```
#Write a Python function that accepts a string and counts the number
of upper and lowercase letters.

def count_upper_lower(string):
    upper_count = 0
    lower_count = 0
    for char in string:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
    return upper_count, lower_count

print(count_upper_lower("Hello World"))
```

```
(2, 8)
```

```
#Sample String : 'The quick Brow Fox'
#Expected Output :
#No. of Upper case characters : 3
#No. of Lower case Characters : 12

def count_upper_lower(string):
    upper_count = 0
    lower_count = 0
    for char in string:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
    return upper_count, lower_count

print(count_upper_lower("The quick Brow Fox"))
```

```
(3, 12)
```

```
#Write a Python function that takes a list and returns a new list with
distinct elements from the first list.

def distinct_elements(list1):
    return list(set(list1))

print(distinct_elements([1,2,3,3,3,3,4,5]))
```

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

```
#Write a Python function to check whether a number is "Perfect" or
not.

def perfect_num(n):
```

```
    sum = 0
    for i in range(1, n):
        if n % i == 0:
            sum += i
    return sum == n

print(perfect_num(28))
```

```
True
```

```
#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 sort_words(words):
    return '-'.join(sorted(words.split('-')))

print(sort_words("1-4-3-2"))
```

```
1-2-3-4
```

```
#WAP to demonstrate the functionality of positional argument in
functions

def positional_arg(a, b, c):
    print(a, b, c)

positional_arg(1, 2, 3)
```

```
1 2 3
```

```
#WAP to demonstrate the functionality of keyword argument in functions

def keyword_arg(a, b, c):
    print(a, b, c)

keyword_arg(a=1, b=2, c=3)
```

```
1 2 3
```

```
#WAP to demonstrate how positional arguments and keyword arguments can
be used in functions

def positional_keyword_arg(a, b, c):
    print(a, b, c)

positional_keyword_arg(1, 2, 3)
positional_keyword_arg(a=1, b=2, c=3)
```

```
1 2 3
1 2 3
```

```python
#WAP to demonstrate the functionality of default argument in functions

def default_arg(a=1, b=2, c=3):
    print(a, b, c)

default_arg()
default_arg(10, 20, 30)
default_arg(a=10, b=20, c=30)
```

```
1 2 3
10 20 30
10 20 30
```

```python
#WAP to demonstrate how variable length arguments are used in
functions.

def variable_length_arg(*args):
    print(args)

variable_length_arg(1, 2, 3, 4, 5)
variable_length_arg("Hello", "World")
```

```
(1, 2, 3, 4, 5)
('Hello', 'World')
```

```python
#WAP to demonstrate how variable length keyword arguments are used in
functions.

def variable_length_keyword_arg(**kwargs):
    print(kwargs)

variable_length_keyword_arg(a=1, b=2, c=3)
variable_length_keyword_arg(name="Saurabh", age=20, city="New York")
```

```
{'a': 1, 'b': 2, 'c': 3}
{'name': 'Saurabh', 'age': 20, 'city': 'New York'}
```

```python
#WAP to demonstrate how to create your own modules for common
mathematical operations and import it and use it.

import math

print(math.sqrt(16))
print(math.pow(2, 3))
print(math.pi)
```

```
4.0
8.0
3.141592653589793
```

```python
#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

print(math.ceil(1.2))
print(math.trunc(1.2))
print(math.floor(1.2))
print(math.factorial(5))
print(math.fabs(-1.2))
print(math.pow(2, 3))
print(math.fmod(10, 3))
print(math.fsum([1, 2, 3, 4, 5]))
print(math.prod([1, 2, 3, 4, 5]))
print(math.sqrt(16))
```

```
2
1
1
120
1.2
8.0
1.0
15.0
120
4.0
```

```
#WAP to demonstrate following functions in random module:
import random

print(random.random())
print(random.randint(1, 10))
print(random.uniform(1, 10))
print(random.choice([1, 2, 3, 4, 5]))
print(random.shuffle([1, 2, 3, 4, 5]))
print(random.randrange(1, 10, 2))
```

```
0.4566720044712216
8
2.226599392158444
5
None
5
```

```python
#Write a Python program to remove duplicates from a list.


def remove_duplicates(list1):
    return list(set(list1))

print(remove_duplicates([1, 2, 3, 3, 3, 3, 4, 5]))
```

