```
# -*- coding: utf-8 -*-
"""assignment 2.pynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1fkEFPB9-Y0b9jAcMuh4kcaQYrOgg85G2

Q1 Write code to reverse string

Ans Here is a simple Python function to reverse a string:
"""

def reverse_string(s):
    return s[::-1]

# Example usage:
string = "Hello, World!"
reversed_string = reverse_string(string)
print(reversed_string)

"""In this code, s[::-1] is a Python slicing technique that returns the string in reverse order.

Q2 Write code to count the number of vowels in string.

Ans Here is a Python code to count the number of vowels in a string:
"""

def count_vowels(string):
    vowels = "aeiouAEIOU"  # Defining both lowercase and uppercase vowels
    count = 0

    for char in string:
        if char in vowels:
            count += 1

    return count

# Example usage:
input_string = "Hello, how many vowels are in this string?"
vowel_count = count_vowels(input_string)
print(f"Number of vowels: {vowel_count}")

"""This code checks each character in the string and increases the count if the character is a vowel. It handles both uppercase and lowercase vowels.

Q3  Write a code to check if a given string is palindrome or not.

Ans Here's a Python code to check if a given string is a palindrome:
"""

def is_palindrome(s):
    # Convert the string to lowercase and remove any spaces or non-alphanumeric characters
    cleaned_string = ''.join(char.lower() for char in s if char.isalnum())

    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Example usage
input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print(f'"{input_string}" is a palindrome.')
else:
    print(f'"{input_string}" is not a palindrome.')

"""The input string is first cleaned by removing non-alphanumeric characters and converting everything to lowercase.
Then, it checks if the cleaned string is the same forwards and backwards by comparing it with its reverse.

Q4  Write a code to check if two given strings are anagrams of each other.

Ans Here's a Python function to check if two given strings are anagrams of each other:
"""

def are_anagrams(str1, str2):
    # Remove any whitespace and convert both strings to lowercase
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Check if both strings have the same sorted characters
    return sorted(str1) == sorted(str2)

# Example usage
string1 = "listen"
string2 = "silent"

if are_anagrams(string1, string2):
    print(f"'{string1}' and '{string2}' are anagrams.")
else:
    print(f"'{string1}' and '{string2}' are not anagrams.")

"""The replace() function removes any spaces in the strings (optional depending on your requirement).
The lower() function converts both strings to lowercase to make the comparison case-insensitive.
The sorted() function sorts the characters in each string and compares the sorted versions.
If they match, the function returns True, meaning they are anagrams. Otherwise, it returns False.

Q5 Write code to find all occurrences of given substring within another string.

Ans Here is a Python code snippet that finds all occurrences of a given substring within another string:
"""

def find_all_occurrences(text, substring):
    occurrences = []
    start = 0
    while True:
        # Find the next occurrence of the substring
        start = text.find(substring, start)
```

```
        # If no more occurrences, break out of the loop
        if start == -1:
            break

        # Add the index to the occurrences list
        occurrences.append(start)

        # Move start index forward for the next search
        start += 1  # Move past this occurrence

    return occurrences

# Example usage:
text = "The quick brown fox jumps over the lazy dog. The dog barks."
substring = "dog"
occurrences = find_all_occurrences(text, substring)
print(occurrences)

"""Explanation:
text.find(substring, start) returns the index of the first occurrence of substring in text starting from start. It returns -1 if no occurrence is found.
Each found index is added to the occurrences list.
The start index is incremented to continue searching beyond the current found occurrence to avoid infinite loops.

Output:
For the example given:
"""

[40, 49]

"""Q6 Write code to perform basic string compression using the counts of repeated characters.

Ans Here's a Python code to perform basic string compression using counts of repeated characters. The compressed version of the string will display the charact
"""

def compress_string(s):
    if not s:
        return ""

    compressed = []
    count = 1

    # Loop through the string and count occurrences of each character
    for i in range(1, len(s)):
        if s[i] == s[i - 1]:
            count += 1
        else:
            compressed.append(s[i - 1] + str(count))
            count = 1

    # Add the last character and its count
    compressed.append(s[-1] + str(count))

    # Convert the list to a string
    compressed_string = ''.join(compressed)

    # Return the original string if compression does not reduce the size
    return compressed_string if len(compressed_string) < len(s) else s


# Example usage
input_str = "aabcccccaaa"
result = compress_string(input_str)
print(f"Original: {input_str}")
print(f"Compressed: {result}")

"""Explanation:
The function compress_string iterates through the input string, counting consecutive occurrences of characters.
It appends each character followed by its count to the compressed list.
Finally, it compares the length of the compressed string with the original and returns the shorter one.
For the input "aabcccccaaa", the compressed string would be "a2b1c5a3".

Q7 Write a code to detertmine if string has all unique characters.

Ans.Here is a Python function to determine if a string has all unique characters:
"""

def has_unique_chars(string):
    # Create a set to store unique characters
    char_set = set()

    # Iterate through each character in the string
    for char in string:
        # If character is already in the set, it's a duplicate
        if char in char_set:
            return False
        # Add character to the set
        char_set.add(char)

    # If no duplicates found, return True
    return True

# Example usage:
input_string = "abcdef"
print(has_unique_chars(input_string))  # Output: True

input_string2 = "abcdeff"
print(has_unique_chars(input_string2))  # Output: False

"""This function checks each character in the string and uses a set to track already seen characters. If it finds a repeated character, it returns False. Other

Q8 Write a code to convert a given string to uppercase or lowercase.

Ans.Here is a Python code that converts a given string to either uppercase or lowercase based on the user's choice:
"""

def convert_string(s, to_upper=True):
```

```
    if to_upper:
        return s.upper()
    else:
        return s.lower()

# Example usage
string = input("Enter a string: ")
choice = input("Convert to uppercase or lowercase? (u/l): ").lower()

if choice == 'u':
    print("Uppercase:", convert_string(string, to_upper=True))
elif choice == 'l':
    print("Lowercase:", convert_string(string, to_upper=False))
else:
    print("Invalid choice! Please enter 'u' for uppercase or 'l' for lowercase.")

"""How it works:
The function convert_string takes two parameters: the string s and a boolean flag to_upper. If to_upper is True, it converts the string to uppercase; otherwise
The user is asked to input a string and specify whether they want it in uppercase or lowercase.

Q9 Write a code to count the number of words in a string.

Ans
Here's a simple Python function to count the number of words in a string:
"""

def count_words(string):
    # Split the string into words based on whitespace
    words = string.split()
    # Return the length of the list of words
    return len(words)

# Example usage
text = "This is an example string with seven words."
word_count = count_words(text)
print(f"Number of words: {word_count}")

"""This code splits the input string into words using spaces as the delimiter and then counts the number of words. You can modify the input text to check other

Q10 Write a code to concatenate two string without using the + operators.

Ans You can concatenate two strings without using the + operator by using various methods in Python. Here are a few approaches:

1. Using join()
"""

str1 = "Hello"
str2 = "World"
result = "".join([str1, str2])
print(result)  # Output: HelloWorld

"""2. Using format()"""

str1 = "Hello"
str2 = "World"
result = "{}{}".format(str1, str2)
print(result)  # Output: HelloWorld

"""3. Using f-strings (Python 3.6+)"""

str1 = "Hello"
str2 = "World"
result = f"{str1}{str2}"
print(result)  # Output: HelloWorld

"""4. Using str() with a list"""

str1 = "Hello"
str2 = "World"
result = str().join([str1, str2])
print(result)  # Output: HelloWorld

"""5. Using reduce()"""

from functools import reduce

str1 = "Hello"
str2 = "World"
result = reduce(lambda x, y: x + y, [str1, str2])
print(result)  # Output: HelloWorld

"""Choose any of these methods based on your preferences or the specific requirements of your task!

Q11 Write a code to removeo all occurrenaces of a specific element from a list.

