Q.1.Write code to reverse string.

```
course = "Data Science"

course[::-1]

'ecneicS ataD'
```

Q.2.Write code to count the number of vowels in string.

```
text = "I want to become a data_scientist"
vowels = "aeiouAEIOU"
vowel_count = sum(1 for char in text if char in vowels)
print(f"The number of vowels in the text is: {vowel_count}")

The number of vowels in the text is: 12

text = "I want to become a data_scientist"
vowels = "aeiouAEIOU"
vowel_count = sum(char in vowels for char in text)
print(f"The number of vowels in the text is: {vowel_count}")

The number of vowels in the text is: 12
```

Q.3.Write code to check if the given string is palindrome or not.

```
word = input("enter a word: ")

palindrome = word[::-1]

if word == palindrome:
    print(f"the given word {word} is a palindrome")
else:
    print(f"the given word {word} is not a palindrome")

enter a word:  civic

the given word civic is a palindrome
```

Q.4.Write code to check if two given string are anagrams of each other.

```
str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")

def anagrams(str1,str2):
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    return sorted(str1) == sorted(str2)
if anagrams(str1,str2):
    print(f"{str1} and {str2} are anagrams")
else:
    print(f"{str1} and {str2} are not anagrams")
```

```
Enter the first string:  listen
Enter the second string:  silent

listen and silent are anagrams
```

Q.5. Write code to find all occurrences of a given substring within another string.

```
import re

def find_substring_occurrences(text, substring):
    # Use re.finditer to find all matches of the substring in the text
    matches = re.finditer(re.escape(substring), text)
    # Get the starting indices of each match
    positions = [match.start() for match in matches]
    return positions

# Example usage
text ='''Jack quickly realized that the big lazy dog was blocking his
path  The sun was setting casting a golden hue over the quiet village
Suddenly a quick brown fox darted across the road making Jack jump
back in surprise.'''
substring = "Jack"
positions = find_substring_occurrences(text, substring)
print(f"The substring '{substring}' occurs at positions: {positions}")
```

```
The substring 'Jack' occurs at positions: [0, 188]
```

Q.6. Write code to perform besic string compression using the counts of repeated characters.

```python
def compress_string(s):
    compressed = []
    count = 1

    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

    compressed.append(s[-1] + str(count))

    compressed_string = ''.join(compressed)

    return compressed_string if len(compressed_string) < len(s) else s

# Example usage
input_string = "aabcccccaaa"
compressed_string = compress_string(input_string)
print(f"Original string: {input_string}")
print(f"Compressed string: {compressed_string}")
```

```
Original string: aabcccccaaa
Compressed string: a2b1c5a3
```

Q.7. Write coode to determine if string has all unique characters.

```python
def has_unique_characters(s):
    char_set = set()
    for char in s:
        if char in char_set:
            return False
        char_set.add(char)
    return True

# Example usage
input_string = "abcdefg"
result = has_unique_characters(input_string)
print(f"The string '{input_string}' has all unique characters:
{result}")

input_string = "hello"
```

```
result = has_unique_characters(input_string)
print(f"The string '{input_string}' has all unique characters: 
{result}")

The string 'abcdefg' has all unique characters: True
The string 'hello' has all unique characters: False
```

Q.8.Write coode to convert a given string to uppercase or lowercase.

```
def convert_to_uppercase(s):
    return s.upper()

def convert_to_lowercase(s):
    return s.lower()

input_string = "Manu Bhakar wins broze, becomes first Indian women to 
win shooting medal at Olympics "

uppercase_string = convert_to_uppercase(input_string)
print(f"Uppercase: {uppercase_string}")

lowercase_string = convert_to_lowercase(input_string)
print(f"Lowercase: {lowercase_string}")

Uppercase: MANU BHAKAR WINS BROZE, BECOMES FIRST INDIAN WOMEN TO WIN 
SHOOTING MEDAL AT OLYMPICS
Lowercase: manu bhakar wins broze, becomes first indian women to win 
shooting medal at olympics
```

Q.9.Write coode to count the number of words in a string.

```
a = "I want to participate in the Olympic Games."
len(a.split())

8
```

Q.10.Write a code to concatenate two strings without using the + operator.

```
def concatenate_strings(str1, str2):
    return f"{str1}{str2}"

string1 = "Play "
string2 = "Store"
```

```
result = concatenate_strings(string1, string2)
print(result)
```

```
Play Store
```

Q.11.Write a code to remove all occurrences of a specific element from a list.

```
l = [1,2,3,4,5,3,5,6,7,5,3]
```

```
set(l)
```

```
{1, 2, 3, 4, 5, 6, 7}
```

```
def occurrences_element(lst, element):
    return list(filter(lambda x: x != element, lst))

# Example usage
my_list = [1, 7, 3, 4, 2, 5, 7,3,3]
element_to_remove = 3
new_list = occurrences_element(my_list, element_to_remove)
print(new_list)
```

```
[1, 7, 4, 2, 5, 7]
```

Q.12.Implement a code to find the second largest number in a given list of integers.

```
list = [2,3,4,5,6,78,56,4,5,67]
```

```
sorted(list)[-2]
```

```
67
```

```
def find_second_largest(numbers):
    return sorted(set(numbers))[-2] if len(set(numbers)) > 1 else None

# Example usage
numbers_list = [111,1232,32,122,564,2231,42334,5555,3333,78867,1991]
second_largest = find_second_largest(numbers_list)
print("The second largest number is:", second_largest)
```

```
The second largest number is: 42334
```

Q.13.Create a code to count the occurrence of each element in a list and return a dictionary with elements as keys and their counts as vlues.

```
from collections import Counter

def count_occurrences(lst):
    return dict(Counter(lst))

# Example usage
elements = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']
occurrences = count_occurrences(elements)
print(occurrences)
```

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

Q.14.Write a code to reverce a list in-place without using any built-in reverce functions.

```
numbers = [1, 2, 3, 4, 5]
numbers[::-1]
```

```
[5, 4, 3, 2, 1]
```

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

    while left < right:
        lst[left], lst[right] = lst[right], lst[left]
        left += 1
        right -= 1
numbers = [10, 20, 30, 45, 50]
reverse_list(numbers)
print(numbers)
```

```
[50, 45, 30, 20, 10]
```

Q.15. Implement a code to find and remove duplicates from a list while preserving the original order of elements.

```
def remove_duplicates(lst):
    numbers = set()
    return [x for x in lst if not (x in numbers or numbers.add(x))]
```

```python
numbers = [1, 2, 2, 3, 4, 3, 5, 6, 6, 7]
unique_numbers = remove_duplicates(numbers)
print(unique_numbers)
```

```
[1, 2, 3, 4, 5, 6, 7]
```

Q.16. Create a code to check if a given list is sorted (either in asending or descending order) or not.

```python
def is_sorted(lst):
    return lst == sorted(lst) or lst == sorted(lst, reverse=True)

list1 = [1, 2, 3, 4, 5]
list2 = [5, 4, 3, 2, 1]
list3 = [1, 3, 2, 4, 5]

print(is_sorted(list1))  # Output: True (ascending)
print(is_sorted(list2))  # Output: True (descending)
print(is_sorted(list3))
```

```
True
True
False
```

Q.17.Write a code to merge two sorted lists into a single sorted list.

```python
def merge_sorted_lists(list1, list2):
    return sorted(list1 + list2)

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

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

Q.18. Implement a code to find the intersection of two given lists.

```python
def find_intersection(list1, list2):
    return [item for item in list1 if item in list2]
```

```python
list1 = [1, 2, 2, 3, 4]
list2 = [2, 3, 5]
result = find_intersection(list1, list2)
print(result)
```

```
[2, 2, 3]
```

Q.19.Create a code to find the union of two lists without duplicates.

```python
def find_union(lst1, lst2):
    seen = set()
    return [item for item in lst1 + lst2 if not (item in seen or
seen.add(item))]

my_list1 = [3, 5, 5, 8, 4]
my_list2 = [3, 5, 9, 6]
result = find_union(my_list1, my_list2)
print(result)
```

```
[3, 5, 8, 4, 9, 6]
```

Q.20. Write a code to shuffle a given list randomaly without using any built-in shuffle functions.

```python
import random
from itertools import permutations

def random_permutation(lst):
    all_permutations = list(permutations(lst))
    return list(random.choice(all_permutations))

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

```
[5, 2, 3, 4, 1]
```

Q.21.Write a code that takes two tuples as input and returns a new tuple containing elements that are common to both input tuples.

```
def common_elements_tuple_ordered(tuple1, tuple2):
    set2 = set(tuple2)

    common_elements = tuple(el for el in tuple1 if el in set2)

    return common_elements
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (4, 5, 6, 7, 8)
result = common_elements_tuple_ordered(tuple1, tuple2)
print(result)
```

```
(4, 5)
```

Q.22.Create a code that prompts the user to enter two sets of integers separated by commas. Then, print the intersection of these two sets.

```
def intersection_of_sets():
    # Prompt the user to enter the first set
    set1_input = input("Enter the first set of integers separated by
commas: ")

    # Prompt the user to enter the second set
    set2_input = input("Enter the second set of integers separated by
commas: ")

    # Convert the input strings to sets of integers
    set1 = set(map(int, set1_input.split(",")))
    set2 = set(map(int, set2_input.split(",")))

    # Find the intersection of the two sets
    intersection = set1.intersection(set2)

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

# Call the function to start the program
intersection_of_sets()
```

```
Enter the first set of integers separated by commas:  1,2,3,4,5
Enter the second set of integers separated by commas:  4,5,6,7,8
```

```
The intersection of the two sets is: {4, 5}
```

Q.23. Write a code to concatenate two tuples. The function should take two tuples as input and return a new tuple containing elements from both input tuples.

```python
def concatenate_tuples(tuple1, tuple2):
    result = tuple1 + tuple2
    return result

tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
result = concatenate_tuples(tuple1, tuple2)
print(result)
```

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

Q.24 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.

```python
set1 = set(input("Enter elements of the first set separated by spaces: 
").split())
set2 = set(input("Enter elements of the second set separated by 
spaces: ").split())

difference = set1 - set2

print("Elements present in the first set but not in the second set:", 
difference)
```

```
Enter elements of the first set separated by spaces:  12,3,3,4,56
Enter elements of the second set separated by spaces:  12,3,4,566,4

Elements present in the first set but not in the second set: 
{'12,3,3,4,56'}
```

Q.25.Create a code that takes a tuple and two integers as input. The function should return a new tuple containing elements from the original tuple within the specified range of indices

```
def slice_tuple(original_tuple, start, end):
    return original_tuple[start:end]

input_tuple = tuple(input("Enter the elements of the tuple separated
by spaces: ").split())
start_index = int(input("Enter the starting index: "))
end_index = int(input("Enter the ending index: "))

result = slice_tuple(input_tuple, start_index, end_index)

print("New tuple containing elements within the specified range:",
result)
```

```
Enter the elements of the tuple separated by spaces:  12 13 51 22 66
17 88
Enter the starting index:  1
Enter the ending index:  4

New tuple containing elements within the specified range: ('13', '51',
'22')
```

Q.26.Write a code that prompts the user to input two sets of characters. Then print the union of these two sets.

```
def get_char_set(prompt):
    user_input = input(prompt)
    return set(user_input)

set1 = get_char_set("Enter characters for the first set (no spaces):
")
set2 = get_char_set("Enter characters for the second set (no spaces):
")

union_set = set1 | set2

print("Union of the two sets:", union_set)
```

```
Enter characters for the first set (no spaces):  abcdef
Enter characters for the second set (no spaces):  efghij
```

```
Union of the two sets: {'f', 'h', 'g', 'j', 'a', 'd', 'c', 'e', 'i',
'b'}
```

Q.27.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.

```
def find_max_min(numbers):

    max_value = max(numbers)
    min_value = min(numbers)
    return max_value, min_value

input_tuple = tuple(map(int, input("Enter integers separated by
spaces: ").split()))

max_val, min_val = find_max_min(input_tuple)
print(f"Maximum value: {max_val}, Minimum value: {min_val}")

Enter integers separated by spaces:  2 45 67 88 9 10 1 44 78 88 65

Maximum value: 88, Minimum value: 1
```

Q.28.Create a code that defines two sets of integers. Then, print the union, intersection , and difference of these two sets.

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

union_set = set1 | set2
intersection_set = set1 & set2
difference_set = set1 - set2

print("Set 1:", set1)
print("Set 2:", set2)
print("Union of the sets:", union_set)
print("Intersection of the sets:", intersection_set)
print("Difference of the sets (Set 1 - Set 2):", difference_set)

Set 1: {1, 2, 3, 4, 5}
Set 2: {4, 5, 6, 7, 8}
Union of the sets: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
Intersection of the sets: {4, 5}
Difference of the sets (Set 1 - Set 2): {1, 2, 3}
```

Q.29.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.

```python
def count_occurrences(tup, element):

    return tup.count(element)

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

element = input("Enter the element to count its occurrences: ")

occurrence_count = count_occurrences(input_tuple, element)

print(f"The element '{element}' occurs {occurrence_count} times in the tuple.")
```

```
Enter the elements of the tuple separated by spaces:  apple banana melon guava grapes kiwi apple banana apple
Enter the element to count its occurrences:  banana

The element 'banana' occurs 2 times in the tuple.
```

Q.30.Develop a code that prompts the user to input two sets of strings. Then print the symmetric difference of these two sets.

```python
def get_input_set(prompt):

    user_input = input(prompt)
    return set(user_input.split())

set1 = get_input_set("Enter elements of the first set separated by spaces: ")
set2 = get_input_set("Enter elements of the second set separated by spaces: ")

symmetric_difference_set = set1 ^ set2
```

```
print("Symmetric difference of the two sets:", 
symmetric_difference_set)

Enter elements of the first set separated by spaces:  apple banana 
guava kiwi cherry
Enter elements of the second set separated by spaces:  orange melon 
apple strawberry papaya

Symmetric difference of the two sets: {'orange', 'papaya', 'kiwi', 
'banana', 'cherry', 'guava', 'strawberry', 'melon'}
```

Q.31. Write a code that takes a list of words as input and returns a dictionary where the keys are unique words and the values are the frequencies of those words in the input list.

```
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

input_words = input("Enter a list of words separated by spaces: 
").split()

frequencies = word_frequencies(input_words)

print("Word frequencies:", frequencies)

Enter a list of words separated by spaces:  apple banana guava kiwi 
orange melon apple strawberry papaya

Word frequencies: {'apple': 2, 'banana': 1, 'guava': 1, 'kiwi': 1, 
'orange': 1, 'melon': 1, 'strawberry': 1, 'papaya': 1}
```

Q.32.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.

```
def merge_dictionaries(dict1, dict2):

    merged_dict = dict1.copy()
    for key, value in dict2.items():
        if key in merged_dict:
            merged_dict[key] += value
        else:
            merged_dict[key] = value
    return merged_dict

dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 3, 'c': 4, 'd': 5}

merged_dict = merge_dictionaries(dict1, dict2)

print("Merged dictionary:", merged_dict)
```

```
Merged dictionary: {'a': 1, 'b': 5, 'c': 7, 'd': 5}
```

Q.33.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 value. If any of the keys do not exist in the dictionary the function should return None.

34.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 descending order.

```
def sort_dict_by_values(input_dict, ascending=True):

    sorted_items = sorted(input_dict.items(), key=lambda item:
item[1], reverse=not ascending)
    return dict(sorted_items)

example_dict = {'apple': 5, 'banana': 2, 'cherry': 7, 'date': 1}
```

```
sorted_dict_ascending = sort_dict_by_values(example_dict,
ascending=True)
print("Sorted dictionary (ascending):", sorted_dict_ascending)

sorted_dict_descending = sort_dict_by_values(example_dict,
ascending=False)
print("Sorted dictionary (descending):", sorted_dict_descending)
```

```
Sorted dictionary (ascending): {'date': 1, 'banana': 2, 'apple': 5,
'cherry': 7}
Sorted dictionary (descending): {'cherry': 7, 'apple': 5, 'banana': 2,
'date': 1}
```

Q.35.Write a code that inverts a dictionary swapping keys and values.Ensure that the inverted dictionary correctly handles cases where multiple keys have the same value by storing the keys as a list in the inverted dictionary.

```
def invert_dictionary(input_dict):

    inverted_dict = {}
    for key, value in input_dict.items():
        if value not in inverted_dict:
            inverted_dict[value] = [key]
        else:
            inverted_dict[value].append(key)
    return inverted_dict

example_dict = {'apple': 1, 'banana': 2, 'cherry': 1, 'date': 2,
'elderberry': 3}

inverted_dict = invert_dictionary(example_dict)

print("Inverted dictionary:", inverted_dict)
```

```
Inverted dictionary: {1: ['apple', 'cherry'], 2: ['banana', 'date'],
3: ['elderberry']}
```