**2. INPUT**

def sum\_of\_squares\_pattern(n):

for i in range (1, n + 1):

sum\_squares = sum (j \*\* 2 for j in range (1, i + 1))

print (f"Sum of squares of first {i} natural numbers: {sum\_squares}")

n = int (input ("Enter a natural number (n ≥ 1): "))

if n >= 1:

sum\_of\_squares\_pattern(n)

else:

print ("Please enter a natural number (n ≥ 1).")

**OUTPUT-**

Enter a natural number (n ≥ 1): 3

Sum of squares of first 1 natural numbers: 1

Sum of squares of first 2 natural numbers: 5

Sum of squares of first 3 natural numbers: 14

**3. # Input: number of values to sum**

N = int (input ("Enter how many numbers you want to sum: "))

# Initialize sum and counter

total = 0

count = 0

while count < N:

num = float (input (f"Enter number {count + 1}: "))

total += num

count += 1

print (f"The sum of the {N} numbers is: {total}")  
  
**OUTPUT-**Enter how many numbers you want to sum: 5

Enter number 1: 5

Enter number 2: 10

Enter number 3: 15

Enter number 4: 20

Enter number 5: 25

The sum of the 5 numbers is: 75.0

**4. # Sample dictionary with duplicate values across keys**

data = {

'a': [1, 2, 3],

'b': [3, 4, 5],

'c': [5, 6, 1]}

# Keep track of seen values

seen = []

for key in data:

unique\_values = []

# Inner loop: iterate through the values of the current key

for value in data[key]:

if value not in seen:

unique\_values.append(value)

seen.append(value)

data[key] = unique\_values

print ("Dictionary after removing duplicates across values:")

print(data)

**OUTPUT-**

Dictionary after removing duplicates across values:

{'a': [1, 2, 3], 'b': [4, 5], 'c': [6]}

**5. # Input from the user**

text = input ("Enter a string: ")

char\_freq = {}

for char in text:

if char in char\_freq:

char\_freq[char] += 1 # Increment count if already exists

else:

char\_freq[char] = 1 # Initialize count if new

print ("Character frequencies:")

for char, freq in char\_freq.items():

print(f"'{char}': {freq}")

**OUTPUT-**

Enter a string: great

Character frequencies:

'g': 1

'r': 1

'e': 1

'a': 1

't': 1

**6. # Sample list and dictionary**

my\_list = ['p', 'q', 'r', 's']

my\_dict = {'p': 12, 'r': 25, 't': 50}

K = input ("Enter the key to check: ")

if K in my\_list and K in my\_dict:

print (f"The value of '{K}' is: {my\_dict[K]}")

else:

print (f"Key '{K}' is not present in both the list and dictionary.")

**OUTPUT**

Enter the key to check: r

The value of 'r' is: 25

**7. INPUT**

def is\_power\_of\_2(n):

if n <= 0:

return False

return (n & (n - 1)) == 0

# Input from the user

num = int (input ("Enter a number: "))

if is\_power\_of\_2(num):

print(f"{num} is a power of 2.")

else:

print(f"{num} is not a power of 2.")

**OUTPUT**

Enter a number: 32

32 is a power of 2.

**8. # Sample list of coordinate tuples**

coordinates = [(2, 4), (1, 3), (6, 8), (7, 2), (10, 12), (1, 4)]

# Iterate and check for even x and y

print ("Coordinates where both x and y are even:")

for x, y in coordinates:

if x % 2 == 0 and y % 2 == 0:

print ((x, y))

**OUTPUT**

Coordinates where both x and y are even:

(2, 4)

(6, 8)

(10, 12)

**9.** def all\_elements\_unique(lst):

return len(lst) == len(set(lst))

my\_list = [1, 2, 3, 4, 5]

print ("All elements unique:", all\_elements\_unique(my\_list)) # True

my\_list2 = [1, 2, 2, 3]

print ("All elements unique:", all\_elements\_unique(my\_list2)) # False

**OUTPUT**

All elements unique: True

All elements unique: False

**10.** def find\_missing\_element (arr1, arr2):

set1 = set(arr1)

set2 = set(arr2)

diff = set1.symmetric\_difference(set2)

return diff.pop () if diff else None

array1 = [1, 2, 3, 4, 5]

array2 = [2, 3, 1, 5]

missing = find\_missing\_element (array1, array2)

print (f"The missing element is: {missing}")

**OUTPUT**

The missing element is: 4

**11. # Sample list of 2-element tuples**

tuple\_list = [(1, 2), (3, 4), (5, 6), (7, 8)]

swapped\_list = [(b, a) for (a, b) in tuple\_list]

print ("Original list:", tuple\_list)

print ("Swapped list: ", swapped\_list)

**OUTPUT**

Original list: [(1, 2), (3, 4), (5, 6), (7, 8)]

Swapped list: [(2, 1), (4, 3), (6, 5), (8, 7)]

**12. # Original tuples of tuples**

data = ((10, 10, 10, 12),

(30, 45, 56, 45),

(81, 80, 39, 32),

(1, 2, 3, 4))

transposed = zip(\*data)

averages = [sum(column) / len(column) for column in transposed]

print ("Average value of the numbers of the said tuple of tuples:")

print(averages)

**OUTPUT**

Average value of the numbers of the said tuple of tuples:

[30.5, 34.25, 27.0, 23.25]