Ans You can remove all occurrences of a specific element from a list in Python using a list comprehension. Here's a simple example:
"""

def remove_occurrences(lst, element):
    return [x for x in lst if x != element]

# Example usage:
my_list = [1, 2, 3, 4, 2, 5, 2]
element_to_remove = 2
updated_list = remove_occurrences(my_list, element_to_remove)

print(updated_list)  # Output: [1, 3, 4, 5]

"""Explanation:
The function remove_occurrences takes two arguments: lst, the original list, and element, the element you want to remove.
It returns a new list containing all elements of lst that are not equal to element.
The for loop in the list comprehension iterates through each item in lst, and the condition if x != element filters out the specified element.

Q12 Implement a code to find the second largest number in a given list of integers.

Ans Here's a simple Python function to find the second largest number in a given list of integers:
```

```
"""

def second_largest(numbers):
    # Remove duplicates by converting the list to a set
    unique_numbers = set(numbers)

    # Check if there are at least two unique numbers
    if len(unique_numbers) < 2:
        return None  # or raise an exception

    # Sort the unique numbers and get the second largest
    sorted_numbers = sorted(unique_numbers, reverse=True)
    return sorted_numbers[1]

# Example usage
numbers = [3, 1, 4, 4, 5, 5, 2]
result = second_largest(numbers)
print(f"The second largest number is: {result}")

"""Explanation:
Remove Duplicates: Convert the list to a set to eliminate duplicate values.
Check Length: Ensure there are at least two unique numbers.
Sort and Access: Sort the unique numbers in descending order and access the second element.
You can replace the numbers list in the example usage with any list of integers you'd like to test.

Q13 Create a code to count the occurrenes of each element in a list and return dictionry with elements keys and their as count values.

Ans You can use Python to create a function that counts the occurrences of each element in a list and returns a dictionary with the elements as keys and their
"""

def count_occurrences(input_list):
    occurrence_dict = {}
    for element in input_list:
        if element in occurrence_dict:
            occurrence_dict[element] += 1
        else:
            occurrence_dict[element] = 1
    return occurrence_dict

# Example usage:
my_list = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']
result = count_occurrences(my_list)
print(result)

"""Explanation:
Function Definition: The function count_occurrences takes a list as an argument.
Dictionary Initialization: An empty dictionary, occurrence_dict, is initialized.
Loop Through List: The function iterates through each element in the input list:
If the element is already a key in the dictionary, it increments the count by 1.
If it's not, it adds the element as a new key with a count of 1.
Return the Dictionary: Finally, the function returns the dictionary containing the counts.
Example Output:
For the input ['apple', 'banana', 'apple', 'orange', 'banana', 'apple'], the output will be:
"""

{'apple': 3, 'banana': 2, 'orange': 1}

"""Q14 Write a code to reverse a list in-placee without using any
 built-in reverse functions.

Ans You can reverse a list in-place in Python by using a simple loop to swap elements. Here's a code snippet that demonstrates how to do this:
"""

def reverse_list_in_place(lst):
    left = 0
    right = len(lst) - 1

    while left < right:
        # Swap the elements
        lst[left], lst[right] = lst[right], lst[left]
        left += 1
        right -= 1

# Example usage
my_list = [1, 2, 3, 4, 5]
reverse_list_in_place(my_list)
print(my_list)  # Output: [5, 4, 3, 2, 1]

"""Explanation:
The reverse_list_in_place function takes a list as an argument.
It initializes two pointers: left at the beginning (0) and right at the end of the list (length - 1).
While the left pointer is less than the right pointer, it swaps the elements at these indices and then moves the pointers closer to the center.
This process continues until the entire list is reversed.

Q15 Implement a code to find and remove duplicates from' list while preserving the original order of elements.

Ans You can use a simple Python function to remove duplicates from a list while preserving the original order of the elements. Here's a sample implementation:
"""

def remove_duplicates(input_list):
    seen = set()  # Create a set to store seen elements
    output_list = []  # Create a list for the output

    for item in input_list:
        if item not in seen:  # Check if the item has already been seen
            seen.add(item)  # Add it to the seen set
            output_list.append(item)  # Append it to the output list

    return output_list

# Example usage
input_list = [1, 2, 2, 3, 4, 4, 5, 1]
result = remove_duplicates(input_list)
print(result)  # Output: [1, 2, 3, 4, 5]

"""Explanation:
```

```
seen: A set that keeps track of elements that have already been encountered.
output_list: A list that will hold the unique elements in their original order.
The function iterates through each item in the input_list. If the item is not in the seen set, it adds it to both seen and output_list.
Finally, it returns the output_list, which contains the unique elements in the order they first appeared.

Q16 Create a code to check if a given list is sorted (either in 'asending or descending order) or not.

Ans Here's a simple Python function that checks if a given list is sorted in either ascending or descending order:
"""

def is_sorted(lst):
    if len(lst) < 2:
        return True  # A list with 0 or 1 element is sorted

    ascending = all(lst[i] <= lst[i + 1] for i in range(len(lst) - 1))
    descending = all(lst[i] >= lst[i + 1] for i in range(len(lst) - 1))

    return ascending or descending

# Example usage:
print(is_sorted([1, 2, 3, 4, 5]))  # True (ascending)
print(is_sorted([5, 4, 3, 2, 1]))  # True (descending)
print(is_sorted([1, 3, 2, 4, 5]))  # False (not sorted)
print(is_sorted([5, 5, 5]))         # True (same elements)

"""Explanation:
The function is_sorted takes a list lst as an argument.
It first checks if the list has fewer than two elements; if so, it returns True, as such lists are considered sorted.
Then, it uses the all() function combined with a generator expression to check if the list is sorted in ascending order (where each element is less than or equ
Finally, it returns True if the list is sorted in either order; otherwise, it returns False.

Q17 Write a code to merge two sorted lists into a single sorted list.

Ans You can merge two sorted lists into a single sorted list using a simple algorithm that compares the elements of both lists and builds the merged list accor
"""

def merge_sorted_lists(list1, list2):
    merged_list = []
    i, j = 0, 0

    # Compare elements from both lists and add the smaller one to merged_list
    while i < len(list1) and j < len(list2):
        if list1[i] < list2[j]:
            merged_list.append(list1[i])
            i += 1
        else:
            merged_list.append(list2[j])
            j += 1

    # If there are remaining elements in list1, add them
    while i < len(list1):
        merged_list.append(list1[i])
        i += 1

    # If there are remaining elements in list2, add them
    while j < len(list2):
        merged_list.append(list2[j])
        j += 1

    return merged_list

# Example usage
list1 = [1, 3, 5, 7]
list2 = [2, 4, 6, 8]
result = merge_sorted_lists(list1, list2)
print(result)  # Output: [1, 2, 3, 4, 5, 6, 7, 8]

"""Explanation:
The function initializes two pointers, i and j, for the two lists.
It compares the current elements of both lists and appends the smaller one to merged_list.
If one list is exhausted, it appends the remaining elements from the other list.
Finally, it returns the merged sorted list.
Feel free to modify the example lists to test the function with different inputs!

Q18  Implement a code to find the intersection of two given lists.

Ans Here's a Python code to find the intersection of two given lists:
"""

def find_intersection(list1, list2):
    # Convert both lists to sets and find their intersection
    intersection = list(set(list1) & set(list2))
    return intersection

# Example usage:
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

result = find_intersection(list1, list2)
print("Intersection:", result)

"""Explanation:
We convert both lists to sets using set().
The & operator is used to find the intersection of the two sets.
Finally, we convert the result back to a list.
In the example above, the output will be:
"""

Intersection: [4, 5]

"""Q19 Create a code to find the union of two lists without duplicates.

Ans You can find the union of two lists without duplicates in Python by using a set, which automatically handles duplicate elements. Here's the code:
"""

def union_lists(list1, list2):
```

```
    # Use set to remove duplicates and find the union
    union_set = set(list1).union(set(list2))
    # Convert back to list if you want the result as a list
    return list(union_set)

# Example usage
list1 = [1, 2, 3, 4]
list2 = [3, 4, 5, 6]
result = union_lists(list1, list2)
print(result)

"""This code converts both lists to sets, performs the union operation, and then converts the result back to a list. The result will contain the union of the t

Q20 Write a code to shuffle a given list randomly without using any built-in shuffle function.

Ans You can shuffle a list randomly by implementing the Fisher-Yates (also known as Knuth) shuffle algorithm. Here's a Python code that shuffles a given list w
"""

import random

def custom_shuffle(lst):
    # Traverse the list from the last element to the second element
    for i in range(len(lst)-1, 0, -1):
        # Generate a random index from 0 to i
        j = random.randint(0, i)
        # Swap the current element with the randomly chosen element
        lst[i], lst[j] = lst[j], lst[i]
    return lst

# Example usage
my_list = [1, 2, 3, 4, 5]
shuffled_list = custom_shuffle(my_list)
print(shuffled_list)

"""Explanation:
We iterate backward through the list.
For each position i, a random index j is generated between 0 and i.
We swap the elements at positions i and j. This approach ensures each element has an equal probability of appearing in any position, achieving a proper random

Q21 Write a code that takes two tuples as input and returns a new tuple containing elements that are common to both input tuples.

Ans Here's a Python function that takes two tuples as input and returns a new tuple containing the elements that are common to both:
"""

def common_elements(tuple1, tuple2):
    # Convert tuples to sets to find common elements
    common_set = set(tuple1).intersection(set(tuple2))
    # Convert the result back to a tuple
    return tuple(common_set)

# Example usage
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (4, 5, 6, 7, 8)

result = common_elements(tuple1, tuple2)
print(result)  # Output will be (4, 5) or (5, 4), since sets are unordered

"""This code works by converting the tuples to sets, finding their intersection, and then converting the result back into a tuple.

Q22 Create a code that prompts the user to enter two sets of integers   seperated by commas. Then, print the intespection of these two sets.

Ans Here's a Python code that prompts the user to enter two sets of integers separated by commas, and then prints the intersection of these two sets:
"""

# Function to get the intersection of two sets
def get_intersection(set1, set2):
    return set1.intersection(set2)

# Prompt the user to enter two sets of integers separated by commas
input_set1 = input("Enter the first set of integers (separated by commas): ")
input_set2 = input("Enter the second set of integers (separated by commas): ")

# Convert the input strings to sets of integers
set1 = set(map(int, input_set1.split(',')))
set2 = set(map(int, input_set2.split(',')))

# Find and print the intersection of the two sets
intersection = get_intersection(set1, set2)
print("The intersection of the two sets is:", intersection)

"""How it works:
The user is prompted to enter two sets of integers as comma-separated values.
The input strings are split by commas, and then converted to sets of integers.
The intersection method is used to find common elements in both sets.
The result is printed.
Example:
"""

Enter the first set of integers (separated by commas): 1,2,3,4
Enter the second set of integers (separated by commas): 3,4,5,6
The intersection of the two sets is: {3, 4}

"""Q23 Write a code to concatenate two tuples. The function should take two tuples input and return a new tuple containing elements from both input tuples.

Ans Here's a Python function to concatenate two tuples:
"""

def concatenate_tuples(tuple1, tuple2):
    # Concatenating the two tuples
    return tuple1 + tuple2

# Example usage:
tuple_a = (1, 2, 3)
tuple_b = (4, 5, 6)
result = concatenate_tuples(tuple_a, tuple_b)
print(result)
```

```
"""In this code, the function concatenate_tuples takes two tuples as input and returns a new tuple containing elements from both. The + operator is used to con

For the example provided, the output will be:
"""

(1, 2, 3, 4, 5, 6)

"""Q24 Develop a code that prompts the user to input two sets of strings. Then print the elements that are present in the first set but not in the second set.

