Python Experiments Code

1. Python Program for factorial of a number

Code –

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result


number = int(input("Enter a number: "))
print("Factorial of", number, "is", factorial(number))
```

O/P-

```
Enter a number:
5
Factorial of 5 is 120
```

2. Python program to print all Prime numbers in an Interval

Code –

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True


def print_primes(start, end):
```

```
    for num in range(start, end + 1):
        if is_prime(num):
            print(num)

start = int(input("Enter the start of the interval: "))
end = int(input("Enter the end of the interval: "))
print("Prime numbers between", start, "and", end, "are:")
print_primes(start, end)
```

O/P-

```
Enter the start of the interval:
1
Enter the end of the interval:
50
Prime numbers between 1 and 50 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
```

43

47

3. Python Program for n-th Fibonacci number

Code –

```
def fibonacci(n):
    if n <= 0:
        return "Invalid input"
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        fib_sequence = [0, 1]
        for i in range(2, n):
            fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
        return fib_sequence[-1]

# Test the function
n = int(input("Enter the value of n: "))
print("The", n, "th Fibonacci number is:", fibonacci(n))
```

O/P-

Enter the value of n:

10

The 10 th Fibonacci number is: 34

4. Python Program for Sum of squares of first n natural numbers

Code –

```
def sum_of_squares(n):
    if n <= 0:
        return "Invalid input"
    else:
        return sum(i**2 for i in range(1, n+1))

# Test the function
n = int(input("Enter the value of n: "))
print("The sum of squares of the first", n, "natural numbers is:", sum_of_squares(n))
```

O/P-

```
Enter the value of n:
5
The sum of squares of the first 5 natural numbers is: 55
```

5. Python Program to find largest element in an array

Code-

```
def find_largest(arr):
    if not arr:
        return "Array is empty"
    else:
        max_element = arr[0]
        for num in arr:
            if num > max_element:
                max_element = num
        return max_element
```

```
# Test the function
array = [int(x) for x in input("Enter the elements of the array: ").strip('[]').split(',')]
print("The largest element in the array is:", find_largest(array))
```

O/P-

```
Enter the elements of the array:
[5,10,14,15,18]
The largest element in the array is: 18
```

6. Python program to find largest number in a list

Code-

```
def find_largest(numbers):
    if not numbers:
        return "List is empty"
    else:
        max_number = numbers[0]
        for num in numbers:
            if num > max_number:
                max_number = num
        return max_number

# Test the function
numbers = [int(x) for x in input("Enter the list of numbers: ").strip('[]').split(',')]
print("The largest number in the list is:", find_largest(numbers))
```

O/P-

```
Enter the list of numbers:
[4,10,14,12,15,144]
The largest number in the list is: 144
```

7. Python program to print all even numbers in a range

Code-

```
def print_even_numbers(start, end):
    if start % 2 != 0:
        start += 1
    for num in range(start, end + 1, 2):
        print(num)

# Test the function
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))
print("Even numbers between", start, "and", end, "are:")
print_even_numbers(start, end)
```

O/P-

```
Enter the start of the range:
0
Enter the end of the range:
20
Even numbers between 0 and 20 are:
0
2
4
6
8
10
12
14
```

16

18

20

8. Remove multiple elements from a list in Python

Code-

```
def remove_elements(lst, indices):
    indices.sort(reverse=True)  # Sort indices in descending order to avoid issues with shifting indexes
    for index in indices:
        del lst[index]
    return lst

# Take list from user
user_input = input("Enter the list of numbers: ")
# Extract the numbers between brackets and split them by comma
user_list = [int(x) for x in user_input.strip('[]').split(',')]

print("Original list:", user_list)

# Take indices to remove from user
indices_to_remove = [int(x) for x in input("Enter the indices of elements to remove: ").split(',')]
result = remove_elements(user_list, indices_to_remove)
print("List after removal:", result)
```

O/P-

Enter the list of numbers:

[1,4,5,6,8,9,3,6,2]

Original list: [1, 4, 5, 6, 8, 9, 3, 6, 2]

Enter the indices of elements to remove:

1,5,2

List after removal: [1, 6, 8, 3, 6, 2]

9. Break a list into chunks of size N in Python

Code-

```
def chunk_list(lst, chunk_size):
    return [lst[i:i + chunk_size] for i in range(0, len(lst), chunk_size)]


# Take list from user
user_input = input("Enter the list of numbers: ")
# Extract the numbers between brackets and split them by comma
user_list = [int(x) for x in user_input.strip('[]').split(',')]


chunk_size = int(input("Enter the chunk size: "))


# Call the function and print the result
chunks = chunk_list(user_list, chunk_size)
print("List after breaking into chunks of size", chunk_size, ":", chunks)
```

O/P-

Enter the list of numbers:

[1,5,6,8,9,3]

Enter the chunk size:

5

List after breaking into chunks of size 5 : [[1, 5, 6, 8, 9], [3]]

10. Python program to multiply two matrices

Code-

```
def matrix_multiply(matrix1, matrix2):
    rows1, cols1 = len(matrix1), len(matrix1[0])
    rows2, cols2 = len(matrix2), len(matrix2[0])

    if cols1 != rows2:
        return "Cannot multiply matrices. Number of columns in the first matrix must be equal to the number of rows in the second matrix."

    result = [[0 for _ in range(cols2)] for _ in range(rows1)]

    for i in range(rows1):
        for j in range(cols2):
            for k in range(cols1):
                result[i][j] += matrix1[i][k] * matrix2[k][j]

    return result

# Function to take a matrix from the user
def take_matrix():
    matrix = []
    rows = int(input("Enter the number of rows: "))
    cols = int(input("Enter the number of columns: "))

    print("Enter the elements of the matrix row-wise:")

    for i in range(rows):
```

```
        row = [int(x) for x in input().split()]
        if len(row) != cols:
            print("Invalid input. Please enter exactly", cols, "elements.")
            return None
        matrix.append(row)

    return matrix

# Test the function
print("Enter the first matrix:")
matrix1 = take_matrix()
print("Enter the second matrix:")
matrix2 = take_matrix()

if matrix1 is not None and matrix2 is not None:
    result = matrix_multiply(matrix1, matrix2)
    if isinstance(result, str):
        print(result)
    else:
        print("Result of matrix multiplication:")
        for row in result:
            print(row)
```

O/P-

```
Enter the first matrix:
Enter the number of rows:
3
Enter the number of columns:
```

3

Enter the elements of the matrix row-wise:

1 2 3

5 6 9

10 14 56

Enter the second matrix:

Enter the number of rows:

3

Enter the number of columns:

3

Enter the elements of the matrix row-wise:

10 12 14

45 56 58

4 5 6

Result of matrix multiplication:

[112, 139, 148]

[356, 441, 472]

[954, 1184, 1288]

11. Python program to check if a string is palindrome or not

Code-

```
def is_palindrome(s):
    return s == s[::-1]


string = input("Enter a string: ")
if is_palindrome(string):
    print("The string is a palindrome.")
else:
```

```
    print("The string is not a palindrome.")
```

O/P-

Enter a string:

ABBA

The string is a palindrome.

12. Ways to remove i’th character from string in Python

Code-

```
def remove_character(string, i):
    return string[:i] + string[i+1:]

string = input("Enter a string: ")
index = int(input("Enter the index of the character to remove: "))
if 0 <= index < len(string):
    result = remove_character(string, index)
    print("String after removing character at index", index, ":", result)
else:
    print("Index out of range.")
```

O/P-

Enter a string:

Collage

Enter the index of the character to remove:

2

String after removing character at index 2 : Colage

13. Python program to accept the strings which contains all vowels

Code-

```
def contains_all_vowels(s):
    vowels = set('aeiouAEIOU')
    return set(s.lower()).issuperset(vowels)

string = input("Enter a string: ")
if contains_all_vowels(string):
    print("The string contains all vowels.")
else:
    print("The string does not contain all vowels.")
```

O/P-

```
Enter a string:
Equatorial
The string does contains all vowels.
```

14. Remove all duplicates from a given string in Python

Code-

```
def remove_duplicates(string):
    return ''.join(sorted(set(string), key=string.index))
string = input("Enter a string: ")
result = remove_duplicates(string)
print("String after removing duplicates:", result)
```

O/P-

```
Enter a string:
College
String after removing duplicates: Coleg
```

15. Python program to split and join a string

Code-

```
def split_and_join(string):
    words = string.split()
    return '-'.join(words)

string = input("Enter a string: ")
result = split_and_join(string)
print("String after split and join:", result)
```

O/P-

```
Enter a string:
college degree
String after split and join: college-degree
```

16. Python – Replace duplicate Occurrence in String

Code-

```
def replace_duplicate_occurrence(string):
    seen = {}
    result = ''
    for char in string:
        if char not in seen:
            seen[char] = 1
            result += char
        else:
            seen[char] += 1
            if seen[char] == 2:
                result += str(seen[char] - 1)
```

```
    return result

string = input("Enter a string: ")
result = replace_duplicate_occurrence(string)
print("String after replacing duplicate occurrence:", result)
```

O/P-

```
Enter a string: College
String after replacing duplicate occurrence: Col1eg1
```

17. Python program to find the sum of all items in a dictionary

Code-

```
def sum_of_dictionary_items(dictionary):
    return sum(dictionary.values())

user_input = input("Enter a dictionary (in the format {'key1': value1, 'key2': value2, ...}): ")
dictionary = eval(user_input)
print(dictionary)
result = sum_of_dictionary_items(dictionary)
print("Sum of dictionary items:", result)
```

O/P-

```
Enter a dictionary (in the format {'key1': value1, 'key2': value2, ...}): {'a': 10, 'b': 20, 'c': 30}
{'a': 10, 'b': 20, 'c': 30}
Sum of dictionary items: 60
```

18. Python – Sort Dictionary key and values List

Code-

```
import json
def sum_of_dictionary_items(dictionary):
    total_sum = 0
    for value_list in dictionary.values():
        total_sum += sum(value_list)
    return total_sum


user_input = input("Enter a dictionary (in the format {'key1': [value1], 'key2': [value2], ...}):
")
dictionary = json.loads(user_input.replace("'", "\""))
print(dictionary)


result = sum_of_dictionary_items(dictionary)
print("Sum of dictionary items:", result)
```

O/P-

```
Enter a dictionary (in the format {'key1': [value1], 'key2': [value2], ...}):  {'a': [3, 2, 1], 'b': [6,
5, 4], 'c': [9, 8, 7]}
{'a': [3, 2, 1], 'b': [6, 5, 4], 'c': [9, 8, 7]}
Sum of dictionary items: 45
```

19. Python Dictionary to find mirror characters in a string

Code-

```
def find_mirror_characters(string):
    mirror_dict = {'b': 'd', 'd': 'b', 'p': 'q', 'q': 'p'}
    mirror_chars = [mirror_dict[char] if char in mirror_dict else char for char in string]
    return ''.join(mirror_chars)
```

```
string = input("Enter a string: ")
result = find_mirror_characters(string)
print("String after finding mirror characters:", result)
```

O/P-

Enter a string: bdpq

String after finding mirror characters: dbqp

20. Python – Adding Tuple to List and vice – versa

Code-

```
def tuple_to_list_and_list_to_tuple(data):
    if isinstance(data, tuple):
        return list(data)
    elif isinstance(data, list):
        return tuple(data)

data = eval(input("Enter a tuple or a list: "))
result = tuple_to_list_and_list_to_tuple(data)
print("Converted data:", result)
```

O/P-

Enter a tuple or a list: (1, 2, 3)

Converted data: [1, 2, 3]

21. Python – Convert Nested Tuple to Custom Key Dictionary

Code-

```
def nested_tuple_to_dict(nested_tuple):
    keys = nested_tuple[::2]
```

```
    values = nested_tuple[1::2]
    return {keys[i]: values[i] for i in range(len(keys))}

user_input = input("Enter a nested tuple: ")
nested_tuple = tuple(eval(user_input))
print(nested_tuple)

result = nested_tuple_to_dict(nested_tuple)
print("Dictionary from nested tuple:", result)
```

O/P-

```
Enter a nested tuple: ('a', 1, 'b', 2, 'c', 3)
('a', 1, 'b', 2, 'c', 3)
Dictionary from nested tuple: {'a': 1, 'b': 2, 'c': 3}
```

22. Python Program to print an Inverted Star Pattern

Code-

```
def inverted_star_pattern(rows):
    for i in range(rows, 0, -1):
        print("*" * i)
rows = int(input("Enter the number of rows: "))
inverted_star_pattern(rows)
```

O/P-

```
Enter the number of rows: 5
*****
****
***
```

**

*

23. Python Program to print double sided stair-case pattern

Code-

```
def double_sided_staircase(rows):
    for i in range(1, rows + 1):
        print('* ' * i)
    for i in range(rows - 1, 0, -1):
        print('* ' * i)

rows = int(input("Enter the number of rows: "))
double_sided_staircase(rows)
```

O/P-

```
Enter the number of rows: 5
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

24. Python program to convert time from 12 hour to 24 hour format

Code-

```
def convert_to_24_hour_format(time):
    if time[-2:] == "AM" and time[:2] == "12":
        return "00" + time[2:-2]
    elif time[-2:] == "AM":
        return time[:-2]
    elif time[-2:] == "PM" and time[:2] == "12":
        return time[:-2]
    else:
        return str(int(time[:2]) + 12) + time[2:-2]

time = input("Enter time in 12-hour format (hh:mm:ss AM/PM): ")
print("Time in 24-hour format:", convert_to_24_hour_format(time))
```

O/P-

```
Enter time in 12-hour format (hh:mm:ss AM/PM): 08:30:02 AM
Time in 24-hour format: 08:30:02
```

25. Python program to find difference between current time and given time

Code-

```
from datetime import datetime

def time_difference(current_time, given_time):
    FMT = '%H:%M:%S'
    delta = datetime.strptime(current_time, FMT) - datetime.strptime(given_time, FMT)
    return delta
current_time_input = input("Enter current time in 24-hour format (hh:mm:ss): ")
```

```
given_time_input = input("Enter given time in 24-hour format (hh:mm:ss): ")


difference = time_difference(current_time_input, given_time_input)
print("Difference between current time and given time:", difference)
```

O/P-

```
Enter current time in 24-hour format (hh:mm:ss):  10:00:00
Enter given time in 24-hour format (hh:mm:ss):  08:00:00
Difference between current time and given time: 2:00:00
```

26. Python – Get number of characters, words, spaces and lines in a file

Code-

Create file.txt file and run code

```
def count_file_details(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()
            num_characters = len(content)
            num_words = len(content.split())
            num_spaces = content.count(' ')
            num_lines = content.count('\n') + 1
        return num_characters, num_words, num_spaces, num_lines
    except FileNotFoundError:
        print("File not found. Please ensure the file is uploaded to the Colab environment.")

# Provide the file path
file_path = "file.txt"

# Call the function to count file details
characters, words, spaces, lines = count_file_details(file_path)

# Display the results
print("Number of characters:", characters)
print("Number of words:", words)
print("Number of spaces:", spaces)
print("Number of lines:", lines)
```

O/P-

```
Number of characters: 34
Number of words: 5
Number of spaces: 5
Number of lines: 1
```

27. Count number of lines in a text file in Python

Code-

```
def count_lines(file_path):
    try:
        with open(file_path, 'r') as file:
            num_lines = sum(1 for line in file)
        return num_lines
    except FileNotFoundError:
        print("File not found. Please ensure the file is uploaded to the Colab environment.")

# Provide the file path
file_path = "file.txt"

# Call the function to count lines
lines = count_lines(file_path)

# Display the result
print("Number of lines in the file:", lines)
```

O/P-

```
Number of lines in the file: 1
```

28. Python program to copy odd lines of one file to other

Code-

```
def copy_odd_lines(input_file, output_file):
    try:
        with open(input_file, 'r') as file_in, open(output_file, 'w') as file_out:
            lines = file_in.readlines()
            odd_lines = [line for i, line in enumerate(lines) if i % 2 != 0]
            file_out.writelines(odd_lines)
        print("Odd lines copied to", output_file)
    except FileNotFoundError:
        print("File not found. Please ensure the file is uploaded to the Colab environment.")


# Provide the input and output file paths
input_file = "file.txt"  # Change to your input file name
output_file = "output.txt"  # Change to your output file name


# Call the function to copy odd lines
copy_odd_lines(input_file, output_file)
```

O/P-

```
Odd lines copied to output.txt
```

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

Code-

```
def reverse_file_content(input_file, output_file):
    try:
        with open(input_file, 'r') as file_in:
            content = file_in.read()
            reversed_content = content[::-1]
        with open(output_file, 'w') as file_out:
```

```
            file_out.write(reversed_content)
        print("Content reversed and stored in", output_file)
    except FileNotFoundError:
        print("File not found. Please ensure the file is uploaded to the Colab environment.")

# Provide the input and output file paths
input_file = "file.txt"  # Change to your input file name
output_file = "reversed_file.txt"  # Change to your desired output file name

# Call the function to reverse file content
reverse_file_content(input_file, output_file)
```

O/P-

Content reversed and stored in reversed_file.txt

30. Python Program for Print Number series without using any loop

Code-

```
def print_numbers(n):
    if n <= 0:
        return
    else:
        print_numbers(n - 1)
        print(n, end=' ')

# Take input from the user
n = int(input("Enter the value of n: "))

# Call the function to print the number series
print_numbers(n)
```

O/P-

Enter the value of n: 25
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

31. Write a program to create bank account class with two attributes. Description: Write a class with 2 attributes(owner and balance). In this assignment you need to maintain a bank account where 2 operations need to be done repeatedly. First one is —deposit‖ and the other operation is —Withdraw‖. If the user selects the withdrawal operation, then you need to check whether the owner has sufficient bank balance or not.

Code-

```
class BankAccount:
    def __init__(self, owner, balance):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposit of", amount, "successful. New balance:", self.balance)

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print("Withdrawal of", amount, "successful. New balance:", self.balance)
        else:
            print("Insufficient balance. Withdrawal failed.")

# Example usage:
account = BankAccount("John Doe", 1000)
account.deposit(500)
account.withdraw(2000)
```

O/P-

```
Deposit of 500 successful. New balance: 1500
Insufficient balance. Withdrawal failed.
```

32. Python program to add two integers with handling expectations Write a Python program input and add two integers only and handle the exceptions. Problem Solution: In this program, we are reading two integers number from the user using int(input()) an handling the following exceptions, ValueError – Occurs when input value is not an integer. ZeroDivisionError – Occurs when divisor is zero. Exception – Any other error.

Code-

```
try:
   num1 = int(input("Enter first integer: "))
   num2 = int(input("Enter second integer: "))
   result = num1 + num2
   print("Sum:", result)
except ValueError:
   print("Error: Please enter integers only.")
except ZeroDivisionError:
   print("Error: Division by zero is not allowed.")
except Exception as e:
   print("An error occurred:", e)
```

O/P-

Enter first integer: 25
Enter second integer: 25.5
Error: Please enter integers only.

33. Write a program to add two numbers by taking these values as inputs and display the sum as the output.

Code-

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

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

sum = num1 + num2

print("Sum:", sum)

O/P-

Enter first number: 3.5

Enter second number: 2.5

Sum: 6.0

34. Write a program to take input of age and depending on the age output a pop-up message showing if the person is eligible to vote or not.

Code-

```
age = int(input("Enter your age: "))
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

O/P-

```
Enter your age: 25
You are eligible to vote.
```

35. Menu Driven program to create a simple calculator.

Code-

```
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error: Division by zero!"
    return x / y

print("Calculator Menu:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
```

```
choice = input("Enter your choice (1/2/3/4): ")

if choice in ('1', '2', '3', '4'):
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == '1':
        print("Result:", add(num1, num2))
    elif choice == '2':
        print("Result:", subtract(num1, num2))
    elif choice == '3':
        print("Result:", multiply(num1, num2))
    elif choice == '4':
        print("Result:", divide(num1, num2))
else:
    print("Invalid choice")
```

O/P-

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter your choice (1/2/3/4): 3
Enter first number: 25
Enter second number: 26
Result: 650.0

36. Creation of simple socket for basic information exchange between server and client.

No need we don't have socket in lab

37. Programs on Threading using python.

Code-

```
import threading
import time

def print_numbers():
```

```
    for i in range(1, 6):
        print(i)
        time.sleep(1)

def print_letters():
    for letter in ['a', 'b', 'c', 'd', 'e']:
        print(letter)
        time.sleep(1)

# Create two threads
t1 = threading.Thread(target=print_numbers)
t2 = threading.Thread(target=print_letters)

# Start both threads
t1.start()
t2.start()

# Wait for both threads to finish
t1.join()
t2.join()

print("Both threads have finished executing")
```

O/P-

```
1
a
2
b
3
c
4
d
5
e
Both threads have finished executing
```

38. Write a function in Python to read lines from a text file "notes.txt". Your function should find and display the occurrence of the word "the". For example: If the content of the file is: "India is the fastest-growing economy. India is looking for more investments around the globe. The whole world is looking at India as a great market. Most of the Indians can foresee the heights that India is capable of reaching." The output should be 5.

Code-

```
def count_word_occurrences(file_path, word):
    with open(file_path, 'r') as file:
        content = file.read()
        occurrences = content.lower().count(word.lower())
    return occurrences

file_path = "notes.txt"  # Replace this with the path to your file
word_to_search = "the"
result = count_word_occurrences(file_path, word_to_search)
print("Number of occurrences of '{}' in the file: {}".format(word_to_search, result))
```

O/P-

```
Number of occurrences of 'the' in the file: 5
```

39. Aditi has used a text editing software to type some text. After saving the article as WORDS.TXT, she realised that she has wrongly typed alphabet J in place of alphabet I everywhere in the article. Write a function definition for JTOI() in Python that would display the corrected version of entire content of the file WORDS.TXT with all the alphabets "J" to be displayed as an alphabet "I" on screen. Note: Assuming that WORD.TXT does not contain any J alphabet otherwise. Example: If Aditi has stored the following content in the file WORDS.TXT: WELL, THJS JS A WORD BY JTSELF. YOU COULD STRETCH THJS TO BE A SENTENCE The function JTOI() should display the following content: WELL, THIS IS A WORD BY ITSELF. YOU COULD STRETCH THIS TO BE A SENTENCE

Code-

```
def replace_j_with_i(file_path):
    with open(file_path, 'r') as file:
        content = file.read()
        corrected_content = content.replace('J', 'I').replace('j', 'i')
    return corrected_content

file_path = "WORDS.TXT"  # Replace this with the path to your file
```

```
corrected_content = replace_j_with_i(file_path)
print(corrected_content)
```

O/P-

WELL, THIS IS A WORD BY ITSELF. YOU COULD STRETCH THIS TO BE A SENTENCE

40. Write a function AMCount() in Python, which should read each character of a text file STORY.TXT, should count and display the occurance of alphabets A and M (including small cases a and m too). For Example: If the file content is as follows: Updated information As simplified by official websites. The EUCount() function should display the output as: A or a:4 M or m :

Code-

```
def count_am_occurrences(file_path):
    with open(file_path, 'r') as file:
        content = file.read()
        a_count = content.lower().count('a')
        m_count = content.lower().count('m')
    return a_count, m_count

file_path = "STORY.TXT"  # Replace this with the path to your file
a_count, m_count = count_am_occurrences(file_path)
print("A or a:", a_count)
print("M or m:", m_count)
```

O/P-

A or a: 4
M or m: 2