In [61]:
```
## Name: Ananya Agarwal
## Batch: 3CO14
## Roll No.: 102083036
##Submitted To: Dr. Sharad Saxena


## Sol 1:
print("Enter elements in the list in which you want to insert more elements or delete elements from it: ")
myList = []
n = int (input ("How many elements do you want to enter in the list initially :- "))
for i in range (n) :
    storeElement = int (input ("Enter an integer number in the list :- "))
    myList.append (storeElement)
print("List elements are: ", myList)

def insert():
    store=int(input("Input list element you want to add: "))
    myList.append (store)
    print("List elements are: ", myList)

def delete_value():
    item=int(input("Enter the item you want to delete: "))
    myList.remove(item)
    print("List elements are: ", myList)

def delete_position():
    pos=int(input("Enter the index of the item you want to delete: "))
    #del myList[pos]
    myList.pop(pos)
    print("List elements are: ", myList)


def delete_slicing():
    pos1=int(input("Enter the starting value from where you want to delete: "))
    pos2=int(input("Enter the ending value (included) till where you want to delete: "))
    del myList[pos1:pos2+1]
    print("List elements are: ", myList)

while True:
    print("\nMAIN MENU")
    print("Press 1 to insert element: ")
    print("Press 2 to delete element: ")
    print("Press 3 to exit: ")
```

```
    option = int(input("Enter your choice: "))

    if option == 1:
        insert()
    elif option == 2:
        print("\nSub choices to delete element are: ")
        print("Press 1 to delete element by value: ")
        print("Press 2 to delete element by position: ")
        print("Press 3 to delete element by slicing: ")
        option2 = int(input("Enter your choice: "))

        if option2 == 1:
            delete_value()
        elif option2 == 2:
            delete_position()
        elif option2 == 3:
            delete_slicing()
        else:
            print("\nIncorrect choice.")
            exit()
    elif option == 3:
        print("\nYou have exited from program. Thank you.")
        break
```

```
Enter elements in the list in which you want to insert more elements or delete elements from it:
How many elements do you want to enter in the list initially :- 4
Enter an integer number in the list :- 3
Enter an integer number in the list :- 2
Enter an integer number in the list :- 4
Enter an integer number in the list :- 1
List elements are:  [3, 2, 4, 1]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 1
Input list element you want to add: 4
List elements are:  [3, 2, 4, 1, 4]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 2
```

```
Sub choices to delete element are:
Press 1 to delete element by value:
Press 2 to delete element by position:
Press 3 to delete element by slicing:
Enter your choice: 3
Enter the starting value from where you want to delete: 4
Enter the ending value (included) till where you want to delete: 1
List elements are:  [3, 2, 4, 1, 4]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 2

Sub choices to delete element are:
Press 1 to delete element by value:
Press 2 to delete element by position:
Press 3 to delete element by slicing:
Enter your choice: 1
Enter the item you want to delete: 3
List elements are:  [2, 4, 1, 4]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 2

Sub choices to delete element are:
Press 1 to delete element by value:
Press 2 to delete element by position:
Press 3 to delete element by slicing:
Enter your choice: 2
Enter the index of the item you want to delete: 0
List elements are:  [4, 1, 4]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 2

Sub choices to delete element are:
Press 1 to delete element by value:
Press 2 to delete element by position:
```

```
Press 3 to delete element by slicing:
Enter your choice: 3
Enter the starting value from where you want to delete: 1
Enter the ending value (included) till where you want to delete: 3
List elements are:  [4]

MAIN MENU
Press 1 to insert element:
Press 2 to delete element:
Press 3 to exit:
Enter your choice: 3

You have exited from program. Thank you.
```

In [33]:

```
## Sol 2:
print("Enter elements in the list in which you want to insert more elements: ")
myList = []
n = int (input ("How many elements do you want to enter in the list initially :- "))
for i in range (n) :
    storeElement = int (input ("Enter an integer number in the list :- "))
    myList.append (storeElement)
print("List elements are: ", myList)

def multiple():
    n1 = int (input ("How many elements do you want to append in the list :- "))
    n = len(myList)
    for i in range (n1) :
        storeElement = int (input ("Enter an integer num :- "))
        myList.append (storeElement)
    print("List elements are: ", myList)
    #list2 = [1,2,3]
    #myList.extend(list2) #extend() adds all the elements of an iterable (list, tuple, string etc.) to end of list.

def single():
    storeelement = int (input ("Enter the element you want to insert in the list:"))
    myList.append (storeelement)
    print("List elements are: ", myList)

while True:
    print("\nMAIN MENU")
    print("Press 1 to insert a list of elements: ")
    print("Press 2 to insert a single element: ")
    print("Press 3 to exit: ")
    option = int(input("Enter your choice: "))
```

```
    if option == 1:
        multiple()
    elif option == 2:
        single()
    elif option ==3:
        print("\nYou have exited from program. Thank you.")
        break
```

```
Enter elements in the list in which you want to insert more elements:
How many elements do you want to enter in the list initially :- 3
Enter an integer number in the list :- 1
Enter an integer number in the list :- 2
Enter an integer number in the list :- 3
List elements are:  [1, 2, 3]

MAIN MENU
Press 1 to insert a list of elements:
Press 2 to insert a single element:
Press 3 to exit:
Enter your choice: 1
How many elements do you want to append in the list :- 5
Enter an integer num :- 5
Enter an integer num :- 6
Enter an integer num :- 4
Enter an integer num :- 3
Enter an integer num :- 2
List elements are:  [1, 2, 3, 5, 6, 4, 3, 2]

MAIN MENU
Press 1 to insert a list of elements:
Press 2 to insert a single element:
Press 3 to exit:
Enter your choice: 2
Enter the element you want to insert in the list:9
List elements are:  [1, 2, 3, 5, 6, 4, 3, 2, 9]

MAIN MENU
Press 1 to insert a list of elements:
Press 2 to insert a single element:
Press 3 to exit:
Enter your choice: 3

You have exited from program. Thank you.
```

