Q.1 Write a code to reverce a string.

```
# Ans.1
S="PWSKILLS"
print(S[::-1])
```

SLLIKSWP

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

```
# Ans.2
S=input('Enter any string')
v='aeiouAEIOU'
count=0
for i in S:
    if i in v:
        count+=1
    else:
        continue
print(count)
```

Enter any string Krishna

2

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

```
# Ans.3
S=input('Enter any string')
if S==S[::-1]:
```

```
    print(S,'is a palindrome')
else:
    print(S,'is not a palindrome')
```

```
Enter any string pkp
pkp is a palindrome
```

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

```
# Ans.4
# Step 3: Compare the sorted strings
if sorted_string1 == sorted_string2:
    print("The strings are anagrams of each other.")
else:
    print("The strings are not anagrams of each other.")
```

```
Enter First string Riya
Enter Second string Iyar
The strings are anagrams of each other.
```

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

```
# Ans.5
# Given strings
main_string = "abracadabra"
substring = "abra"

# Lengths of the main string and the substring
```

```
main_length = len(main_string)
sub_length = len(substring)

# List to store the starting indices of all occurrences
occurrences = []

# Loop through the main string
for i in range(main_length - sub_length + 1):
    # Check if the substring is found at the current position
    if main_string[i:i + sub_length] == substring:
        occurrences.append(i)

# Print the list of starting indices
print("Occurrences of the substring:", occurrences)
```

Occurrences of the substring: [0, 7]

Q.6 Write a code to basic string compression using the counts repeated characters.

```
# Ans.6
# Given string
input_string = "aabccccccaaa"

# Initialize variables
compressed_string = ""
count = 1
```

```
# Iterate through the string
for i in range(1, len(input_string)):
    # If the current character is the same as the previous one, increment the count
    if input_string[i] == input_string[i - 1]:
        count += 1
    else:
        # Otherwise, append the character and its count to the compressed string
        compressed_string += input_string[i - 1] + str(count)
        count = 1

# Append the last character and its count
compressed_string += input_string[-1] + str(count)

# Print the compressed string
print("Compressed string:", compressed_string)
```

Compressed string: a2b1c5a3

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

```
# Ans.7
# Given string
input_string =input("Enter your string")
# Initialize a set to keep track of seen characters
seen_characters = set()

# Iterate through the string
```

```
for char in input_string:
    # If the character is already in the set, it means it's a duplicate
    if char in seen_characters:
        print("The string does not have all unique characters.")
        break
    # Add the character to the set
    seen_characters.add(char)
else:
    # If the loop completes without finding duplicates, all characters are unique
    print("The string has all unique characters.")
```

```
Enter your string Krishna
The string has all unique characters.
```

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

```
# Ans.8
# Given string
input_string = "Hello, World!"

# Convert the string to uppercase
uppercase_string = input_string.upper()

# Convert the string to lowercase
lowercase_string = input_string.lower()
```

```
# Print the results
print("Original string:", input_string)
print("Uppercase string:", uppercase_string)
print("Lowercase string:", lowercase_string)
```

```
Original string: Hello, World!
Uppercase string: HELLO, WORLD!
Lowercase string: hello, world!
```

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

```
S=input('Enter any string')
count=0
for i in S:
    count+=1
print(count)
```

```
Enter any string Krishna
7
```

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

```
# Given strings
string1 =input('Enter First string')
string2 = input('Enter Second string')

# Concatenate strings using join()
concatenated_string = "".join([string1, string2])
```

```
# Print the result
print("Concatenated string using join():", concatenated_string)
```

```
Enter First string krishna
Enter Second string dubey
Concatenated string using join(): krishnadubey
```

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

```
# Ans.11
# List and element to be removed
my_list = [1, 2, 3, 4, 3, 5, 3, 6]
element_to_remove = 3

# Create a new list to store the result
new_list = []

# Iterate through the original list
for item in my_list:
    # If the current item is not the element to remove, add it to the new list
    if item != element_to_remove:
        new_list.append(item)

# Update the original list to the new list without the element
my_list = new_list
```

```
# Print the updated list
print("Updated list:", my_list)
```

Updated list: [1, 2, 4, 5, 6]

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

```
# Ans.12
# List of integers
my_list = [12, 35, 1, 10, 34, 1]

# Check if the list has less than 2 elements
if len(my_list) < 2:
    print("List should have at least two distinct elements")
else:
    # Initialize the largest and second largest elements
    largest = second_largest = float('-inf')

    # Iterate through the list to find the largest and second largest elements
    for num in my_list:
        if num > largest:
            second_largest = largest
            largest = num
        elif num > second_largest and num != largest:
            second_largest = num
```

```
    # Check if we found a valid second largest element
    if second_largest == float('-inf'):
        print("There is no second largest element")
    else:
        print("The second largest element is:", second_largest)
```

The second largest element is: 34

Q.13 Create a code to count the occurrences of each element in a list and return a dictionary with elements as key and their counts as values.

```
# Ans.13
def count_elements(lst):
    element_count = {}
    for element in lst:
        if element in element_count:
            element_count[element] += 1
        else:
            element_count[element] = 1
    return element_count

# Example usage
example_list = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
print(count_elements(example_list))
```

{1: 1, 2: 2, 3: 3, 4: 4}

Q.14 write a code to reverse a list in place without using any built-in reverse function.

```
# Ans.14
def reverse_list(lst):
    left = 0
    right = len(lst) - 1

    while left < right:
        # Swap the elements at the left and right indices
        lst[left], lst[right] = lst[right], lst[left]
        # Move the pointers towards the center
        left += 1
        right -= 1

# Example usage
example_list = [1, 2, 3, 4, 5]
reverse_list(example_list)
print(example_list)
```

[5, 4, 3, 2, 1]

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

```
# Ans.15
def remove_duplicates(lst):
    seen = set()
    result = []
```

```
    for element in lst:
        if element not in seen:
            result.append(element)
            seen.add(element)

    return result

# Example usage
example_list = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
unique_list = remove_duplicates(example_list)
print(unique_list)
```

[1, 2, 3, 4]

Q.16 Create a code to check if the given list is soted(either in ascending or decending order) or not.

```
# Ans.16
L = [1,2,3,4]
a = all(L[i] <= L[i+1] for i in range(len(L)-1))
d = all(L[i] >= L[i+1] for i in range(len(L)-1))
if (a or d) == True:
    print("The list is sorted")
else :
    print("this is not a sorted list")
```

The list is sorted

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

```
# Ans.17


#Let take two sorted list

list_1 =[1,3,5,7,9]

list_2 =[0,2,4,6,8]


i=0

j=0


# Create an another list to merge two list

merge_list =[]


while i<len(list_1) and j<len(list_2):
    if list_1[i] < list_2[j]:
        merge_list.append(list_1[i])
        i+=1
    else:
        merge_list.append(list_2[j])
        j+=1


while i<len(list_1):
    merge_list.append(list_1[i])
    i+=1
while i<len(list_2):
```

```
    merge_list.append(list_2[j])
    j+=1

print("Merged list :", merge_list)
```

Merged list : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

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

```
#Ans.18
list_1 =[1,2,3,4,5,6]
list_2 =[3,4,5,6,7,8]
intersect_list = [item for item in list_1 if item in list_2]
print(intersect_list)
```

[3, 4, 5, 6]

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

```
#Ans.19
list_1 =[1,2,3,4,5,6]
list_2 =[3,4,5,6,7,8]
union = []
for i in list_1:
    if i not in union:
        union.append(i)
for j in list_2:
    if j not in union:
        union.append(j)
```

```
print(union)
```

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

Q.20 Write a code to shuffle a given list randomly without using any build-in shuffle function.

```
#Ans.20

import random

random.seed()

list_1 =[1,2,3,4,5,6]

n=len(list_1)

Shuffle = []

for i in range(n):

    j = random.randint(0,n-1)

    list_1[i],list_1[j] = list_1[j] , list_1[i]

print(list_1)
```

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

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

```
# Define two tuples

tuple1 = tuple(input("Enter the value of first tuple"))

tuple2 = tuple(input("Enter the value of second tuple"))


# Find common elements using set intersection

common_elements = tuple(set(tuple1) & set(tuple2))


# Print the result
```

print(common_elements)

Enter the value of first tuple 1,2,3,4,5

Enter the value of second tuple 4,5,6,7,8

('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.

#Ans.22

```
# Prompt the user to enter the first set of integers

input1 = input("Enter the first set of integers separated by commas: ")


# Prompt the user to enter the second set of integers

input2 = input("Enter the second set of integers separated by commas: ")


# Convert the input strings to sets of integers

set1 = set(map(int, input1.split(',')))

set2 = set(map(int, input2.split(',')))


# Find the intersection of the two sets

intersection_set = set1 & set2


# Print the result

print("The intersection of the two sets is:", intersection_set)
```

Enter the first set of integers separated by commas: 12345

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.

```
tuple1 = tuple(input("Enter the value of first tuple"))

tuple2 = tuple(input("Enter the value of second tuple"))

concate_tuple = tuple1 + tuple2

print(concate_tuple)
```

Enter the value of first tuple 123

Enter the value of second tuple 456

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

Q.24 Develop a code that prompt 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.24

S1=set(input("Enter the elements of first String"))

S2=set(input("Enter the lements of Second String"))

for i in S1:

   if i in S2:

      continue

   else:

      print(i)
```

Enter the elements of first String 1234

Enter the lements of Second String 4567

3

2

1

Q.25 Create a code that take 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 New_tuple(T1,N1,N2):
    return T1[N1:N2]
New_tuple((1,2,3,4,5,6),2,4)
```

(3, 4)

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

```
S1=set(input("Enter the elements of first set"))
S2=set(input("Enter the lements of Second set"))
result = S1 | S2
print(result)
```

Enter the elements of first set asdf

Enter the lements of Second set sdfg

{'s', 'f', 'd', 'g', 'a'}

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 unpacking.

```
# Input: Tuple of integers
input_tuple = tuple(map(int, input("Enter a tuple of integers separated by commas: ").split(',')))

# Unpacking the tuple to find the maximum and minimum values
max_value = max(input_tuple)
min_value = min(input_tuple)
```

```
# Output the results

print("The maximum value in the tuple is:", max_value)

print("The minimum value in the tuple is:", min_value)
```

```
Enter a tuple of integers separated by commas:  1,2,3,4

The maximum value in the tuple is: 4

The minimum value in the tuple is: 1
```

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

```
# Define two sets of integers

set1 = {1, 2, 3, 4, 5}

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


# Calculate the union of the two sets

union_set = set1 | set2


# Calculate the intersection of the two sets

intersection_set = set1 & set2


# Calculate the difference of the two sets

difference_set1 = set1 - set2  # Elements in set1 but not in set2

difference_set2 = set2 - set1  # Elements in set2 but not in set1


# Print the results

print("Union of set1 and set2:", union_set)
```

```
print("Intersection of set1 and set2:", intersection_set)

print("Difference of set1 and set2 (set1 - set2):", difference_set1)

print("Difference of set2 and set1 (set2 - set1):", difference_set2)
```

Union of set1 and set2: {1, 2, 3, 4, 5, 6, 7, 8}

Intersection of set1 and set2: {4, 5}

Difference of set1 and set2 (set1 - set2): {1, 2, 3}

Difference of set2 and set1 (set2 - set1): {8, 6, 7}

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.

```
# Input: Tuple of elements

input_tuple = tuple(input("Enter a tuple of elements separated by commas: ").split(','))


# Input: Element to count

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


# Count occurrences of the element in the tuple

count = 0

for item in input_tuple:

    if item == element:

        count += 1


# Output the result

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

Enter a tuple of elements separated by commas: a,s,f,h,j,k,l,e,r,t,u,i,g,d,a,a,s,d,f,s

Enter the element to count: a

The element 'a' occurs 3 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.

```
# Prompt the user to enter the first set of strings

input1 = input("Enter the first set of strings separated by commas: ")


# Prompt the user to enter the second set of strings

input2 = input("Enter the second set of strings separated by commas: ")


# Convert the input strings to sets of strings

set1 = set(input1.split(','))

set2 = set(input2.split(','))


# Calculate the symmetric difference of the two sets

symmetric_difference_set = set1 ^ set2


# Print the result

print("The symmetric difference of the two sets is:", symmetric_difference_set)
```

Enter the first set of strings separated by commas: 1,2,3

Enter the second set of strings separated by commas: 4,5,6

The symmetric difference of the two sets is: {'1', '6', '3', '4', '5', '2'}

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.

```
# Input: List of words

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


# Initialize an empty dictionary to store word frequencies

word_frequencies = {}


# Count the frequency of each word in the list

for word in words:

   if word in word_frequencies:

      word_frequencies[word] += 1

   else:

      word_frequencies[word] = 1


# Output the result

print("Word frequencies:", word_frequencies)
```

```
Enter a list of words separated by spaces:  k r i s h n a d u b e y s i r

Word frequencies: {'k': 1, 'r': 2, 'i': 2, 's': 2, 'h': 1, 'n': 1, 'a': 1, 'd': 1, 'u': 1, 'b': 1, 'e': 1, 'y': 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.

```
# Input: Two dictionaries

dict1 = eval(input("Enter the first dictionary: "))
```

```
dict2 = eval(input("Enter the second dictionary: "))


# Initialize the result dictionary
merged_dict = dict1.copy()


# Merge the dictionaries
for key, value in dict2.items():
    if key in merged_dict:
        merged_dict[key] += value
    else:
        merged_dict[key] = value


# Output the result
print("Merged dictionary:", merged_dict)
```

```
Enter the first dictionary:  {'a': 2, 'b': 3, 'c': 5}
Enter the second dictionary:  {'b': 4, 'c': 2, 'd': 7}
Merged dictionary: {'a': 2, 'b': 7, 'c': 7, 'd': 7}
```

Q.33 Write a code to access a value in a nested dictionary. The function should take the dictionary and the 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.

```
def get_nested_value(nested_dict, keys):
    current_dict = nested_dict
    for key in keys:
        if key in current_dict:
```

```
            current_dict = current_dict[key]
        else:
            return None
    return current_dict

# Example usage:
nested_dict = {
    'a': {
        'b': {
            'c': 42,
            'd': 10
        },
        'e': 99
    },
    'f': 7
}

keys = input("Enter a list of keys separated by commas: ").split(',')

value = get_nested_value(nested_dict, keys)

if value is not None:
    print(f"The value at the specified path is: {value}")
else:
    print("One or more keys do not exist in the dictionary.")
```

Enter a list of keys separated by commas: a

The value at the specified path is: {'b': {'c': 42, 'd': 10}, 'e': 99}

Q.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):
    # Sort the dictionary by values
    sorted_items = sorted(input_dict.items(), key=lambda item: item[1], reverse=not ascending)

    # Convert the sorted items back to a dictionary
    sorted_dict = dict(sorted_items)

    return sorted_dict

# Example usage:
input_dict = eval(input("Enter the dictionary: "))

order = input("Sort in ascending or descending order? (asc/desc): ").strip().lower()

# Determine the order based on user input
ascending = order == "asc"

# Get the sorted dictionary
sorted_dict = sort_dict_by_values(input_dict, ascending)
```

```
# Output the result

print("Sorted dictionary:", sorted_dict)
```

```
Enter the dictionary:  {'apple': 5, 'banana': 2, 'cherry': 7}

Sort in ascending or descending order? (asc/desc):  asc

Sorted dictionary: {'banana': 2, 'apple': 5, 'cherry': 7}
```

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 in inverted_dict:
            inverted_dict[value].append(key)
        else:
            inverted_dict[value] = [key]

    return inverted_dict

# Example usage:

input_dict = eval(input("Enter the dictionary: "))

# Get the inverted dictionary

inverted_dict = invert_dictionary(input_dict)
```

```
# Output the result

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

```
Enter the dictionary:  {'apple': 1, 'banana': 2, 'cherry': 1, 'date': 3, 'fig': 2}

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