1.Demonstrate programs using basic constructs of Python.

1. AIM : Enter two integers and perform all arithmetic operatons.

PROGRAM :

```
#ARITHMATIC OPERATIONS
a= int(input("Enter the first number: "))
b= int(input("Enter the second number:"))
print("Sum is: ",a+b)
print("Difference is: ", a-b)
print("Product is: ", a*b)
print("Division is: ", a/b)
print("Remainder is: ", a%b)
print("Exponent is: ",a**b)
```

OUTPUT:

```
Enter the first number: 15
Enter the second number: 10
Sum is: 25
Difference is: 5
Product is: 150
Division is: 1.5
Remainder is: 5
Exponent is: 576650390625
```

2.AIM: Program to find area of the triangle

PROGRAM :

```
base = int(input("Enter the base: "))
height= int(input("Enter the height: "))
area = (base*height)/2
print("Area of the triangle: ", area)
```

OUTPUT :

```
Enter the base: 2
Enter the height: 3
Area of the triangle: 3.0
```

3.AIM: Program to calculate area and circumference of a circle

PROGRAM :

```
radius = int(input("Enter the radius: "))
area= 3.14*radius*radius
circumference = 2*3.14*radius
print("Area : ", area)
print("Circumference : ", circumference)
```

OUTPUT :

```
Enter the radius: 2
Area : 12.56
Circumference : 12.56
```

4.AIM ; Develop a program to calculate simple and compound interest

Program :

```
p = int(input("Enter principle amount: "))
t= int(input("Enter the time in years: "))
r= int(input("Enter the rate: "))
si = (p*t*r)/100
a = p*(1+r/100)**t
ci= p-a
print("Simple interest is : ", si)
print('Compund interest is : ", ci)
```

OUTPUT :

```
Enter the principle amount: 1000
Enter the time in years: 1
Enter the rate: 1
Simple interest is : 10
Compound interest is : 10
```

5.AIM : Write a program to check the greatest of the three numbers

PROGRAM ;

```
a = int(input("Enter the first number :"))
b = int(input("Enter the first number : "))
c = int(input("Enter the third number : "))
max = a
if(b > max):
      max = b
elif(c > max)
      max = c
print("Maximum of three numbers is : ", max)
```

OUTPUT :

```
Enter the first number : 3
Enter the first number : 10
Enter the third number : 12
Maximum of three numbers is : 12
```

6.AIM : Check if the given number is positive, negative or zero(0).

PROGRAM :

```
n = int(input("Enter n value : "))
if(n<0):
    print("Negative")
elif(n>0):
    print("Positive")
else:
    print("Equal to Zero")
```

OUTPUT :

```
Enter n value : 10
Positive
```

7.AIM : Check if the number is prime or not

PROGRAM :

```
a = int(input("Enter a value: "))
if(a<2):
     print("Not Prime")
else:
     flag = 0
     i = 2
     while(i<a/2):
          if(a%i == 0):
               flag = 1
               break
     if(flag == 1);
          print("Not Prime")
     else:
          print("Prime")
```

OUTPUT :

```
Enter a value : 3
Prime
```

8.AIM : Write a program to print n natural numbers

PROGRAM :

```
n = int(input("Enter n value : "))
i = 1
while(i<=n):
      print(i)
      i=i+1
```

OUTPUT :

```
Enter n value : 4
1
2
3
4
```

9.AIM : Write a program to convert binary to decimal

PROGRAM :

```
n = int(input("Enter a binary number : "))
i = 0
sum = 0
while(n>0):
     r = n%10
     sum = sum + r * pow(2,1)
     i=i+1
     n=n/10
print(sum)
```

OUTPUT :

```
Enter a binary number : 101
5
```

10.AIM : Write a program to find multiplication table of the given number

PROGRAM :

```
n = int(input("Enter n value : "))
i=1
while(i<=10):
     print(n,"*",i,"=",n*i)
     i=i+1
```

OUTPUT :

```
Enter n value: 5
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
```

11.AIM : Write a program to find the factorial of the given number

PROGRAM :

```
n = int(input("Enter n value : "))
fact = 1
while(n>0):
     fact = fact * n
     n= n-1
print("Factorial of the given number : ",fact)
```

OUTPUT :

```
Enter n value : 5
Factorial of the given number : 120
```

2.Build modular programs using functions.