In [13]:
```
## Sol 3:
alist = []
```

```
alist2 = []
alist3 = []

n = int(input("Enter the no. of elements you want to input: "))
if n >= 9:
    for i in range(n):
        inp = int(input("Enter an element: "))
        alist.append(inp)
    mytuple = tuple(alist)
    print("Entered tuple is: ", mytuple)
    alist2 = mytuple[-1:0:-3]
    #for i in range(len(mytuple)):
        #if i+1 > 3 and i+1 < 9 and (i+1) % 2 != 0:
            #alist3.append(mytuple[i])
    alist3 = mytuple[4:8:2]

    tup1 = tuple(alist2)
    tup2 = tuple(alist3)
    print("\nTuple containing every third element in reverse order, starting from last is: ")
    print(tup1)
    print("\nTuple containing alternate element between 3rd and 9th element is: ")
    print(tup2)

else:
    print("You must enter atleast 9 elements.")
```

```
Enter the no. of elements you want to input: 9
Enter an element: 2
Enter an element: 3
Enter an element: 4
Enter an element: 5
Enter an element: 6
Enter an element: 7
Enter an element: 8
Enter an element: 9
Enter an element: 1
Entered tuple is:  (2, 3, 4, 5, 6, 7, 8, 9, 1)

Tuple containing every third element in reverse order, starting from last is:
(1, 7, 4)

Tuple containing alternate element between 3rd and 9th element is:
(6, 8)
```

In [34]:

```
## Sol 4:
n = int (input ("Enter the total number of students: "))
```

```
list1=[]
for i in range(n):
    email=input("Enter email: ")
    list1.append(email)

tuple1=tuple(list1)

names=[]
domains=[]

for i in tuple1:
    #name,domain = i.split("@")
    #names.append(name)
    #domains.append(domain)
    temp = i.split("@")
    names.append(temp[0])
    domains.append(temp[1])

names = tuple(names)
domains = tuple(domains)

print("Original Tuple = ",tuple1)
print("Names = ",names)
print("Domains = ",domains)
```

```
Enter the total number of students: 2
Enter email: ananyaag06@gmail.com
Enter email: aag3_be19@thapar.edu
Original Tuple =  ('ananyaag06@gmail.com', 'aag3_be19@thapar.edu')
Names =  ('ananyaag06', 'aag3_be19')
Domains =  ('gmail.com', 'thapar.edu')
```

In [9]:

```
## Sol 5:
d={}

n=int(input('Enter number of the winners :'))

for i in range (n):

   a=input('Enter name of the winner:')

   b=int(input('Enter the number of wins of the winner:'))

   d[a]=b
```

```
print('\n')

print('name of the winners','\t','number of wins')

for i in d:

   print(i,'\t','\t','\t',d[i])

print('\n')
```

```
Enter number of the winners :5
Enter name of the winner:a
Enter the number of wins of the winner:2
Enter name of the winner:v
Enter the number of wins of the winner:3
Enter name of the winner:b
Enter the number of wins of the winner:4
Enter name of the winner:t
Enter the number of wins of the winner:5
Enter name of the winner:g
Enter the number of wins of the winner:5


name of the winners      number of wins
a                        2
v                        3
b                        4
t                        5
g                        5
```

In [25]:

```
## Sol 6:
dict1 = {0:'Zero',1:'One',2:'Two',3:'Three'}
value = input("Enter the value you want to match: ")
flag = 0

for i in dict1.keys():
    if dict1[i].lower() == value.lower():
        print(i)
        flag = 1
        break
if flag == 0:
    print("error!!")
```

```
Enter the value you want to match: five
error!!
```

In [32]:

```python
## Sol 7:
d={}
n=int(input("Enter the number of students: "))
print("\nEnter the name, class, roll no of students: ")

for i in range(n):
    print("Enter Details of student No.", i+1)
    rollno = int(input("\nEnter roll no: "))
    name = input("\nEnter name: ")
    marks = int(input("\nEnter marks: "))

    d[rollno] = [name,marks]

for i in d.values():
    if i[1] > 75:
        print(i[0])
```

```
Enter the number of students: 3

Enter the name, class, roll no of students:
Enter Details of student No. 1

Enter roll no: 1

Enter name: ananya

Enter marks: 500
Enter Details of student No. 2

Enter roll no: 2

Enter name: b

Enter marks: 4
Enter Details of student No. 3

Enter roll no: 3

Enter name: c

Enter marks: 4
ananya
```

In [62]:
```
## Sol 8:
sent = input("Enter the sentence: ")
dict = {}

for i in sent:
    if (i >= "a" and i <= "z" ) or (i >= "A" and i <= "Z") or (i >= "0" and i <= "9"):
        if i not in dict.keys():
            dict[i] = 1
        else:
            dict[i]+=1

print(dict)
```

```
Enter the sentence: Ananyaagarwal065
{'A': 1, 'n': 2, 'a': 5, 'y': 1, 'g': 1, 'r': 1, 'w': 1, 'l': 1, '0': 1, '6': 1, '5': 1}
```

In [63]:
```
## Sol 9:
str1 = input("Enter a string : ").lower()
vowels = ('a', 'e', 'i', 'o', 'u')
sub_str = ""
for i in range(len(str1)):
  temp = ""
  if str1[i] >= 'a' and str1[i] <= 'z' and str1[i] not in vowels:
    temp += str1[i]
    for j in range(i+1, len(str1)):
      if str1[j] >= 'a' and str1[j] <= 'z' and str1[j] not in vowels:
        temp += str1[j]
      else:
        break
      if len(temp) > len(sub_str):
        sub_str = temp
print(sub_str)
```

```
Enter a string : xprqaxeije
xprq
```

In [64]:
```
## Sol 10:
def Count(str):
    upper, lower, digit, symbol, alphabet  = 0, 0, 0, 0, 0
    for i in range(len(str)):
        if str[i].isupper():
            upper += 1
        if str[i].islower():
            lower += 1
```

```
        if str[i].isdigit():
            digit += 1
        if str[i].isalpha():
            alphabet += 1
        if not str[i].isalnum():
            symbol += 1
#Python isalnum() only returns true if a string contains alphanumeric characters, without symbols.


    print('\nUpper case letters:', upper)
    print('\nAlphabets:', alphabet)
    print('\nLowercase letters:', lower)
    print('\nDigits:', digit)
    print('\nSymbols:', symbol)


line = input("Enter a line: ")
Count(line)
```

```
Enter a line: anANy12@#'

Upper case letters: 2

Alphabets: 5

Lowercase letters: 3

Digits: 2

Symbols: 3
```