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

```python
#Write a Python func;on that takes two lists and returns True if they
have at least one common member.

def common_member(list1, list2):
    return set(list1).intersection(set(list2))

print(common_member([1, 2, 3, 4, 5], [5, 6, 7, 8, 9]))
```

```
{5}
```

```python
#Write a Python program to print the numbers of a specified list after
removing even numbers from it.

def remove_even(list1):
    return [i for i in list1 if i % 2 != 0]

print(remove_even([1, 2, 3, 4, 5, 6, 7, 8, 9]))
```

```
[1, 3, 5, 7, 9]
```

```python
#Write a Python program to find the second smallest number in a list.

def second_smallest(list1):
    return sorted(list1)[1]

print(second_smallest([4,5,6,8]))
```

```
5
```

```python
#Write a Python program to split a list every Nth element.

def split_list(list1, n):
    return [list1[i::n] for i in range(n)]

print(split_list([1,2,3,4,5,6,7,8,9], 3))
```

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

```python
#Write a Python func;on to find the union and intersec;on of two
lists.

def union_intersection(list1, list2):
    return set(list1).union(set(list2)),
```

```
set(list1).intersection(set(list2))

print("Union lists and intersection lists: ", union_intersection([1,
2, 3, 4, 5], [4, 5, 6, 7, 8]))
```

```
Union lists and intersection lists:  ({1, 2, 3, 4, 5, 6, 7, 8}, {4,
5})
```

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

def palindrome(list1):
    return list1 == list1[::-1]

print(palindrome([3,5,3]))
print(palindrome([3,5,4]))
```

```
True
False
```

```
def menu():
    list1 = []
    while True:
        print("1. Insertion")
        print("2. Deletion")
        print("3. Access")
        print("4. Updation")
        print("5. Traversal")
        print("6. Exit")
        choice = input("Enter your choice: ").lower()

        if choice in ['1', 'insertion']:
            list1.append(input("Enter the element to be inserted: "))
            print(list1)
        elif choice in ['2', 'deletion']:
            element = input("Enter the element to be deleted: ")
            if element in list1:
                list1.remove(element)
            else:
                print("Element not found in the list.")
            print(list1)
        elif choice in ['3', 'access']:
            try:
                index = int(input("Enter the index of the element to
be accessed: "))
                if 0 <= index < len(list1):
                    print(list1[index])
                else:
                    print("Invalid index.")
            except ValueError:
```

```
                print("Please enter a valid integer for the index.")
        elif choice in ['4', 'updation']:
            try:
                index = int(input("Enter the index of the element to
be updated: "))
                if 0 <= index < len(list1):
                    list1[index] = input("Enter the new element: ")
                    print(list1)
                else:
                    print("Invalid index.")
            except ValueError:
                print("Please enter a valid integer for the index.")
        elif choice in ['5', 'traversal']:
            for i in list1:
                print(i)
        elif choice in ['6', 'exit']:
            print("Exiting the program.")
            return
        else:
            print("Invalid choice. Please try again.")

menu()
```

```
1. Insertion
2. Deletion
3. Access
4. Updation
5. Traversal
6. Exit
['3']
1. Insertion
2. Deletion
3. Access
4. Updation
5. Traversal
6. Exit
Element not found in the list.
['3']
1. Insertion
2. Deletion
3. Access
4. Updation
5. Traversal
6. Exit
[]
1. Insertion
2. Deletion
3. Access
4. Updation
5. Traversal
```

```
6. Exit
Exiting the program.

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

def nested_list():
    list1 = []
    while True:
        print("1. Create")
        print("2. Access")
        print("3. Add elements")
        print("4. Delete elements")
        print("5. Modify elements")
        print("6. Exit")
        choice = int(input("Enter your choice: "))
        if choice in ['1', 'create']:
            list1 = []
            print(list1)
        elif choice in ['2', 'access']:
            element = input("Enter the element to be accessed: ")
            if element in list1:
                print((element))
            else:
                print("Element not found in the list.")
        elif choice in ['3', 'add elements']:
            list1.append(input("Enter the element to be added: "))
            print(list1)
        elif choice in ['4', 'delete elements']:
            element = input("Enter the element to be deleted: ")
            if element in list1:
                list1.remove(element)
            else:
                print("Element not found in the list.")
            print(list1)
        elif choice in ['5', 'modify elements']:
            try:
                index = int(input("Enter the index of the element to
be modified: "))
                if 0 <= index < len(list1):
                    list1[index] = input("Enter the new element: ")
                    print(list1)
                else:
                    print("Invalid index.")
            except ValueError:
                print("Please enter a valid integer for the index.")
        elif choice in ['6', 'exit']:
            print("Exiting the program.")
            return
        else:
```

```
            print("Invalid choice. Please try again.")

nested_list()



#Write a program for various list slicing operation.
a= [10,20,30,40,50,60,70,80,90,100]
#i. Print Complete list
print(a[:])
#ii. Print 4th element of list
print(a[3])
#iii. Print list from 0th to 4th index.
print(a[0:4])
#iv. Print list -7th to 3rd element
print(a[-7:3])
#v. Appending an element to list.
a.append(110)
print(a)
#vi. Sorting the element of list.
a.sort()
print(a)
#vii. Popping an element.
a.pop()
print(a)
#viii. Removing Specified element.
a.remove(20)
print(a)
#ix. Entering an element at specified index.
a.insert(0, 5)
print(a)
#x. Counting the occurrence of a specified element.
print(a.count(10))
#xi. Extending list.
a.extend([11,12,13])
print(a)
#xii. Reversing the list.
a.reverse()
print(a)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
40
[10, 20, 30, 40]
[]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[10, 30, 40, 50, 60, 70, 80, 90, 100]
[5, 10, 30, 40, 50, 60, 70, 80, 90, 100]
1
```

```
[5, 10, 30, 40, 50, 60, 70, 80, 90, 100, 11, 12, 13]
[13, 12, 11, 100, 90, 80, 70, 60, 50, 40, 30, 10, 5]
```

```
#WAP to add two matrices using nested list

list1 = [[1,2,3], [4,5,6], [7,8,9]]
list2 = [[3,4,5], [4,5,7], [7,4,8]]
result = [[list1[i][j] + list2[i][j]
           for j in range(len(list1[0]))]
           for i in range(len(list1))]
print(result)
```

```
[[4, 6, 8], [8, 10, 13], [14, 12, 17]]
```

```
#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,'z',9,'a',0,4]
l2 = [i for i in l1 if type(i) == int]
print(l2)
```

```
[1, 4, 9, 0, 4]
```

```
#Write a Python program to compute the element-wise sum of given
tuples.

tuple1 = (1,2,3)
tuple2 = (4,5,6)
result = tuple(map(sum, zip(tuple1, tuple2)))
print(result)
```

```
(5, 7, 9)
```

```
#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)
result = tuple(map(sum, zip(tuple1, tuple2, tuple3)))
print(result)
```

```
(6, 9, 8, 6)
```

```python
#Write a Python program to convert a given list of tuples to a list of lists.

list_of_tuples = [(1, 2), (2, 3), (3, 4)]
list_of_lists = [list(tup) for tup in list_of_tuples]
print(list_of_lists)
```

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