12.AIM : Write a function is_prime() that returns a 1 if the argument passed to it is a prime number and 0 otherwise

PROGRAM :

```
def is_prime(n):
if(n==1):
    return 0
for i in range(2, n//2):
     if(n%i == 0):
          return 0
     return 1
x = int(input("Enter number : "))
if(is_prime(x) == 0):
     print("Not Prime Number")
else:
     print("Prime Number")
```

OUTPUT :

```
Enter number : 4
Not Prime Number
```

13.AIM : Write a menu driven program to add , subtract, multiply and divide two integers using functions

PROGRAM :

```
def add(a,b) :
   print("Addition : ", a+b)
def sub(a,b) :
   print("Subtraction : ", a-b)
def mul(a,b) :
   print("Multiply : ", a*b)
def div(a,b) :
   print("Division : ", a//b)
a=int(input("Enter a : "))
b=int(input("Enter b : "))
n=int(input("Enter your choice : "))
while(n!=-1) :
   if(n==1) :
      add(a,b)
   elif(n==2) :
      sub(a,b)
   elif(n==3) :
      mul(a,b)
    elif(n==4) :
      div(a,b)
 n=int(input("Enter your choice : "))
```

OUTPUT :

```
Enter a : 5
Enter b : 10
Enter your choice : 1
Addition :  15
Enter your choice : 2
Subtraction :  -5
Enter your choice : 3
Multiply :  50
Enter your choice : 4
Division :  0
Enter your choice : -1
```

14.AIM : Write a program that passes lambda function as an argument to another function to compute the cube of a number.

PROGRAM :

```
c = lambda x : x**3
a=int(input("Enter a : "))
print("Cube of number : ", c(a))
```

OUTPUT :

```
Enter a : 3
Cube of number :  27
```

15.AIM : Program to calculate sum of series 1/1! + 4/2! + 27/3! + …..

PROGRAM :

```
def fact(n):
  f=1
  for i in range(1,n+1) :
    f=f*i
  return f
def power(n) :
  p=1
  for i in range(1,n+1) :
    p=p*n;
  return p
n=int(input("Enter number : "))
sum=0;
for i in range(1,n+1) :
   sum=sum+(power(i)/fact(i))
print("Sum of series : ", sum)
```

OUTPUT :

```
Enter number : 3
Sum of series :  7.5
```

16.AIM : Write a function that converts temperature given in Celcius to Fahrenheit

PROGRAM :

```
def celsius_fahrenheit(c) :
   return c*(9/5) + 32
c=int(input("Enter temperature in Celsius : "))
print("Temperature in Fahrenheit : ", celsius_fahrenheit(c))
```

OUTPUT :

```
Enter temperature in Celsius : 98
Temperature in Fahrenheit :  208.4
```

17.AIM : Write a function that accepts three integers and returns true if they are sorted else it returns false.

PROGRAM :

```
def sortedOrNot(a,b,c) :
   if(a<b and b<c) :
      return "true"
   else :
      return "false"
a=int(input("Enter a : "))
b=int(input("Enter b : "))
c=int(input("Enter c : "))
if(sortedOrNot(a,b,c)=="true") :
  print(a,b,c,"are sorted")
```

```
else:
  print(a,b,c,"are not sorted")
```

OUTPUT :

```
Enter a : 1
Enter b : 6
Enter c : 3
1 6 3 are not sorted
```

3.Implement Python Programs using lists, tuples and dictionaries

18.AIM : Write a program that accepts a list from user and print the alternate element of the list