In [ ]:

In [5]:
```
## Name: Ananya Agarwal
## Batch: 3CO14
## Roll No.: 102083036
##Submitted To: Dr. Sharad Saxena

##Sol 1:
class Emp:
    EmpId = 0
    EmpName = None
    Points = 0
    Group = None
    Average_Points = 0


    def __init__(self,EmpId = None, EmpName = None):
        self.EmpId = EmpId
        self.EmpName = EmpName

    def addPoints(self,Points):
        self.Points = self.Points + Points

    def removePoints(self,Points):
        self.Points = self.Points - Points
        if self.Points < 0:
            self.Points = 0

    def computeGroup(self):
        if self.Points <= 100:
            self.Group = "Silver"
        elif self.Points > 100 and self.Points <= 500:
            self.Group = "Gold"
        elif self.Points > 500 and self.Points <= 1000:
            self.Group = "Platinum"
        elif self.Points > 1000:
            self.Group = "Diamond"

    def count_Groupwise(self):
        if self.Group == "Silver":
            lSilver.append(self.EmpName)
        elif self.Group == "Gold":
            lGold.append(self.EmpName)
```

```
        elif self.Group == "Platinum":
            lPlatinum.append(self.EmpName)
        elif self.Group == "Diamond":
            lDiamond.append(self.EmpName)

    def display_details(self):
        print("\nEmployee ID of the employee is: ",self.EmpId)
        print("Name of the employee is: ",self.EmpName)
        print("Points of the employee is: ",self.Points)
        print("Group of the employee is: ",self.Group)

lSilver = []
lGold = []
lPlatinum = []
lDiamond = []


e1 = Emp(1,"Ananya")
e2 = Emp(2,"Vasu")
e3 = Emp(3,"Pooja")

e1.addPoints(5000)
e1.removePoints(4999)
e1.computeGroup()

e2.addPoints(1000)
e2.removePoints(20)
e2.computeGroup()

e3.addPoints(700)
e3.removePoints(2)
e3.computeGroup()

e1.count_Groupwise()
e2.count_Groupwise()
e3.count_Groupwise()

print("Total no. of employees are 3!!")
n=3

print("\nThe employees with group Silver are: ",lSilver)
print("The employees with group Gold are: ",lGold)
```

```
print("The employees with group Platinum are: ",lPlatinum)
print("The employees with group Diamond are: ",lDiamond)

e1.display_details()
e2.display_details()
e3.display_details()


Average_Points = (e1.Points+e2.Points+e3.Points)/n
print("\nAverage Points of the 3 employees are: ",Average_Points)
```

```
Total no. of employees are 3!!

The employees with group Silver are:  ['Ananya']
The employees with group Gold are:  []
The employees with group Platinum are:  ['Vasu', 'Pooja']
The employees with group Diamond are:  []

Employee ID of the employee is:  1
Name of the employee is:  Ananya
Points of the employee is:  1
Group of the employee is:  Silver

Employee ID of the employee is:  2
Name of the employee is:  Vasu
Points of the employee is:  980
Group of the employee is:  Platinum

Employee ID of the employee is:  3
Name of the employee is:  Pooja
Points of the employee is:  698
Group of the employee is:  Platinum

Average Points of the 3 employees are:  559.6666666666666
```

In [2]:
```
##Sol 2:
class Property:
    square_footage = 1500
    no_bedrooms = 4
    no_bathrooms = 4

    def __init__(self,square_footage = 1200, no_bedrooms = 3, no_bathrooms = 3):
        self.sqaure_footage = square_footage
        self.no_bedrooms = no_bedrooms
        self.no_bathrooms = no_bathrooms

class House(Property):
    no_stories = 2
    garage = None
    yard_fenced = False

    def __init__(self,no_stories = 3, garage = "Attached", yard_fenced = True):
        self.no_stories = no_stories
        self.garage = garage
        self.yard_fenced = yard_fenced

    def display_House(self):
        print("\nFollowing are the details of the property house: ")
        print("Square footage of the house: ",self.square_footage)
        print("No. of bedroooms in the house: ",self.no_bedrooms)
        print("No. of bathroooms in the house: ",self.no_bathrooms)
        print("No. of stories in the house: ",self.no_stories)
        print("Type of garage in the house: ",self.garage)
        print("Is the yard of the house fenced or not: ",self.yard_fenced)

class Apartment(Property):
    balcony = True
    laundry = "coin"

    def __init__(self,balcony = False, laundry = "en-suite"):
        self.balcony= balcony
        self.laundry = laundry

    def display_Apartment(self):
        print("\nFollowing are the details of the property Apartment: ")
        print("Square footage of the Apartment: ",self.square_footage)
```

```
        print("No. of bedroooms in the Apartment: ",self.no_bedrooms)
        print("No. of bathroooms in the Apartment: ",self.no_bathrooms)
        print("Is there a balcony present in the Apartment: ",self.balcony)
        print("Type of laundry in the Apartment: ",self.laundry)

class Rental:
    rent_cost = 20000
    is_furnished = True
    utility_included = False

    def __init__(self,rent_cost = 15000, is_furnished = False, utility_included = True):
        self.rent_cost = rent_cost
        self.is_furnished = is_furnished
        self.utility_included = utility_included

    def display_rental(self):
        print("\nFollowing are the details of the properties being rented: ")
        print("Rent per month is: ",self.rent_cost)
        print("Is the property furnished or not: ",self.is_furnished)
        print("Are the utilities included or not: ",self.utility_included)

class HouseRental(House,Rental):
    pass

class ApartmentRental(Apartment,Rental):
    pass

class Purchase:
    purchase_price = 1000000
    annual_tax = 20000

    def __init__(self,purchase_price = 10000000, annual_tax = 1000000):
        self.purchase_price = purchase_price
        self.annual_tax = annual_tax

    def display_purchase(self):
        print("\nFollowing are the details of the properties being purchased: ")
        print("Purchase price is: ",self.purchase_price)
        print("estimated annual property taxes are: ",self.annual_tax)

class HousePurchase(House,Purchase):
    pass
```

```
class ApartmentPurchase(Apartment,Purchase):
    pass

def insert():

    while True:
        print("\nMenu to insert more properties in the agent list: ")
        print("Which class object do you want to create: ")

        print("Press 1 to add more properties(Houses) in the agent which are of type rented: ")
        print("Press 2 to add more properties(Houses) in the agent which are of type purchased: ")
        print("Press 3 to add more properties(Apartments) in the agent which are of type rented: ")
        print("Press 4 to add more properties(Apartments) in the agent which are of type purchased: ")
        print("Press 5 to exit: ")

        option = int(input("Enter your choice: "))

        if option == 1:
            o_hr_1 = HouseRental()
            print("New object of HouseRental class created at memory location: \n",o_hr_1)
            agent.append (o_hr_1)
            print("\nAfter appending one more object of HouseRental type, properties the agent now has are: \n", agent)

        elif option == 2:
            o_hp_1 = HousePurchase()
            print("New object of HousePurchase created at memory location: \n",o_hp_1)
            agent.append (o_hp_1)
            print("\nAfter appending one more object of HousePurchase type, Properties the agent now has are: \n", agent

        elif option == 3:
            o_ar_1 = ApartmentRental()
            print("New object of ApartmentRental created at memory location: \n",o_ar_1)
            agent.append (o_ar_1)
            print("\nAfter appending one more object of AppartmentRental type, Properties the agent now has are: \n", ag

        elif option == 4:
            o_ap_1 = ApartmentPurchase()
            print("New object of ApartmentPurchase created at memory location: \n",o_ap_1)
            agent.append (o_ap_1)
            print("\nAfter appending one more object of ApartmentPurchase type, Properties the agent now has are: \n", a
```

```
        elif option == 5:
            print("\nYou have exited from insertion option. Redirecting you to main menu!!")
            break



o_hr = HouseRental(1500,"Attached",True)
o_hp = HousePurchase(1600,"Dettached",False)
o_ar = ApartmentRental(True,"coin")
o_ap = ApartmentPurchase(False,"en-suite")

agent = [o_hr, o_ar, o_hp, o_ap]

while True:
    print("\nMAIN MENU")
    print("Press 1 to add more objects (houses and apartments on rent or purchase) in the agent: ")
    print("Press 2 display all the data related to house which are to be put on rent: ")
    print("Press 3 display all the data related to house which are to be put for purchase: ")
    print("Press 4 display all the data related to apartment which are to be put on rent: ")
    print("Press 5 display all the data related to apartment which are to be put for purchase: ")
    print("Press 6 to exit: ")

    option = int(input("Enter your choice: "))

    if option == 1:
        insert()

    elif option == 2:
        o_hr.display_House()
        o_hr.display_rental()

    elif option == 3:
        o_hp.display_House()
        o_hp.display_purchase()

    elif option == 4:
        o_ar.display_Apartment()
        o_ar.display_rental()

    elif option == 5:
        o_ap.display_Apartment()
```

```
        o_ap.display_purchase()

    elif option == 6:
        print("\nYou have exited from program. Thank you.")
        break
```

```
MAIN MENU
Press 1 to add more objects (houses and apartments on rent or purchase) in the agent:
Press 2 display all the data related to house which are to be put on rent:
Press 3 display all the data related to house which are to be put for purchase:
Press 4 display all the data related to apartment which are to be put on rent:
Press 5 display all the data related to apartment which are to be put for purchase:
Press 6 to exit:
Enter your choice: 1

Menu to insert more properties in the agent list:
Which class object do you want to create:
Press 1 to add more properties(Houses) in the agent which are of type rented:
Press 2 to add more properties(Houses) in the agent which are of type purchased:
Press 3 to add more properties(Apartments) in the agent which are of type rented:
Press 4 to add more properties(Apartments) in the agent which are of type purchased:
Press 5 to exit:
Enter your choice: 4
New object of ApartmentPurchase created at memory location:
```

In [ ]:

In [ ]:

In [ ]:

In [5]:
```
#sol1
import numpy as np

arr=np.array([67,56,45,89,99,76])
even = arr % 2 == 0
odd = arr % 2 != 0
print("All even elements of the original array arr:",arr[even])
print("All odd elements of the original array arr:",arr[odd])
arr = np.where(arr % 2 == 0,0,arr)
print("Original array in which 0 is there at the place of even number:",arr)
```

```
All even elements of the original array arr [56 76]
All odd elements of the original array arr [67 45 89 99]
Original array in which 0 is there at the place of even number [67  0 45 89 99  0]
```

In [30]:
```
#sol2
import numpy as np

arr=np.array([67,56,45,89,99,76])
copy_arr = np.where(arr % 2 != 0,'odd',arr)
print("Original array is: ",arr)
print("Array after required change is: ",copy_arr)
```

```
Original array is:  [67 56 45 89 99 76]
Array after required change is:  ['odd' '56' 'odd' 'odd' 'odd' '76']
```

In [45]:
```
a = np.array(['How','Are','You'])
b = np.repeat(a, 3)
c = np.tile(a, 3)
b = np.append(b,c,0) #list without comma
joined_string = ",".join(b) #string with comma
my_list = joined_string.split(",") #list with comma
print(my_list)
```

```
['How', 'How', 'How', 'Are', 'Are', 'Are', 'You', 'You', 'You', 'How', 'Are', 'You', 'How', 'Are', 'You', 'How', 'Are',
'You']
```

In [15]:
```
#intersect1d method
array1 = np.array(['a','A','E','b','c','B'])
print("Array1: ",array1)
array2 = np.array(['a','Y','B'])
print("Array2: ",array2)
print("Common values between two arrays:")
print(np.intersect1d(array1, array2))
```

```
Array1:  ['a' 'A' 'E' 'b' 'c' 'B']
Array2:  ['a' 'Y' 'B']
Common values between two arrays:
['B' 'a']
```

In [20]:
```
my_array = np.arange(12).reshape(3, 4)
print("Original array:")
print(my_array)

my_array[:,[2,3]] = my_array[:,[3,2]]
print("\nAfter swapping last 2 columns of the given array: ")
print(my_array)

my_array[[0,1],:] = my_array[[1,0],:]
print("\nAfter swapping starting 2 rows of the given array: ")
print(my_array)
```

```
Original array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

After swapping last 2 columns of the given array:
[[ 0  1  3  2]
 [ 4  5  7  6]
 [ 8  9 11 10]]

After swapping starting 2 rows of the given array:
[[ 4  5  7  6]
 [ 0  1  3  2]
 [ 8  9 11 10]]
```

In [25]:
```
#The uniform() method returns random floating number between the two specified numbers (both included).
rand_arr = np.random.uniform(6,12, size=(5,3))
print(rand_arr)
```

```
[[10.56216378 11.01722648  8.05407174]
 [ 7.66141252 11.1867239   8.99687268]
 [11.6872732   7.76161484 10.58513173]
 [ 6.74894499  6.40590414  8.35119122]
 [11.89893807 11.62035001  7.55383214]]
```

In [52]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

iris_2d = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
#print(iris_2d[:4]) #4 rows and all columns

print(iris_2d) #all rows and all columns
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
```

In [53]:
```
sepallength = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0])
mu, med, sd = np.mean(sepallength), np.median(sepallength), np.std(sepallength)
print(mu, med, sd)
```

```
5.843333333333334 5.8 0.8253012917851409
```

In [ ]:
```

```

```
# Q1

import numpy as np

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
iris_1d = np.genfromtxt(url, delimiter=',', dtype = "float,"*4 + "U20", names=True)
iris_2d = np.array([row.tolist()[:4] for row in iris_1d])
iris_class = np.array([row.tolist()[4] for row in iris_1d])

sepal_len = float(input("Enter the sepal length : "))
sepal_width = float(input("Enter the sepal width : "))
petal_len = float(input("Enter the petal length : "))
petal_width = float(input("Enter the petal width : "))
arr = np.array([sepal_len, sepal_width, petal_len, petal_width])
print(arr)

i = 0
dic = {}
for row in iris_2d:
  d = np.linalg.norm(row - arr)
  dic[i] = d
  i += 1
  #i=149
dic = dict(sorted(dic.items(), key=lambda item: item[1]))

cnt = 0
print("\nClass of closest 5 samples : ")
for i in dic.keys():
  cnt += 1
  print(iris_class[i])
  if cnt == 5:
    break

#np.linalg.norm() is called on an array-like input to compute the square root of the sum of squared elements
#humne x means key value pair diya for all the items kyuki apne aap d.items se ek loop chal rha hai on all elements
#and x[1] means value is picked up and then sorted and stored in sorted dictionary d
```

```
#dictionary ki key hai excel ka serial number and value hai distance jo aayi
#same k varible ko use krenge toh excel ka class aajeyga by output_column[k]
```

```
     Enter the sepal length : 2
     Enter the sepal width : 3
     Enter the petal length : 2
     Enter the petal width : 2

     Class of closest 5 samples :
     Iris-setosa
     Iris-setosa
     Iris-setosa
     Iris-setosa
     Iris-setosa
```

```
# Q2

import numpy as np

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
iris_1d = np.genfromtxt(url, delimiter=',', dtype = "float,"*4 + "U20", names=True)
iris_2d = np.array([row.tolist()[:4] for row in iris_1d])
iris_class = np.array([row.tolist()[4] for row in iris_1d])

sepal_len = float(input("Enter the sepal length : "))
sepal_width = float(input("Enter the sepal width : "))
petal_len = float(input("Enter the petal length : "))
petal_width = float(input("Enter the petal width : "))
arr = np.array([sepal_len, sepal_width, petal_len, petal_width])

from scipy.spatial import distance

i = 0
dic = {}
for row in iris_2d:
  d = distance.cityblock(row, arr)
  dic[i] = d
  i += 1
```

```
    i += 1
dic = dict(sorted(dic.items(), key=lambda item: item[1]))

cnt = 0
print("\nClass of closest 5 samples : ")
for i in dic.keys():
  cnt += 1
  print(iris_class[i])
  if cnt == 5:
    break

  #scipy.spatial.distance.cityblock computes the City Block (Manhattan) distance between vectors u and v
```

```
 Enter the sepal length : 2
 Enter the sepal width : 3
 Enter the petal length : 2
 Enter the petal width : 2

 Class of closest 5 samples :
 Iris-setosa
 Iris-setosa
 Iris-setosa
 Iris-setosa
 Iris-setosa
```

```
# Q3

import numpy as np

dic = {"best case" : [], "avg case" : [], "worst case" : []}
limit = 1

df = np.genfromtxt("student.csv", delimiter = ',', dtype = int, names = True)
rno = np.array([row.tolist()[0] for row in df])
marks = np.array([row.tolist()[1:] for row in df])

for i in range(len(rno)):
  print("Mean of marks of student with roll no ", rno[i], " : ", np.mean(marks[i]))
  if abs(np.mean(marks[i]) - np.mean(marks)) <= limit:
```

```
      dic["avg case"].append(rno[i])
    elif np.mean(marks[i]) > np.mean(marks):
      dic["best case"].append(rno[i])
    else:
      dic["worst case"].append(rno[i])

print("\nTotal mean of data stored : ", np.mean(marks))

print("\n", dic)
```

```
     Mean of marks of student with roll no  1  :  63.6
     Mean of marks of student with roll no  2  :  62.4
     Mean of marks of student with roll no  3  :  67.4
     Mean of marks of student with roll no  4  :  95.6
     Mean of marks of student with roll no  5  :  67.4
     Mean of marks of student with roll no  6  :  65.0
     Mean of marks of student with roll no  7  :  41.4
     Mean of marks of student with roll no  8  :  45.6
     Mean of marks of student with roll no  9  :  54.4
     Mean of marks of student with roll no  10  :  76.0

     Total mean of data stored :  63.88

      {'best case': [3, 4, 5, 6, 10], 'avg case': [1], 'worst case': [2, 7, 8, 9]}
```

```
# Q4

import numpy as np
import random as r

n= int(input("Enter the no of elements required in random sequence : "))

ran = []
for i in range(n):
  ran.append(r.randint(0, 149))
ran = np.array(ran)

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

```
iris_1d = np.genfromtxt(url, delimiter=',', dtype = "float,"*4 + "U20", names=True)
iris_2d = np.array([row.tolist()[:4] for row in iris_1d])

df = []
for i in ran:
  df.append(iris_2d[i])
df = np.array(df)

print("\nMean of randomly selected samples : ", np.mean(df))
print("Standard Deviation of randomly selected samples : ", np.std(df))
```

```
     Enter the no of elements required in random sequence : 6

     Mean of randomly selected samples :  3.775
     Standard Deviation of randomly selected samples :  2.002342378315956
```

In [1]:
```
#sol1
try:
    fp = open('abc.txt')    # Open the file in reading mode
    c=0
    vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

    for char in fp.read():
        if char in vowels:
            c = c + 1
    fp.close()

    print("\nTotal Vowels in the file are:")
    print(c)

except:
    print("Error!! No such file exist")
```

```
Total Vowels in the file are:
11
```

In [13]:
```
#sol2:
readFp = open('abc.txt')
writeFp = open('result.txt','w')

c = readFp.read()

for i in range(len(c)):
    if i % 2 == 0:
        writeFp.write(c[i])

print ("Writing done !! \nOpen result.txt to view the content")
writeFp.close()

writeFp = open('result.txt', 'r')
c1 = writeFp.read()
print(c1)

readFp.close()
writeFp.close()
```

```
Writing done !!
Open result.txt to view the content
Hlo ynm sAay gra.hehlohe
```

In [4]:
```
#sol 3:
readFp = open('abc.txt', 'r')
writeFp = open('result.txt', 'w')

c = readFp.readlines()

for i in range(0,len(c)):
    if(i % 2 == 0):
        writeFp.write(c[i])
    else:
        pass

print ("Writing done !! \nOpen result.txt to view the content")
writeFp.close()

#not necessary..can print the copied data here as well.
writeFp = open('result.txt', 'r')
c1 = writeFp.read()
print(c1)

readFp.close()
writeFp.close()
```

```
Writing done !!
Open result.txt to view the content
Hello! My name is Ananya Agarwal.
hello
```

In [17]:
```python
#Sol 4:

readFp = open('abc.txt')
writeFp = open('result.txt','w')

c = readFp.readlines()

for i in range(len(c)):
    if (len(c[i]) > 50):
        writeFp.write(c[i])

print ("Writing done !! \nOpen result.txt to view the content")
writeFp.close()

#not necessary..can print the copied data here as well.
writeFp = open('result.txt', 'r')
c1 = writeFp.read()
print(c1)

readFp.close()
writeFp.close()
```

```
Writing done !!
Open result.txt to view the content
Hello! My name is Ananya Agarwaliiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii.
```

In [29]:
```
#Sol 5:

#import matplotlib.pyplot as plt

readFp = open('abc.txt')
c = readFp.read()

d={}
for i in c:
    if i in d.keys():
        d[i]+=1
    else:
        d[i]=1

print(d)

#plt.hist(d.values())
#plt.bar(d.keys(), d.values(), 1, color='r')
```

```
{'a': 5, 'b': 2, 'c': 2}
```

In [51]:
```
#Sol 6:

import pandas as pd
import math
df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_copy.csv")

# Mean
n = len(df.iloc[:,1]) #1 means 2nd column since indexed by 0
get_sum = sum(df.iloc[:,1])
mean = get_sum / n
print('Mean: ',mean)

# Standard Deviation
var  = sum(pow(x-mean,2) for x in df.iloc[:,1]) / n  # variance
std_var  = math.sqrt(var)  # standard deviation
print('Standard deviation: ',std_var)

#Median
df.iloc[:,1].sort_values()

if n % 2 == 0:
    median1 = df.iloc[:,1][(n-1)//2]
    median2 = df.iloc[:,1][(n+1)//2]
    median = (median1 + median2)/2
else:
    median = df.iloc[:,1][n//2]

print('Median: ',median)
```

```
Mean:  3.0540000000000007
Standard deviation:  0.4321465800705435
Median:  2.95
```

In [3]:
```
#Sol 7:
import math
import pandas as pd
import numpy as np
df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_copy.csv")


# Mean X
x = len(df.iloc[:,0])
get_sum_x = sum(df.iloc[:,0])
get_sum_x_sq1 = get_sum_x * get_sum_x;

# Mean Y
y = len(df.iloc[:,1])
get_sum_y = sum(df.iloc[:,1])
get_sum_y_sq1 = get_sum_y * get_sum_y;

df1 = pd.DataFrame({"a": df.iloc[:,0], "b": df.iloc[:,1]})

x_y = df1["a"] * df1["b"]
x_y_sum = sum(x_y)

x_sq = df1["a"]*df1["a"]

y_sq = df1["b"]*df1["b"]

get_sum_x_sq = sum(x_sq)

get_sum_y_sq = sum(y_sq)

r = ((x*x_y_sum) - (get_sum_x*get_sum_y))/(math.sqrt( ((x * get_sum_x_sq)-get_sum_x_sq1)*((x *get_sum_y_sq)-get_sum_y_sq

print(r)
import scipy.stats
u = scipy.stats.pearsonr(df.iloc[:,0], df.iloc[:,1])[0]
print(u)
```

```
-0.10936924995067286
-0.10936924995064935
```

