In [ ]:
```
#1. Write a code to reverse a string
```

In [1]:
```
a = "pwskills"
print(a[::-1])
```

```
sllikswp
```

In [ ]:
```
#2.Write a code to count the number of vowels in a string.
```

In [2]:
```
def count_vowels(s):
    vowels = "aeiou" or "AEIOU"
    return sum(1 for i in s if i in vowels)
s = input("enter a string here")
print("Number of vowels:" ,
count_vowels(s))
```

```
Number of vowels: 3
```

In [ ]:
```
#3. Write a code to check if a given string is a palindrome or not.
```

In [3]:
```
def check_palindrome(s):
    return s ==s[::-1]
s = input("enter a string here")
if check_palindrome(s):
    print("It is a palindrome")
else:
    print("it is not a plindrome")
```

```
It is a palindrome
```

In [ ]:
```
#4. Write a code to check if two given string are anagrams of each other
```

In [4]:
```
a = "heart"
b = "earth"

if sorted (a.lower()) == sorted (b.lower()):
    print("both string are angrams")
else:
    print("both string are not anagrams")
```

```
both string are angrams
```

In [5]:
```
#5.Write a code to find all occurrences of a given substring within another string.
```

In [6]:
```
text = "Hello world ,  Hello guys"
word = "Hello"
print(word, "occurs" , text.count(word))
```

```
Hello occurs 2
```

In [ ]:
```
#6. Write a code to perform basic string compression using the counts of repeated characters.
```

In [35]:
```
a = "CCCCDDDD"
count = 1

for i in range(1,len(a)):
    if a[i] == a[i-1]:
        count+=1
else:
    print(a[i-1]+ str(count) , end = "") count =1
print(a[-1]+ str(count))
```

```
D7D1
```

In [ ]:
```
#7. Write a code to determine if a string has all unique characters.
```

In [16]:
```
a = "abcdefghijk"
print(len(a) == len(set(a)))
```

```
True
```

In [ ]:
```
#8. Write a code to convert a given string to uppercase or lowercase.
```

In [7]:
```
a = "devendra singh"
print(a.upper())
```

```
DEVENDRA SINGH
```

In [ ]:
```
#9. Write a code to count the number of words in string
```

In [8]:
```
a = "koshish karne walon ki kabhi haar nahi hoti"
print(len(a.split()))
```

```
8
```

In [ ]:
```
#10. Write a code to count to concatenate two string without using the + operator
```

In [9]:
```
a = "pw"
b = "skills"

output = "".join([a , b])
print(output)
```

```
pwskills
```

In [ ]:
```
#11.Write a code to remove all occurrences of a specific element from a list.
```

In [46]:
```
def remove_element(lst , element):
    new_list = [x for x in lst if x != element]
    return new_list
my_list = [1,2,3,4,5]
element_to_remove = 2
new_list = remove_element(my_list,element_to_remove)
print(new_list)
```

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

In [ ]:
```
#12.Implement a code to find the second largest number in a given list of integers
```

In [47]:
```
num = [10 , 20 , 30 ,40 ,50]
print(sorted(set(num))[-2])
```

```
40
```

In [ ]:
```
#13. Create a code to count the occurrences of each element in a list and return a dictionary with elements as keys and their counts as values
```

In [53]:
```
numbers = [1,2,2,3,3,3,4,4,4,4]
print({i:numbers.count(i) for i in set(numbers)})
```

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

In [ ]:
```
#14. Write a code to reverse a list in-place without using any built-in reverse functions.
```

In [55]:
```
my_list= [1,2,3,4,5]
my_list = my_list[::-1]
print(my_list)
```

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

In [ ]:
```
#15. Implement a code to find and remove duplicates from a list while preserving the original order of elements.
```

In [10]:
```
students = ["Rohan" , "Virat" ,"Virat" , "Rohit" , "Rohit" , "Bhuvan" , "Bhuvan", "Tarun" ,"Tarun"]
print(list(set(students)))
```

```
['Rohit', 'Bhuvan', 'Virat', 'Tarun', 'Rohan']
```

In [ ]:
```
#16. Create a code to check if a given list is sorted(either in ascending or descending order) or not
```

In [58]:
```
lst = [1,2,3,4,5]
print(lst ==sorted(lst)or lst==sorted(lst,reverse = True))
```

```
True
```

In [ ]:
```
#17. Write a code to merge two sorted lists into a single sorted list.
```

In [61]:
```
list1 = [1,3,5]
list2 = [2,4,6]
merged_list = sorted(list1 + list2)
print(merged_list)
```

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

In [ ]:
```
#18.Implement a code to find the intersection of two given list.
```

In [62]:
```
list1 = [1,2,3,4,5]
list2 = [4,5,6,7,8]
intersection  = [element for element in list1 if element in list2]
print(intersection)
```

```
[4, 5]
```

In [ ]:
```
#19. Create a code to find the union of two lists without duplicates.
```

In [63]:
```
list1 = [1,2,3,4,5]
list2 = [4,5,6,7,8]
union = list(set(list1+list2))
print(union)
```

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

In [ ]:
```
#20. Write a code to shuffle a given list randomly without using any built-in shuffle functions.
```

In [65]:
```
import random
list1 = [1,2,3,4,5]
random.shuffle(list1)
print(list1)
```

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

In [ ]:
```
#21. Write a code that takes two tuples as input and returns a new tuples containing elements that are common to both input tuples.
```

In [69]:
```
tuple1 = (1,2,3,4,5)
tuple2 = (4,5,6,7,8)

common_tuples = tuple(set(tuple1)&set(tuple2))
print(common_tuples)
```

```
(4, 5)
```

In [ ]:
```
#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.
```

In [70]:
```
a = input("Enter first set of integer(separated by commas):")
b = input("Enter second set of integer(separated by commas):")
print("Intersection:",set(a.split(','))&set(b.split(',')))
```

```
Intersection: {'4', '5'}
```

In [ ]:
```
#23. Write a code to concatenate two tuples.The function should take two tuples as input ans return a new tuple containing elements from both input tuples.
```

In [71]:
```
t1 = (1,2,3)
t2 = (4,5,6)
result = t1+t2
print(result)
```

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

In [ ]:
```
#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.
```

In [75]:
```
set1 = set(input("Enter first set of strings:").split())
set2= set(input("Enter second set of strings:").split())
print(set1-set2)
```

```
{'banana', 'mango'}
```

In [ ]:
```
#25. Create a code that takes a tuple and two integers as input. The function should return a new tuple containing elements from the
```

In [77]:
```
tup = (1,2,3,4,5,6,7,8,9)
start = 3
end = 7
new_tup = tup[start : end]
print(new_tup)
```

```
(4, 5, 6, 7)
```

In [ ]:
```
#26. Write a code that prompts the user to input two sets of characters. Then, print the union of these two sets.
```

In [115...
```
print("union of two sets: " , set(input("enter first set of characters: "))|
      set(input("enter second set of characters: ")))
```

```
union of two sets:  {'b', 'c', 'a', 'd'}
```

In [ ]:
```
#27. Develop a code that takes a tuple odf integers as input.The function should return the maximum and minimum values from the tuple using tuple unpacking.
```

In [80]:
```
a ,b = input("Enter two integers separated by space: ").split()
print("Maximum value:" , max(int(a), int(b)))
print("Maximum value:" , min(int(a), int(b)))
```

```
Maximum value: 100
Maximum value: 50
```

In [ ]:
```
#28. Create a code that defines two sets of integers. Then , print the union , intersection , and difference of these two sets.
```

In [84]:
```
set1 = {1,2,3,4,5}
set2 = {4,5,6,7,8}

print("Union:" , set1|set2)
print("Intersection:" , set1&set2)
print("Difference:" ,set1^set2)
```

```
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference: {1, 2, 3, 6, 7, 8}
```

In [ ]:
```
#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.
```

In [89]:
```
tup = tuple(input("Enter a tuple(elements seperated by comma):").split(','))
element = input("Enter an element: ")
print(tup.count(element))
```

```
2
```

In [ ]:
```
#30. Develop a code that prompts the user to input two sets of strings. Then, print the symmetric difference of these two sets.
```

In [92]:
```
set1 = set(input("Enter first set of strings(elements seperated by comma):").split(','))
set2 = set(input("Enter second set of strings(elements seperated by comma):").split(','))
print("Symmetrical difference" , set1^set2)
```

```
Symmetrical difference {'6', '5', '1', '2'}
```

In [ ]:
```
#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.
```

In [112...
```
word_list = ["apple", "banana", "apple", "orange", "banana", "banana"]
print({word: word_list.count(word)for word in set(word_list)})
```

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

In [ ]:
```
#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.
```

In [114...
```
dict1 = {"a": 1 , "b": 2 , "c": 3}
dict2 = {"b":4 ,"c": 5 , "d": 6}
merged_dict = {k:dict1.get(k,0)+dict2.get(k,0) for k in set (dict1)|set(dict2)}
print(merged_dict)
```

```
{'b': 6, 'c': 8, 'a': 1, 'd': 6}
```

In [ ]:
```
#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
```

In [105...
```
nested_dict = {"a":{"b":{"c":"value"}}}
keys  = ["a" , "b" , "c" ]
value = nested_dict
for key in keys:
    if isinstance(value, dict) and key in value:
        value = value[key]
    else:
        print("Unknown")
        break
else:
    print(value)
```

```
value
```

In [ ]:
```
#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.
```

In [108...
```
dictionary = {"apple": 5 , "banana": 10 , "cherry" : 3}
print(dict(sorted(dictionary.items(),key = lambda item:item[1])))
print(dict(sorted(dictionary.items(), key = lambda item:item[1],reverse = True)))
```

```
{'cherry': 3, 'apple': 5, 'banana': 10}
{'banana': 10, 'apple': 5, 'cherry': 3}
```

In [ ]:
```
#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
```

In [109...
```
dictionary = {"a": 1 , "b": 2 , "c": 1, "d": 3}

inverted_dict = {}
for key, value in dictionary.items():
```

```
    inverted_dict.setdefault(value,[]).append(key)
print(inverted_dict)
```

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

In [ ]: