Functions and Modules

Q1. **Write a Python function that accepts a string and counts the number of upper and lower case letters.** Sample String : 'The quick Brow Fox' Expected Output : No. of Upper case characters : 3 No. of Lower case Characters : 12

```
def count_letters(s):
    uppercase_count=0
    lowercase_count=0

    for char in s:
        if char.isupper():
             uppercase_count += 1
        elif char.islower():
            lowercase_count += 1
    print(f"No. of Upper case character :{uppercase_count}")
    print(f"No. of lower case character :{lowercase_count}")

s = "The quick Brow Fox"
count_letters(s)
```

Q2. **Write a Python function that takes a list and returns a new list with distinct elements from the first list.** Sample List : [1,2,3,3,3,3,4,5] Unique List : [1, 2, 3, 4, 5]

```
def unique_list(lst):
    return list(set(lst))

sample_list= [1,2,3,3,3,3,4,5]
unique_list= unique_list(sample_list)
print(f'Unique List :{unique_list}')
```

Q3.**Write a Python function to check whether a number is "Perfect" or not. Hint: In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). Example : The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6) / 2 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 +

1. This is followed by the perfect numbers 496 and 8128.**

```
def number_isperfect(n):
    if n < 1:
        return False
    sum_divisors = 0
    for i in range(1, n):
        if n % i == 0:
            sum_divisors += i
    return sum_divisors == n

n= int(input("Enter a number to check if it is perfect : "))

if number_isperfect(n):
    print(f'{n} is a perfect number.')
else :
    print(f'{n} is not a perfect number.')
```

Q4. **Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically.**

```
def hypen_separated_sequence(sequence):
    words = sequence.split('-')
    words.sort()
    return '-'.join(words)

u= input("Enter a hyphen-separated sequence of words:")
sorted_sequence = hypen_separated_sequence(u)
print(f"hyphen-separated sequence of words: {sorted_sequence}")
```

Q5. **WAP to demonstrate the functionality of positional argument in functions ?**

```
def details_student(name,id,birth_date,cohort):
    print( f"Details of the Student is:\nName of the student:{name}\
nid:{id}\nBirth_date: {birth_date}\ncohort name:{cohort}\n")

details_student('Raj koli ', id='1234',birth_date='31/10/2006',
cohort= 'Elon Musk')
```

Q6.**WAP to demonstrate the functionality of keyword argument in functions ?**

```
def details_student(name,id,birth_date,cohort):
    print(f'Details of the Student is:\nName of the student:{name}\
nid:{id}\nBirth_date: {birth_date}\ncohort name:{cohort}\n')
```

```
details_student(name='Raj koli ', id='13345255', birth_date=
'31/10/2006', cohort='Elon Musk')
```

Q7. **WAP to demonstrate how positional arguments and keyword arguments can be used in functions**

```
def details_student(name,id,birth_date,cohort):
    Detail1 = {
        'Name' : name,
        'id': id,
        'birth_date' : birth_date,
        'cohort': cohort
     }
    return Detail1

by_using_positional_arg= details_student('Raj koli', '1234' ,
'31/10/2006', 'Elon Musk')
by_using_keyword_arg= details_student('Raj koli', '12344' ,
'31/10/2006', 'Elon Musk')

print(f"Details of the Student using Positional Arg is:
{by_using_positional_arg}\n ")
print(f"Details of the Student using keyword Arg is:
{by_using_keyword_arg} ")
```

Q8. **WAP to demonstrate the functionality of default argument in functions**

```
def greet_user(name , greeting= 'Hello'):
    print(f'{greeting}, {name}!')

greet_user('Pranav')
greet_user('Kale' , 'Welcome')
```

Q9. **WAP to demonstrate how variable length arguments are used in functions.**

```
def sum_number(*args):
    return sum(args)
print(sum_number(1,2,3,4,5,6,7))
```

Q10. **WAP to demonstrate how variable length keyword arguments are used in functions.**

```
def dislay_student_info(**kwargs):
    print("Student Information: ")
    for key, value in kwargs.items():
                print(f'{key}: {value}')

dislay_student_info(name='Raj',age=17,
city='Mumbai',Subject='Computer-Science')
```

Q 11. **WAP to demonstrate how to create your own modules for common mathematical operations and import it and use it.**

```
import Module_Operations as math

a=10
b=20

print(f'{a} + {b} = {math.add(a,b)}')
print(f'{a} - {b} = {math.subtract(a,b)}')
print(f'{a} * {b} = {math.multiply(a,b)}')
print(f'{a} / {b} = {math.divide(a,b)}')
```

Q12. **WAP to demonstrate following functions in math module:**

a. Ceil

b. Trunc

c. Floor

d. Factorial

e. Fabs

f. Pow

g. Fmod

h. Fsum

i. Prod

j. sqrt

```
import math
from math import prod

a= 5.75
b= -3.4
c=5
```

```
d=10

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

print(f'Ceil of {a}: {math.ceil(a)}')

print(f'Trunc of {a}: {math.trunc(a)}')

print(f'Floor of {a}: {math.floor(a)}')

print(f'Factorial of {a}: {math.factorial(c)}')

print(f'Absolute of {a}: {math.fabs(a)}')

print(f'{c}raised to the power of {d}: {math.pow(c,d)}')

print(f' Fmod of {d} and {c}: {math.fmod(d,c)}')

print(f'Fsum of {numbers} and {c}: {math.fsum(numbers)}')

print(f'Prod of {numbers}: {prod(numbers)}')

print(f"Square root of {c}: {math.sqrt(c)}")
```

Q 13. **WAP to demonstrate following functions in random module:**

a. random()

b. randint()

c. uniform()

d. choice()

e. shuffle()

f. randrange()

```
import random

print("random(): ,random.random()")

print('randint(1,10):)', random.randint(1,10))

print("uniform(1.0 ,10.0):", random.uniform(1.0,10.0))

sequence = ['apple', 'banana','cherry','date']
print("choice(sequence):", random.uniform(1.0,10.0))
```

```
random.shuffle(sequence)
print('choice(sequnce):', sequence)

print('randrange(1,10,2):', random.randrange(1,10,2))
```

List

Q14. **Write a Python program to remove duplicates from a list.**

```
def remove_duplicates(lst):
    return list(set(lst))

original_list = [1,2,3,3,4,5,6,6,7,8]
unique_list=  remove_duplicates(original_list)
print('Original list:', original_list)
print("List without duplicates:",original_list)
```

Q15.**Write a Python function that takes two lists and returns True if they have at least one common member.**

```
def have_common_member(list1, list2):
    for element in list1:
        if element in list2:
            return True
    return False

list_1 = [1,2,3,4,5]
list_2 = [2,3,1,2,4,4,5]
print(have_common_member(list_1, list_2))

list_3 = [1,2,3,4,5]
list_4 = [7,6,8,9,7]
print(have_common_member(list_3, list_4))
```

Q16. **Write a Python program to print the numbers of a specified list after removing even numbers from it.**

```
def print_odd_number(number_list):
    for number in number_list:
        if number %2 !=0 :
            print(number)

my_list =[ 1,2,3,4,5,6,7,8,9]
print_odd_number(my_list)
```

Q17. **Write a Python program to find the second smallest number in a list.**

```
def second_smallest(input_list):
    if len(input_list)< 2 :
        return None
    unique_member = list(set(input_list))
    unique_member.sort()
    return unique_member[1]

sample_list = [12,13,1,10,34,9,1]
second_smallest = second_smallest(sample_list)
print(f'The Second smallest number is: {second_smallest}')
```

Q18. **Write a Python program to split a list every Nth element.**

```
def split_list_nth(lst,n):
    return [lst[i:i+n] for i in range(0,len(lst),n)]
sample_list = [1,2,3,4,5,6,7,8,9,10]
n=2
split_list= split_list_nth(sample_list,n)

print(f'Original list: {sample_list}')
print(f'List split every {n}th element: {split_list}')
```

Q19. **Write a Python function to find the union and intersection of two lists.**

```
def union_intersection(list_1,list_2):
    union = list(set(list_1).union(set(list_2)))
    intersection = list(set(list_1).intersection(set(list_2)))
    return union , intersection

list_1= [1,2,3,4,5,6,]
list_2= [4,5,6,7,8,9]
union_result, intersection_result = union_intersection(list_1,list_2)

print("Union:",union_result)
print("Intersection:",intersection_result)
```

Q20. **Write a Python func;on to check if a list is a palindrome or not. Return true otherwise false.**

```
def is_palindrome(lst):
    return lst == lst[::-1]
```

```
list1=[1,2,3,2,1]
list2=[1,2,4,5,6,4,5]
print(is_palindrome(list1))
print(is_palindrome(list2))
```

Q21. **WAP to create a menu based program for insertion, deletion, access, updation, traversal of list elements.**

```
def display_menu():
    print("\nMenu:")
    print("1. Insert an element")
    print("2. Delete an element")
    print("3. Access an element")
    print("4. Update an element")
    print("5. Traverse the list")
    print("6. Exit")

def main():
    my_list = []

    while True:
        display_menu()
        choice = input("Choose an option (1-6): ")

        if choice == '1':
            element = input("Enter the element to insert: ")
            my_list.append(element)
            print(f"Inserted: {element}")

        elif choice == '2':
            element = input("Enter the element to delete: ")
            if element in my_list:
                my_list.remove(element)
                print(f"Deleted: {element}")
            else:
                print(f"{element} not found in the list.")

        elif choice == '3':
            index = int(input("Enter the index of the element to
access: "))
            if 0 <= index < len(my_list):
                print(f"Element at index {index}: {my_list[index]}")
            else:
                print("Index out of range.")

        elif choice == '4':
            index = int(input("Enter the index of the element to
update: "))
            if 0 <= index < len(my_list):
```

```
                new_value = input("Enter the new value: ")
                my_list[index] = new_value
                print(f"Updated index {index} to {new_value}.")
            else:
                print("Index out of range.")

        elif choice == '5':
            print("List elements:")
            for i, elem in enumerate(my_list):
                print(f"Index {i}: {elem}")

        elif choice == '6':
            print("Exiting the program.")
            break

        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

Q22. **WAP to create create, access, add elements, delete, modify elements in nested list.**

```
def display_menu():
    print("\nMenu:")
    print("1. Create a nested list")
    print("2. Access an element")
    print("3. Add an element")
    print("4. Delete an element")
    print("5. Modify an element")
    print("6. Display the nested list")
    print("7. Exit")

def main():
    nested_list = []

    while True:
        display_menu()
        choice = input("Choose an option (1-7): ")

        if choice == '1':
            # Create a nested list
            nested_list = [
                ["apple", "banana"],
                ["carrot", "daikon"],
                ["eggplant", "fig"]
            ]
            print("Nested list created.")
```

```
        elif choice == '2':
            # Access an element
            row = int(input("Enter the row index: "))
            col = int(input("Enter the column index: "))
            try:
                print(f"Element at [{row}][{col}]: {nested_list[row]
[col]}")
            except IndexError:
                print("Index out of range.")

        elif choice == '3':
            # Add an element
            row = int(input("Enter the row index to add an element:
"))
            element = input("Enter the element to add: ")
            try:
                nested_list[row].append(element)
                print(f"Added '{element}' to row {row}.")
            except IndexError:
                print("Row index out of range.")

        elif choice == '4':
            # Delete an element
            row = int(input("Enter the row index to delete an element:
"))
            col = int(input("Enter the column index: "))
            try:
                deleted_element = nested_list[row].pop(col)
                print(f"Deleted element '{deleted_element}' from
[{row}][{col}].")
            except IndexError:
                print("Index out of range.")

        elif choice == '5':
            # Modify an element
            row = int(input("Enter the row index of the element to
modify: "))
            col = int(input("Enter the column index: "))
            try:
                new_value = input("Enter the new value: ")
                nested_list[row][col] = new_value
                print(f"Modified element at [{row}][{col}] to
'{new_value}'.")
            except IndexError:
                print("Index out of range.")

        elif choice == '6':
            # Display the nested list
            print("Current nested list:")
            for row in nested_list:
```