In [28]:
```
# QUESTION - 1 - iris_01.csv
import pandas as pd

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df)

df1 = df.drop("Unnamed: 0", axis=1)#column 1 just 1,2,3,4....unnamed bcs column header is missing
print(df1)

df1.isnull().sum()#total count of null values got

#mean
mean_sl = df1['sepal length'].mean()
mean_sl = round(mean_sl, 1)#round of by 1 decimal place
df1['sepal length'].fillna(value=mean_sl, inplace=True)

mean_sw = df1['sepal width (cm)'].mean()
mean_sw = round(mean_sw, 1)
df1['sepal width (cm)'].fillna(value=mean_sw, inplace=True)

mean_pl = df1['petal length (cm)'].mean()
mean_pl = round(mean_pl, 1)
df1['petal length (cm)'].fillna(value=mean_pl, inplace=True)

mean_pw = df1['petal width (cm)'].mean()
mean_pw = round(mean_pw, 1)
df1['petal width (cm)'].fillna(value=mean_pw, inplace=True)

print("Mean sepal length: ", mean_sl)
print("Mean sepal width: ", mean_sw)
print("Mean petal length: ", mean_pl)
print("Mean petal width: ", mean_pw)

print(df1)


#median
df2 = df.drop("Unnamed: 0", axis=1)

med_sl = df2['sepal length'].median()
med_sl = round(med_sl, 1)
```

```
df2['sepal length (cm)'].fillna(value=med_sl, inplace=True)

med_sw = df2['sepal width (cm)'].median()
med_sw = round(med_sw, 1)
df2['sepal width (cm)'].fillna(value=med_sw, inplace=True)

med_pl = df2['petal length (cm)'].median()
med_pl = round(med_pl, 1)
df2['petal length (cm)'].fillna(value=med_pl, inplace=True)

med_pw = df2['petal width (cm)'].median()
med_pw = round(med_pw, 1)
df2['petal width (cm)'].fillna(value=med_pw, inplace=True)

print("Median sepal length: ", med_sl)
print("Median sepal width: ", med_sw)
print("Median petal length: ", med_pl)
print("Median petal width: ", med_pw)

print(df2)


#mode
df3 = df.drop("Unnamed: 0", axis=1)

#Similar to iloc, in that both provide integer-based lookups.
#Use iat if you only need to get or set a single value in a DataFrame or Series.
#sirf mode nahi chalta

mode_sl = df3['sepal length'].mode().iat[0]
mode_sl = round(mode_sl, 1)
df3['sepal length (cm)'].fillna(value=mode_sl, inplace=True)

mode_sw = df3['sepal width (cm)'].mode().iat[0]
mode_sw = round(mode_sw, 1)
df3['sepal width (cm)'].fillna(value=mode_sw, inplace=True)

mode_pl = df3['petal length (cm)'].mode().iat[0]
mode_pl = round(mode_pl, 1)
df3['petal length (cm)'].fillna(value=mode_pl, inplace=True)

mode_pw = df3['petal width (cm)'].mode().iat[0]
```

```
mode_pw = round(mode_pw, 1)
df3['petal width (cm)'].fillna(value=mode_pw, inplace=True)

print("Mode sepal length: ", mode_sl)
print("Mode sepal width: ", mode_sw)
print("Mode petal length: ", mode_pl)
print("Mode petal width: ", mode_pw)

print(df3)

#zero value
df4 = df.drop("Unnamed: 0", axis=1)

df4['sepal length'].fillna(value=0, inplace=True)
df4['sepal width (cm)'].fillna(value=0, inplace=True)
df4['petal length (cm)'].fillna(value=0, inplace=True)
df4['petal width (cm)'].fillna(value=0, inplace=True)

print(df4)

#replace with maximum value
df5 = df.drop("Unnamed: 0", axis=1)

max_sl = df5['sepal length'].max()
df5['sepal length (cm)'].fillna(value=max_sl, inplace=True)

max_sw = df5['sepal width (cm)'].max()
df5['sepal width (cm)'].fillna(value=max_sw, inplace=True)

max_pl = df5['petal length (cm)'].max()
df5['petal length (cm)'].fillna(value=max_pl, inplace=True)

max_pw = df5['petal width (cm)'].max()
df5['petal width (cm)'].fillna(value=max_pw, inplace=True)

print("Maximum sepal length: ", max_sl)
print("Maximum sepal width: ", max_sw)
print("Maximum petal length: ", max_pl)
print("Maximum petal width: ", max_pw)

print(df5)
```

```
146           6.3              2.5               5.0               1.9
147           6.5              3.0               5.2               2.0
148           6.2              0.0               5.4               2.3
149           5.9              3.0               5.1               1.8

     Class  Predicted_class
0        0                0
1        0                0
2        0                0
3        0                0
4        0                0
..     ...              ...
145      2                2
146      2                2
147      2                2
148      2                0
149      2                2

[150 rows x 6 columns]
```

In [7]:

```
# QUESTION - 2
import pandas as pd
import numpy as np

df5 = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df5)

iris_y = pd.DataFrame(df5, columns=['Class', 'Predicted_class'])
print(iris_y)

from sklearn.metrics import confusion_matrix
y_true = iris_y['Class']
y_pred = iris_y['Predicted_class']

cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix => ")
print(cm)

TP = np.diag(cm)
print("TRUE POSITIVE => ",TP)

FP = cm.sum(axis=0) - np.diag(cm)
print("FALSE POSITIVE => ",FP)

FN = cm.sum(axis=1) - np.diag(cm)
print("FALSE NEGATIVE => ",FN)

TN = cm.sum() - (FP + FN + TP)
print("TRUE NEGATIVE => ",TN)

TPR = TP/(TP+FN)
print("TRUE POSITIVE RATE => ",TPR)

TNR = TN/(TN+FP)
print("TRUE NEGATIVE RATE => ",TNR)

FPR = FP/(FP+TN)
print("FALSE POSITIVE RATE => ",FPR)

FNR = FN/(FN+TP)
```

```
print("FALSE NEGATIVE RATE => ",FNR)

#accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print("ACCURACY => ",ACC)

#F1 and F beta score by code
precision = TP/TP+FP
recall = TP/TP+FN
print("precision: ",precision)
print("recall: ",recall)

F1 = (2*precision*recall)/(precision+recall)
print("By code F1 score value is: ",F1)

beta = 0.5
F_beta = ((1+(beta*beta))*(precision*recall))/((beta*beta*precision) + recall)
print("By code F beta score value is: ",F_beta)
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4
1             1           4.9               3.0                1.4
2             2           4.7               3.2                1.3
3             3           4.6               3.1                1.5
4             4           5.0               3.6                1.4
..          ...           ...               ...                ...
145         145           0.0               3.0                5.2
146         146           6.3               2.5                5.0
147         147           6.5               3.0                5.2
148         148           6.2               0.0                5.4
149         149           5.9               3.0                5.1

     petal width (cm)  Class  Predicted_class
0                 0.2      0                0
1                 0.2      0                0
2                 0.2      0                0
3                 0.2      0                0
4                 0.2      0                0
..                ...    ...              ...
145               2.3      2                2
```

```
146              1.9      2               2
147              2.0      2               2
148              2.3      2               0
149              1.8      2               2

[150 rows x 7 columns]
     Class  Predicted_class
0        0                0
1        0                0
2        0                0
3        0                0
4        0                0
..     ...              ...
145      2                2
146      2                2
147      2                2
148      2                0
149      2                2

[150 rows x 2 columns]
Confusion Matrix =
[[46  2  2]
 [ 0 45  5]
 [ 2  4 44]]
TRUE POSITIVE =>  [46 45 44]
FALSE POSITIVE =>  [2 6 7]
FALSE NEGATIVE =>  [4 5 6]
TRUE NEGATIVE =>  [98 94 93]
TRUE POSITIVE RATE =>  [0.92 0.9  0.88]
TRUE NEGATIVE RATE =>  [0.98 0.94 0.93]
FALSE POSITIVE RATE =>  [0.02 0.06 0.07]
FALSE NEGATIVE RATE =>  [0.08 0.1  0.12]
ACCURACY =>  [0.96       0.92666667 0.91333333]
inbuilt F1 score:  0.9003839159426148
inbuilt F beta score for beta = 0.5:  0.9007939090258347
precision:  [3. 7. 8.]
recall:  [5. 6. 7.]
By code F1 score value is:  [3.75       6.46153846 7.46666667]
By code F beta score value is:  [3.26086957 6.77419355 7.77777778]
```

In [31]:

```
# QUESTION - 3

import pandas as pd
import numpy as np

df5 = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df5)

#creating model

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(iris_x, iris_true, test_size=0.2, random_state=3)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression(multi_class='ovr') #one-vs-rest use krke model banayae
model.fit(x_train, y_train)
#model created


from sklearn.metrics import confusion_matrix

y_true = y_test
y_pred = model.predict(x_test)

cm2 = confusion_matrix(y_true, y_pred)
print("Confusion Matrix => ")
print(cm2)

print("y testing was: ")
print(y_true)
print("y predcited is: ")
print(y_pred)

TP = np.diag(cm2)
FP = cm2.sum(axis=0) - np.diag(cm2)
FN = cm2.sum(axis=1) - np.diag(cm2)
TN = cm2.sum() - (FP + FN + TP)
print("TRUE POSITIVE => ", TP)
print("FALSE POSTITVE => ", FP)
print("FALSE NEGATIVE => ", FN)
```

```
print("TRUE NEGATIVE => ", TN)

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("TRUE POSITIVE RATE => ", TPR)
print("TRUE NEGATIVE RATE => ", TNR)
print("FALSE POSITIVE RATE => ", FPR)
print("FALSE NEGATIVE RATE => ", FNR)

ACC = (TP+TN)/(TP+FP+FN+TN)
print("ACCURACY => ",ACC)

#F1 and F beta score by code
precision = TP/TP+FP
recall = TP/TP+FN
print("precision: ",precision)
print("recall: ",recall)

F1 = (2*precision*recall)/(precision+recall)
print("By code F1 score value is: ",F1)

beta = 0.5
F_beta = ((1+(beta*beta))*(precision*recall))/((beta*beta*precision) + recall)
print("By code F beta score value is: ",F_beta)


#same way for ques 6
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4
1             1           4.9               3.0                1.4
2             2           4.7               3.2                1.3
3             3           4.6               3.1                1.5
4             4           5.0               3.6                1.4
..          ...           ...               ...                ...
145         145           0.0               3.0                5.2
146         146           6.3               2.5                5.0
147         147           6.5               3.0                5.2
148         148           6.2               0.0                5.4
```

```
149         149           5.9                3.0                 5.1

     petal width (cm)  Class  Predicted_class
0                 0.2      0                0
1                 0.2      0                0
2                 0.2      0                0
3                 0.2      0                0
4                 0.2      0                0
..                ...    ...              ...
145               2.3      2                2
146               1.9      2                2
147               2.0      2                2
148               2.3      2                0
149               1.8      2                2

[150 rows x 7 columns]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-b1a4a79ae885> in <module>
     10
     11 from sklearn.model_selection import train_test_split
---> 12 x_train, x_test, y_train, y_test = train_test_split(iris_x, iris_true, test_size=0.2, random_state=3)
     13
     14 from sklearn.linear_model import LogisticRegression

NameError: name 'iris_x' is not defined
```

In [30]:

```
#Ques 4

new_feature=[]
new_feature_1=[]

import pandas as pd
empty = pd.DataFrame()

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
shape=df.shape
rows=shape[0]

feature=df["sepal length"]
max_val=feature.max()
min_val=feature.min()

feature_1=df["sepal width (cm)"]
max_val_1=feature_1.max()
min_val_1=feature_1.min()

for i in range(rows):
    normal=(feature[i]-min_val)/(max_val-min_val)
    new_feature.append(normal)

empty["feature_1"]=new_feature

for i in range(rows):
    normal=(feature_1[i]-min_val_1)/(max_val_1-min_val_1)
    new_feature_1.append(normal)

empty["feature_2"]=new_feature_1

empty.to_csv("hi.csv") #will be created at desktop
```

In [19]:
```
# QUESTION - 5

import pandas as pd
import numpy as np

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df)

X=df[['sepal length','sepal width (cm)','petal length (cm)','petal width (cm)']]
Y=df[['Class']]

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
print(X_scaled)
Transposed=X_scaled.T
Covariance_matrix=np.cov(Transposed)
print(Covariance_matrix)
value,vector=np.linalg.eig(Covariance_matrix)
print(value)
print(vector)
percentage_values=[]
for i in range(len(value)):
    percentage_values.append(value[i]/np.sum(value))
print(percentage_values)
projected_1=X_scaled.dot(vector.T[0])
projected_2=X_scaled.dot(vector.T[1])
res=pd.DataFrame(projected_1,columns=['PC1'])
res['PC2']=projected_2
res["Species"]=Y
print(res)
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4
1             1           4.9               3.0                1.4
2             2           4.7               3.2                1.3
3             3           4.6               3.1                1.5
4             4           5.0               3.6                1.4
..          ...           ...               ...                ...
145         145           0.0               3.0                5.2
146         146           6.3               2.5                5.0
```

```
147         147          6.5               3.0               5.2
148         148          6.2               0.0               5.4
149         149          5.9               3.0               5.1

     petal width (cm)  Class  Predicted_class
0                 0.2      0                0
1                 0.2      0                0
2                 0.2      0                0
3                 0.2      0                0
4                 0.2      0                0
```

In [ ]: