## PROGRAM-1

**a. Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages.**

```
    studentname = input('Enter the Name of the student: ')
    usn = input('Enter the USN of the student: ')
    m1 = int(input('Enter the mark in first subject: '))
     m2 = int(input('Enter the mark in second subject: '))
    m3 = int(input('Enter the mark in third subject: ')) total = m1+m2+m3
    percentage = round((total/300)*100)

    print('\nName of the Student: ',
    studentname)print('USN: ', usn)
    print('Mark in Subject 1: ', m1)
    print('Mark in Subject 2: ', m2)
    print('Mark in Subject 3: ', m3)
    print('Total Score = ', total)
     print('Percentage = ', percentage)

    if percentage >=90:
    print('First Class Exemplary.')
elif percentage >=75 and percentage <90:
print('First Class with Distinction.')
elif percentage >=60 and percentage <75:
print('First Class.')
elif percentage >=35 and percentage <60:
print('Second Class.')
    else:
    print('Fail.')
```

output:

```
Enter student name: ARUN
Enter USN: 1DT20CS001

Enter mark in subject 1: 85
Enter mark in subject 2: 90
Enter mark in subject 3: 75

Student Details:
Name: ARUN
USN: 1DT20CS001

Total Marks: 250.0
Percentage: 83.33%
```

**b. Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.**

```
personname = input('Enter the Name of the
person: ')yearofbirth = int(input('Enter the
Year of Birth: ')) currentyear = 2023
age = currentyear - yearofbirth

print('\nName of the person: ',
personname) print('Year of Birth of the
person: ', yearofbirth)print('Age of the
person: ', age)

if age >= 60:
print('The person is a Senior
Citizen.')else:
print('The person is not a Senior Citizen.')
```

Output:

```
Enter the Name of the
person: SabaEnter the Year
of Birth: 1986

Name of the person:  Saba
Year of Birth of the person:
1986Age of the person:  37
The person is not a Senior Citizen.
```

## PROGRAM-2

**a. Develop a program to generate Fibonacci sequence of length (N). Read N from the console.**

```
f0 = 0
f1 = 1
N = int(input('Enter the Length of required Fibonacci Sequence: '))

if N<=0:
   print ('Enter Positive Integer value: ')
else:
      i=0
      print('Fibonacci Sequence for N = '+ str(N) + ' is: ')
while i<N:
      print(f0)
      fth = f0 + f1
      f0 = f1
      f1 = fth
      i += 1
```

**Output:**

```
Enter the Length of required Fibonacci
Sequence: 10Fibonacci Sequence for N =
10 is:
0
1
1
2
3
5
8
13
21
34
```

**b. Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N and R).**

```
# function to calculate the factorial of a number
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

# read N and R from the console
N = int(input("Enter the value of N: "))
R = int(input("Enter the value of R: "))

# calculate the binomial coefficient
binomial_coef = factorial(N) // (factorial(R) * factorial(N-R))

# print the result
print("The binomial coefficient of", N, "and", R, "is:", binomial_coef)
```

output:

```
Enter the value of N: 7
Enter the value of R: 3
The binomial coefficient of 7 and 3 is: 35
```

## PROGRAM-3

**Read N numbers from the console and create a list. Develop a program to print mean, variance and standard deviation with suitable messages.**

```
import math
# read N from the console
N = int(input("Enter the number of elements: "))
# read N numbers from the console and create a list
num_list = []
for i in range(N):
    num = float(input(f"Enter element {i+1}: "))
    num_list.append(num)

# calculate mean
mean = sum(num_list) / N

# calculate variance
variance = sum([((num - mean) ** 2) for num in num_list]) / N
# calculate standard deviation
std_dev = math.sqrt(variance)
# print the results
print("The list is:", num_list)
print("The mean is:", mean)
print("The variance is:", variance)
print("The standard deviation is:", std_dev)
```

output:

```
Enter the number of elements: 5
Enter element 1: 3.2
Enter element 2: 4.8
Enter element 3: 2.5
Enter element 4: 6.1
Enter element 5: 5.9
The list is: [3.2, 4.8, 2.5, 6.1, 5.9]
The mean is: 4.5
The variance is: 1.6900000000000002
The standard deviation is: 1.299038105676658
```

## PROGRAM-4

**4. Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.**

```
# read the number as a string from the console
num_str = input("Enter a multi-digit number: ")

# initialize a dictionary to store the frequency of each digit
digit_freq = {}

# loop through each character in the string and update the frequency count
for char in num_str:
    if char.isdigit():
        if char in digit_freq:
            digit_freq[char] += 1
        else:
            digit_freq[char] = 1

# print the frequency of each digit
print("The frequency of each digit in the number is:")
for digit in digit_freq:
    print(f"{digit}: {digit_freq[digit]}")
```

output:

```
Enter a multi-digit number: 1234567890
The frequency of each digit in the number is:
1: 1
2: 1
3: 1
4: 1
5: 1
6: 1
7: 1
8: 1
9: 1
0: 1
```

## PROGRAM-5

**Develop a program to print 10 most frequently appearing words in a text file. [Hint: Use dictionary with distinct words and their frequency of occurrences. Sort the dictionary in the reverse order of frequency and display dictionary slice of first 10 items]**

```
# read the file name from the console
filename = input("Enter the file name: ")

# open the file and read its contents
f=open("exp.txt","r")
text = f.read()

# remove punctuation and convert to lowercase
text = text.lower()
for char in '.,?!:;()':
    text = text.replace(char, '')

# split the text into words
words = text.split()

# initialize a dictionary to store the frequency of each word
word_freq = {}

# loop through each word and update the frequency count
for word in words:
    if word in word_freq:
        word_freq[word] += 1
    else:
        word_freq[word] = 1

# sort the dictionary by frequency in descending order
sorted_freq = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)

# print the 10 most frequently appearing words
print("The 10 most frequently appearing words are:")
for word, freq in sorted_freq[:10]:
    print(f"{word}: {freq}")
```

**output:**

```
Enter the file name: sample.txt
The 10 most frequently appearing words are:
the: 6
and: 5
to: 5
in: 4
of: 4
a: 3
is: 3
that: 3
with: 3
```

## PROGRAM-6

**Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Hint: Use string methods strip(), len(), list methods sort(), append(), and file methods open(), readlines(), and write()].**

```
# read the input and output file names from the console
input_filename = input("Enter the input file name: ")
output_filename = input("Enter the output file name: ")

# open the input file and read its contents into a list
with open("exp.txt", "r") as input_file:
    lines = input_file.readlines()

# remove leading and trailing whitespace from each line
lines = [line.strip() for line in lines]

# sort the lines in alphabetical order
lines.sort()

# open the output file and write the sorted lines to it
with open(output_filename, "w") as output_file:
    for line in lines:
        output_file.write(line + "\n")

print("File has been sorted and written to", output_filename)
```

Here's how the program works:

1. We first read the input and output file names from the console using the **input()** function.
2. We then open the input file using the **open()** function in read mode (**"r"**) and read its contents into a list using the **readlines()** method.
3. We remove leading and trailing whitespace from each line using the **strip()** method.
4. We sort the lines in alphabetical order using the **sort()** method.
5. We open the output file using the **open()** function in write mode (**"w"**) and write the sorted lines to it using a **for** loop and the **write()** method. Note that we add a newline character (**"\n"**) after each line to separate them.
6. Finally, we print a message to the console to confirm that the file has been sorted and written to the output file.

## PROGRAM-7

**Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.**

```
import zipfile
import os

# set the name of the folder to back up and the name of the ZIP file
folder_name = "my_folder"
zip_file_name = "my_folder_backup.zip"

# create a new ZIP file in write mode
zip_file = zipfile.ZipFile(zip_file_name, "w")

# walk the directory tree and add each file to the ZIP file
for foldername, subfolders, filenames in os.walk(folder_name):
    for filename in filenames:
        # get the full path of the file
        file_path = os.path.join(foldername, filename)
        # add the file to the ZIP file
        zip_file.write(file_path, os.path.relpath(file_path, folder_name))

# close the ZIP file
zip_file.close()

print("Backup complete. Created", zip_file_name)
```

Here's how the program works:

1. We first set the name of the folder to back up and the name of the ZIP file.
2. We create a new ZIP file in write mode using the **ZipFile()** function from the **zipfile** module.
3. We use the **os.walk()** function to recursively walk the directory tree of the folder to back up. For each file found, we get its full path using the **os.path.join()** function and add it to the ZIP file using the **write()** method. Note that we use the **os.path.relpath()** function to get the relative path of the file with respect to the folder to back up, so that the ZIP file retains the directory structure.
4. Finally, we close the ZIP file and print a message to the console to confirm that the backup is complete.

Note that this program only backs up files, not directories. If you want to back up directories as well, you can modify the program to add them to the ZIP file using the **write()** method with the **arcname** argument set to the directory name (i.e., **zip_file.write(directory_path, arcname=directory_name)**).

## PROGRAM-8

**Write a function named DivExp which takes TWO parameters a, b and returns a value c (c=a/b). Write suitable assertion for a>0 in function DivExp and raise an exception for when b=0. Develop a suitable program which reads two values from the console and calls a function DivExp.**

```
def DivExp(a, b):
    assert a > 0, "Value of 'a' must be greater than zero"
    if b == 0:
        raise ValueError("Value of 'b' cannot be zero")
    c = a / b
    return c

# read two values from the console
a = float(input("Enter value for 'a': "))
b = float(input("Enter value for 'b': "))

# call the DivExp() function with the input values
try:
    result = DivExp(a, b)
    print("Result:", result)
except AssertionError as e:
    print("Error:", e)
except ValueError as e:
    print("Error:", e)
```

Here's how the program works:

1. We define the **DivExp()** function with two parameters **a** and **b**. The function first checks the assertion **a > 0** using the **assert** statement, which raises an **AssertionError** if the assertion is false. Next, the function checks whether **b** is zero and raises a **ValueError** if it is. If both checks pass, the function calculates **c** as the quotient of **a** and **b** and returns it.
2. We use the **input()** function to read two values from the console and store them in variables **a** and **b**.
3. We call the **DivExp()** function with the input values using a **try** block. If the function call raises an **AssertionError** or a **ValueError**, the corresponding **except** block is executed to print an error message to the console. If the function call succeeds, the result is printed to the console.

## PROGRAM-9

**Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers. Define a suitable class 'Complex' to represent the complex number. Develop a program to read N (N >=2) complex numbers and to compute the addition of N complex numbers.**

```
class Complex:
    def __init__(self, real=0, imag=0):
        self.real = real
        self.imag = imag

    def __add__(self, other):
        return Complex(self.real + other.real, self.imag + other.imag)

    def __str__(self):
        return f"{self.real} + {self.imag}i"

def add_complex(a, b):
    return a + b

n = int(input("Enter the number of complex numbers: "))
complex_list = []

for i in range(n):
    real = float(input(f"Enter real part of complex number {i+1}: "))
    imag = float(input(f"Enter imaginary part of complex number {i+1}: "))
    complex_list.append(Complex(real, imag))

result = complex_list[0]

for i in range(1, n):
    result = add_complex(result, complex_list[i])

print("Sum of complex numbers:", result)
```

1. We define a **Complex** class with two attributes **real** and **imag** to represent a complex number. The **__add__()** method is defined to add two complex numbers using the **+** operator, and the **__str__()** method is defined to return a string representation of the complex number.
2. We define a **add_complex()** function that takes two **Complex** objects and returns a new **Complex** object representing the sum of the two complex numbers.
3. We read the number of complex numbers **n** from the console using the **input()** function, and create an empty list **complex_list** to store the **n** complex numbers.
4. We use a **for** loop to read the real and imaginary parts of each complex number from the console, create a new **Complex** object with the input values, and append it to the **complex_list**.
5. We initialize the **result** variable to the first complex number in the list, and use another **for** loop to add the remaining **n-1** complex numbers to the **result** using the **add_complex()** function.
6. We print the sum of the **n** complex numbers to the console.

## PROGRAM-10

**Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details. [Hint: Use list to store the marks in three subjects and total marks. Use __init__() method to initialize name, USN and the lists to store marks and total, Use getMarks() method to read marks into the list, and display() method to display the score card details.]**

```
class Student:
    def __init__(self, name, usn):
        self.name = name
        self.usn = usn
        self.marks = []
        self.total = 0

    def getMarks(self):
        for i in range(3):
            mark = float(input(f"Enter marks in subject {i+1}: "))
            self.marks.append(mark)
            self.total += mark

    def display(self):
        print("-----------------------------------------------")
        print("                 Score Card                  ")
        print("-----------------------------------------------")
        print(f"Name: {self.name}")
        print(f"USN: {self.usn}")
        print("Marks:")
        for i, mark in enumerate(self.marks):
            print(f"Subject {i+1}: {mark}")
        print(f"Total: {self.total}")
        percentage = self.total/3
        print(f"Percentage: {percentage:.2f}%")
        print("-----------------------------------------------")
# Main program
name = input("Enter name: ")
usn = input("Enter USN: ")

student = Student(name, usn)
student.getMarks()
student.display()
```

Here's how the program works:

1. We define a **Student** class with four attributes: **name**, **usn**, **marks**, and **total**. The **__init__()** method is defined to initialize the **name**, **usn**, and the empty **marks** list and **total** to 0. The **getMarks()** method is defined to prompt the user to enter marks in three subjects, store them in the **marks** list and calculate the **total**. The **display()** method is

defined to display the score card details including name, USN, marks in each subject, total marks and percentage.

2. We prompt the user to enter the **name** and **usn** using the **input()** function.
3. We create a new **Student** object with the input **name** and **usn**.
4. We call the **getMarks()** method of the **student** object to read marks in three subjects from the console.
5. We call the **display()** method of the **student** object to display the score card details to the console.