 0

digit\_count = 0

for char in sentence:

    if char.isalpha():

        letter\_count += 1

    elif char.isdigit():

        digit\_count += 1

print("Number of letters:", letter\_count)

print("Number of digits:", digit\_count)

OR

Q2. 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 compute\_area(self):

        return math.pi \* (self.radius \*\* 2)

    def compute\_circumference(self):

        return 2 \* math.pi \* self.radius

radius = float(input("Enter the radius of the circle: "))

circle = Circle(radius)

area = circle.compute\_area()

circumference = circle.compute\_circumference()

print(f"Area of the circle: {area:.2f}")

print(f"Circumference of the circle: {circumference:.2f}")

Slip 12 :

Q1. Write a program which checks whether given element exists within a tuple. [10 marks]

def element\_exists\_in\_tuple(my\_tuple, element):

    if element in my\_tuple:

        return True

    else:

        return False

my\_tuple = (1, 2, 3, 4, 5)

element\_to\_check = int(input("Enter an 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.")

Q2. Write a Python program to find the greatest common divisor (gcd) of two integers.

[20 marks]

def find\_gcd(a, b):

    while b:

        a, b = b, a % b

    return a

num1 = int(input("Enter the first integer: "))

num2 = int(input("Enter the second integer: "))

gcd = find\_gcd(num1, num2)

print("The greatest common divisor (GCD) is:", gcd)

OR

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

    def display(self):

        print(f"Roll Number: {self.rollno}")

        print(f"Name: {self.name}")

        print(f"Age: {self.age}")

        print(f"Gender: {self.gender}")

class Test(Student):

    def \_\_init\_\_(self, rollno, name, age, gender, marks):

        super().\_\_init\_\_(rollno, name, age, gender)

        self.marks = marks

    def display(self):

        super().display()

        print(f"Marks: {self.marks}")

        print(f"Percentage: {sum(self.marks) / len(self.marks)}%")

test1 = Test("A1", "Alice", 18, "Female", [85, 92, 78])

test2 = Test("B2", "Bob", 19, "Male", [76, 88, 95])

test3 = Test("C3", "Charlie", 20, "Male", [90, 84, 79])

print("Details of Student 1:")

test1.display()

print("\nDetails of Student 2:")

test2.display()

print("\nDetails of Student 3:")

test3.display()

Slip 13 :

Q1. Write a python program to find the repeated items of a tuple. [10 marks]

def find\_repeated\_items(input\_tuple):

    repeated\_items = []

    seen\_items = set()

    for item in input\_tuple:

        if input\_tuple.count(item) > 1 and item not in seen\_items:

            repeated\_items.append(item)

            seen\_items.add(item)

    return repeated\_items

my\_tuple = (1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 7)

repeated\_items = find\_repeated\_items(my\_tuple)

print("Repeated items in the tuple:", repeated\_items)

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\_characters(input\_string):

    result = ""

    for i in range(len(input\_string)):

        if i % 2 == 0:

            result += input\_string[i]

    return result

input\_string = input("Enter a string: ")

new\_string = remove\_odd\_index\_characters(input\_string)

print("String with odd-index characters removed:", new\_string)

OR

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

    def display(self):

        print(f"Name: {self.name}")

        print(f"Address: {self.address}")

class Employee(Person):

    def \_\_init\_\_(self, name, address, staffed, salary):

        super().\_\_init\_\_(name, address)

        self.staffed = staffed

        self.salary = salary

    def display(self):

        super().display()

        print(f"Staff ID: {self.staffed}")

        print(f"Salary: {self.salary}")

n = int(input("Enter the number of employees: "))

employees = []

for i in range(n):

    print(f"Enter details for Employee {i + 1}:")

    name = input("Name: ")

    address = input("Address: ")

    staffed = input("Staff ID: ")

    salary = float(input("Salary: ")

    employee = Employee(name, address, staffed, salary)

    employees.append(employee)

print("\nDetails of Employees:")

for i, employee in enumerate(employees, 1):

    print(f"Employee {i}:")

    employee.display()

    print()

slip 14 :

Q.1 Write a python program that prints out all the elements of the list that are less than 25.

[10 marks]

# Define a list of numbers

numbers = [12, 30, 15, 10, 40, 5, 22, 18, 8, 28]

# Iterate through the list and print elements less than 25

print("Elements less than 25:")

for number in numbers:

    if number < 25:

        print(number)

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 compute\_area(self):

        return math.pi \* self.radius \*\* 2

    def compute\_perimeter(self):

        return 2 \* math.pi \* self.radius

# Create a Circle object with a given radius

radius = float(input("Enter the radius of the circle: "))

circle = Circle(radius)

# Calculate and display the area and perimeter of the circle

area = circle.compute\_area()

perimeter = circle.compute\_perimeter()

print(f"Area of the circle: {area:.2f}")

print(f"Perimeter of the circle: {perimeter:.2f}")

OR

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.

# Initialize a dictionary to store the counts of each name

name\_counts = {}

# Open the .txt file for reading

with open("names.txt", "r") as file:

    # Read each line in the file

    for line in file:

        # Split the line into names (assuming names are separated by spaces)

        names = line.strip().split()

        # Iterate through the names

        for name in names:

            # Update the count for each name in the dictionary

            name\_counts[name] = name\_counts.get(name, 0) + 1

# Print the counts for each name

for name, count in name\_counts.items():

    print(f"{name}: {count} times")

Slip 15 :

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

# Input a string from the user

input\_string = input("Enter a string: ")

# Call the function to reverse the string

reversed\_result = reverse\_string(input\_string)

# Print both the original and reversed strings

print("Original string:", input\_string)

print("Reversed string:", reversed\_result)

Q2. Write a python program to implement binary search to search the given element using

function. [20 marks]

🡪

def binary\_search(arr, target):

    left = 0

    right = len(arr) - 1

    while left <= right:

        mid = (left + right) // 2

        if arr[mid] == target:

            return mid  # Element found, return its index

        elif arr[mid] < target:

            left = mid + 1

        else:

            right = mid - 1

    return -1  # Element not found

# Sorted list

sorted\_list = [1, 3, 5, 7, 9, 11, 13, 15]

# Element to search for

element = int(input("Enter the element to search for: "))

# Call the binary\_search function to search for the element

result = binary\_search(sorted\_list, element)

# Display the search result

if result != -1:

    print(f"Element {element} found at index {result}")

else:

    print(f"Element {element} not found in the list.")

OR

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):

    # Convert the lists into sets to remove duplicates

    set1 = set(list1)

    set2 = set(list2)

    # Find the intersection of the sets

    common\_set = set1.intersection(set2)

    # Convert the result back to a list

    common\_list = list(common\_set)

    return common\_list

# Example lists

list1 = [1, 2, 3, 4, 5, 5, 6]

list2 = [4, 5, 6, 7, 8]

# Call the function to find common elements

common\_elements = find\_common\_elements(list1, list2)

# Print the common elements without duplicates

print("Common elements:", common\_elements)

Slip 17 :

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

# Input a string from the user

input\_string = input("Enter a string: ")

# Call the function to count the characters

character\_count = count\_characters(input\_string)

# Print the character count

print("Number of characters in the string:", character\_count)

Q2. 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. [20 marks]

class Person:

    def \_\_init\_\_(self, name, address):

        self.name = name

        self.address = address

class Employee(Person):

    def \_\_init\_\_(self, name, address, staffid, salary):

        super().\_\_init\_\_(name, address)

        self.staffid = staffid

        self.salary = salary

# Create 'n' Employee objects

n = int(input("Enter the number of employees: "))

employees = []

for i in range(n):

    print(f"Enter details for Employee {i + 1}:")

    name = input("Name: ")

    address = input("Address: ")

    staffid = input("Staff ID: ")

    salary = float(input("Salary: "))

    employee = Employee(name, address, staffid, salary)

    employees.append(employee)

# Find the highest salaried employee

highest\_salary = 0

highest\_salary\_employee = None

for employee in employees:

    if employee.salary > highest\_salary:

        highest\_salary = employee.salary

        highest\_salary\_employee = employee

# Display details of the highest salaried employee

if highest\_salary\_employee:

    print("\nDetails of the Highest Salaried Employee:")

    print(f"Name: {highest\_salary\_employee.name}")

    print(f"Address: {highest\_salary\_employee.address}")

    print(f"Staff ID: {highest\_salary\_employee.staffid}")

    print(f"Salary: {highest\_salary\_employee.salary}")

else:

    print("No employees entered.")

OR

Q2. Write a python program to check if a given key already exists in a dictionary. If key

exists replace with another key/value pair.

# Sample dictionary

my\_dict = {"name": "Alice", "age": 25, "city": "New York"}

# Prompt the user for a key to check

key\_to\_check = input("Enter a key to check: ")

# Check if the key exists in the dictionary

if key\_to\_check in my\_dict:

    new\_key = input("Enter a new key: ")

    new\_value = input("Enter a new value: ")

    my\_dict[new\_key] = new\_value

    del my\_dict[key\_to\_check]

else:

    print(f"The key '{key\_to\_check}' does not exist in the dictionary.")

# Print the updated dictionary

print("Updated dictionary:", my\_dict)

Slip 18 :

.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]

string1 = input("Enter the first string: ")

string2 = input("Enter the second string: ")

if len(string1) >= 2 and len(string2) >= 2:

    result = string2[:2] + string1[2:] + " " + string1[:2] + string2[2:]

    print("Result:", result)

else:

    print("Both strings must have at least two characters.")

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, staffid, salary):

        super().\_\_init\_\_(name, address)

        self.staffid = staffid

        self.salary = salary

# Create 'n' Employee objects

n = int(input("Enter the number of employees: "))

employees = []

for i in range(n):

    print(f"Enter details for Employee {i + 1}:")

    name = input("Name: ")

    address = input("Address: ")

    staffid = input("Staff ID: ")

    salary = float(input("Salary: "))

    employee = Employee(name, address, staffid, salary)

    employees.append(employee)

# Find the highest salaried employee

highest\_salary = 0

highest\_salary\_employee = None

for employee in employees:

    if employee.salary > highest\_salary:

        highest\_salary = employee.salary

        highest\_salary\_employee = employee

# Display details of the highest salaried employee

if highest\_salary\_employee:

    print("\nDetails of the Highest Salaried Employee:")

    print(f"Name: {highest\_salary\_employee.name}")

    print(f"Address: {highest\_salary\_employee.address}")

    print(f"Staff ID: {highest\_salary\_employee.staffid}")

    print(f"Salary: {highest\_salary\_employee.salary}")

else:

    print("No employees entered.")

OR

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)

# Input the number of elements 'n'

n = int(input("Enter the number of elements: "))

# Initialize an empty list to store the elements

elements = []

# Input 'n' numbers and add them to the list

for i in range(n):

    num = float(input(f"Enter element {i + 1}: "))

    elements.append(num)

# Initialize variables to store the maximum, minimum, and sum

maximum = elements[0]

minimum = elements[0]

total = 0

# Iterate through the elements to find maximum, minimum, and calculate the sum

for element in elements:

    if element > maximum:

        maximum = element

    if element < minimum:

        minimum = element

    total += element

# Create a tuple from the list of elements

elements\_tuple = tuple(elements)

# Print the maximum, minimum, and sum

print("Elements in the tuple:", elements\_tuple)

print("Maximum: ", maximum)

print("Minimum: ", minimum)

print("Sum: ", total)

Slip 19 :

Q.1 Write a python program to accept and convert string in uppercase or vice versa.

[10 marks]

# Input a string from the user

input\_string = input("Enter a string: ")

# Prompt the user for the conversion choice

conversion\_choice = input("Enter 'U' to convert to uppercase or 'L' to convert to lowercase: ")

# Perform the conversion based on the user's choice

if conversion\_choice == 'U':

    converted\_string = input\_string.upper()

    print("Uppercase:", converted\_string)

elif conversion\_choice == 'L':

    converted\_string = input\_string.lower()

    print("Lowercase:", converted\_string)

else:

    print("Invalid choice. Please enter 'U' or 'L' for conversion.")

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  # Constructor does not require any specific initialization

    def add(self, num1, num2):

        return num1 + num2

    def subtract(self, num1, num2):

        return num1 - num2

    def multiply(self, num1, num2):

        return num1 \* num2

    def divide(self, num1, num2):

        if num2 == 0:

            return "Cannot divide by zero"

        return num1 / num2

    def remainder(self, num1, num2):

        if num2 == 0:

            return "Cannot find remainder with zero"

        return num1 % num2

# Create a Calculator object

calculator = Calculator()

# Test the calculator operations

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

print("Addition:", calculator.add(num1, num2))

print("Subtraction:", calculator.subtract(num1, num2))

print("Multiplication:", calculator.multiply(num1, num2))

print("Division:", calculator.divide(num1, num2))

print("Remainder:", calculator.remainder(num1, num2))

OR

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.

# Create two sets

set1 = {1, 2, 3, 4, 5}

set2 = {3, 4, 5, 6, 7}

# Perform set operations

union\_result = set1.union(set2)              # Union of set1 and set2

intersection\_result = set1.intersection(set2)  # Intersection of set1 and set2

difference\_result = set1.difference(set2)     # Set difference (elements in set1 but not in set2)

symmetric\_difference\_result = set1.symmetric\_difference(set2)  # Symmetric difference

# Print the results

print("Set 1:", set1)

print("Set 2:", set2)

print("Union of Set 1 and Set 2:", union\_result)

print("Intersection of Set 1 and Set 2:", intersection\_result)

print("Set Difference (Set 1 - Set 2):", difference\_result)

print("Symmetric Difference of Set 1 and Set 2:", symmetric\_difference\_result)

Slip 20