Exp:1- Simple Calculator

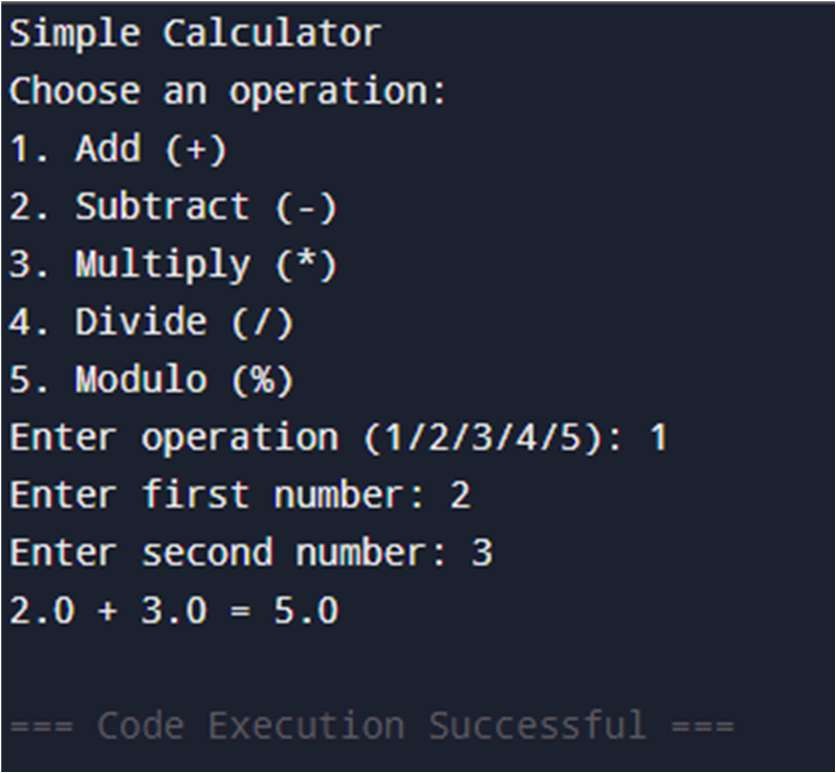
Anas Tanvar, 47, Batch-3 print("Simple Calculator") print("Choose an operation:") print("1. Add (+)") print("2. Subtract (-)") print("3. Multiply (\*)") print("4. Divide (/)") print("5. Modulo (%)")

operation = input("Enter operation (1/2/3/4/5): ") num1 = float(input("Enter first number: ")) num2 = float(input("Enter second number: ")) if operation == '1': result = num1 + num2 print(f"{num1} + {num2} = {result}") elif operation == '2': result = num1 - num2 print(f"{num1} - {num2} = {result}") elif operation == '3': result = num1 \* num2 print(f"{num1} \* {num2} = {result}") elif operation == '4': if num2 != 0:

result = num1 / num2 print(f"{num1} / {num2} = {result}") else:

print("Error! Division by zero.") elif operation == '5': result = num1 % num2 print(f"{num1} % {num2} = {result}") else:

print("Invalid input. Please choose a valid operation.")



List operation

AnasTanvar,

47, Batch-3

# Initialize Lists list1 = [] list2 = [] while True:

print("\nMENU") print("0. Create a list") print("1. Display Lists") print("2. Find Length of a List") print("3. Check if an Element is in the List") print("4. Concatenate Two Lists") print("5. Replace an Element in a List") print("6. Delete an Element from a List") print("7. Work with Nested Lists") print("8. Exit") choice = input("Enter your choice: ")if choice == "0":

for i in range(3): ele = input("Enter values of list1: ") list1.append(ele) for i in range(3): ele = input("Enter values of list2: ") list2.append(ele) elif choice == "1":

print("\n1. Display Lists")

print("list1:", list1) print("list2:", list2) elif choice == "2":

print("\n2. Find Length of a List") print("Length of list1:", len(list1)) print("Length of list2:", len(list2)) elif choice == "3":

print("\n3. Check if an Element is in the List") element = input("Enter the element to check: ") if element in list2:

print(f"'{element}' is in list2.") else:

print(f"'{element}' is not in list2.") elif choice == "4":

print("\n4. Concatenate Two Lists") result\_list = list1 + list2 print("Result of concatenation:", result\_list) elif choice == "5":

print("\n5. Replace an Element in a List") print("Original list1:", list1)

try:

index = int(input("Enter the index to replace: ")) new\_value = input("Enter the new value: ") if 0 <= index < len(list1): list1[index] = new\_value print("Updated list1:", list1) else:

print("Invalid index!") except ValueError:

print("Invalid input! Please enter a valid index.") elif choice == "6":

print("\n6. Delete an Element from a List") print("Original list1:", list1)

try:

index = int(input("Enter the index to delete: "))

if 0 <= index < len(list1):

del list1[index] print("Updated list1:", list1) else:

print("Invalid index!") except ValueError:

print("Invalid input! Please enter a valid index.") elif choice == "7":

print("\n7. Work with Nested Lists") books\_list = [("C", 896), ("C++", 599), ("Python", 1269)] while True:

print("\nNested List Operations:") print("1. Show Books List") print("2. Add a Book") print("3. Back to Main Menu") nested\_choice = input("Enter your choice: ") if nested\_choice == "1": print("\nBook\t\tPrice") for book, price in books\_list: print(f"{book}\t\t{price}") elif nested\_choice == "2":

book\_name = input("\nEnter the name of the book: ")

try:

book\_price = int(input("Enter its price: ")) books\_list.append((book\_name, book\_price)) print("Book added successfully!") except ValueError:

print("Invalid price! Please enter a numeric value.") elif nested\_choice == "3":

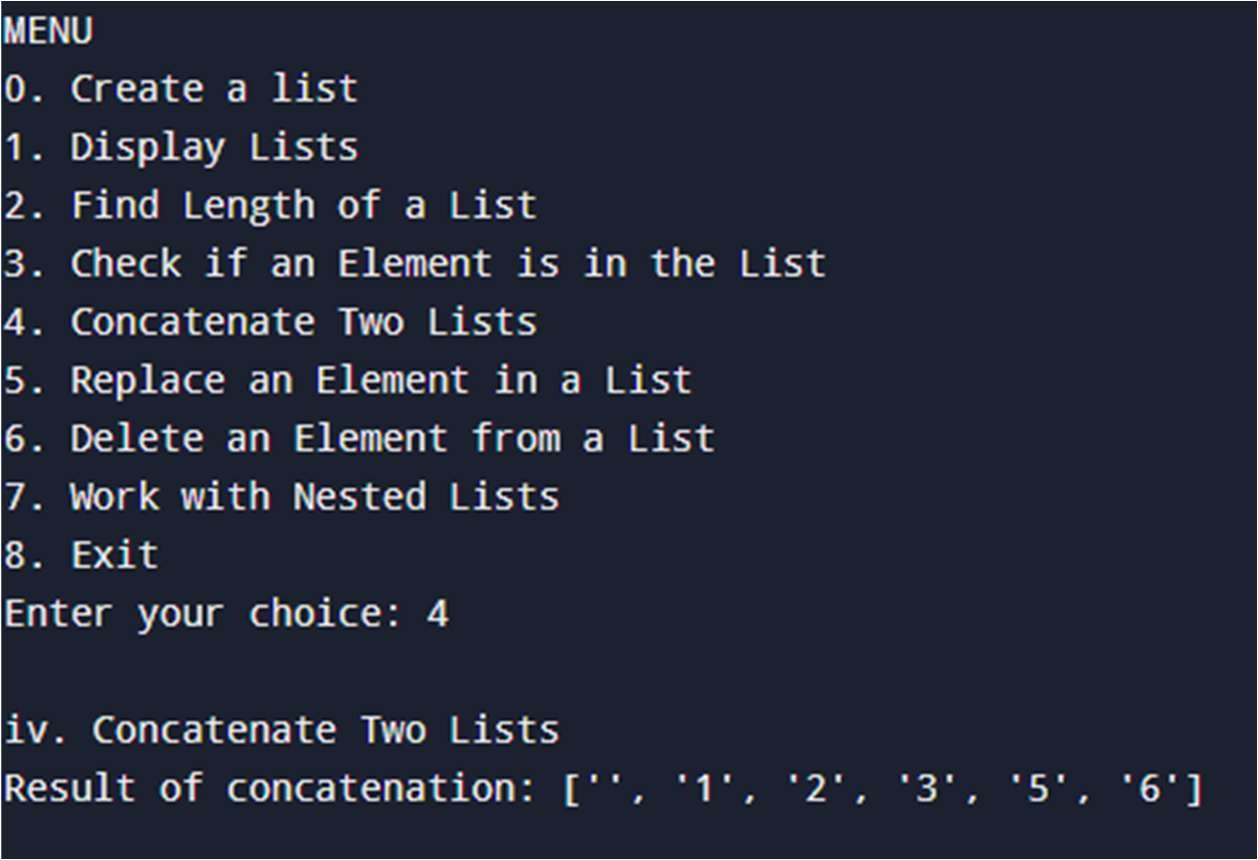
break else:

print("Invalid choice! Please try again.") elif choice == "8":

print("Exiting the program. Goodbye!")

break

else: print("Invalid choice! Please select a valid option.")



Tuples operation

Anas Tanvar, 47, Batch-3

def create\_tuple():

elements = input("Enter elements separated by space: ").split() return tuple(elements) def display\_tuple(t):

print(f"The tuple is: {t}") def sort\_tuple(t):

return tuple(sorted(t)) def reverse\_tuple(t): return t[::-1] def replicate\_tuple(t): times = int(input("Enter the number of times to replicate: ")) return t \* times def sum\_tuple(t):

return sum(map(int, t)) # Assuming the tuple contains numeric elements def count\_element(t): element = input("Enter the element to count: ") return t.count(element) def main(): t = ()

while True:

print("\nMenu:") print("1. Create tuple") print("2. Display tuple") print("3. Sort the tuple") print("4. Reverse the tuple") print("5. Replicate the tuple") print("6. Find the sum of elements of a tuple") print("7. Display frequency of an element using count method")

print("8. Exit")

choice = input("Enter your choice: ") if choice == '1': t = create\_tuple() elif choice == '2': display\_tuple(t) elif choice == '3': t = sort\_tuple(t) print("Sorted tuple:", t) elif choice == '4':

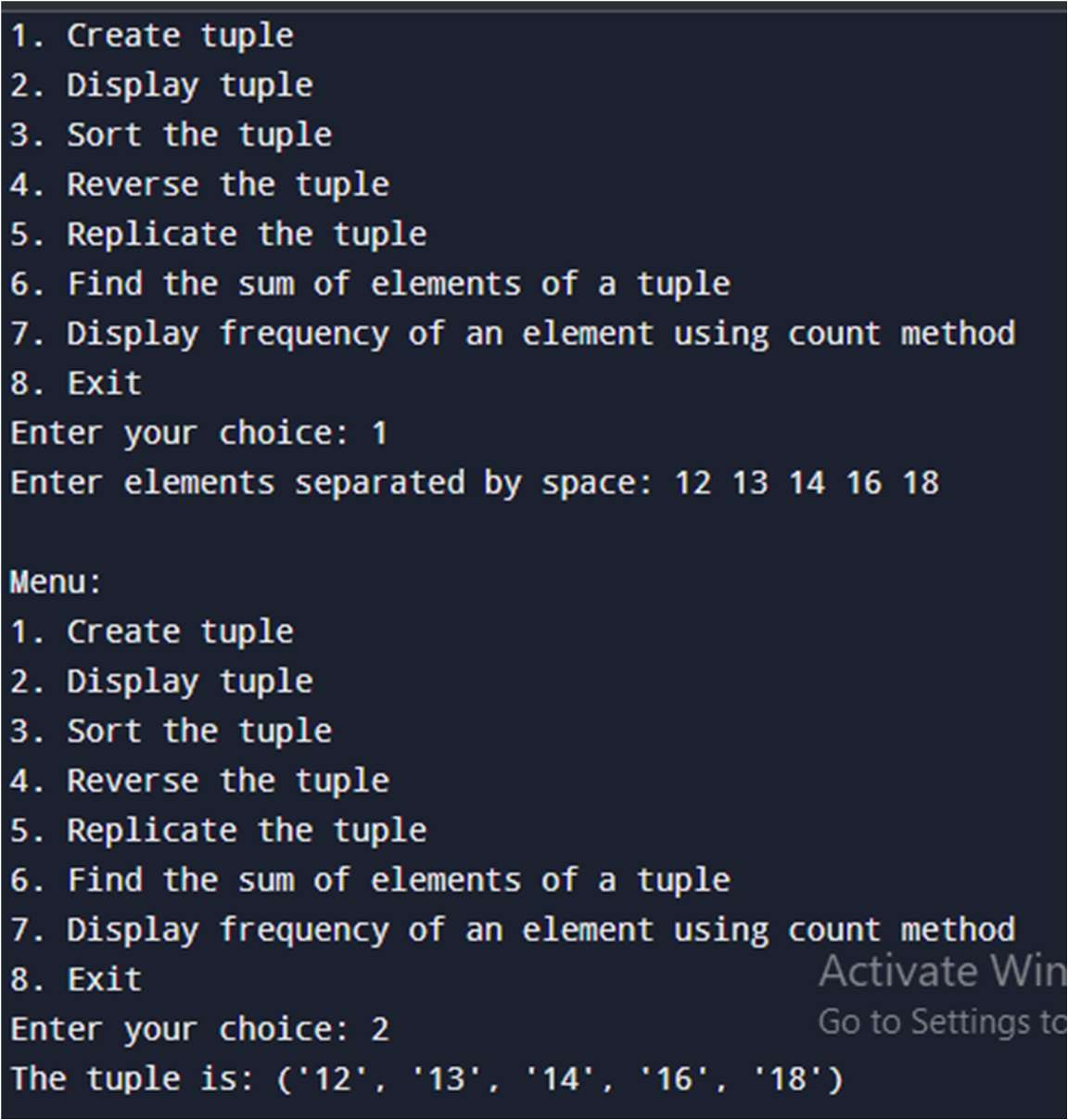
t = reverse\_tuple(t) print("Reversed tuple:", t) elif choice == '5':

t = replicate\_tuple(t) print("Replicated tuple:", t) elif choice == '6': print("Sum of elements:", sum\_tuple(t)) elif choice == '7': print("Frequency of element:", count\_element(t)) elif choice == '8':

print("Exiting the program.") break else:

print("Invalid choice. Please try again.") if \_\_name\_\_ == "\_\_main\_\_":

main()



Set-operation

Anas Tanvar, 47, Batch-3

A, B = set(), set() while True:

print("\n\*\*\*\*\*\*\*\*\*\* Set Operations Menu \*\*\*\*\*\*\*\*\*\*") print("1. Create Sets")

print("2. Union, Intersection, Difference, Symmetric Difference") print("3. Modify Set") print("4. Remove Elements from Set") print("5. Use Pop and Clear") print("6. Display frequency of an element") print("7. Exit")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") choice = input("Enter your choice (1-7): ") if choice == '1':

A = {12, 54, 78, 36} B = {1, 2, 3, 4, 5} print("Set A:", A) print("Set B:", B) elif choice == '2':

print("Union of A and B:", A | B)

print("Intersection of A and B:", A & B) print("Difference of A and B (A - B):", A - B) print("Symmetric Difference of A and B:", A ^ B) elif choice == '3': my\_set = {1, 3} print("Initial Set:", my\_set) my\_set.add(2)

print("After Adding 2:", my\_set) my\_set.update([2, 3, 4])

print("After Adding Multiple Elements:", my\_set) my\_set.update([4, 5], {1, 6, 8}) print("After Adding List and Set:", my\_set) elif choice == '4':

my\_set = {1, 3, 4, 5, 6} print("Initial Set:", my\_set) my\_set.discard(4)

print("After Discarding 4:", my\_set) my\_set.remove(6) print("After Removing 6:", my\_set) my\_set.discard(2) # No error if not present print("After Discarding 2:", my\_set) elif choice == '5':

my\_set = set("HelloWorld") print("Initial Set:", my\_set) print("Popped Element:", my\_set.pop()) print("Set after pop:", my\_set) my\_set.clear()

print("Set after clear:", my\_set) elif choice == '6':

element = input("Enter the element to count: ") count = 0 for e in A:

if str(e) == element: # Ensure comparison works for different data types

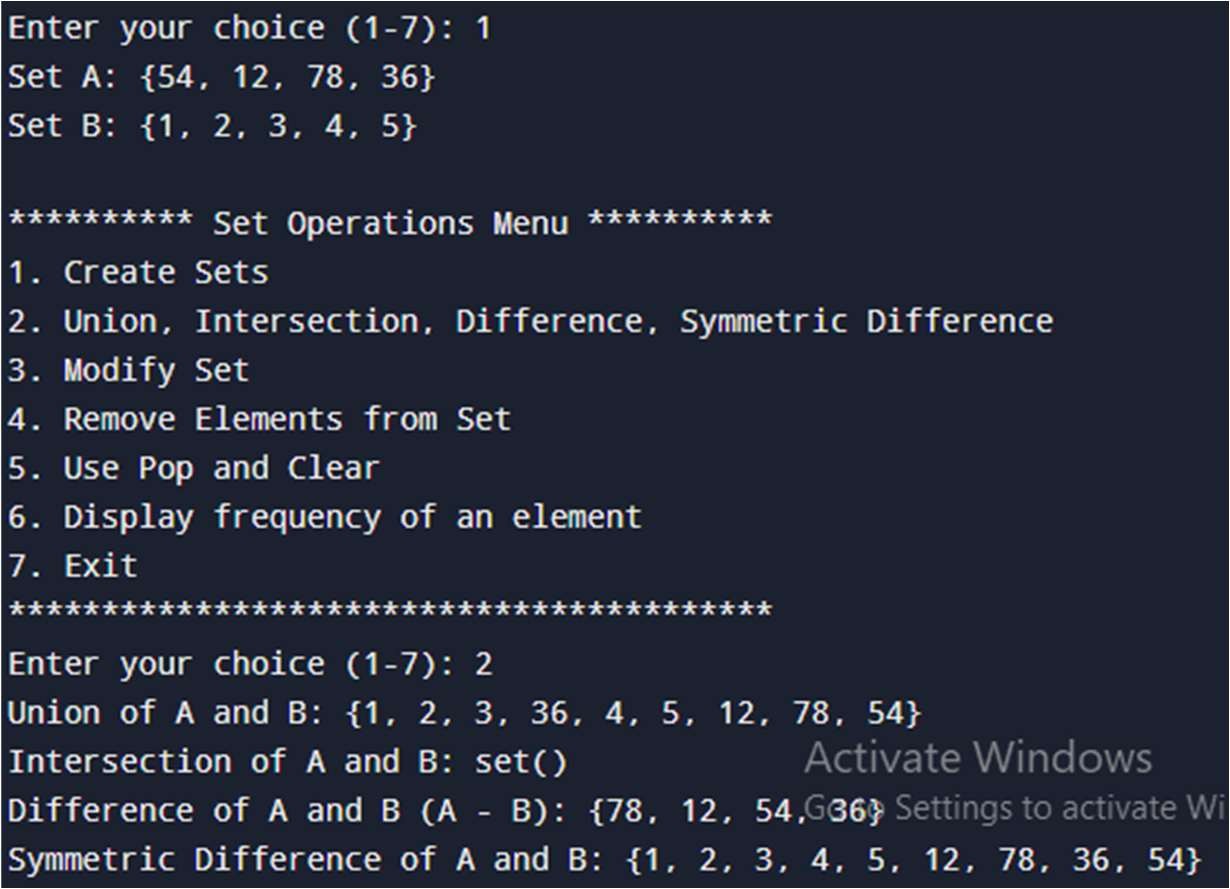
count += 1

print(f"Frequency of element '{element}':", count)

elif choice == '7':

print("Exiting Program. Goodbye!") break else:

print("Invalid choice! Please enter a number between 1 and 7.")



Dictionary-operation

Anas Tanvar, 47, Batch-3 create\_dictionary():

n = int(input("Enter number of key-value pairs: "))

d = {} for \_ in range(n):

key = input("Enter key: ") value = input("Enter value: ") d[key] = value return d def iterate\_values(d):

print("Values in the dictionary:") for value in d.values():

print(value) def update\_value(d):

key = input("Enter the key whose value you want to update: ") if key in d:

value = input("Enter new value: ") d[key] = value

print("Updated dictionary:", d) else:

print("Key not found.") def add\_key\_value(d):

key = input("Enter new key: ") value = input("Enter new value: ") d[key] = value print("Updated dictionary:", d) def delete\_key\_value(d):

key = input("Enter the key to delete: ") if key in d: del d[key]

print("Updated dictionary:", d) else:

print("Key not found.") def set\_default\_value(d):

key = input("Enter the key for default value: ") default\_value = input("Enter default value: ") value = d.setdefault(key, default\_value) print(f"Value for key '{key}':", value) def main(): d = {} while True:

print("\n\*\*\*\*\*\*\*\* Dictionary Operations Menu \*\*\*\*\*\*\*\*") print("1. Create dictionary") print("2. Iterate values from a dictionary") print("3. Update value of any key") print("4. Add a new key-value pair in a dictionary") print("5. Delete key-value pair from a dictionary") print("6. Set default value and display") print("7. Exit")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") choice = input("Enter your choice (1-7): ") if choice == '1':

d = create\_dictionary()

elif choice == '2': iterate\_values(d) elif choice == '3': update\_value(d) elif choice == '4': add\_key\_value(d) elif choice == '5':

delete\_key\_value(d) elif choice == '6':

set\_default\_value(d) elif choice == '7':

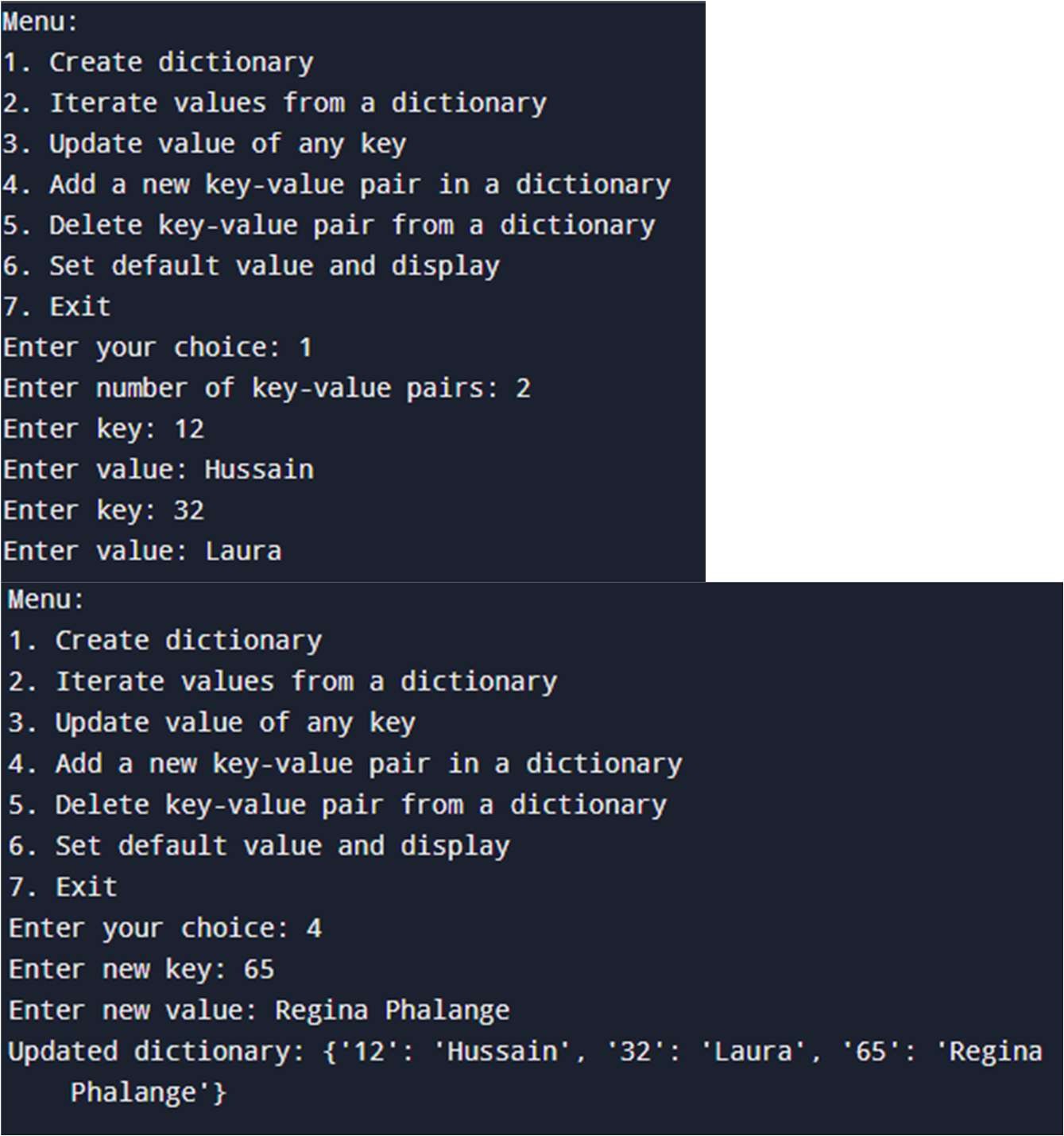
print("Exiting the program. Goodbye!")

break else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()



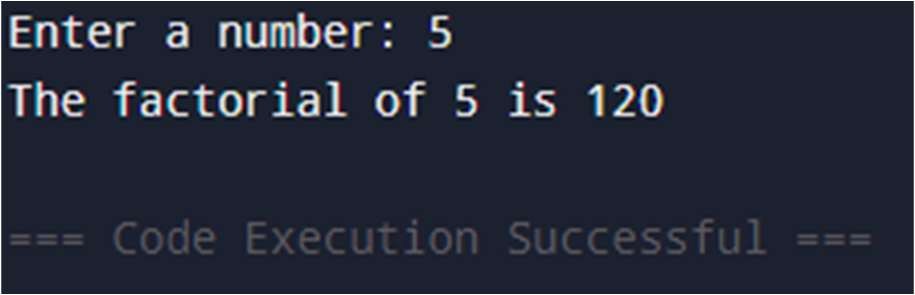
Post-lab factorial:

Anas Tanvar, 47,

Batch-3

def factorial(n): if n == 0: return 1 else:

return n \* factorial(n - 1) num = int(input("Enter a number: ")) print(f"The factorial of {num} is {factorial(num)}")



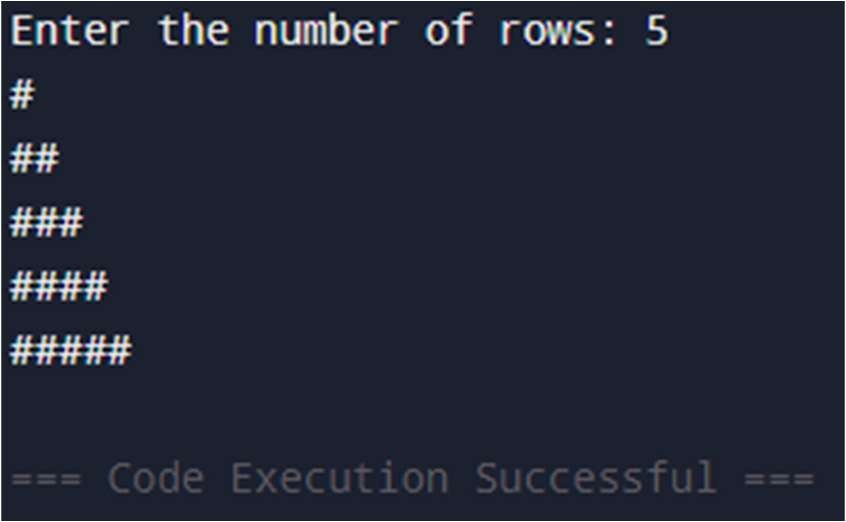
Post-lab patterns

Anas Tanvar, 47, Batch-3

def print\_triangle(n): for i in range(1, n + 1):

print('#' \* i)

num = int(input("Enter the number of rows: ")) print\_triangle(num)



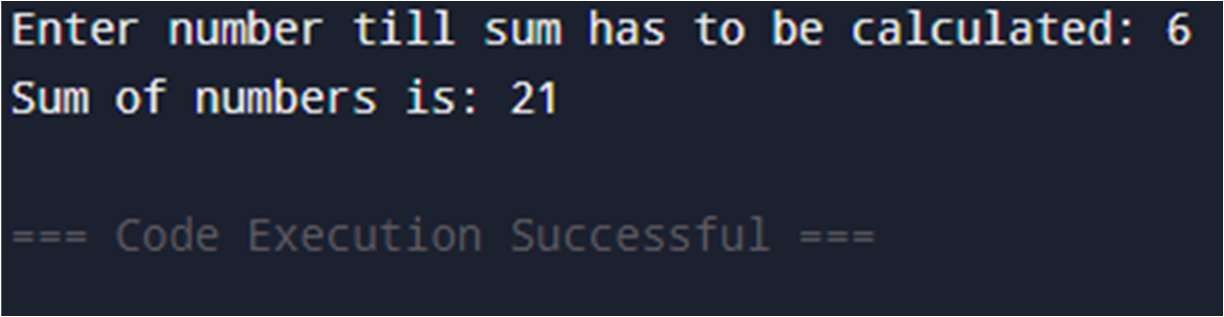
Exp-2 Sum of n numbers

Anas Tanvar, 47, Batch-3

def sum(n):

if n < 1: return n else:

return n + sum(n - 1) n = int(input("Enter number till sum has to be calculated: ")) print("Sum of numbers is:", sum(n))



Accept roll no and check student present or not

# Name: Anas Tanvar

# Roll no:47

def detail(roll, attendance\_list): if roll in attendance\_list:

print("Roll number {roll} is present")

else:

print("Roll number {roll} is absent") def presentees():

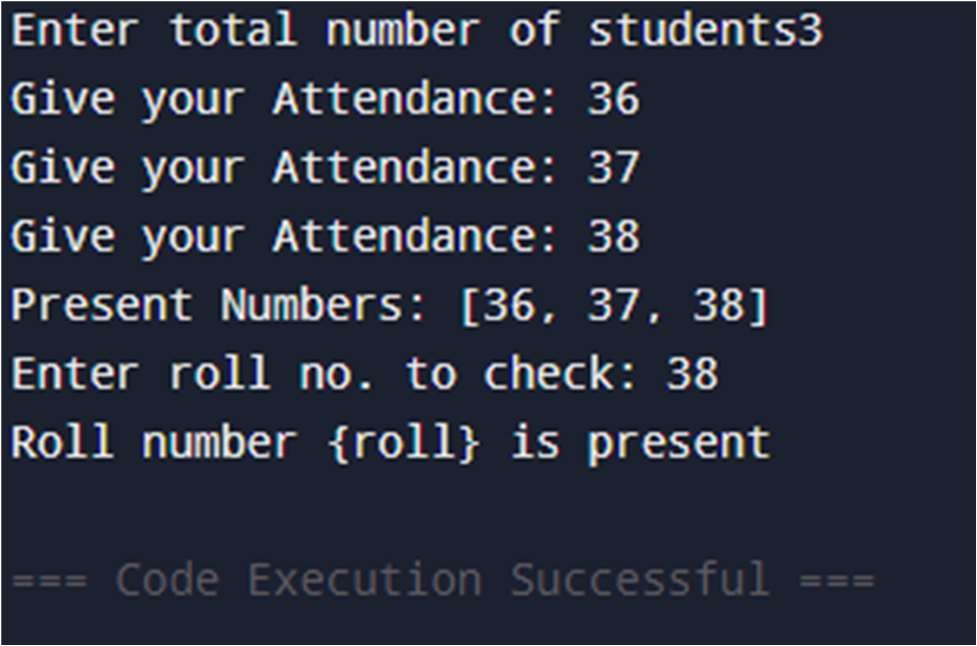
p=[]

no\_of\_student=int(input("Enter total number of students")) for i in range(1, no\_of\_student+1):

roll\_no=int(input("Give your Attendance: "))

p.append(roll\_no) print(f"Present Numbers: {p}") return p

attendance\_list=presentees() roll=int(input("Enter roll no. to check: ")) detail(roll, attendance\_list)



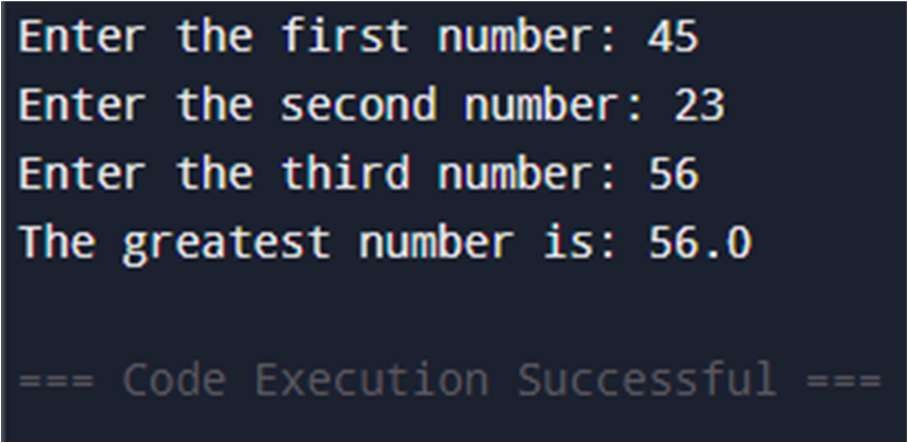
Post—lab: Greatest number

Anas Tanvar, 47, Batch-3

def find\_greatest(a, b, c): return max(a, b, c) try:

num1 = float(input("Enter the first number: ")) num2 = float(input("Enter the second number: ")) num3 = float(input("Enter the third number: ")) greatest = find\_greatest(num1, num2, num3) print("The greatest number is:", greatest) except ValueError: print("Please enter valid numerical values.")

Bubble sort



Anas Tanvar, 47, Batch-3

def bubble\_sort(arr): n = len(arr) for i in range(n): swapped = False for j in range(0, n-i-1): if arr[j] > arr[j+1]:

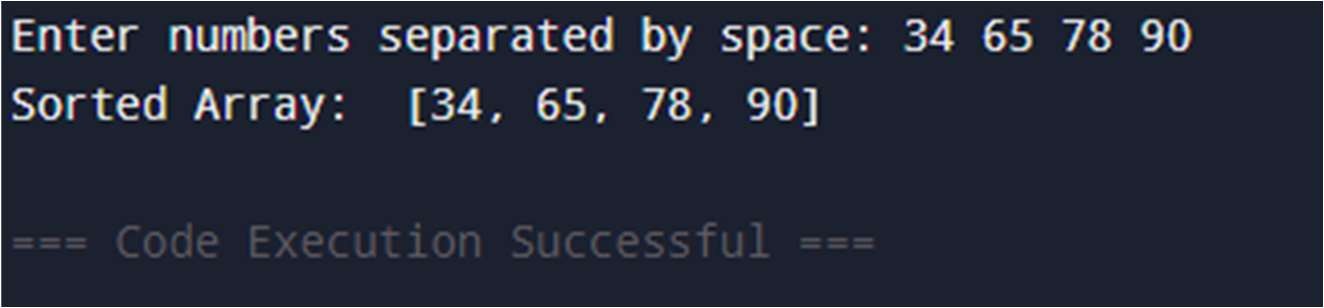
arr[j], arr[j+1] = arr[j+1], arr[j] # Swap swapped = True if not swapped:

break

def get\_input\_and\_sort():

arr = list(map(int, input("Enter numbers separated by space: ").split())) bubble\_sort(arr) print("Sorted Array: ", arr)

get\_input\_and\_sort()



Exp-3: WAP in python to implement following multiple inheritances

Anas Tanvar, 47, Batch-3

# Employee class class Employee:

def \_\_init\_\_(self, id): self.id = id self.name = None def Setname(self, name): self.name = name def Getname(self): return self.name def Getid(self): return self.id class Student: def \_\_init\_\_(self, college): self.college = college def Getcollege(self): return self.college

class Intern(Employee, Student): def \_\_init\_\_(self, id, college, period):

# Initialize both parent classes

Employee.\_\_init\_\_(self, id) Student.\_\_init\_\_(self, college) self.period = period self.name = None def Setdetails(self, name):

self.name = name def Getdetails(self): return {

"name": self.name,

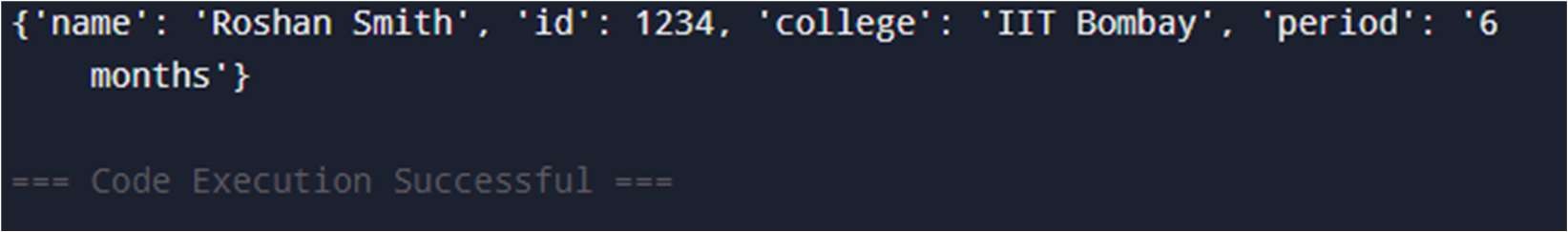
"id": self.id,

"college": self.college,

"period": self.period

}

intern = Intern(1234, "IIT Bombay", "6 months") intern.Setdetails("Roshan Smith") details = intern.Getdetails() print(details)



Write a program to calculate volume of sphere using multilevel inheritance. Anas Tanvar, 47, Batch-3

import math class Base: def \_\_init\_\_(self):

self.radius = float(input("Enter the radius of the sphere: ")) class Circle(Base):

def area\_of\_circle(self):

return math.pi \* (self.radius \*\* 2) # Area of circle = πr² class Sphere(Circle):

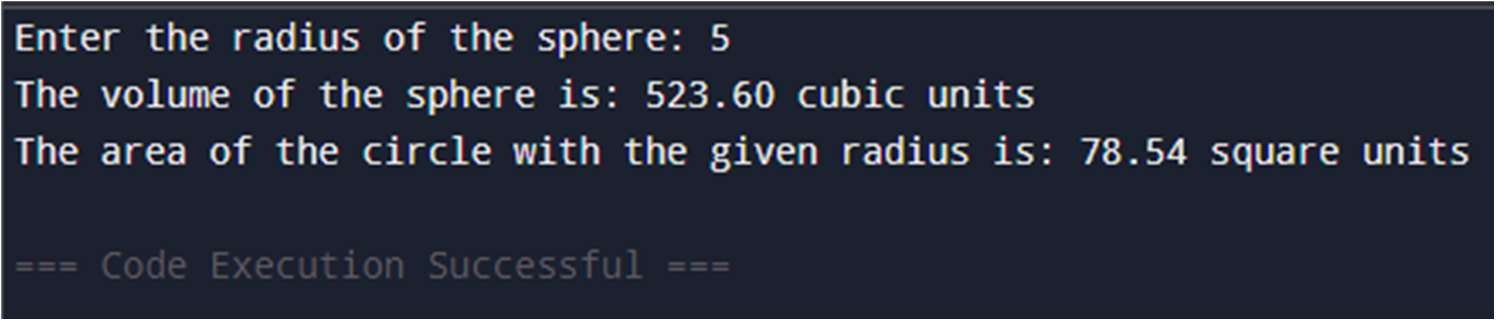
def volume\_of\_sphere(self):

return (4/3) \* math.pi \* (self.radius \*\* 3) def display\_volume(self):

volume = self.volume\_of\_sphere() print(f"The volume of the sphere is: {volume:.2f} cubic units") def main():

sphere = Sphere() sphere.display\_volume() area = sphere.area\_of\_circle() print(f"The area of the circle with the given radius is: {area:.2f} square units")

main()



Using method overriding

Anas Tanvar, 47, Batch-3

import math class Base: def \_\_init\_\_(self):

self.radius = float(input("Enter the radius of the sphere: ")) def get\_radius(self): return self.radius class Circle(Base): def get\_radius(self): return self.radius def area\_of\_circle(self):

return math.pi \* (self.get\_radius() \*\* 2) # Area of circle = πr² class Sphere(Circle): def get\_radius(self):

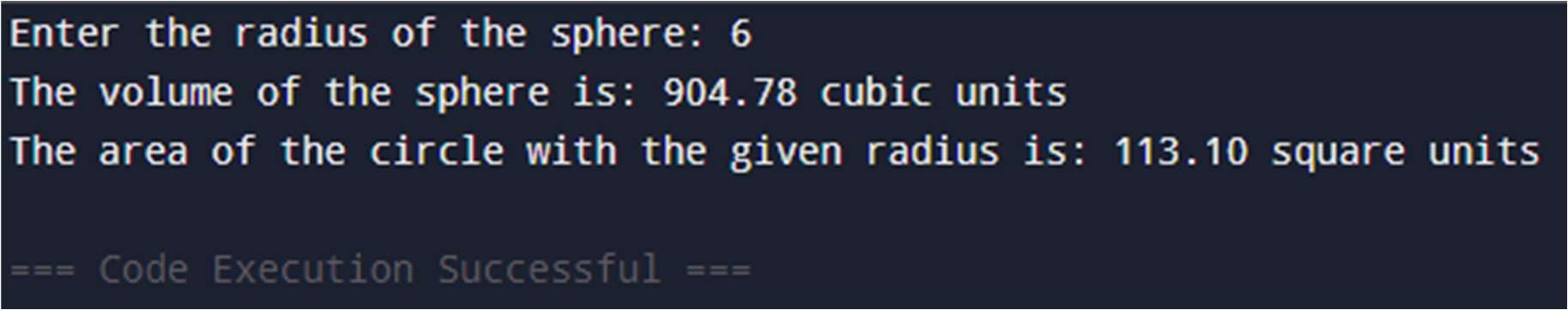
return self.radius

def volume\_of\_sphere(self):

return (4/3) \* math.pi \* (self.get\_radius() \*\* 3) # Volume of sphere = (4/3)πr³ def display\_volume(self):

volume = self.volume\_of\_sphere() print(f"The volume of the sphere is: {volume:.2f} cubic units") def main():

sphere = Sphere() sphere.display\_volume() area = sphere.area\_of\_circle() print(f"The area of the circle with the given radius is: {area:.2f} square units") main()



EXP 4: Write a menu driven program on files to add, delete and display movie from text file.

Anas Tanvar, 47, Batch-3

import os

FILE\_NAME = "movies.txt" def add\_movie(movie):

with open(FILE\_NAME, "a") as file:

file.write(movie + "\n") def delete\_movie(movie):

if not os.path.exists(FILE\_NAME): return with open(FILE\_NAME, "r") as file: lines = file.readlines() with open(FILE\_NAME, "w") as file: for line in lines: if line.strip().lower() != movie.lower():

file.write(line) def display\_movies(): if not os.path.exists(FILE\_NAME):

print("No movies found.")

return with open(FILE\_NAME, "r") as file:

print("\nMovies List:") print(file.read()) def main(): while True:

print("1. Add Movie 2. Delete Movie 3. Display Movies 4. Exit") choice = input("Enter choice: ") if choice == "4":

break

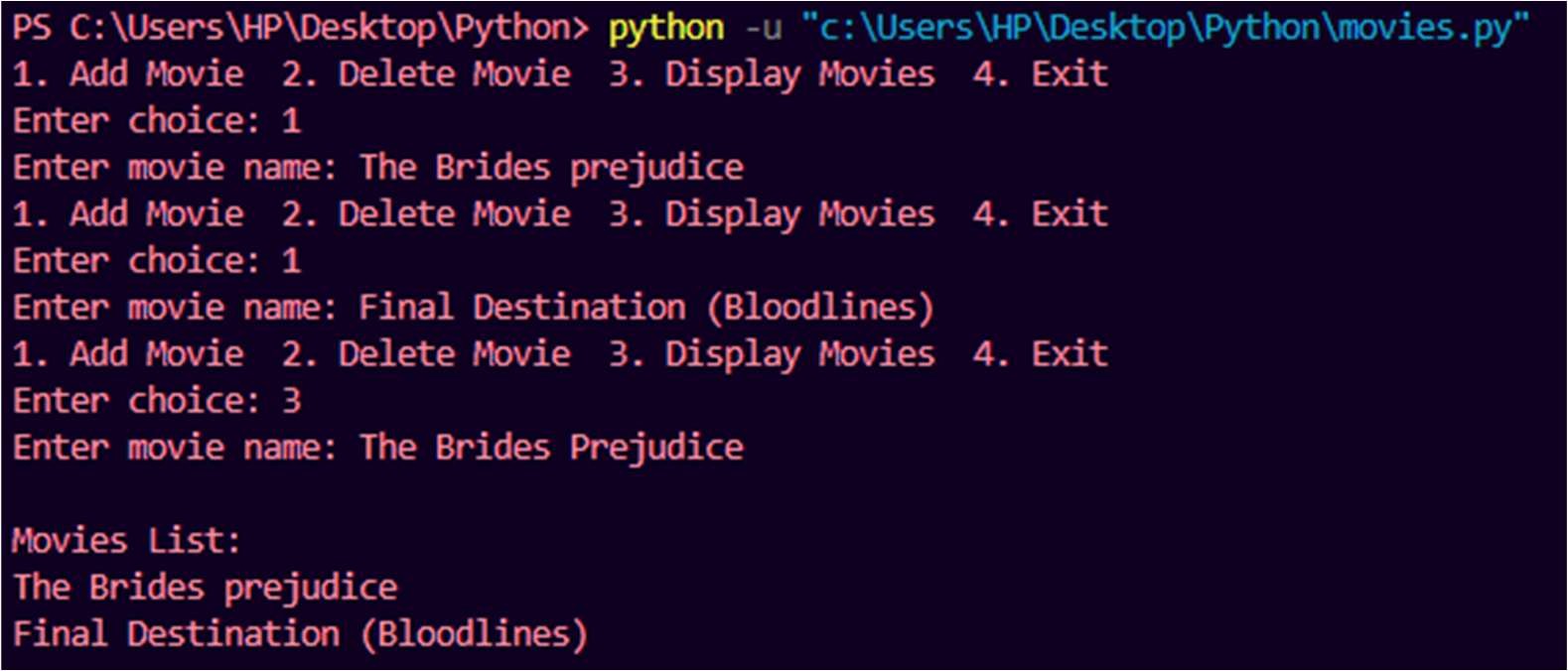
movie = input("Enter movie name: ") if choice == "1":

add\_movie(movie) elif choice == "2":

delete\_movie(movie) elif choice == "3": display\_movies() else:

print("Invalid choice.") if \_\_name\_\_ == "\_\_main\_\_":

main()



Write Python program to count number of lines, words and characters in a file.

Anas Tanvar, 47, Batch-3

def count\_file\_contents(file\_path): with open(file\_path, 'r', encoding='utf-8') as file:

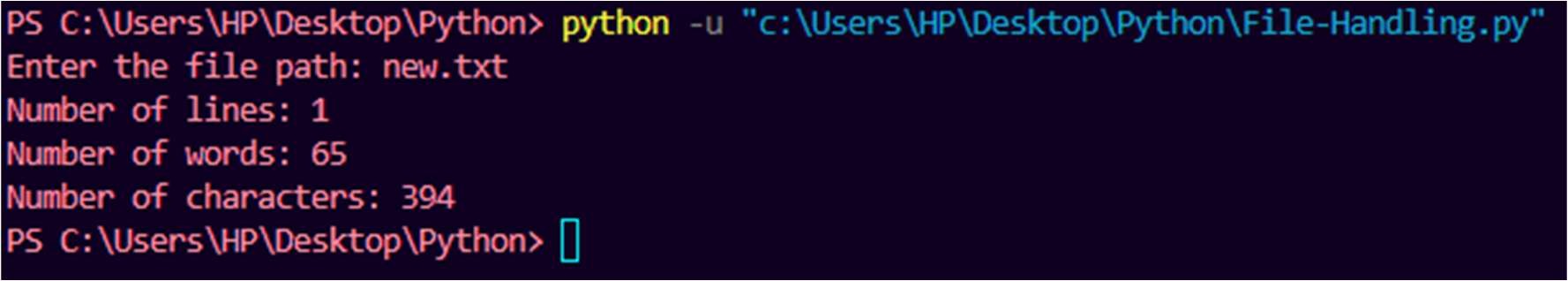
lines = file.readlines() num\_lines = len(lines) num\_words = sum(len(line.split()) for line in lines) num\_chars = sum(len(line) for line in lines)

print(f"Number of lines: {num\_lines}") print(f"Number of words: {num\_words}")

print(f"Number of characters: {num\_chars}")

# Example usage

file\_path = input("Enter the file path: ") count\_file\_contents(file\_path)



Quiz program

Anas Tanvar, 47, Batch-3

def load\_questions(): with open("Question.txt", "r") as q\_file:

questions = q\_file.read().strip().split("\n\n") # Splitting questions by blank lines return questions def load\_answers(): with open("Answers.txt", "r") as a\_file:

answers = [line.strip().upper() for line in a\_file.readlines()] return answers def quiz():

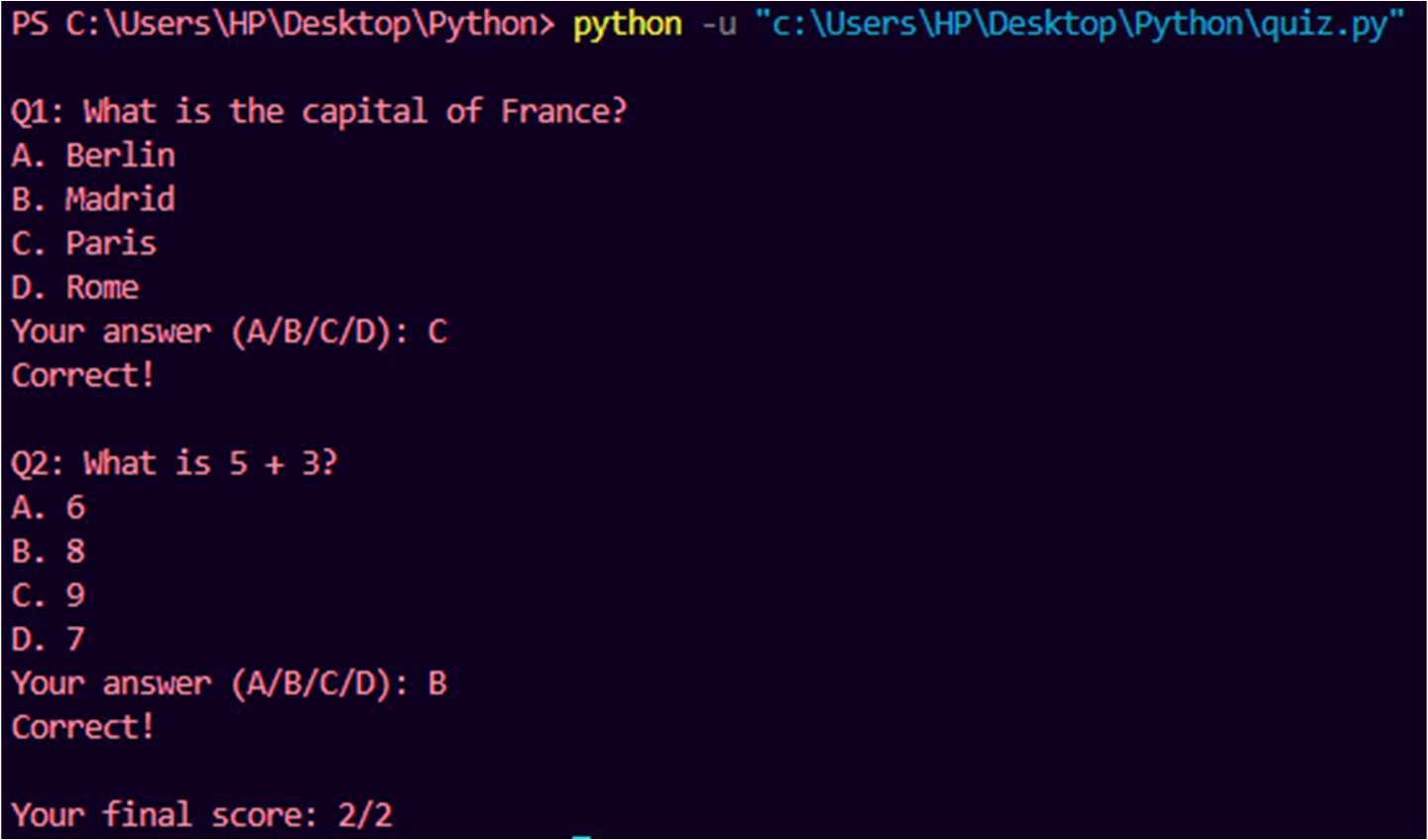
questions = load\_questions() answers = load\_answers()

score = 0 for i, question in enumerate(questions):

print(f"\nQ{i+1}: {question}") # Display question with options user\_answer = input("Your answer (A/B/C/D): ").strip().upper() if user\_answer == answers[i]: # Check if the answer is correct print("Correct!") score += 1 else:

print(f"Wrong! Correct answer: {answers[i]}") print(f"\nYour final score: {score}/{len(questions)}") if \_\_name\_\_ == "\_\_main\_\_":

quiz()



Operations of file handling:

Anas Tanvar, 47, Batch-3

def read\_file(filename): with open(filename, "r") as file:

content = file.read()

print("\nFile Content:\n", content)

def write\_file(filename):

content = input("Enter content to write into the file: ") with open(filename, "w") as file:

file.write(content)

print("Content written successfully.")

def append\_file(filename): content = input("Enter content to append into the file: ") with open(filename, "a") as file:

file.write("\n" + content)

print("Content appended successfully.")

def main():

filename = "sample.txt" # Default file name while True:

print("\n1. Read File 2. Write File 3. Append File 4. Exit") choice = input("Enter choice: ") if choice == "1":

read\_file(filename) elif choice == "2":

write\_file(filename) elif choice == "3":

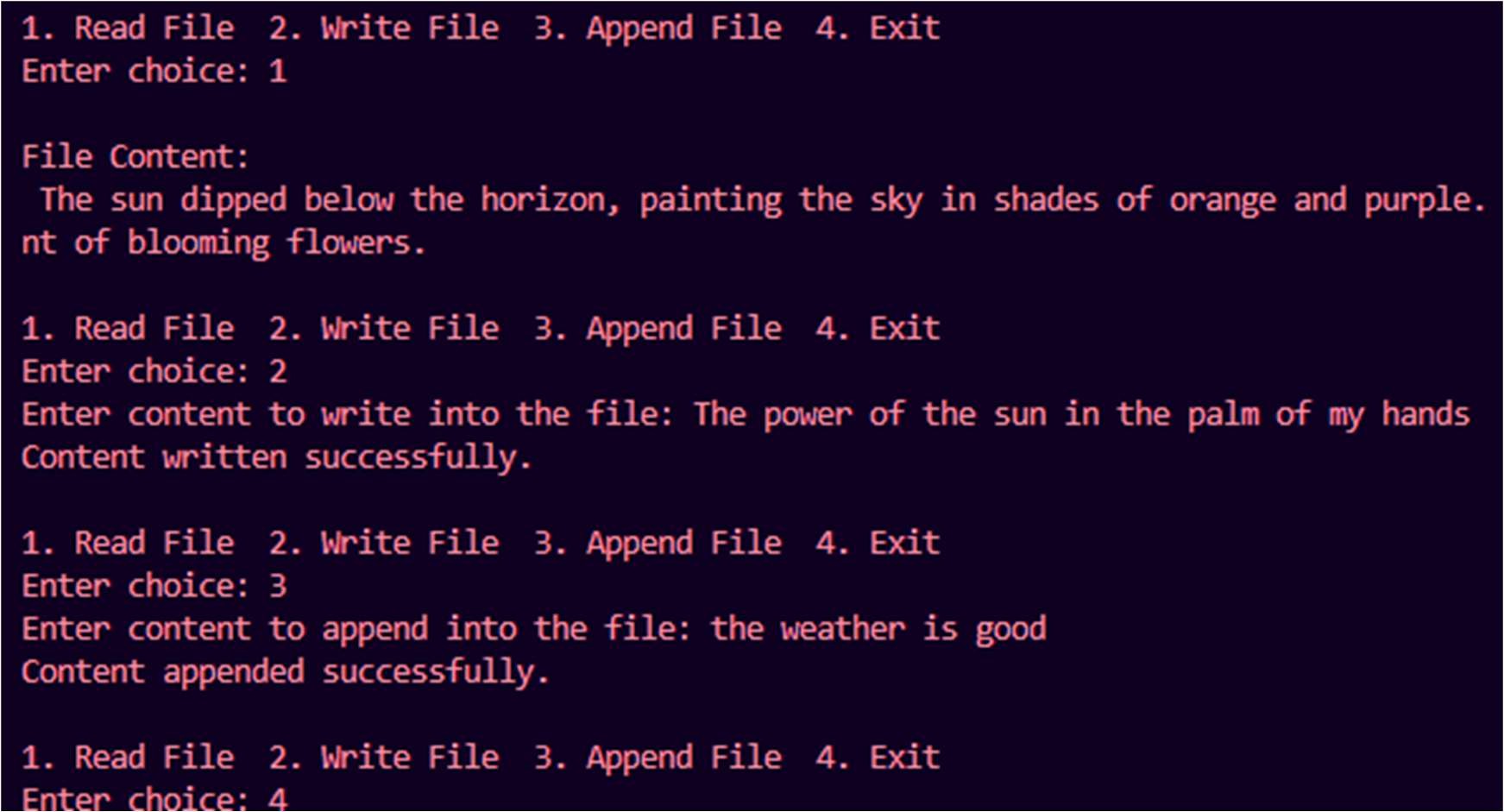
append\_file(filename) elif choice == "4":

break else:

print("Invalid choice. Try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()



Exp-5 Interactive Calculator

Mohammed Sadriwala, 28, Batch-2 def parse\_input(user\_input): input\_list = user\_input.split() if len(input\_list) != 3: raise Exception("Input must consist of three elements (e.g., '1 + 2')") n1, op, n2 = input\_list

try:

n1 = float(n1) n2 = float(n2) if not (n1.is\_integer() and n2.is\_integer()):

raise Exception("One of the two operands is not an integer") except ValueError: raise Exception("Both inputs must be numbers") return n1, op, n2 def calculate(n1, op, n2):

if op == '+':

return n1 + n2 elif op == '-': return n1 - n2 elif op == '\*': return n1 \* n2 elif op == '/': if n2 == 0:

raise Exception("Division by zero is not allowed") return n1 / n2 else: raise Exception(f"'{op}' is not a valid operator. Use +, -, \*, or /") while True:

user\_input = input('>>> ') if user\_input.lower() == "quit":

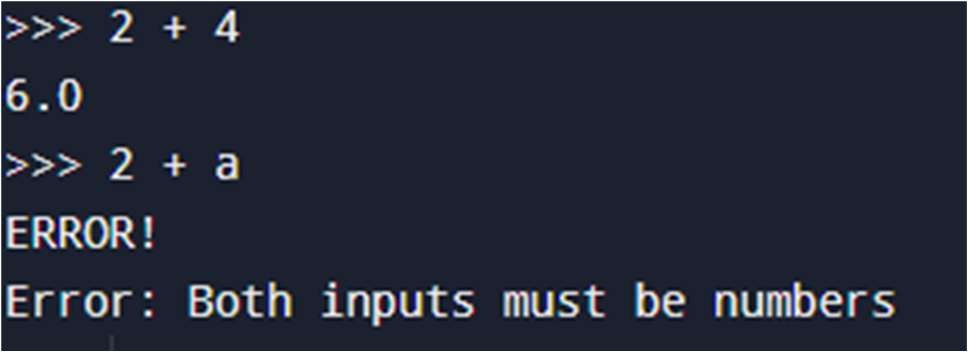
break try:

n1, op, n2 = parse\_input(user\_input)

result = calculate(n1, op, n2) print(result)

except Exception as e:

print(f"Error: {e}")



Guess the number

Mohammed Sadriwala, 28, Batch-2 class IncorrectNumberError(Exception): def \_\_init\_\_(self, message="The number you entered is incorrect!"):

self.message = message super().\_\_init\_\_(self.message) def number\_checker(): correct\_number = 42

while True:

try:

user\_input = int(input("Please enter the number: ")) if user\_input != correct\_number:

raise IncorrectNumberError

else:

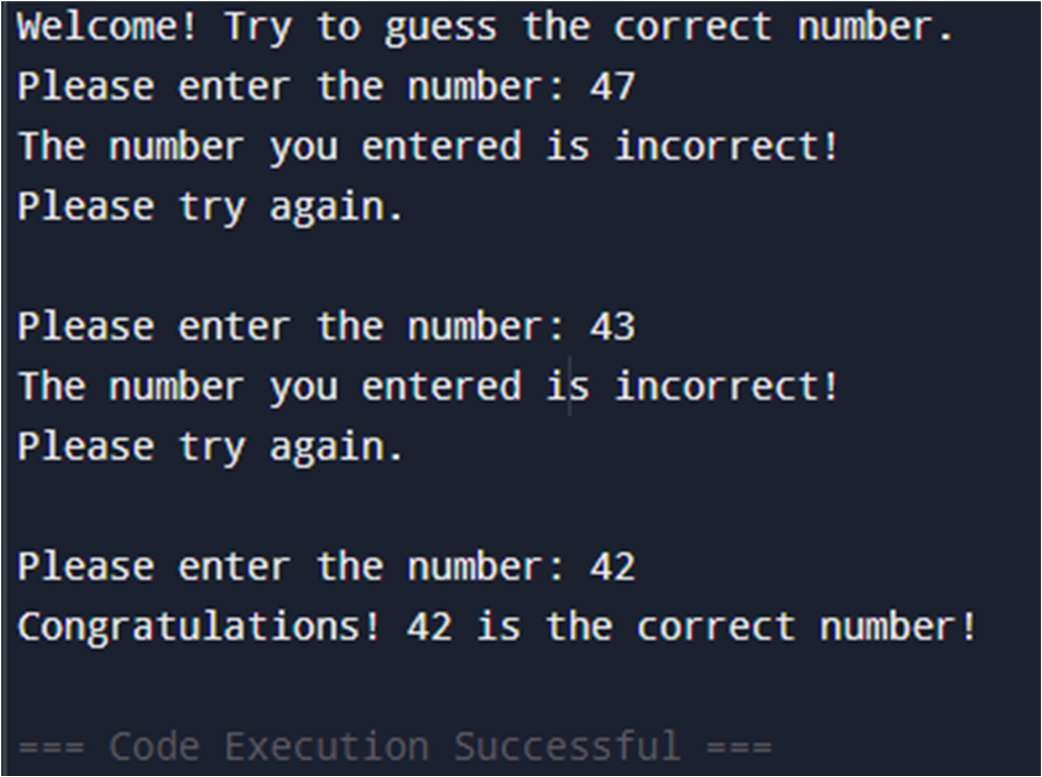
print(f"Congratulations! {correct\_number} is the correct number!")

break # Exit loop if correct except IncorrectNumberError as e:

print(e) print("Please try again.\n") except ValueError:

print("Invalid input! Please enter a valid integer.\n") if \_\_name\_\_ == "\_\_main\_\_":

print("Welcome! Try to guess the correct number.") number\_checker()



Validate name and age

Mohammed Sadriwala, 28, Batch-2

class InvalidAgeError(Exception):

"""Custom exception for invalid age input."""

pass

class InvalidNameError(Exception):

"""Custom exception for invalid name input."""

pass

def validate\_name(name):

if not name.replace(" ", "").isalpha():

raise InvalidNameError("Invalid name! Name should contain only alphabets.") def validate\_age(age):

if age < 0: raise InvalidAgeError("Age cannot be negative!")

elif age < 18:

raise InvalidAgeError("You are not eligible to vote. Must be at least 18 years old.") def main():

try:

name = input("Enter your name: ") validate\_name(name) # Validate name age = int(input("Enter your age: ")) validate\_age(age) # Validate age print(f"Hello {name}, you are eligible to vote!") except InvalidNameError as e:

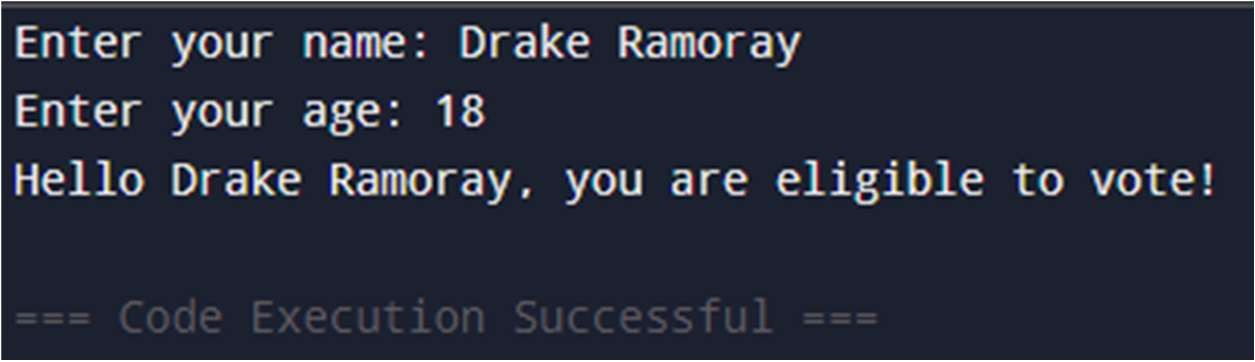
print(e) except InvalidAgeError as e:

print(e) except ValueError:

print("Invalid input! Age should be a valid number.")

if \_\_name\_\_ == "\_\_main\_\_":

main()



Program to demonstrate user defined exception Mohammed Sadriwala, 28, Batch-2 class InvalidMonthError(Exception):

"""Custom exception for invalid month number."""

pass

def get\_month\_name(month\_no):

months = {

1: "January", 2: "February", 3: "March", 4: "April",

5: "May", 6: "June", 7: "July", 8: "August",

9: "September", 10: "October", 11: "November", 12: "December"

}

if month\_no not in months: raise InvalidMonthError("Invalid month number! Please enter a number between 1 and 12.") return months[month\_no] def main():

try:

month\_no = int(input("Enter a month number (1-12): ")) month\_name = get\_month\_name(month\_no) print(f"The month is: {month\_name}") except InvalidMonthError as e:

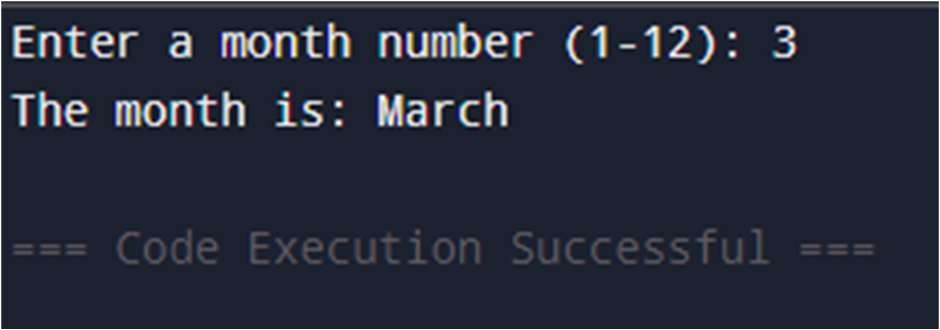
print(e)

except ValueError:

print("Invalid input! Please enter a numeric value.")

if \_\_name\_\_ == "\_\_main\_\_":

main()



Exp-6:

Mohammed Sadriwala, 28, Batch-2 Client: # A TCP/IP server that sends messge to client import socket

# take server name and port number host='localhost' port=5000

# create a socket at server side using TCP/IP protocol s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) print("Socket created") #bind the socket with server and port number

s.bind((host,port))

# allow maximum one client to the Socket

s.listen() print("waiting for connections...") # wait till a client accept a connection c,addr=s.accept() # display client address print("Connection from:",str(addr))

# send msg to client after encoding into binary string

c.send(b"Hellow client ! How are you") msg="Bye!"

c.send(msg.encode()) # disconnect the server c.close()

Sever:

#A TCP/IP client that recieves messge from server import socket

# take server name and port number host='localhost' port=5000

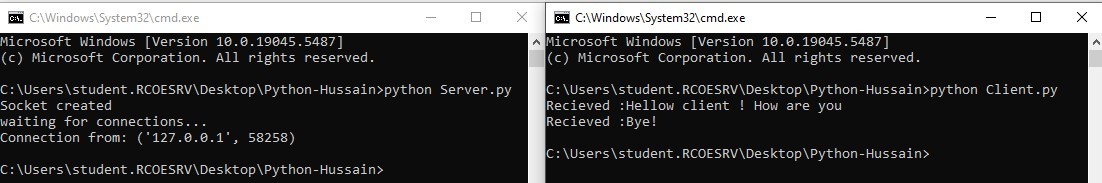
# create a socket at server side using TCP/IP protocol s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) # connect it to server and port number s.connect((host,port))

# recieve msg string from server at a time 1024 B msg=s.recv(1024)

# repeat as long as msg string are not empty while msg:

print("Recieved :"+msg.decode()) msg=s.recv(1024) # disconnect the server

s.close()



Write a python program to create simple socket for basic information exchange between server and client.

Mohammed Sadriwala, 28, Batch-2 Server 1:

import socket

def start\_server(): # Create a socket object server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Define host and port

host = '127.0.0.1' # Localhost address port = 12345 # Arbitrary port number # Bind the socket to the address and port server\_socket.bind((host, port))

# Enable the server to accept connections (max 5 clients in the waiting queue) server\_socket.listen(5) print(f"Server started on {host}:{port}. Waiting for client to connect...") try:

# Wait for a client to connect

client\_socket, client\_address = server\_socket.accept() print(f"Connection established with

{client\_address}") # Receive data from the client data = client\_socket.recv(1024).decode('utf-8') print(f"Received from client: {data}")

# Send a response back to the client response = "Hello from server!" client\_socket.send(response.encode('utf-8'))

# Close the connection client\_socket.close() except Exception as e:

print(f"Error: {e}") finally:

# Close the server socket

server\_socket.close() if name == ' main ':

start\_server()

Client 1:

import socket def start\_client(): # Create a socket object client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Define server details (localhost and port) host = '127.0.0.1' # Localhost address port = 12345 # Same port as the server try: # Connect to the server print(f"Connecting to server {host}:{port}...") client\_socket.connect((host, port))

# Send a message to the server message = "Hello, Server!" client\_socket.send(message.encode('utf-8')) print(f"Sent to server: {message}")

# Receive the server's response response = client\_socket.recv(1024).decode('utf-8') print(f"Received from server: {response}") except Exception as e: print(f"Error: {e}") finally:

# Close the connection

client\_socket.close() if name == ' main ': start\_client()

