1)Write a program to short list in descending order

```
list1=[1,4,3,5,6,7,8,]
sorted_list=sorted(list1,reverse=True)
print("the sorted array",sorted_list)
```

2)write a program to remove all accurance of a specified element from the list

```
list1=[1,2,2,3,2,4,2,5,2,6,2,7,3]
element_to_remove=2
list1=[x for x in list1 if x !=element_to_remove]
print("the list after removing removing element",list1)
```

3)concanate of two numbers

```
string1=[1,2,3]
string2=[4,5,6]
string3=[string1+string2]
print("the concanate two numbers",string3)
```

4)write A program sum square to print the list of odd and even numbers

```
num=[int(x) for x in input("enter the numbers:").split()]
even_square=sum(x**2 for x in num if x%2 == 0)
odd_square=sum(x**2 for x in num if x%2 !=0)
print("the even square of numbers:",even_square)
print("the odd square of numbers:",odd_square)
```

5)Write a program to provide bonus mark if the category is sports
in python

```
category = input("Enter the category: ")
current_marks = float(input("Enter the current marks: "))
bonus_mark = 5 if category.lower() == "sports" else 0
total_marks = current_marks + bonus_mark
print("Total marks with bonus:", total_marks)
```

6)Sum of n numbers in python

```
n = int(input("Enter the number of elements: "))
total_sum = sum(float(input()) for _ in range(n))
print("Sum of the entered numbers:", total_sum)
```

7)sum of the digits

```
num=int(input("entr the num:"))
digit_sum=sum(int(digit)for digit in str(abs(num)))
print("sum of digits:",digit_sum)
```

8) write a program revers of given number

```
num=int(input("enter theb number:"))
reverse_num=input(str(abs(num))[::-1])
print("the reversed number:",reversed_num)
```

9)Write a program to find the factorial of a given number

```
def factorial (num):
    return 1 if num==0 else num*factorial(num-1)
num=int(input("enter the number:"))
result=factorial(num)
print("factorial of {} is:{}".format(num,result))
```

10)Write a program to print all the Non-Prime numbers between A and B? Sample Input: A = 12 B = 19

Sample Output:
14, 15, 16, 18

```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
A = int(input("Enter the value of A: "))
B = int(input("Enter the value of B: "))
```

```
non_prime_numbers = [num for num in range(A, B + 1) if not is_prime(num)]
print("Non-prime numbers between {} and {}: {}".format(A, B,non_prime_numbers))
```

11)Find the year of the given anniversary is sleep or not. If leap year, then print the next Anniversary, if not leap year then print the previous Anniversary. Sample Input:
Enter Date: 04/11/1947 Sample Output:
Given Anniversary Year: Non Leap Year. Anniversary Date: 04/11/1946

```
anniversary_date = input("Enter Date (DD/MM/YYYY): ")
anniversary_year = int(anniversary_date.split('/')[-1])
next_anniversary_year = anniversary_year + 1 if anniversary_year % 4 == 0 and (anniversary_year % 100 != 0 or anniversary_year % 400 == 0) else anniversary_year - 1
leap_status = "Leap Year" if next_anniversary_year > anniversary_year else "Non Leap Year"
print("Given Anniversary Year: {}. Anniversary Date: {}/{}".format(leap_status, anniversary_date[:-4], next_anniversary_year)
```

12) Write a program to print the given number is Perfect number or not?

```
Number = int(input(" Please Enter any Number: "))
Sum = 0
for i in range(1, Number):
    if(Number % i == 0):
        Sum = Sum + i
        if (Sum == Number):
            print(" %d is a Perfect Number" %Number)
        else:
            print(" %d is not a Perfect Number" %Number)
```

13)Write a program to find the sum of digits of N digit number (sum should be single digit)
Sample Input: Enter N value : 3 Enter 3 digit number: 143
Sample Output: Sum of 3 digit number: 8

```
N = int(input("Enter N value: "))
number = int(input("Enter {} digit number: ".format(N)))
sum_of_digits = number % 9 or 9
print("Sum of {} digit number: {}".format(N, sum_of_digits))
```