```python
#Write a Python program to remove an empty tuple(s) from a list of tuples.

list_of_tuples = [(1, 2), (2, 3), (3, 4), (), ()]
list_of_tuples = [tup for tup in list_of_tuples if tup]
print(list_of_tuples)
```

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

```python
#Write a Python program to convert a given string to a tuple

string = "Hello World"
tuple_string = tuple(string)
print(tuple_string)
```

```
('H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd')
```

```python
#Write a Python program to calculate the product, multiplying all the numbers in a given tuple.

tuple_numbers = (1, 2, 3, 4, 5)
product = 1
for number in tuple_numbers:
    product *= number
print(product)
```

```
120
```

```python
#WAP to create a list of square of numbers from 1 to 20 using List comprehension

square_numbers = [i**2 for i in range(1, 21)]
print(square_numbers)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400]
```

```python
#Write a Python program to remove an item from a set if it is present in the set.

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

```
if item in set1:
    set1.remove(item)
print(set1)
```

```
{1, 2, 4, 5}
```

```
#Write a Python program to check if two given sets have no elements in common

set1 = {1, 2, 3, 4, 5}
set2 = {6, 7, 8, 9, 10}
if set1.isdisjoint(set2):
    print("Two sets have no elements in common")
else:
    print("Two sets have elements in common")
```

```
Two sets have no elements in common
```

```
#Get Only unique items from two sets

set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
unique_items = set1.symmetric_difference(set2)
print(unique_items)
```

```
{1, 2, 3, 6, 7, 8}
```

```
#Write a Python program to Convert Set to one String

set1 = {1, 2, 3, 4, 5}
string = ''.join(str(item) for item in set1)
print(string)
```

```
12345
```

```
#WAP to count number of vowels using sets in given string

string = "Hello World"
vowels = set("aeiouAEIOU")
count = sum(1 for char in string if char in vowels)
print(count)
```

```
3
```

```
#WAP to create a set of cubes of even numbers from 2 to 12 using set comprehension.

cubes_of_even_numbers = {i**3 for i in range(2, 13) if i % 2 == 0}
print(cubes_of_even_numbers)
```

```
{512, 64, 1728, 8, 1000, 216}
```

```
#Write a Python script to sort (ascending and descending) a dic;onary
by value.

d = {'a': 1, 'b': 2, 'c': 3}
sorted_d = sorted(d.items(), key=lambda x: x[1])
print(sorted_d)
sorted_d = sorted(d.items(), key=lambda x: x[1], reverse=True)
print(sorted_d)

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

d1 = {'a': 100, 'b': 200, 'c':300}
d2 = {'a': 300, 'b': 200, 'd':400}
d1.update(d2)
print(d1)
```

```
{'a': 300, 'b': 200, 'c': 300, 'd': 400}
```

```
#Write a Python program to create a dic;onary from a string. ( Track
the count of the letters from the string.)

string = "Hello World"
letter_count = {letter: string.count(letter) for letter in string}
print(letter_count)
```

```
{'H': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'W': 1, 'r': 1, 'd': 1}
```

```
#Write a Python program to match key and values both, in two
dic;onaries.

d1 = {'a': 1, 'b': 2, 'c': 3}
d2 = {'a': 1, 'b': 2, 'd': 4}
if d1.keys() == d2.keys() and d1.values() == d2.values():
    print("Both dictionaries have the same keys and values")
else:
    print("Both dictionaries do not have the same keys and values")
```

```
Both dictionaries do not have the same keys and values
```

```
#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}
new_price = {item: value*0.95 for (item, value) in old_price.items()}
print(new_price)
```

```
{'milk': 0.969, 'coffee': 2.375, 'bread': 2.375}
```

```
#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}
```

```
even_age_dict = {key: value for (key, value) in original_dict.items()
if value % 2 == 0}
print(even_age_dict)

{'jack': 38, 'michael': 48}
```