PROGRAM :

```
l1=[]
n=int(input("Enter no of elements : "))
print("Enter elements : ")
for i in range(0,n) :
   x=int(input())
   l1.append(x)
print("Alternate elements are : ")
for j in range(0,n) :
   if(j%2!=0) :
      print(l1[j])
```

OUTPUT :

```
Enter no of elements : 5
Enter elements :
1
2
3
4
5
Alternate elements are :
2
4
```

19.AIM : Find and display the largest number of a list without builtin function max()

PROGRAM :

```
l1 = []
n=int(input("Enter no of elements : "))
print("Enter elements : ")
for i in range(0,n) :
   x=int(input())
   l1.append(x)
max=l1[0]
for i in range(1,n) :
   if(l1[i]>max) :
    max=l1[i]
print("Greatest Element of list is : ", max)
```

OUTPUT :

```
Enter no of elements : 5
Enter elements :
1
2
3
4
5
Greatest Element of list is :  5
```

20.AIM : Write a program that rotates the element of a list so that the element at the first index moves to the second index, the element in the second index moves to the third index, etc., and the element in the last index moves to the first index.

PROGRAM :

```
l1 = []
n=int(input("Enter no of elements : "))
print("Enter elements : ")
```

```
for i in range(0,n) :
    x=int(input())
    l1.append(x)
i=n-1
x=l1[n-1]
while(i>0) :
    l1[i]=l1[i-1]
    i=i-1
l1[0]=x
print("Elements after rearranging : ")
print(l1)
```

OUTPUT :

```
Enter no of elements : 5
Enter elements :
3
7
6
11
9
Elements after rearranging :
[9, 3, 7, 6, 11]
```

21.AIM : Write a program to search an element in a given list

PROGRAM :

```
l1 = []
n=int(input("Enter no of elements : "))
print("Enter elements : ")
for i in range(0,n) :
    x=int(input())
    l1.append(x)
flag=0;
k=int(input("Enter the element to search : "))
for i in range(0,n) :
    if(l1[i]==k) :
     flag=1
     break;
if(flag==1) :
 print("Element found in list")
else :
 print("Element not found in list")
```

OUPUT :

```
Enter no of elements : 5
Enter elements :
1
2
3
4
5
Enter the element to search : 10
Element not found in list
```

22.AIM : Write a program to remove duplicates from a list

PROGRAM :

```
list = [1, 6, 8, 6, 10]
i = 0
while i<n:
      j = 0
      while j < n:
```

```
        if i != j and list[i] == list[j]:
                list.pop()
                n = n-1
        j =  j+1
        I =  i+1
    print(list)
```

OUTPUT :

[1, 6, 8, 10]

23.AIM : Write a program to add two matrices

PROGRAM :

```
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
res = [[0, 0], [0, 0]]
for i in range(len(A)):
       for j in range(len(B[0])):
            res[i][j] = A[i][j] + B[i][j]
for r in res:
     print(r)
```

OUTPUT :

[6, 8]
[10, 12]

24.AIM : Write a program to multiply two matrices

PROGRAM :

```
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
res = [[0, 0], [0, 0]]
for i in range(len(A)):
       for j in range(len(B[0])):
            for k in range(len(B)):
                 res[i][j] += A[i][j] + B[i][j]
for r in res:
     print(r)
```

OUTPUT :

[19, 22]
[43, 50]

25.AIM : Write a program to transpose the matrix

PROGRAM :

```
A = [[1, 2], [3, 4], [5, 6]]
res = [[0, 0, 0], [0, 0, 0]]
for i in range (len(A)):
      for i in range (len(A[0])):
           res[j][i] = A[i][j]
for r in res :
      print(r)
```

OUTPUT :

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

26.AIM : Write a program to convert two lists into a dictionary

PROGRAM :

```
list1 = ['Bandaru', 'Kanaka', 'Aparna']
list2 = [1, 2, 3]
print("list1 : ", list1)
```

```
print("list2 :", list2)
res = { }
for key in list1:
    for value in list2:
        res[key] = value
        list2.remove(value)
        break
print("Dictionary converted from two lists : \n", res)
```

OUTPUT :

```
list1 = ['Bandaru', 'Kanaka', 'Aparna']
list2 = [1, 2, 3]
Dictionary converted from two lists :
['Bandaru' : 1, 'Kanaka' : 2, 'Aparna' : 3]
```

27.AIM : Write a program with class name rectangle constructed by a length and width and a method which will compute the area of a rectangle

PROGRAM :

```
class Rectangle :
    def __init__(self,length,width) :
        self.length = length
        self.width = width
    def area(self) :
        return self.length*self.width
l = int(input("Enter length of rectangle : "))
b = int(input("Enter width of rectangle : "))
a=Rectangle(l,b)
print(a.area())
```

OUTPUT :

```
Enter length of rectangle : 10
Enter width of rectangle : 12
120
```

28.AIM : Write a program to find the three elements that sum to zero from a set of n real numbers

PROGRAM :

```
class SumTOZero :
    def sum(a,b,c) :
        if(a+b+c == 0) :
            return 1
        else :
            return 0
l = [1,-3,2,4,-5,3,-6,8,0,9]
n=len(l)
for i in range(0,n-2) :
    for k in range(i+1, n-1) :
        for j in range(k+1, n) :
            x = SumTOZero.sum(l[i],l[k],l[j])
            if(x==1) :
                print(l[i],l[k],l[j])
```

OUTPUT :

```
1 -3 2
1 4 -5
-3 -5 8
-3 3 0
-3 -6 9
2 4 -6
2 -5 3
```

29.AIM : Write a program tp convert an integer into a roman numeral and write a python class to convert an integer to a roman numeral.

PROGRAM :

```
#integer to roman
class roman:
    def int_to_Roman(self, num):
        val = [
                1000, 900, 500, 400,
                100, 90, 50, 40,
                10, 9, 5, 4,
                1
              ]
        sym = [
```

```
                "M", "CM", "D", "CD",
                "C", "XC", "L", "XL",
                "X", "IX", "V", "IV",
                 "I"
              ]
        roman_num = ' '
        i = 0
        while num > 0:
            for _ in range(nu,//val[i]):
                roman_num += syb[i]
                num -= val[i]
            i += 1
        return roman_num
print(roman().int_to_Roman(4000))
```

OUTPUT :

```
MMMM
```

```
#roman to numeral
class solution:
    def roman_to_int(self, s):
        rom_val = {'I':1, 'V': 5, 'X': 10, 'L':50, 'C':100, 'D':500, 'M':1000}
        int_val = 0
        for i in range(len(s)):
            if i > 0 and rom_val[s[i]] > rom_val[s[i - 1]]:
                int_val += rom_val[s[i]] - 2 * rom_val[s[i - 1]]
            else:
                int_val += rom_val[s[i]]
        return int_val
print(solution().roman_to_int('MMMM')
```

OUTPUT :

```
4000
```

30.AIM : Write a program with class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

PROGRAM :

```
class Circle :
    def __init__(self,r):
        self.r=r
    def perimeter(self) :
        return 2*3.14*r
    def area(self) :
        return 3.14*r*r
r = int(input("Enter radius of Cirlcle : "))
o = Circle(r)
print("Perimeter of Circle : ", o.perimeter())
print("area of Circle : ", o.area())
```

OUTPUT :

```
Enter radius of Cirlcle : 5
Perimeter of Circle :  31.400000000000002
area of Circle :  78.5
```

5.Develop programs to perform searching and sorting

31.AIM : Write a program to sort the elements using Bubble Sort

PROGRAM :

```
def BubbleSort(l1,n) :
    for i in range(0,n-1) :
        for j in range(0, n-i-1) :
            if(l1[j]>l1[j+1]) :
                t = l1[j]
                l1[j] = l1[j+1]
                l1[j+1] = t
    print(l1)
l1 = [8,5,7,3,2]
n = len(l1)
BubbleSort(l1,n)
```

OUTPUT :

```
[2, 3, 5, 7, 8]
```

32.AIM : Write a program to sort the elements using Insertion Sort

PROGRAM :

```
def InsertionSort(l1,n) :
    for i in range(1,n) :
        j=i-1
        t=l1[i];
        while(j>=0 and t<=l1[j]):
            l1[j+1]=l1[j]
            j=j-1
        l1[j+1]=t
    print(l1)
l1 = [8,5,7,3,2]
n = len(l1)
InsertionSort(l1,n)
```

OUTPUT :

```
[2, 3, 5, 7, 8]
```

33.AIM : Write a program to sort the elements using Selection Sort

PROGRAM :

```
def SelectionSort(l1,n) :
    for i in range(0,n-1) :
        index=i
        for j in range(i+1, n) :
            if(l1[j]<l1[index]) :
                index = j
        t = l1[index]
        l1[index] = l1[i]
        l1[i] = t
    print(l1)
l1 = [8,5,7,3,2]
n = len(l1)
SelectionSort(l1,n)
```

OUTPUT :

```
[2, 3, 5, 7, 8]
```

34.AIM : Write a program to sort the elements using Merge Sort

PROGRAM :

```
def Sort(l1,low,mid,high) :
    h=low
    j=mid+1
    l2=[]
```

```
    while(h<=mid and j<=high) :
        if(l1[h]<=l1[j]) :
            l2.append(l1[h])
            h=h+1
        else :
            l2.append(l1[j])
            j=j+1
    if(h>mid) :
        for k in range(j,high+1) :
            l2.append(l1[k])
    else :
        for k in range(h,mid+1) :
            l2.append(l1[k])
    i=0
    for k in range(low,high+1) :
        l1[k]=l2[i]
        i=i+1
def Merge(l1,low,high) :
    if(low<high) :
        mid=(low+high)//2
        Merge(l1,low,mid)
        Merge(l1,mid+1,high)
        Sort(l1,low,mid,high)
l1 = [8,5,7,3,2]
n = len(l1)
Merge(l1,0,n-1)
print(l1)
```

OUTPUT :

[2, 3, 5, 7, 8]

35.AIM : Write a program to sort the elements using Quick Sort

PROGRAM :

```
def QuickSort(l1,lb,rb) :
    if(lb<rb) :
        i=lb
        j=rb+1
        while(1) :
            i=i+1
            while(i<=rb and l1[i]<l1[lb]) :
                i=i+1
            j=j-1
            while(j>=lb and l1[j]>l1[lb]) :
                j=j-1
            if(i>j) :
                break
            else :
                t=l1[i]
                l1[i]=l1[j]
                l1[j]=t
        t=l1[lb]
        l1[lb]=l1[j]
        l1[j]=t
        QuickSort(l1,lb,j-1)
        QuickSort(l1,j+1,rb)
l1 = [8,5,7,3,2]
n = len(l1)
QuickSort(l1,0,n-1)
print(l1)
```

OUTPUT :

[2, 3, 5, 7, 8]

3

36.AIM : Write a program to implement Linear Search

PROGRAM :

```
def LinearSearch(l1,n,key) :
    for i in range(0,n) :
        if(l1[i]==key) :
            return i
    return -1
l1 = []
n=int(input("Enter no of elements : "))
print("Enter elements : ")
for i in range(0,n) :
    x=int(input())
    l1.append(x)
key = int(input("Enter element to search : "))
flag = LinearSearch(l1,n,key)
if(flag==-1) :
    print("Element not found")
else :
    print("Element fount at index ", flag)
```

OUTPUT :

```
Enter no of elements : 5
Enter elements :
10
25
36
48
69
Enter element to search : 15
Element not found
```