14)Program to find whether the given number is Armstrong number or not Sample Input: Enter number: 153
Sample Output: Given number is Armstrong number

```
def is_armstrong_number(number):
    num_digits = len(str(number))
    total = sum(int(digit) ** num_digits for digit in str(number))
    return total == number
number = int(input("Enter number: "))
if is_armstrong_number(number):
    print("Given number is Armstrong number")
else:
    print("Given number is not Armstrong number")
```

15)Program to find whether the given number is Harshad number or not Sample Input: Enter number: 21
Sample Output: Given number is Harshad number

```
def is_harshad_number(number):
    digit_sum = sum(int(digit) for digit in str(number))
    return number % digit_sum == 0
number = int(input("Enter number: "))
if is_harshad_number(number):
    print("Given number is Harshad number")
else:
    print("Given number is not Harshad number")
```

16)Program to find whether the given number is Happy number or not Sample Input: Enter number: 19
Sample Output: Given number is happy number

```
def is_happy_number(number):
    seen_numbers = set()
    while number != 1 and number not in seen_numbers:
        seen_numbers.add(number)
        number = sum(int(digit)**2 for digit in str(number))
    return number == 1
number = int(input("Enter number: "))
if is_happy_number(number):
    print("Given number is happy number")
```

```
else:
    print("Given number is not happy number")
```

17)Program to find whether the given number is Tech number or not Sample Input: Enter number: 3025
Sample Output: Given number is Tech number

```
def is_tech_number(number):
    digits = [int(digit) for digit in str(number)]
    digit_sum = sum(digits)
    digit_product = 1 if 0 in digits else 1
    for digit in digits:
        digit_product *= digit
    return digit_sum == digit_product
number = int(input("Enter number: "))
if is_tech_number(number):
    print("Given number is Tech number")
else:
    print("Given number is not Tech number")
```

18)Write a program using function to calculate the simple interest. Suppose the customer is a senior citizen. She is being offered 15 percent rate of interest; he is being offered 12 percent rate of interest for all other customers, the ROI is 10 percent. Sample Input Enter the principal amount: 200000
Enter the no of years: 3 Gender (m/f): m
Is customer senior citizen (y/n): n Sample Output:
Interest: 60000

```
principal = float(input("Enter the principal amount: "))
years = int(input("Enter the number of years: "))
gender = input("Gender (m/f): ").lower()
is_senior_citizen = input("Is the customer a senior citizen (y/n): ").lower() == 'y'
rate_of_interest = 0.15 if is_senior_citizen else 0.12 if gender == 'm' else 0.1
interest = principal * rate_of_interest * years
print("Interest:", interest)
```

19)Find the number of factors for the given number and print the 1st N factors of the given number.
Sample Input: Given number: 100
N: 4Sample Output: Number of factors = 91st 4 factors are: 1, 2, 4, 5

```
number = int(input("Given number: "))
N = int(input("N: "))
factors = [i for i in range(1, number + 1) if number % i == 0]
print("Number of factors =", len(factors))
print("1st {} factors are: {}".format(N, ', '.join(map(str, factors[:N]))))
```

20)Write a program to print number of factors and to print nth factor of the given number.
Sample Input: Given Number: 100
N =4Sample Output:Number of factors = 9 4th factor of 100 = 5

```
number = int(input("Given number: "))
N = int(input("N: "))
factors = [i for i in range(1, number + 1) if number % i == 0]
print("Number of factors =", len(factors))
print("1st {} factors are: {}".format(N, ', '.join(map(str, factors[:N]))))
```

