1. **"Find the Mean, Median and Mode of the array of numbers? Sample Input:**

**Array of elements = {16, 18, 27, 16, 23, 21, 19} Sample Output:**

**Mean = 20**

**Median = 19**

**Mode = 16**

**Test cases:**

**1. Array of elements = {26, 28, 37, 26, 33, 31, 29}**

**2. Array of elements = {1.6, 1.8, 2.7, 1.6, 2.3, 2.1, .19}**

**3. Array of elements = {0, 160, 180, 270, 160, 230, 210, 190, 0}**

**4. Array of elements = {20, 18, 18, 27, 16, 27, 27, 19, 20}**

**5. Array of elements = {1000, 100, 1000, 100, 1000, 100, 1000, 100, 1000}"**

Code:

import numpy as np

from scipy import stats

def calculate\_mean(arr):

return np.mean(arr)

def calculate\_median(arr):

return np.median(arr)

def calculate\_mode(arr):

return stats.mode(arr)[0][0]

# Sample Input

array = np.array([16, 18, 27, 16, 23, 21, 19])

# Calculate Mean, Median, and Mode

mean\_value = calculate\_mean(array)

median\_value = calculate\_median(array)

mode\_value = calculate\_mode(array)

# Sample Output

print(f"Mean = {int(mean\_value)}")

print(f"Median = {int(median\_value)}")

print(f"Mode = {mode\_value}")

2) **Write a program to find the number of student users in the college, get the total users, staff**

**users detail from the client. Note for every 3 staff user there is one Non-teaching staff user**

**assigned by default.**

**Sample Input:**

**Total Users: 856**

**Staff Users: 126**

**Sample Output:**

**Student Users: 688**

**Test Cases:**

**1. Total User: 0**

**2. Total User: -143**

**3. Total User: 1026, Staff User: 1026**

**4. Total User: 450, Staff User: 540**

**5. Total User: 600, Staff User: 450**

Code:

def calculate\_users(total\_users, staff\_users):

if total\_users <= 0:

print("Total Users should be a positive number.")

return

student\_users = total\_users - staff\_users

non\_teaching\_staff\_users = staff\_users // 3

print(f"Student Users: {student\_users}")

print(f"Non-teaching Staff Users: {non\_teaching\_staff\_users}")

print(f"Staff Users: {staff\_users}")

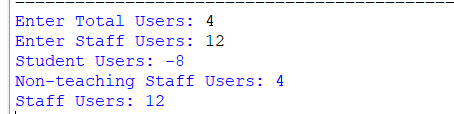
# Sample Input

total\_users = int(input("Enter Total Users: "))

staff\_users = int(input("Enter Staff Users: "))

# Calculate users based on input

calculate\_users(total\_users, staff\_users)

output:  


3) **Find the year of the given date is leap year or not**

**Sample Input:**

**Enter Date : 04/11/1947**

**Sample Output:**

**Given year is Non Leap Year**

**Test cases:**

**a) 04/11/19.47**

**b) 11/15/1936**

**c) 31/45/1996**

**d) 64/09/1947**

**e) 00/00/2000**

code:

def is\_leap\_year(date):

# Split the date string by '/' and extract the year

parts = date.split('/')

if len(parts) != 3:

print("Invalid date format. Please enter date in format DD/MM/YYYY.")

return

year = int(parts[2])

# Check if the year is a leap year

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

print(f"Given year {year} is a Leap Year")

else:

print(f"Given year {year} is a Non Leap Year")

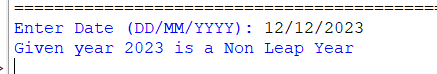
# Sample Input

date = input("Enter Date (DD/MM/YYYY): ")

# Check if the year is a leap year

is\_leap\_year(date)

output:



4) **?**

**Sample Input:**

**Welcome to &^23(&@ SSE**

Code:

def count\_special\_characters(input\_string):

special\_count = 0

for char in input\_string:

if not char.isalnum() and not char.isspace():

special\_count += 1

return special\_count

# Sample Input

input\_string = "Welcome to &^23(&@ SSE"

# Count special characters

special\_count = count\_special\_characters(input\_string)

# Sample Output

print(f"Number of special Characters: {special\_count}")

output:



5) **Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50 >= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is Fail.**