Ans Here is a Python code that prompts the user to input two sets of strings, then prints the elements that are present in the first set but not in the second:
"""

# Prompt the user to input two sets of strings
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 difference between the two sets
difference = set1 - set2

# Print the elements that are in set1 but not in set2
print("Elements in the first set but not in the second set:")
for element in difference:
    print(element)

"""How it works:
The user inputs two sets of strings (each string separated by a space).
The strings are split into individual elements and stored as sets.
The program calculates the difference between the two sets (set1 - set2).
It then prints the elements that are present in the first set but not in the second.
You can run this code in any Python environment

Q25 Create a code that takes a tuples and two integers as input. The function should return a new tuple containing elements from the original tuple within the

Ans Here's a Python function that takes a tuple and two integers as input and returns a new tuple containing elements within the specified range of indices:
"""

def slice_tuple(input_tuple, start_index, end_index):
    # Return a sliced portion of the tuple based on start and end index
    return input_tuple[start_index:end_index]

# Example usage
original_tuple = (1, 2, 3, 4, 5, 6, 7)
start = 2
end = 5

new_tuple = slice_tuple(original_tuple, start, end)
print(new_tuple)  # Output: (3, 4, 5)

"""Explanation:
input_tuple[start_index:end_index] returns a slice from start_index to one before end_index.
In the example above, it extracts elements from index 2 to index 4.

Q26 Write a code that prompts the user to input two sets of characters. Then print the union of this two sets.

Ans Here is a Python code that prompts the user to input two sets of characters and prints their union:
"""

# Prompt user to input two sets of characters
set1 = set(input("Enter the first set of characters (without spaces): "))
set2 = set(input("Enter the second set of characters (without spaces): "))

# Calculate the union of the two sets
union_set = set1.union(set2)

# Print the union of the two sets
print("The union of the two sets is:", union_set)

"""How it works:
The input() function takes user input as strings, and set() is used to convert them into sets of characters.
The union() method computes the union of the two sets.
Finally, it prints the union.

Q27  Develop a code that takes a tuple of integers as input. The function should return the maximum and  minimum values from the tuple using tuple unpacking.

Ans Here is a Python function that takes a tuple of integers as input and returns the maximum and minimum values using tuple unpacking:
"""

def find_max_min(input_tuple):
    # Unpack the tuple into a list
    *rest, = input_tuple

    # Use built-in functions to find the max and min
    max_value = max(rest)
    min_value = min(rest)

    return max_value, min_value

# Example usage:
numbers = (5, 12, 7, 9, 3, 15, 1)
max_value, min_value = find_max_min(numbers)
print(f"Maximum: {max_value}, Minimum: {min_value}")

"""Explanation:
The *rest, = input_tuple unpacks the tuple into a list called rest, allowing us to manipulate the elements.
max(rest) and min(rest) are used to find the maximum and minimum values from the unpacked list.
The function returns both the maximum and minimum values.
Let me know if you'd like further details!

Q28 Create a code that defines two sets of integers. Then print the union intersection and difference of these two sets.

Ans Here's a Python code snippet that defines two sets of integers and then prints their union, intersection, and difference:
"""

# Define two sets of integers
set1 = {1, 2, 3, 4, 5}
```

```
set2 = {4, 5, 6, 7, 8}

# Calculate union, intersection, and difference
union = set1.union(set2)
intersection = set1.intersection(set2)
difference_set1_set2 = set1.difference(set2)
difference_set2_set1 = set2.difference(set1)

# Print the results
print("Set 1:", set1)
print("Set 2:", set2)
print("Union:", union)
print("Intersection:", intersection)
print("Difference (Set 1 - Set 2):", difference_set1_set2)
print("Difference (Set 2 - Set 1):", difference_set2_set1)

"""Explanation:
Union combines all unique elements from both sets.
Intersection returns elements that are common to both sets.
Difference returns elements that are in the first set but not in the second (and vice versa for the second difference).
You can run this code in any Python environment to see the results.

Q29 write a code that takes a tuple and an element as input. The function should return the count of occurrences of the given element in the tuple.

Ans You can achieve this by defining a function in Python that takes a tuple and an element as input, then counts the occurrences of that element in the tuple.
"""

def count_occurrences(input_tuple, element):
    return input_tuple.count(element)

# Example usage:
my_tuple = (1, 2, 3, 2, 4, 2)
element_to_count = 2
count = count_occurrences(my_tuple, element_to_count)
print(f"The element {element_to_count} occurs {count} times in the tuple.")

"""In this code:

The count_occurrences function uses the count() method of tuples to find the number of times the specified element appears in the tuple.
You can replace my_tuple and element_to_count with any tuple and element you want to test.

Q30 develop a code that prompts the user to input two sets of strings. Then print the symmetric difference of these two sets.

Ans Here's a simple Python code snippet that prompts the user to input two sets of strings and then prints the symmetric difference between them:
"""

# Function to get user input and create sets
def input_set(prompt):
    user_input = input(prompt)
    return set(user_input.split())

# Get two sets of strings from the user
set1 = input_set("Enter the first set of strings (separated by spaces): ")
set2 = input_set("Enter the second set of strings (separated by spaces): ")

# Calculate the symmetric difference
symmetric_difference = set1.symmetric_difference(set2)

# Print the result
print("The symmetric difference is:", symmetric_difference)

"""How it Works:
The input_set function prompts the user for input and splits the input string into individual strings, creating a set.
The program then calculates the symmetric difference between the two sets using the symmetric_difference method.
Finally, it prints the symmetric difference.
Example Usage:
If the user inputs apple banana cherry for the first set and banana cherry date for the second set, the output will be:
"""

The symmetric difference is: {'apple', 'date'}

"""Q31 Write a code that takes a list of words as input and returns a dictionary where the keys are unique word and the values are the frequencies of those wor

Ans Here's a Python function that takes a list of words as input and returns a dictionary with the unique words as keys and their frequencies as values:
"""

def word_frequencies(word_list):
    frequency_dict = {}

    for word in word_list:
        if word in frequency_dict:
            frequency_dict[word] += 1
        else:
            frequency_dict[word] = 1

    return frequency_dict

# Example usage
input_words = ["apple", "banana", "apple", "orange", "banana", "apple"]
result = word_frequencies(input_words)
print(result)  # Output: {'apple': 3, 'banana': 2, 'orange': 1}

"""Explanation:
The function word_frequencies initializes an empty dictionary frequency_dict.
It iterates over each word in the input list.
If the word is already a key in the dictionary, it increments its count. If not, it adds the word to the dictionary with a count of 1.
Finally, it returns the frequency dictionary.
Feel free to modify the input list to test with different words!

Q32 Write a code that takes two dictionaries as input and merges them into a single dictionary. If there are common keys the values should be added together.

Ans Here's a Python function that merges two dictionaries, summing the values of any common keys:
"""

def merge_dictionaries(dict1, dict2):
    merged_dict = dict1.copy()  # Start with a copy of the first dictionary
```

```
    for key, value in dict2.items():
        if key in merged_dict:
            merged_dict[key] += value  # Add values if key exists
        else:
            merged_dict[key] = value  # Add new key-value pair

    return merged_dict

# Example usage:
dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 3, 'c': 1, 'd': 4}

result = merge_dictionaries(dict1, dict2)
print(result)  # Output: {'a': 1, 'b': 5, 'c': 4, 'd': 4}

"""Explanation:
The function merge_dictionaries takes two dictionaries as input.
It creates a copy of the first dictionary (dict1) to avoid modifying the original.
It then iterates over the second dictionary (dict2).
If a key from dict2 already exists in merged_dict, it adds the value to the existing value.
If the key does not exist, it adds the new key-value pair to merged_dict.
Finally, it returns the merged dictionary.

Q33 Write a code to access a value in a nested dictionary. The function should take the dictionary and a list of keys as input and return the corresponding val

Ans Here's a Python function that accesses a value in a nested dictionary using a list of keys. If any of the keys do not exist in the dictionary, it will retu
"""

def get_nested_value(nested_dict, keys):
    """Access a value in a nested dictionary using a list of keys.

    Args:
        nested_dict (dict): The nested dictionary to search.
        keys (list): A list of keys to traverse the dictionary.

    Returns:
        The value corresponding to the keys, or None if any key is not found.
    """
    current_value = nested_dict
    for key in keys:
        if isinstance(current_value, dict) and key in current_value:
            current_value = current_value[key]
        else:
            return None
    return current_value

# Example usage:
nested_dictionary = {
    'a': {
        'b': {
            'c': 42
        }
    }
}

keys_list = ['a', 'b', 'c']
print(get_nested_value(nested_dictionary, keys_list))  # Output: 42

keys_list_not_found = ['a', 'b', 'd']
print(get_nested_value(nested_dictionary, keys_list_not_found))  # Output: None

"""Explanation:
The function get_nested_value takes a nested dictionary and a list of keys.
It initializes current_value with the input dictionary and iterates over the list of keys.
If the current value is still a dictionary and the key exists, it updates current_value.
If a key does not exist, it returns None.
Finally, if all keys are found, it returns the corresponding value.

Q34 Write a code that takes a dictionary as input and returns a sorted version of it based on the values. You can choose whether to sort in ascending or descen

Ans Here's a Python function that takes a dictionary as input and returns a sorted version of it based on the values. You can specify whether to sort in ascend
"""

def sort_dict_by_value(input_dict, ascending=True):
    """
    Sorts a dictionary by its values.

    Parameters:
        input_dict (dict): The dictionary to be sorted.
        ascending (bool): If True, sorts in ascending order; if False, sorts in descending order.

    Returns:
        dict: A new dictionary sorted by its values.
    """
    # Sort the dictionary by its values
    sorted_dict = dict(sorted(input_dict.items(), key=lambda item: item[1], reverse=not ascending))
    return sorted_dict

# Example usage
example_dict = {'a': 3, 'b': 1, 'c': 2}
sorted_dict_ascending = sort_dict_by_value(example_dict, ascending=True)
sorted_dict_descending = sort_dict_by_value(example_dict, ascending=False)

print("Sorted in ascending order:", sorted_dict_ascending)
print("Sorted in descending order:", sorted_dict_descending)

"""Explanation:
The function sort_dict_by_value takes a dictionary (input_dict) and a boolean parameter (ascending).
It sorts the dictionary using sorted(), which sorts based on the second item of the tuples (the values).
The reverse parameter is set to not ascending, so if ascending is True, it sorts in ascending order; if False, it sorts in descending order.
Finally, it converts the sorted items back into a dictionary and returns it.

Q35 Write a code that inverts a dictionary swapping keys and values. Ensure that the inverted dictionary correctly handles cases where multiple keys have the s

Ans
```

```
You can invert a dictionary in Python by swapping keys and values, and handling cases where multiple keys have the same value by storing the keys in a list. He
"""

def invert_dictionary(original_dict):
    inverted_dict = {}
    for key, value in original_dict.items():
        # Check if the value is already a key in the inverted dictionary
        if value in inverted_dict:
            # If it is, append the current key to the list of keys
            inverted_dict[value].append(key)
        else:
            # If it isn't, create a new list with the current key
            inverted_dict[value] = [key]
    return inverted_dict

# Example usage
original_dict = {
    'a': 1,
    'b': 2,
    'c': 1,
    'd': 3
}

inverted = invert_dictionary(original_dict)
print(inverted)

"""Explanation:
The function invert_dictionary takes an input dictionary original_dict.
It iterates through each key-value pair in the original dictionary.
For each value, it checks if that value already exists as a key in inverted_dict.
If it exists, it appends the current key to the list associated with that value.
If it doesn't exist, it creates a new entry with the value as the key and initializes a list containing the current key.
Finally, it returns the inverted dictionary.
Output:
For the provided original_dict, the output will be:
"""

{1: ['a', 'c'], 2: ['b'], 3: ['d']}

"""This output shows that both keys 'a' and 'c' share the same value 1, so they are stored in a list under the key 1."""
```