```
                print(row)

        elif choice == '7':
            print("Exiting the program.")
            break

        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

```
Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Index out of range.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
```

```
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
```

```
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
```

```
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
```

```
6. Display the nested list
7. Exit
Invalid choice, please try again.

Menu:
1. Create a nested list
2. Access an element
3. Add an element
4. Delete an element
5. Modify an element
6. Display the nested list
7. Exit
```

Q23. **Write a program for various list slicing operation.** a= [10,20,30,40,50,60,70,80,90,100]

i. Print Complete list

ii. Print 4th element of list

iii. Print list from 0th to 4th index.

iv. Print list -7th to 3rd element

v. Appending an element to list.

vi. Sorting the element of list.

vii. Popping an element.

viii. Removing Specified element.

ix. Entering an element at specified index.

x. Counting the occurrence of a specified element.

xi. Extending list.

xii. Reversing the list.

```
a = [10,20,30,40,50,60,70,80,90,100]

print("Compare list: ",a)

print('4th element of list: ', a[3])

print('List from 0th th 4th index: ', a[0:5])

print('List from -7th to 3rd element: ', a[-7:4])

a.append(110)
print("List after appending 110: " , a)

a.sort()
```

```
print("List after sorting:",a)

popped_element = a.pop()
print("Popped element: ", popped_element)
print("list after popping an element: ", a)

a.remove(40)
print("List after removing 40:",a)

a.insert(2,45)
print("List after inserting 45 at index 2: ",a)

count_60= a.count(60)
print("Count of 60 in list: ", count_60)

a.extend([120,130,140])
print("List after extending: ",a)



a.reverse()
print("List after reversing: ",a)
```

Q24. **WAP to add two matrices using nested list ?**

```
def add_matrices(matrix1 , matrix2):
    result = []

    for i in range(len(matrix1)):
        result.append([matrix1[i][j] + matrix2[i][j] for j in
range(len(matrix1[0]))])
    return result

matrix1= [
    [1,2,3],
    [4,5,6],
    [7,8,9]
]

matrix2 = [
    [9,8,7],
    [6,5,4],
    [3,2,1]
]

result = add_matrices(matrix1,matrix2)

print("Sum of matrices: ")
for row in result:
```

```
    print(row)
```

Q25. **Consider a list with mixed type of elements such as l1=[1,'x',4,5.6,'z',9,'a',0,4]. Create another list l2 using list comprehension which consist of only integer element present within list l1.**

```
l1 = [1,'x',4, 5.6, 'y',9,'z',0.4]

l2 = [item for item in l1 if isinstance(item,int) ]

print('List of integers from 11:', 12)
```

Tuple

Q26. **Write a Python program to compute the element-wise sum of given tuples.** Original : (1, 2, 3, 4) (3, 5, 2, 1) (2, 2, 3, 1) Element-wise sum of the said tuples: (6, 9, 8, 6)

```
tuple1 = (1,2,3,4)
tuple2 = (3,5,2,1)
tuple3 = (2,2,3,1)

elementwise_sum = []
for i in range(len(tuple1)):
    elementwise_sum.append(tuple1[i] + tuple2[i]  + tuple3[i])

elementwise_sum = tuple(elementwise_sum)
print("Element_wise sum of the said tuples: ", elementwise_sum)
```

Q27. **Write a Python program to convert a given list of tuples to a list of lists. Original list of tuples: [(1, 2), (2, 3), (3, 4)] . Convert the said list of tuples to a list of lists: [[1, 2], [2, 3], [3, 4]]**

```
original_tuples = [(1,2),(2,3),(3,4)]
list_of_list = [list(t) for t in  original_tuples ]
print("Convert the said list of tuples to a list of lists:
",list_of_list)
```

Q28.**Write a Python program to remove an empty tuple(s) from a list of tuples**

```
tuples_list= [(1,2),(),(3,4),(),(5,)]

filtered_list = [t for t in tuples_list if t ]
print("List after removing empty tuples: ", filtered_list)
```

Q29.**Write a Python program to convert a given string to a tuple**

```
string = "hello world this is Python"
```

```
tuple_of_words = tuple(string.split())
print("Tuple of words: ", tuple_of_words)
```

Q30. **Write a Python program to calculate the product, mul;plying all the numbers in a given tuple.**

```
num= (1,2,3,4,5)
prod=1

for num in num:
    prod *= num

print("Product of the number in the tuple: ",prod)
```

Q31. **WAP to create a list of square of numbers from 1 to 20 using List comprehension**

```
squares=[]

for x in range(1,21):
    squares.append(x**2)

print("List of squares from 1 to 20: ", squares)
```

Set

Q32. **Write a Python program to remove an item from a set if it is present in the set.**

```
my_set = {1,2,3,4,5}
item_to_remove = 4

if item_to_remove in my_set:
    my_set.remove(item_to_remove)

print("Set after removing the items",my_set)
```

Q33.**Write a Python program to check if two given sets have no elements in common.**

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

if set1 & set2 == set():
    print("The two sets have no elements in common. ")
else:
    print("The two sets have some elements in common. ")
```

Q34. **Get Only unique items from two sets**

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

unique_list = set1.union(set2) - set1.intersection(set2)
print(unique_list)
```

Q35.**Write a Python program to Convert Set to one String**

```
my_set = {'apple','banana','cherry'}
set_as_string = ', '.join(my_set)
print(set_as_string)
```

Q36.**WAP to count number of vowels using sets in given string**

```
input_string = input("Enter the string:")
vowels = 'aeiouAEIOU'
count=0

for char in input_string:
    if char in vowels:
        count += 1
print(count)
```

Q37.**WAP to create a set of cubes of even numbers from 2 to 12 using set comprehension.**

```
cubes_of_even_num = {x**3 for x in range(2,13,2)}
print(sorted(cubes_of_even_num))
```

Dictionary

Q38.**Write a Python script to sort (ascending and descending) a dictionary by value.**

```
my_dict = { 'apple':3, 'banana' :1 ,'cherry':2, 'date':4}

sorted_asc ={k: v for k, v in sorted(my_dict.items(), key=lambda item:
item[1])}
print("Sorted by value (ascending):", sorted_asc)

sorted_desc = {k: v for k, v in sorted(my_dict.items(), key=lambda
item: item[1], reverse=True)}
print("Sorted by value (descending):", sorted_desc)
```

Q39.**Write a Python program to remove duplicates from the dictionary.**

```
my_dict = {
    'apple':1,
    'banana': 2,
    'cherry': 2,
```

```
    'date': 3,
    'elderberry': 1
}

unique_dict = {}
for key, value in my_dict.items():
    if value not in unique_dict.values():
        unique_dict[key] = value


print("Dictionary after removing duplicates:", unique_dict)
```

Q40. **Write a Python program to combine two dic;onary by adding values for common keys.**

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


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

result = combine_dictionaries(dict1, dict2)
print(result)
```

Q41.**Write a Python program to create a dictionary from a string. (Track the count of the letters from the string.)**

```
def count_letters(input_string):
    letter_count = {}  # Initialize the dictionary here

    for char in input_string:
        if char.isalpha():
            char = char.lower()
            if char in letter_count:
                letter_count[char] += 1
            else:
                letter_count[char] = 1

    return letter_count

input_string = "Hello, World!"
```

```
result = count_letters(input_string)
print(result)
```

Q42.**Write a Python program to match key and values both, in two dictionaries.**

```
def match_dicts(dict1, dict2):
    matches = {}

    for key in dict1:
        if key in dict2 and dict1[key] == dict2[key]:
            matches[key] = dict1[key]

    return matches


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

result = match_dicts(dict1, dict2)
print(result)
```

Q43.**Use dic;onary comprehension to convert the price of following dic;onary from dollar to pound** old_price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}

```
conversion_rate = 0.73

old_price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}

new_price = {item: price * conversion_rate for item, price in
old_price.items()}

print(new_price)
```

Q44. **Use dic;onary comprehension to create a dic;onary to store only key value pairs having even age.** original_dict = {'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}

```
original_dict = {'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}

even_age_dict = {name: age for name, age in original_dict.items() if
age % 2 == 0}

print(even_age_dict)
```