21)Write a program to print unique permutations of a given number Sample Input: Given Number: 143 Sample Output:
Permutations are:
134143314341413431

```
from itertools import permutations
def unique_permutations(given_number):
    unique_permutations_set = set(permutations(str(given_number)))
    print("Permutations are:")
    for perm in unique_permutations_set:
        print(''.join(perm))
given_number = int(input("Given Number: "))
unique_permutations(given_number)
```

22)Write a program to find the square, cube of the given decimal number Sample Input: Given Number: 0.6
Sample Output: Square Number: 0.36 Cube Number:0.216

```
given_number = float(input("Given Number: "))
square_number = given_number ** 2
cube_number = given_number ** 3
print("Square Number:", square_number)
print("Cube Number:", cube_number)
```

23) Write a program to convert the Binary to Decimal, Octal Sample Input: Given Number: 1101 Sample Output:
Decimal Number: 13 Octal: 15

```
binary_number = input("Given Number: ")
decimal_number = int(binary_number, 2)
octal_number = oct(decimal_number)
print("Decimal Number:", decimal_number)
print("Octal:", octal_number[2:])
```

24)Add BinaryGiven two binary strings a and b, return their sum as a binary string. a and b consist only of '0' or '1' characters.
Each string does not contain leading zeros except for the zero itself. **Test cases:** 1.Input: a = "11", b = "1"

```
a, b = "11", "1"
result = bin(int(a, 2) + int(b, 2))[2:]
print("Input:", a, b)
print("Output:", result)
```

25)Python program to find the greatest of three binary numbers

```
binary1 = input("Enter the first binary number: ")
binary2 = input("Enter the second binary number: ")
binary3 = input("Enter the third binary number: ")

greatest_binary = max(binary1, binary2, binary3)
print("The greatest binary number is:", greatest_binary)
```

26)Write a program for matrix multiplication? Sample Input:
Mat1 =
Mat2 =Sample Output: Mat Sum =

```
mat1 = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

mat2 = [
    [9, 8, 7],
    [6, 5, 4],
    [3, 2, 1]
]

result_matrix = [[mat1[i][j] + mat2[i][j] for j in range(len(mat1[0]))] for i in range(len(mat1))]

print("Mat Sum =", result_matrix)
```

27)Find the LCM and GCD of n numbersSample Input: N value = 2 Number 1 = 16 Number 2 = 20Sample Output: LCM = 80 GCD = 4

```
import math
N = int(input("N value: "))
numbers = [int(input(f"Number {i + 1} = ")) for i in range(N)]
lcm_result = math.lcm(*numbers)
gcd_result = math.gcd(*numbers)
print("LCM =", lcm_result)
print("GCD =", gcd_result)
```

28) Program to find row, column and diagonal sum in Matrix a = [[1, 2, 3],[4, 5, 6,[7, 8, 9]] o/p:Sum of 1 row: 6 Sum of 2 row: 15 Sum of 3 row: 24 Sum of 1 column: 12 Sum of 2 column: 15 Sum of 3 column: 18 Diagonal sum 15

```
 matrix = [

    [1, 2, 3],

    [4, 5, 6],

    [7, 8, 9]

]

row_sums = [sum(row) for row in matrix]

col_sums = [sum(col) for col in zip(*matrix)]
```

```
diagonal_sum = sum(matrix[i][i] for i in range(len(matrix)))

for i, row_sum in enumerate(row_sums):

    print(f"Sum of {i + 1} row: {row_sum}")

for j, col_sum in enumerate(col_sums):

    print(f"Sum of {j + 1} column: {col_sum}")

print("Diagonal sum:", diagonal_sum)
```

29)Given three integers **M, N** and **K**. Consider a grid of **M * N**, where **mat[i][j] = i * j** (1 based index). The task is to return the **Kth** smallest element in the **M * N** multiplication table.

```
def kth_smallest_in_multiplication_table(m, n, k):
    low, high = 1, m * n
    while low < high:
        mid = low + (high - low) // 2
        count = sum(min(mid // i, n) for i in range(1, m + 1))
        if count < k:
            low = mid + 1
        else:
            high = mid
   return low
M, N, K = map(int, input("Enter M, N, and K: ").split())
result = kth_smallest_in_multiplication_table(M, N, K)
print("Kth smallest element:", result)
```

30)Print the sum of boundary elements of a matrix

```
def sum_of_boundary_elements(matrix):
    return sum(matrix[0]) + sum(matrix[-1]) + sum(row[0] + row[-1] for row in matrix[1:-1])
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
result = sum_of_boundary_elements(matrix)
print("Sum of boundary elements:", result)
```

31)Print the given matrix in spiral order

```
def spiral_order(matrix):
    return [elem for row in matrix for elem in row] if not matrix or not matrix[0] else matrix.pop(0) + spiral_order(list(zip(*matrix))[::-1])
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
result = spiral_order(matrix)
print("Matrix in spiral order:", result)
```

32)Write a python program to find the sum of 12+22+.......N2 numbers Sample input: N=6
Sample output: Sum=91

```
N = int(input("Enter N: "))
sum_series = N * (N + 1) * (2 * N + 1) // 6
print("Sum =", sum_series)
```

33)factorial

```
num = int(input("Enter a number: "))
result = 1 if num == 0 or num == 1 else num * factorial(num - 1)
print(f"The factorial of {num} is: {result}")
```

34)Write a python program to find the sum of 1!+2!+.......N! numbers Sample input: N=4Sample output: Sum=33

```
def factorial(n):
    return 1 if n == 0 or n == 1 else n * factorial(n - 1)
N = int(input("Enter N: "))
sum_factorials = sum(factorial(i) for i in range(1, N + 1))
print("Sum =", sum_factorials)
```

35)Write a python program to find the sum of 1!/1+2!/2+.......N!/N numbers Sample input: N=5
Sample output: Sum=34

```
def factorial(n):
    return 1 if n == 0 or n == 1 else n * factorial(n - 1)
N = int(input("Enter N: "))
sum_series = sum(factorial(i) / i for i in range(1, N + 1))
print("Sum =", sum_series)
```

36)Write a python program to find the difference between sum of square and square of sum N numbersSample input: N=5 Sample output: Diff=170

```
N = int(input("Enter N: "))
sum_of_squares = sum(i**2 for i in range(1, N + 1))
square_of_sum = sum(range(1, N + 1))**2
diff = square_of_sum - sum_of_squares
print("Diff =", diff)
```

36)Write a python program to find the sum of all digits in a triangle

```
triangle = [
    [1],
    [2, 3],
    [4, 5, 6],
    [7, 8, 9, 10]
]
sum_of_digits = sum(int(digit) for row in triangle for digit in row)
print("Sum of all digits in the triangle:", sum_of_digits)
```

37)fabnic series

```
def fibonacci(n):
    fib_series = [0, 1]
    while len(fib_series) < n: fib_series.append(fib_series[-1] + fib_series[-2])
    return fib_series

num_terms = int(input("Enter the number of Fibonacci terms: "))
print("Fibonacci Series:", fibonacci(num_terms))
```

38)You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

```
def climb_stairs(n):
    if n <= 2:
        return n
    prev1, prev2 = 1, 2
    for _ in range(3, n + 1):
        prev1, prev2 = prev2, prev1 + prev2
    return prev2
num_steps = int(input("Enter the number of steps: "))
print(f"Distinct ways to climb {num_steps} steps:", climb_stairs(num_steps))
```

39)vehical and children program

```
class Vehicle:
    def __init__(self, name):
        self.name = name

    def move(self):
        print(f"{self.name} is moving")

class Child:
    def __init__(self, name):
        self.name = name

    def play(self, vehicle):
        print(f"{self.name} is playing with {vehicle.name}")
        vehicle.move()

# Example usage
car = Vehicle("Car")
bike = Vehicle("Bike")

Child("Alice").play(car)
Child("Bob").play(bike)
```