**Write a Python program to accept n numbers in list and remove duplicates from a list**

# Accept numbers in a list and remove duplicates

def remove\_duplicates():

n = int(input("Enter the number of elements in the list: "))

numbers = []

for i in range(n):

num = int(input(f"Enter number {i + 1}: "))

numbers.append(num)

# Remove duplicates by converting the list to a set and back to a list

unique\_numbers = list(set(numbers))

print("List after removing duplicates:", unique\_numbers)

# Call the function

remove\_duplicates()

**Write a Python program to merge two lists and remove any duplicates.**

# Merge two lists and remove duplicates

def merge\_and\_remove\_duplicates():

# Accept the first list from the user

n1 = int(input("Enter the number of elements in the first list: "))

list1 = [int(input(f"Enter element {i + 1} for the first list: ")) for i in range(n1)]

# Accept the second list from the user

n2 = int(input("Enter the number of elements in the second list: "))

list2 = [int(input(f"Enter element {i + 1} for the second list: ")) for i in range(n2)]

# Merge both lists and remove duplicates

merged\_list = list(set(list1 + list2))

print("Merged list without duplicates:", merged\_list)

# Call the function

merge\_and\_remove\_duplicates()

**Write a Python program to reverse a list without using built-in functions like reverse() or without slicing. .**

# Reverse a list without using reverse() or slicing

def reverse\_list():

# Accept the list from the user

n = int(input("Enter the number of elements in the list: "))

numbers = [int(input(f"Enter element {i + 1}: ")) for i in range(n)]

# Reverse the list manually

reversed\_list = []

for i in range(len(numbers) - 1, -1, -1):

reversed\_list.append(numbers[i])

print("Reversed list:", reversed\_list)

# Call the function

reverse\_list()

**Write a Python program to rotate a list to the right by a given number of positions. For example, rotating [1, 2, 3, 4, 5] by 2 positions would result in [4, 5, 1, 2, 3].**

# Rotate a list to the right by a given number of positions

def rotate\_list():

# Accept the list from the user

n = int(input("Enter the number of elements in the list: "))

numbers = [int(input(f"Enter element {i + 1}: ")) for i in range(n)]

# Accept the number of positions to rotate

k = int(input("Enter the number of positions to rotate: "))

# Calculate the effective number of rotations (handles cases where k > n)

k = k % n

# Rotate the list to the right by k positions manually

rotated\_list = numbers[-k:] + numbers[:-k]

print("Rotated list:", rotated\_list)

# Call the function

rotate\_list()

**Write a Python program that combines two sorted lists into a single sorted list without using built-in sorting methods.**

# Combine two sorted lists into a single sorted list

def merge\_sorted\_lists():

# Accept the first sorted list from the user

n1 = int(input("Enter the number of elements in the first sorted list: "))

list1 = [int(input(f"Enter element {i + 1} for the first sorted list: ")) for i in range(n1)]

# Accept the second sorted list from the user

n2 = int(input("Enter the number of elements in the second sorted list: "))

list2 = [int(input(f"Enter element {i + 1} for the second sorted list: ")) for i in range(n2)]

# Merge the two sorted lists

merged\_list = []

i, j = 0, 0

while i < n1 and j < n2:

if list1[i] < list2[j]:

merged\_list.append(list1[i])

i += 1

else:

merged\_list.append(list2[j])

j += 1

# Add any remaining elements from both lists

while i < n1:

merged\_list.append(list1[i])

i += 1

while j < n2:

merged\_list.append(list2[j])

j += 1

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

# Call the function

merge\_sorted\_lists()

**Write a Python program to swap the values of two variables using a tuple.**

# Swap two variables using a tuple

def swap\_variables():

# Accept two variables from the user

a = input("Enter the value of a: ")

b = input("Enter the value of b: ")

# Swap values using a tuple

a, b = b, a

print(f"After swapping: a = {a}, b = {b}")

# Call the function

swap\_variables()

**Write a program to concatenate two tuples and find the length of the resulting tuple.**

# Concatenate two tuples and find the length

def concatenate\_tuples():

# Accept two tuples from the user

tuple1 = tuple(input("Enter the elements of the first tuple (separated by spaces): ").split())

tuple2 = tuple(input("Enter the elements of the second tuple (separated by spaces): ").split())

# Concatenate the tuples

concatenated\_tuple = tuple1 + tuple2

# Find the length of the resulting tuple

length = len(concatenated\_tuple)

print("Concatenated tuple:", concatenated\_tuple)

print("Length of the concatenated tuple:", length)

# Call the function

concatenate\_tuples()

**Write a Python program to find the intersection of two sets without using the built-in intersection() function.**

# Find the intersection of two sets

def find\_intersection():

# Accept two sets from the user

set1 = set(input("Enter the elements of the first set (separated by spaces): ").split())

set2 = set(input("Enter the elements of the second set (separated by spaces): ").split())

# Find the intersection manually

intersection = {element for element in set1 if element in set2}

print("Intersection of the two sets:", intersection)

# Call the function

find\_intersection()

**Using a set, write a Python program that removes duplicate elements from a list and prints the unique elements in the order they appeared in the list**

# Remove duplicates from a list while preserving order

def remove\_duplicates\_preserve\_order():

# Accept the list from the user

n = int(input("Enter the number of elements in the list: "))

elements = [input(f"Enter element {i + 1}: ") for i in range(n)]

# Use a set to track seen elements

seen = set()

unique\_elements = []

for element in elements:

if element not in seen:

unique\_elements.append(element)

seen.add(element)

print("List with unique elements in order:", unique\_elements)

# Call the function

remove\_duplicates\_preserve\_order()

**Create a Python program that takes a set of numbers and returns a new set containing only the even numbers.**

# Filter even numbers from a set

def filter\_even\_numbers():

# Accept a set of numbers from the user

numbers = set(map(int, input("Enter the numbers in the set (separated by spaces): ").split()))

# Use a set comprehension to filter even numbers

even\_numbers = {num for num in numbers if num % 2 == 0}

print("Set of even numbers:", even\_numbers)

# Call the function

filter\_even\_numbers()

**Write a Python program to merge two dictionaries and resolve any key conflicts by summing the values for common keys**

# Merge two dictionaries and sum values for common keys

def merge\_dictionaries():

# Accept the first dictionary from the user

dict1 = eval(input("Enter the first dictionary (e.g., {'a': 1, 'b': 2}): "))

# Accept the second dictionary from the user

dict2 = eval(input("Enter the second dictionary (e.g., {'b': 3, 'c': 4}): "))

# Merge dictionaries with conflict resolution

merged\_dict = dict1.copy() # Start with the first dictionary

for key, value in dict2.items():

if key in merged\_dict:

merged\_dict[key] += value # Sum values for common keys

else:

merged\_dict[key] = value # Add new key-value pair

print("Merged dictionary:", merged\_dict)

# Call the function

merge\_dictionaries()

o/p:

Enter the first dictionary (e.g., {'a': 1, 'b': 2}): {'a': 1, 'b': 2}

Enter the second dictionary (e.g., {'b': 3, 'c': 4}): {'b': 3, 'c': 4}

Merged dictionary: {'a': 1, 'b': 5, 'c': 4}

**Write a Python program that takes a dictionary of student names as keys and their scores as values. Return the name of the student with the highest score.**

# Find the student with the highest score

def find\_top\_student():

# Accept the dictionary of student scores from the user

scores = eval(input("Enter the dictionary of student scores (e.g., {'Alice': 85, 'Bob': 90}): "))

# Find the student with the highest score

top\_student = max(scores, key=scores.get)

print("Student with the highest score:", top\_student)

# Call the function

find\_top\_student()

o/p:-

Enter the dictionary of student scores: {'Alice': 85, 'Bob': 90}

Student with the highest score: Bob

**Using a dictionary, write a Python program that groups the list of tuples by the first element of the tuple. For example, given [('a', 1), ('b', 2), ('a', 3)], group it as {'a': [1, 3], 'b': [2]}**

# Group list of tuples by the first element

def group\_tuples\_by\_first\_element():

# Accept the list of tuples from the user

tuples\_list = eval(input("Enter the list of tuples (e.g., [('a', 1), ('b', 2), ('a', 3)]): "))

# Create an empty dictionary to group the tuples

grouped\_dict = {}

for key, value in tuples\_list:

if key not in grouped\_dict:

grouped\_dict[key] = [] # Initialize an empty list for new keys

grouped\_dict[key].append(value) # Append the value to the appropriate key's list

print("Grouped dictionary:", grouped\_dict)

# Call the function

group\_tuples\_by\_first\_element()

**Write a Python program to check if a given key already exists in a dictionary. If key exists replace with another key/value pair**

# Check if a key exists and replace it with another key/value pair

def replace\_key\_in\_dictionary():

# Accept the dictionary from the user

dictionary = eval(input("Enter the dictionary (e.g., {'a': 1, 'b': 2}): "))

# Accept the key to check

key\_to\_check = input("Enter the key to check: ")

# Accept the new key and value

new\_key = input("Enter the new key: ")

new\_value = input("Enter the new value: ")

# Check if the key exists and replace it

if key\_to\_check in dictionary:

dictionary.pop(key\_to\_check) # Remove the old key

dictionary[new\_key] = new\_value # Add the new key-value pair

print(f"Key '{key\_to\_check}' found and replaced with ({new\_key}: {new\_value})")

else:

print(f"Key '{key\_to\_check}' not found in the dictionary.")

# Print the updated dictionary

print("Updated dictionary:", dictionary)

# Call the function

replace\_key\_in\_dictionary()

**Write a Python script to generate and print a dictionary which contains a number (between 1 and n) in the form(x, x\*x). Sample Dictionary (n=5) Expected Output: {1:1, 2:4, 3:9, 4:16, 5:25}**

# Generate a dictionary with numbers and their squares

def generate\_square\_dictionary():

# Accept the value of n from the user

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

# Create the dictionary using dictionary comprehension

square\_dict = {x: x \* x for x in range(1, n + 1)}

print("Generated dictionary:", square\_dict)

# Call the function

generate\_square\_dictionary()

**Write a Python program to convert a tuple of string values to a tuple of integer values. Original tuple values: (('333', '33'), ('1416', '55')) New tuple values: ((333, 33), (1416, 55))**

# Convert a tuple of string values to a tuple of integer values

def convert\_tuple():

# Accept the tuple of string values from the user

str\_tuple = eval(input("Enter a tuple of string values (e.g., (('333', '33'), ('1416', '55'))): "))

# Convert to a tuple of integer values

int\_tuple = tuple(tuple(int(num) for num in inner\_tuple) for inner\_tuple in str\_tuple)

print("Converted tuple:", int\_tuple)

# Call the function

convert\_tuple()

**Write a Python program to compute element-wise sum of given tuples. Original lists: (1, 2, 3, 4) (3, 5, 2, 1) (2, 2, 3, 1) Element-wise sum of the said tuples: (6, 9, 8, 6)**

# Input tuples

tuple1 = (1, 2, 3, 4)

tuple2 = (3, 5, 2, 1)

tuple3 = (2, 2, 3, 1)

# Element-wise sum

result = tuple(a + b + c for a, b, c in zip(tuple1, tuple2, tuple3))

print("Element-wise sum of the tuples:", result)

**. Write a python program to count repeated characters in a string. Sample string: 'the quick brown fox jumps over the lazy dog' Expected output: o-4, e-3, u-2, h-2, r-2, t-2, ‘ ’-7.**

from collections import Counter

# Sample string

sample\_string = 'the quick brown fox jumps over the lazy dog'

# Count characters

char\_count = Counter(sample\_string)

# Filter characters with more than 1 occurrence

repeated\_chars = {char: count for char, count in char\_count.items() if count > 1}

# Display output

for char, count in repeated\_chars.items():

print(f"{repr(char)}-{count}", end=", ")

**. Write a python script to find the repeated items of a tuple**

# Input tuple

input\_tuple = (1, 2, 3, 4, 5, 1, 2, 6, 3, 7, 4)

# Find repeated items

repeated\_items = {item for item in input\_tuple if input\_tuple.count(item) > 1}

print("Repeated items in the tuple:", repeated\_items)

**Regular Expression\*\*\*🡺**

**Write a program to check if the input year is leap year or not. Validate the input.**

import re

def is\_leap\_year(year):

if re.match(r'^\d{4}$', year):

year = int(year)

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

return True

else:

return False

else:

return False

year = input("Enter a year: ")

if is\_leap\_year(year):

print("Leap Year")

else:

print("Not a Leap Year")

**. Write a Python code to validate email address using regular expression**

import re

def validate\_email(email):

return bool(re.match(r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email))

email = input("Enter email: ")

if validate\_email(email):

print("Valid Email")

else:

print("Invalid Email")

. .

def validate\_ip(ip):

return bool(re.match(r'^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$', ip))

def validate\_phone\_number(phone):

return bool(re.match(r'^\+?[1-9][0-9]{9,14}$', phone))

ip = input("Enter IP Address: ")

if validate\_ip(ip):

print("Valid IP Address")

else:

print("Invalid IP Address")

phone = input("Enter Phone Number: ")

if validate\_phone\_number(phone):

print("Valid Phone Number")

else:

print("Invalid Phone Number")

**. Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' ']'. These brackets must be close in the correct order. for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid**

import re

def validate\_parentheses(string):

return bool(re.match(r'^[\(\)\{\}\[\]]\*$', string))

parentheses = input("Enter string of parentheses: ")

if validate\_parentheses(parentheses):

print("Valid Parentheses")

else:

print("Invalid Parentheses")

**. Write python code to validate URL (e.g. https://www.google.com or http://xyz.com or** [**ftp://localhost:2000**](ftp://localhost:2000)

import re

def validate\_url(url):

return bool(re.match(r'^(https?|ftp)://[^\s/$.?#].[^\s]\*$', url))

url = input("Enter URL: ")

if validate\_url(url):

print("Valid URL")

else:

print("Invalid URL")

**. Write python code to validate password. Password should contain at least 1 Capital Letter, 1 digit, 1 special character and the length of password should be min 8 characters and max length should be 15.**

import re

def validate\_password(password):

return bool(re.match(r'^(?=.\*[A-Z])(?=.\*\d)(?=.\*[\W\_]).{8,15}$', password))

password = input("Enter Password: ")

if validate\_password(password):

print("Valid Password")

else:

print("Invalid Password")