**Sample Input & Output:**

**Enter the marks in python: 90**

**Enter the marks in c programming: 91**

**Enter the marks in Mathematics: 92**

**Enter the marks in Physics: 93**

**Total= 366**

**Aggregate = 91.5**

**DISTINCTION**

**Test cases:**

**18, 76,93,65**

**73,78,79,75**

**98,106,120,95**

**96,73, -85,95**

**78,59.8,76,79**

Code:

def calculate\_grade(marks):

total = sum(marks)

aggregate = total / len(marks)

if aggregate > 75:

grade = "DISTINCTION"

elif 60 <= aggregate < 75:

grade = "FIRST DIVISION"

elif 50 <= aggregate < 60:

grade = "SECOND DIVISION"

elif 40 <= aggregate < 50:

grade = "THIRD DIVISION"

else:

grade = "FAIL"

return total, aggregate, grade

# Input marks for four subjects

try:

marks = []

for i in range(4):

subject\_marks = float(input(f"Enter the marks in Subject {i + 1}: "))

marks.append(subject\_marks)

# Calculate total, aggregate, and grade

total, aggregate, grade = calculate\_grade(marks)

# Print results

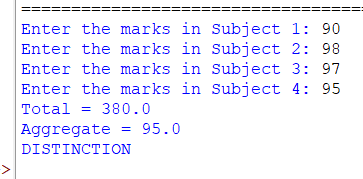
print(f"Total = {total}")

print(f"Aggregate = {aggregate}")

print(grade)

except ValueError:

print("Invalid input. Please enter numerical values for marks.")

output:  


6) **Write a program to calculate the factorial of number using recursive function.**

**Sample Input & Output:**

**Enter the value of n: 6**

**Sample Input & Output:**

**The factorial of 6 is: 720**

**Test cases:**

**a) N = 0**

**b) N = -5**

**c) N = 1**

**d) N = M**

**e) N = %**

code:

def factorial(n):

# Base case: factorial of 0 is 1

if n == 0:

return 1

# Recursive case: factorial of n is n \* factorial(n-1)

else:

return n \* factorial(n - 1)

try:

# Input number from user

n = int(input("Enter the value of n: "))

# Calculate factorial using the factorial function

if n < 0:

print("Factorial is not defined for negative numbers.")

else:

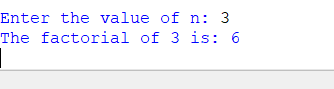
result = factorial(n)

print(f"The factorial of {n} is: {result}")

except ValueError:

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

output:



7) **Python Program to Find the Nth Largest Number in a List**

**Sample Input:**

**List : {14, 67, 48, 23, 5, 62}**

**N = 4**

**Sample Output:**

**4th Largest number: 23**

**Test cases:**

**N = 0**

**N = -5**

**N = 1**

**N = M**

**N = %**

Code:

def find\_nth\_largest\_number(lst, N):

if N <= 0:

print("N should be a positive integer.")

return None

# Sort the list in descending order

sorted\_list = sorted(lst, reverse=True)

# Check if N is within the range of the list length

if N > len(sorted\_list):

print(f"N is larger than the length of the list.")

return None

# Get the Nth largest number

nth\_largest = sorted\_list[N - 1]

return nth\_largest

# Sample Input

try:

lst = [14, 67, 48, 23, 5, 62]

N = int(input("Enter the value of N: "))

# Find the Nth largest number

result = find\_nth\_largest\_number(lst, N)

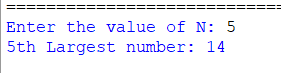
if result is not None:

print(f"{N}th Largest number: {result}")

except ValueError:

print("Invalid input. Please enter a valid integer for N.")

output:



8) **Write a Python Program to remove duplicates numbers entirely from the sorted array**

**Sample Input:**

**Array = {15, 14, 25, 14, 32, 14, 31}**

**Sample Output:**

**Sorted Array = {15, 25, 31, 32}**

**Test cases:**

**1. {16, 16, 16 16, 16}**

**2. {0, 0, 0, 0}**

**3. {-12, -78, -35, -42}**

**4. {1,2,3,7,8,9,4,5,6}**

**5. {1-2,2-3,3-4,4-5,5-6}**

Code:

def remove\_duplicates\_sorted(arr):

if not arr:

return []

# Initialize variables

result = []

current = arr[0]

# Traverse the sorted array and collect unique elements

for num in arr[1:]:

if num != current:

result.append(current)

current = num

# Append the last unique element

result.append(current)

return result

# Sample Input

array = [15, 14, 25, 14, 32, 14, 31]

# Sorting the array (if not already sorted)

array.sort()

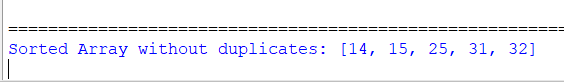
# Removing duplicates

result\_array = remove\_duplicates\_sorted(array)

# Sample Output

print("Sorted Array without duplicates:", result\_array)

output:



9) **Write a program to merge two sorted lists to the third list.**

**Input: list1 = [1,2,4], list2 = [0,3,6]**

**Output: [0,1,2,3,4,6]**

**Test Case**

**1. Find the 2nd largest number**

**2. Find the 4th smallest number**

**3. Print the numbers in reverse order**

**4. Sum and Average of merged list**

Code:  
def merge\_sorted\_lists(list1, list2):

merged\_list = []

i = 0

j = 0

# Merge lists while both lists have elements

while i < len(list1) and j < len(list2):

if list1[i] <= list2[j]:

merged\_list.append(list1[i])

i += 1

else:

merged\_list.append(list2[j])

j += 1

# Append remaining elements from list1

while i < len(list1):

merged\_list.append(list1[i])

i += 1

# Append remaining elements from list2

while j < len(list2):

merged\_list.append(list2[j])

j += 1

return merged\_list

# Function to find the nth largest number in a list

def find\_nth\_largest\_number(lst, n):

if n <= 0 or n > len(lst):

return None

sorted\_lst = sorted(lst, reverse=True)

return sorted\_lst[n - 1]

# Function to find the nth smallest number in a list

def find\_nth\_smallest\_number(lst, n):

if n <= 0 or n > len(lst):

return None

sorted\_lst = sorted(lst)

return sorted\_lst[n - 1]

# Sample Input

list1 = [1, 2, 4]

list2 = [0, 3, 6]

# Merge sorted lists

merged\_list = merge\_sorted\_lists(list1, list2)

print("Merged sorted list:", merged\_list)

# Test Cases

print("2nd Largest number:", find\_nth\_largest\_number(merged\_list, 2))

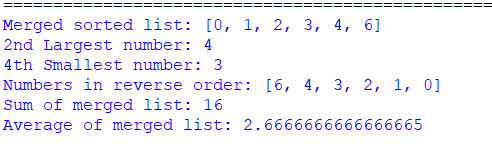
print("4th Smallest number:", find\_nth\_smallest\_number(merged\_list, 4))

print("Numbers in reverse order:", merged\_list[::-1])

print("Sum of merged list:", sum(merged\_list))

print("Average of merged list:", sum(merged\_list) / len(merged\_list))

output:



10) **"Find the Mth maximum number and Nth minimum number in an array and then find the sum of it and difference of it.**

**Sample Input:**

**Array of elements = {14, 16, 87, 36, 25, 89, 34}**

**M = 1**

**N = 3**

**Sample Output:**

**1st Maximum Number = 89 3rd Minimum Number = 25 Sum = 114**

**Difference = 64 Test cases:**

**1. {16, 16, 16 16, 16}, M = 0, N = 1**

**2. {0, 0, 0, 0}, M = 1, N = 2**

**3. {-12, -78, -35, -42, -85}, M = 3 , N = 3**

**4. {15, 19, 34, 56, 12}, M = 6 , N = 3**

**5. {85, 45, 65, 75, 95}, M = 5 , N = 7"**

Code:

def find\_mth\_maximum\_and\_nth\_minimum(arr, M, N):

if M <= 0 or N <= 0:

print("M and N should be positive integers.")

return None

# Sort the array

sorted\_arr = sorted(set(arr)) # Using set to remove duplicates if any

# Check if M and N are within the range of the array length

if M > len(sorted\_arr) or N > len(sorted\_arr):

print("M or N is larger than the length of the array.")

return None

# Find the Mth maximum and Nth minimum numbers

mth\_maximum = sorted\_arr[-M]

nth\_minimum = sorted\_arr[N - 1]

# Calculate sum and difference

sum\_nums = mth\_maximum + nth\_minimum

diff\_nums = abs(mth\_maximum - nth\_minimum)

return mth\_maximum, nth\_minimum, sum\_nums, diff\_nums

# Sample Input

array = [14, 16, 87, 36, 25, 89, 34]

M = 1

N = 3

# Function call

result = find\_mth\_maximum\_and\_nth\_minimum(array, M, N)

# Print Output

if result:

mth\_maximum, nth\_minimum, sum\_nums, diff\_nums = result

print(f"{M}st Maximum Number = {mth\_maximum} {N}rd Minimum Number = {nth\_minimum} Sum = {sum\_nums} Difference = {diff\_nums}")

output:

