Assignment 3

Github: Github/bha411

1. Reverse string:

```
#REVERSE STRING
String1 = "BHAVYA"
for i in range(-1,-(len(String1))-1,-1):
    print(String1[i],end="")
```

[8] ✓ 0.0s

```
AYVAHB
```

2. Number of Vowels in string:

```
#2. Number of Vowels in string:

String1="bhavya"
count=0
vowels = ["a","e","i","o","u"]
for i in String1:
    if i in vowels:
        count+=1
    else:
        pass
print(f"total vowels={count}")
```

[13]

```
total vowels=2
```

3. String Palindrome:

```
#3. String Palindrome
String2 = "racecar"
if String2== String2[::-1]:
    print(f"The word {String2} is a palindrome")
else :
    print("Not a palindrome")
```

[7] ✓ 0.0s

```
The word racecar is a palindrome
```

4. String anagram:

```
#4. String anagram
String3 = "Listen"
String4 = "Silent"

if sorted(String3.lower()) ==  sorted(String4.lower()):
    print(f"{String3} and {String4} are anagrams")

else:
    print("not anagrams")
```

[14] ✓ 0.0s

```
Listen and Silent are anagrams
```

5. Occurance of a substring:

```
#5. Occurance of a substring
String5 = " hi my name is bhavya gupta. this is an assignment"
substr5 = "is"
count = 0
pos = []
while True:
    count = String5.find(substr5,count)

    if count == -1:
        break
    pos.append(count)
    count+=1

print(f"substr {substr5} is found at {pos} position")
```

[22] ✓ 0.0s

```
substr is is found at [12, 31, 34] position
```

6. String Compression:

```
String6= "aaabccdddd"
compressed_string = ""
count = 1
for i in range(len(String6) - 1):
    if String6[i] == String6[i + 1]:
        count += 1
    else:

        compressed_string += String6[i] + str(count)
        count = 1


compressed_string += String6[-1] + str(count)

print(f"Compressed string: {compressed_string}")
```

[25] ✓ 0.0s

```
Compressed string: a3b1c2d4
```

7. String has unique chars:

```
#7. string has unique chars
String7 = "Bhavya Gupta"
chars = []

for i in String7:
    chars.append(i)

if len(chars)==len(set(chars)):
    print("All chars are unique")

else:
    print("Duplicates found")
```

[29] ✓ 0.0s

```
Duplicates found
```

8. String to Upper or Lower

```
#8. String Upper or Lower
String8 = " This is a MIXED case StrINg"

print("the lower case will be ",String8.lower())
print("the upper case will be ",String8.upper())
```

[30] ✓ 0.0s

```
the lower case will be   this is a mixed case string
the upper case will be   THIS IS A MIXED CASE STRING
```

9. Counting the number of words in string:

```
#9. Counting the number of words in a string
String9 = "This is the string with seven words"

print("total number of words in the string =",len(String9.split()))
```

[33] ✓ 0.0s

```
total number of words in the string = 7
```

10. Concatenation of 2 strings without + operator.

```
#10. Concat 2 str without +
String10 = "My name is "
String1 = "Bhavya Gupta"
combined_string = "".join([String10,String1])

print("the concatenated string = ",combined_string)
```

[38] ✓ 0.0s

```
the concatenated string =  My name is Bhavya Gupta
```

11. Removal of all occurrences of a specific element:

```
#11 remove all occuerances of an element:

List11 = [1,1,1,2,3,4,5,6,7,5,3,2,2]
element = 2
while element in List11 :
    List11.remove(element)

print(List11)
```

[47] ✓ 0.0s

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

12. Find Second largest in a list of int:

```
#12. Second largest int in a list

List12 = [1,2,4,5,6,7,8,9,0]
element = max(List12)
while element in List12:
    List12.remove(element)

print(f"second largest number after {element} is  {max(List12)}")
```

[53] ✓ 0.0s

```
second largest number after 9 is  8
```

13. Count the occurrence of each element and return a dict for them with frequency and element as the key value pair.

```
#13. Count the occurance of each element in a list and then creating a dictionary for the same.
List13 = [1,1,2,3,4,4,4,5,6,6,6,6,7,7]
dict = {}

for element in List13:
    if element in dict:
        dict[element] +=1
    else:
        dict[element] = 1

print(dict)
```

[56] ✓ 0.0s

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

14. Reversing a string without using the builtin functions for the same:

```
#14. Reversing a list without using builtin fucntions:

List14 = [1,1,2,3,4,4,5,5,6,7]

for i in range(len(List14)//2):
    List14[i],List14[len(List14)-i-1] = List14[len(List14)-i-1],List14[i]

print(List14)
```

[59] ✓ 0.0s

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

15. Removal of duplicates, preserving the original Order:

```
#15. Find and remove the duplicates, preserving the original order

List15 = [1,2,3,3,3,4,5,6,6,6,7,7,8,8]
empt_list = []
for element in (List15):
    if element in empt_list:
        continue
    else:
        empt_list.append(element)

print(empt_list)
```

[65] ✓ 0.0s

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

16. To check if a given list is sorted or not.

```python
#16 To check if a given list is sorted or not.

List16 = [2,3,4,5,6,78]
Copy_lst = List16.copy()
Copy_lst2 = List16.copy()
List16.sort()
Copy_lst.sort(reverse=True)
# print(List16,Copy_lst,Copy_lst2)
if List16==Copy_lst2:
    print("List is sorted in ASCENDING order")
elif Copy_lst2 == Copy_lst:
    print("List is sorted in DESCENDING order")
else:
    print("List is not sorted")
```

[104] ✓ 0.0s

```
List is sorted in ASCENDING order
```

17. Merge 2 sorted lists.

```python
#17. Merging 2 sorted lists
Sorted_list1 = [1,2,3,4,5]
Sorted_list2 = [6,7,8,9]

i,j = 0,0
merged = []

while i<len(Sorted_list1) and j<len(Sorted_list2):
    if Sorted_list1[i]>= Sorted_list2[j]:
        merged.append(Sorted_list2[j])
        j+=1
    else:
        merged.append(Sorted_list1[i])
        i+=1

while i<len(Sorted_list1):
    merged.append(Sorted_list1[i])
    i+=1

while j<len(Sorted_list2):
    merged.append(Sorted_list2[j])
    j+=1

print(merged)
```

[105] ✓ 0.0s

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

18. Intersection of 2 lists:

```
#18 Intersection of 2 strings:
List18_1 = [1,2,2,2,2,3,4,5,6,7]
List18_2 = [2,3,4,6,1,8,9,6,2,1]

intersection = []

for i in List18_1:
    if i in List18_2:
        intersection.append(i)

print(intersection)
```

[109] ✓ 0.0s

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

19. Union of 2 lists without duplicate:

```
#19 Union of 2 lists without duplicates:

List19_1 = [1,2,3,4,4,5,6,1,2,8,9]
List19_2 = [1,2,34,5,6,7,9,7,5,343,2,1,1]

union = []

for i in List19_1:
    if i in List19_2:
        union.append(i)

print(set(union))
```

[111] ✓ 0.0s

```
{1, 2, 5, 6, 9}
```

20. Shuffling a list without using any inbuilt methods:

```
#20 Shuffling the list randomly without susing any builting methods

import random

List20 = [1,2,3,4,5,6]

for i in List20:
    j = random.randint(0, len(List20) - 1)
    List20[i], List20[j] = List20[j], List20[i]

print(List20)
```

[113] ✓ 0.0s

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

\

21. Taking 2 tuples as input and returning a new tuple with unique elemts form both:

```
#21. Taking 2 tuples as inout and returning a new one with no duplicates:

Tuple21_1 = (1,2,3,4,3,6,6)
Tuple21_2 = (2,3,4,5,6,7,1,1,1)

def new_tup(Tuple21_1,Tuple21_2):
    return tuple(set(Tuple21_1 + Tuple21_2))
print(new_tup(Tuple21_1,Tuple21_2))
```

[115] ✓ 0.0s

```
(1, 2, 3, 4, 5, 6, 7)
```

22. Taking 2 sets as input and ouput the intersection:

```
#22. taking 2 sets as input and returing the intersection of 2

Set22_1 = set(input("enter the set"))#input given was 1,2,3,4
Set22_2 = set(input("enter the 2nd set"))#input given was 1,2,3,4,5,6,7,8
print(Set22_1&Set22_2)
```

[122] ✓ 11.3s

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

23. Concat 2 tuples:

```python
#23. concatenation of 2 tuples and getting a single:

def concat_tup(t1,t2):
    return t1 + t2

tuple1 = (1,2,3)
tuple2 = (2,3,4)

concat_tuple = concat_tup(tuple1,tuple2)
print(concat_tuple)
```

[123] ✓ 0.0s

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

24. Taking 2 tupples as input and fining out the different elements in the two:

```python
#24. finfing different elements that belong to 2 cutom tuples:
set1_input = input("Enter the first set of strings separated by commas: ") #input given: ('bhavya','ai', '
set2_input = input("Enter the second set of strings separated by commas: ") #input given :(bhavya, artific


set1 = set(set1_input.split(','))
set2 = set(set2_input.split(','))

difference = set1 - set2


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

[2] ✓ 42.2s Python

```
Elements in the first set but not in the second set: {'ai', 'pw', 'gupta', 'pwskills', 'ml'}
```

25. Taking a tuple and 2 integers as input and extracting the range of mentioned indices.

```python
#25. Taking a tuple as input and extractng the range of indices
def tup_range(original_tuple,start_index,end_index):
    return original_tuple[start_index:end_index]




Tuple25 = input("enter the set of elements sepearted by a comma ") #1,2,3,4,5,6,7,8,9
int25_start = int(input("enter the starting index")) #2
int25_end = int(input("enter hte endING index")) #5

Final_tup = tup_range(Tuple25,int25_start,int25_end)

print(Final_tup)
```

[10] ✓ 12.8s

```
2,3
```

26. *Taking 2 sets of characters as input and printing the union of the two.

```
#26. Taking 2 sets of characters as input and printing the union of the two

Set26_1 = set(input("enter the values separated by comma")) #bhavya
Set26_2 = set(input("enter the values separated by comma")) #physics wallah

print(Set26_1.union(Set26_2))
```

[14] ✓ 7.2s

```
{'l', 'a', 's', 'y', 'i', ' ', 'w', 'p', 'h', 'b', 'c', 'v'}
```

27. Taking a tuple as input and printing the min and max of the tuple:

```
#27. Function that return min and max of the tuple taken as input
def find_max_min(values_tuple):
    min_value, max_value = min(values_tuple), max(values_tuple)
    return min_value, max_value

Tuple27 = input("Enter the tuple of integers separated by commas: ") #3,5,43,543,545,3,3,22,35,1

values_tuple = tuple(map(int, Tuple27.split(',')))

min_val, max_val = find_max_min(values_tuple)

print("Minimum value:", min_val)
print("Maximum value:", max_val)
```

[17] ✓ 9.8s

```
Minimum value: 1
Maximum value: 545
```

28. Prining union,intersection and difference of 2 sets:

```
#28. Prining union,intersection and difference of 2 sets
set28_1 = {1, 2, 3, 4, 5}
set28_2 ={4, 5, 6, 7, 8}

union_set = set28_1.union(set28_2)

intersection_set = set28_1.intersection(set28_2)

difference_set = set28_1.difference(set28_2)

print("Set 1:", set28_1)
print("Set 2:", set28_2)
print("Union:", union_set)
print("Intersection:", intersection_set)
print("Difference (Set 1 - Set 2):", difference_set)
```

[19] ✓ 0.0s

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

29. Function to return the count of an element in the tuple:

```
#29. Fucntion to return the count of an element in the tuple:

tup29 = input("enter the values sepeated by a comma") #1,2,3,4,1,2,1,1

tup29 = tuple(map(int, tup29.split(',')))

element = int(input("enter the vlaue to search for")) #1
count=0
for i in tup29:
    if i == element:
        count+=1

print(count)
```

[23] ✓ 13.6s

```
4
```

30. Printing the symmetric diff:

```python
#prinint the symmentric difference of 2 sets
Set30_1 = input("Enter the first set of strings separated by commas: ") #bhavya,gupta,ml,ai

Set30_2 = input("Enter the second set of strings separated by commas: ") #bhavya,physics wallah, ds

set1 = set(Set30_1.split(','))
set2 = set(Set30_2.split(','))

symmetric_diff = set1.symmetric_difference(set2)

print("Symmetric Difference between the two sets:", symmetric_diff)
```

[25] ✓ 33.7s

```
Symmetric Difference between the two sets: {'ai', 'gupta', 'ml', 'physics wallah', ' ds'}
```

31. Printing the frequency of the word in a lsit and plotting it to dictionary:

```python
#31. Printing the frequency of the word in a lsit and plotting it to dictionary
words_input = input("Enter a list of words separated by spaces: ")#Bhavya Gupta ai ai ml ds

words_list = words_input.split()

word_frequencies = {}

for word in words_list:
    if word in word_frequencies:
        word_frequencies[word] += 1
    else:
        word_frequencies[word] = 1

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

[28] ✓ 20.6s

```
Word Frequencies: {'Bhavya': 1, 'Gupta': 1, 'ai': 2, 'ml': 1, 'ds': 1}
```

32. Merging 2 dictionaries into one and then adding the values with common keys:

```python
#32. Merging 2 dictionaries into one and then adding the values with common keys
dict1 = {'apple': 3, 'banana': 2, 'orange': 5}
dict2 = {'banana': 3, 'orange': 1, 'kiwi': 2}

merged_dict = {}

for key, value in dict1.items():
    merged_dict[key] = value

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

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

[29] ✓ 0.0s

```
Merged Dictionary: {'apple': 3, 'banana': 5, 'orange': 6, 'kiwi': 2}
```

33. Accessing the value in a nested dict

```python
#33. Accessing the value in a nested dict.
nested_dict = {
    'person': {
        'name': 'Bhavya',
        'age': 21,
        'address': {
            'city': 'Delhi',
            'zipcode': 110070
        }
    }
}

def access_nested_value(dictionary, keys):
    current_value = dictionary
    for key in keys:
        if key in current_value:
            current_value = current_value[key]
        else:
            return None
    return current_value

keys_to_find = ['person', 'address', 'city']
result = access_nested_value(nested_dict, keys_to_find)

print(f"Value for the keys {keys_to_find}: {result}")
```

[31] ✓ 0.0s

```
Value for the keys ['person', 'address', 'city']: Delhi
```

34. Function to sort the dict by its values:

```python
#34. Function to sort the dict by its values:
def sort_dict_by_values(input_dict, ascending=True):
    sorted_items = sorted(input_dict.items(), key=lambda x: x[1], reverse=not ascending)

    sorted_dict = dict(sorted_items)

    return sorted_dict

sample_dict = {'apple': 5, 'banana': 3, 'orange': 8, 'kiwi': 2}

sorted_dict_asc = sort_dict_by_values(sample_dict, ascending=True)
print("Sorted in ascending order:", sorted_dict_asc)

sorted_dict_desc = sort_dict_by_values(sample_dict, ascending=False)
print("Sorted in descending order:", sorted_dict_desc)
```

[32] ✓ 0.0s

```
Sorted in ascending order: {'kiwi': 2, 'banana': 3, 'apple': 5, 'orange': 8}
Sorted in descending order: {'orange': 8, 'apple': 5, 'banana': 3, 'kiwi': 2}
```

35. Fucntion to invert the key value pairs of a dictionary:

```python
#35 Fucntion to inver the key value pairs:

def invert_dict(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

sample_dict = {'apple': 5, 'banana': 3, 'orange': 5, 'kiwi': 3, 'mango': 8}
inverted = invert_dict(sample_dict)
print("Inverted dictionary:", inverted)
```

[33] ✓ 0.0s

```
Inverted dictionary: {5: ['apple', 'orange'], 3: ['banana', 'kiwi'], 8: ['mango']}
```