**Problem 1: String Manipulation**

Write a Python program that takes a user-input string and checks if it contains only alphabetical

characters (no numbers or symbols). If the input is valid, print the string in uppercase; otherwise, print

an error message.

**Ans:**

def check\_alphabet(input\_string):

return input\_string.isalpha()

# Take user input

user\_input = input("Enter a string: ")

# Check if the input contains only alphabetical characters

if check\_alphabet(user\_input):

# Print the string in uppercase

uppercase\_string = user\_input.upper()

print("Uppercase:", uppercase\_string)

else:

print("Error: Input should contain only alphabetical characters.")

**Problem 2: Palindrome Checker**

Create a Python function that determines if a given word is a palindrome. Allow the user to input a

word, and then use your function to check and print whether the word is a palindrome or not.

**Ans:**

def check\_palindrome(word):

# Remove spaces and convert to lowercase for case-insensitive comparison

cleaned\_word = ''.join(word.split()).lower()

# Check if the cleaned word is equal to its reverse

return cleaned\_word == cleaned\_word[::-1]

# Take user input

user\_word = input("Enter a word: ")

# Check if the word is a palindrome and display the result

if check\_palindrome(user\_word):

print(f"{user\_word} is a palindrome!")

else:

print(f"{user\_word} is not a palindrome.")

**Problem 3: List Operations**

Write a Python program that takes a list of integers as input and calculates the sum of all prime

numbers in the list. Display the result.

**Ans:**

def is\_prime(number):

if number < 2:

return False

for i in range(2, int(number\*\*0.5) + 1):

if number % i == 0:

return False

return True

def sum\_of\_primes(numbers):

prime\_numbers = [num for num in numbers if is\_prime(num)]

return sum(prime\_numbers)

# Example: Take a list of integers as input

user\_input = input("Enter a list of integers separated by spaces: ")

input\_list = [int(num) for num in user\_input.split()]

# Calculate the sum of prime numbers in the list

result = sum\_of\_primes(input\_list)

# Display the result

print("Sum of Prime Numbers:", result)

**Problem 4: List Manipulation**

Create a function that accepts two lists of integers and returns a new list containing common elements between the two input lists. Display the resulting list.

**Ans.**

def common\_elements(list1, list2):

common\_elements = list(set(list1) & set(list2))

return common\_elements

# Example lists

list1 = input("Enter List1[]")

list2 = input("Enter List2[]")

# Find and display common elements

result = common\_elements(list1, list2)

print("Common Elements:", result)

**Problem 5: String Formatting**

Develop a program that reads a sentence from the user and replaces all occurrences of the word "not"

with "good" in the sentence. Print the modified sentence.

**Ans:**

def replace\_not\_with\_good(sentence):

# Split the sentence into words

words = sentence.split()

# Iterate through the words and replace "not" with "good"

modified\_words = [word if word.lower() != "not" else "good" for word in words]

# Join the modified words to form the modified sentence

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

return modified\_sentence

# Take user input

user\_sentence = input("Enter a sentence: ")

# Replace "not" with "good" in the sentence

modified\_sentence = replace\_not\_with\_good(user\_sentence)

# Display the modified sentence

print("Modified Sentence:", modified\_sentence)

**Problem 6: Loop and String Manipulation**

Write a Python program that takes a string as input and prints each character along with its ASCII

value on a new line.

**Ans:**

user\_input = input("Enter a string: ")

def ascii\_value\_of\_string(input\_string):

for char in input\_string:

ascii\_value = ord(char)

print(f" {char} {ascii\_value}")

# Display characters and their ASCII values

ascii\_value\_of\_string(user\_input)

**Problem 7: List Comprehension**

Create a Python program that generates a list of squares of even numbers from 1 to 10 using list

comprehension. Display the resulting list.

**Ans:**

v\_list=[i\*\*2 for i in range(1,10)if i%2==0]

print(v\_list)

**Problem 8: While Loop**

Implement a Python program that calculates the factorial of a user-input positive integer using a while

loop. Display the result.

**Ans:**

v\_given\_number=int(input("Enter Positive integer for factorial: "))

def cal\_factorial(v\_given\_number):

if v\_given\_number < 0:

return "Factorial is not defined for negative numbers."

elif v\_given\_number == 0 or v\_given\_number == 1:

return 1

else:

result = 1

while v\_given\_number > 1:

result \*= v\_given\_number

v\_given\_number -= 1

return result

cal\_factorial(0)

factorial\_result = cal\_factorial(v\_given\_number)

print(f"The factorial of {v\_given\_number} is {factorial\_result}")

**Problem 9: Nested Loops**

Write a program that prints a multiplication table for numbers 1 to 5. Use nested loops to achieve this.

**Ans:**

start = 1

end = 5

# Nested loop to print the multiplication table

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

for j in range(start, end+1): # Multiplying by numbers 1 to 5

product = i \* j

print(f"{i} \* {j} = {product}")

**Problem 10: Advanced String Formatting**

Design a program that takes a list of names and their corresponding ages. Display the information in a

formatted table with columns for names and ages.

**Ans:**

def formatting\_advance\_string(data):

# Print the table header

print("{:<15} {:<5}".format("Name", "Age"))

print("-" \* 20)

# Print each row of the table

for emp in data:

print("{:<15} {:<5}".format(emp["name"], emp["age"]))

# Example data: a list of dictionaries with names and ages

emp\_data = [

{"name": "Mr. Mamun", "age": 25},

{"name": "Mr. Abu taher", "age": 30},

{"name": "Mr. Abu Saleh", "age": 22},

{"name": "Mr. Tuhin", "age": 28},

]

# Display the information in a formatted table

formatting\_advance\_string(emp\_data)