Python

Program 1

list1=input("enter the list1")

list2=input("enter the list2")

merge\_list=list1+list2

sorted\_list=sorted(merge\_list)

print("the merged list after sorted is:",sorted\_list)

OUTPUT:

enter the list1[12,34,32]

enter the list2[33,23,45]

the merged list after sorted is: [',', ',', ',', ',', '1', '2', '2', '2', '3', '3', '3', '3', '3', '4', '4', '5', '[', '[',

']', ']']

Program 2

def are\_isomorphic(s, t):

if len(s) != len(t):

return False

mapping = {}

used\_characters = set()

for i in range(len(s)):

char\_s = s[i]

char\_t = t[i]

if char\_s in mapping

if mapping[char\_s] != char\_t:

return False

else:

if char\_t in used\_characters:

return False

mapping[char\_s] = char\_t

used\_characters.add(char\_t)

return True

s = input("Enter the first string: ")

t = input("Enter the second string: ")

if are\_isomorphic(s, t):

print("The strings are isomorphic.")

else:

print("The strings are not isomorphic.")

OUTPUT:

Enter the first string: add

Enter the second string: egg

The strings are isomorphic.

Program 3

def sumsquare(l):

odd\_sum = 0

even\_sum = 0

for num in l:

if num % 2 == 0:

even\_sum += num \*\* 2

else:

odd\_sum += num \*\* 2

return [odd\_sum, even\_sum]

numbers = [int(x) for x in input("Enter a list of integers separated by spaces: ").split()]

result = sumsquare(numbers)

print("Sum of squares of odd numbers:", result[0])

print("Sum of squares of even numbers:", result[1])

OUTPUT:

Enter a list of integers separated by spaces: 22 434 54 37 59

Sum of squares of odd numbers: 4850

Sum of squares of even numbers: 191756

Program 4

from itertools import permutations

def generate\_combinations(digits):

digit\_permutations = permutations(digits)

combinations = [''.join(p) for p in digit\_permutations]

return combinations

input\_digits = input("Enter 3 digits: ")

combinations = generate\_combinations(input\_digits)

for combo in combinations:

print(combo)

OUTPUT:

Enter 3 digits: 121

121

112

211

211

112

121

Program 5

def mirror\_image(input\_str):

mirrored\_str = input\_str + input\_str[::-1]

return mirrored\_str

input\_string = input("Enter a string: ")

mirror\_result = mirror\_image(input\_string)

print("Mirror image:", mirror\_result)

OUTPUT:

Enter a string: 123456

Mirror image: 123456654321

Program 6

def smaller\_numbers\_than\_current(nums):

sorted\_nums = sorted(nums)

count\_dict = {}

result = []

for i, num in enumerate(sorted\_nums):

if num not in count\_dict:

count\_dict[num] = i

for num in nums:

result.append(count\_dict[num])

return result

nums = [int(x) for x in input("Enter the numbers separated by spaces: ").split()]

output = smaller\_numbers\_than\_current(nums)

print("Count of numbers smaller than each element:", output)

OUTPUT

Enter the numbers separated by spaces: 8 6 5 4

Count of numbers smaller than each element: [3, 2, 1, 0]

Program 7

def delchar(s, c):

if len(c) != 1:

return s

new\_string = ""

for char in s:

if char != c:

new\_string += char

return new\_string

s = input("Enter a string: ")

c = input("Enter a character to delete: ")

result = delchar(s, c)

print("Result:", result)

OUTPUT:

Enter a string: hello world

Enter a character to delete: o

Result: hell wrld

Program 8

def reverse\_words(s):

words = s.split()

reversed\_words = words[::-1]

reversed\_string = ' '.join(reversed\_words)

return reversed\_string

input\_string = input("Enter a string: ")

reversed\_result = reverse\_words(input\_string)

print("Reversed words:", reversed\_result)

OUTPUT:

Enter a string: "sky is blue"

Reversed words: blue" is "sky

Program 9

def generate\_pascals\_triangle(num\_rows):

triangle = []

for n in range(num\_rows):

row = [1]

if n > 0:

prev\_row = triangle[n - 1]

for k in range(1, n):

row.append(prev\_row[k - 1] + prev\_row[k])

row.append(1)

triangle.append(row)

return triangle

def sum\_of\_nth\_row(triangle, n):

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

return 0

return sum(triangle[n])

num\_rows = int(input("Enter the number of rows: "))

pascals\_triangle = generate\_pascals\_triangle(num\_rows)

row\_number = int(input("Enter the row number: "))

row\_sum = sum\_of\_nth\_row(pascals\_triangle, row\_number)

print("Pascal's Triangle:")

for row in pascals\_triangle:

print(' '.join(map(str, row)))

print("Sum of elements in row", row\_number, "is:", row\_sum)

OUTOUT:

Enter the number of rows: 3

Enter the row number: 4

Pascal's Triangle:

1

1 1

1 2 1

Sum of elements in row 4 is: 0

Program 10

def get\_season(month, day):

if (month == "Mar" and day >= 20) or (month == "Apr" or month == "May") or (month == "Jun" and day < 21):

return "Spring"

elif (month == "Jun" and day >= 21) or (month == "Jul" or month == "Aug") or (month == "Sep" and day < 22):

return "Summer"

elif (month == "Sep" and day >= 22) or (month == "Oct" or month == "Nov") or (month == "Dec" and day < 21):

return "Fall"

else:

return "Winter"

month = input("Enter the month (First three letters, e.g., Jan, Feb, Mar): ")

day = int(input("Enter the day within the month: "))

season = get\_season(month, day)

print("The season associated with the date is:", season)

OUTPUT:

Enter the month (First three letters, e.g., Jan, Feb, Mar): mar

Enter the day within the month: 15

The season associated with the date is: Winter