37.AIM : Write a program to implement Binary Search

PROGRAM :

```
def BinarySearch(l1,low,high,key) :
    while(low<high) :
        mid=(low+high)//2
        if(l1[mid]==key) :
            return mid
        elif(l1[mid]<key) :
            low=mid+1
        else :
            high=mid-1
    return -1
l1 = []
n=int(input("Enter no of elements : "))
print("Enter elements : ")
for i in range(0,n) :
    x=int(input())
    l1.append(x)
key = int(input("Enter element to search : "))
flag = BinarySearch(l1,0,n,key)
if(flag==-1) :
    print("Element not found")
else :
    print("Element fount at index ", flag)
```

OUTPUT :

```
Enter no of elements : 5
```

```
Enter elements :
10
25
36
48
15
Enter element to search : 36
Element fount at index  2
```

6.Implement Python Programs to perform operations on stack

38.AIM : Write a program to implement stacks

PROGRAM :

```
l1= []
n=int(input("Enter no of elements : "))
top=-1
choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
while(choice!=-1) :
   if(choice==1) :
      x=int(input("Enter the element to insert : "))
      if(top+1==n) :
         print("Overflow")
      else :
         top=top+1
         l1.insert(top,x)
   if(choice==2) :
      if(top==-1) :
         print("Underflow")
      else :
         print(top)
         l1.pop(top)
         top=top-1
   if(choice==3) :
      print(l1)
   choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
```

OUTPUT :

```
Enter no of elements : 5
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 10
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 25
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 29
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 2
2
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 3
[10, 25]
Enter the choice (1=push or 2=pop or 3=display or -1=exit): -1
```

39.AIM : Write a program to implement Towers of Hanoi

PROGRAM :

```
def hanoi(n,a,b,c) :
   if(n==1) :
      print("Disk ",n," moves from ",a," to ",c)
   else :
      hanoi(n-1,a,c,b)
      print("Disk ",n," moves from ",a," to ",c)
      hanoi(n-1,b,a,c)
n=int(input("Enter no of disks : "))
hanoi(n,'A','B','C')
```

OUTPUT :

```
Enter no of disks : 4
Disk  1  moves from  A  to  B
Disk  2  moves from  A  to  C
Disk  1  moves from  B  to  C
Disk  3  moves from  A  to  B
```

```
Disk 1 moves from C to A
Disk 2 moves from C to B
Disk 1 moves from A to B
Disk 4 moves from A to C
Disk 1 moves from B to C
Disk 2 moves from B to A
Disk 1 moves from C to A
Disk 3 moves from B to C
Disk 1 moves from A to B
Disk 2 moves from A to C
Disk 1 moves from B to C
```

7.Implement Python Programs to perform operations on queue

40.AIM : Write a program to implement Queue Operations

PROGRAM :

```
def enqueue(x) :
    global r,f,n,q
    if(r==n-1) :
        print("\nOverflow")
    else :
        r=r+1;
        q[r]=x
        if(f==-1) :
            f=f+1
def dequeue() :
    global q,r,f
    if(f==-1) :
        print("\nUnderflow")
        return -1
    else :
        y = q[f]
        if(r==f) :
            r=f=-1
        else :
            f=f+1
        return y
def display() :
    global q,r,f
    if(f==-1) :
        print("\nStack is empty")
    else :
        for i in range(f,r+1) :
            print(q[i])
n=int(input("Enter no of elements : "))
r=f=-1
q=[]
for i in range(0,n) :
    q.append(0)
choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
while(choice!=-1) :
    if(choice==1) :
        x=int(input("Enter the element to insert : "))
        enqueue(x)
    if(choice==2) :
        res = dequeue()
        if(res!=-1) :
            print("Deleted element int stack is ", res);
    if(choice==3) :
        display()
    choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
```

OUTPUT :

```
Enter no of elements : 5
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 3
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 6
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 4
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 2
```

```
Deleted element int stack is  3
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 3
6
4
Enter the choice (1=push or 2=pop or 3=display or -1=exit): -1
```

41.AIM : Write a program to implement Circular Queue Operations

PROGRAM :

```
def Cenqueue(x) :
    global r,f,n,cq
    if((r==n-1 and f==0) or (r+1 == f)) :
        print("\nOverflow");
    else :
        if(r==n-1) :
            r=0
        else :
            r=r+1
        cq[r]=x
        if(f==-1) :
            f=0
def Cdequeue() :
    global cq,r,f,n
    if(f==-1) :
        print("\nUnderflow")
        return -1
    else :
        y = cq[f]
        if(r==f) :
            r=f=-1
        else :
            if(f==n-1) :
                f=0
            else :
                f=f+1
    return y
def Cdisplay() :
    global cq,r,f
    if(f==-1) :
        print("\nUnderflow");
    else :
        if(r<f) :
            for i in range(f,n) :
                print(cq[i])
            for i in range(0,r+1) :
                print(cq[i])
        else :
            for i in range(f,r+1) :
                print(cq[i])
n=int(input("Enter no of elements : "))
r=f=-1
for i in range(0,n) :
    cq.append(0)
choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
while(choice!=-1) :
    if(choice==1) :
        x=int(input("Enter the element to insert : "))
        Cenqueue(x)
    if(choice==2) :
        res = Cdequeue()
        if(res!=-1) :
```

```
        print("Deleted element int stack is ", res);
  if(choice==3) :
     Cdisplay()
  choice=int(input("Enter the choice (1=push or 2=pop or 3=display or -1=exit): "))
```

OUTPUT :

```
Enter no of elements : 5
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 10
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 12
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 1
Enter the element to insert : 32
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 2
Deleted element int stack is  10
Enter the choice (1=push or 2=pop or 3=display or -1=exit): 3
12
32
Enter the choice (1=push or 2=pop or 3=display or -1=exit): -1
```

8.Implement Python Programs to perform operations on linked list

42.AIM : Write a program to implement Single Linked List

PROGRAM :

```
class node() :
    def __init__(self,data,nextnode) :
        self.data = data
        self.nextnode = nextnode
class SLL() :
    def __init__(self,head) :
        self.head=head
    def insertion_front(self,x) :
        p=node(x,self.head)
        self.head=p
    def insertion_end(self,x) :
        p=node(x,None)
        r=self.head
        while(r.nextnode!=None) :
            r=r.nextnode
        r.nextnode=p
    def insertion_middle(self,x,pos) :
        if(pos==1) :
            self.insertion_front(x)
        elif(pos==self.count()+1) :
            self.insertion_end(x)
        else :
            t=r=self.head
            for i in range(1,pos) :
                t = r
                r = r.nextnode
            p=node(x,r)
            t.nextnode = p
    def count(self) :
        r=self.head
        c=0
        while(r!=None) :
            c=c+1
            r=r.nextnode
        return c
    def deletion_front(self) :
        print("Deleted element is : ", self.head.data)
        self.head=self.head.nextnode
    def deletion_end(self) :
        r=t=self.head
        while(r.nextnode!=None) :
            t=r
            r=r.nextnode
        t.nextnode=None
        print("Deleted element is : ", r.data)
    def deletion(self,pos) :
        if(self.head==None) :
            print("Underflow\n")
        else :
            if(pos==1) :
                self.deletion_front()
            elif(pos==self.count()) :
                self.deletion_end()
            else :
```

```
            t=r=self.head
            for i in range(1,pos) :
               t = r
               r = r.nextnode
            t.nextnode = r.nextnode
            print("Deleted element is : ", r.data)
   def display(self) :
      r=self.head
      while(r!=None) :
         print(r.data)
         r=r.nextnode
s=SLL(None)
op = int(input("1.insertion_front\n2.insertion_end\n3.insertion_middle\n4.deletion\n5.display\n-
1.exit\nEnter your choice : "))
while(op!=-1) :
   if(op==1) :
      x=int(input("Enter the element to insert : "))
      s.insertion_front(x)
   if(op==2) :
      x=int(input("Enter the element to insert : "))
      s.insertion_end(x)
   if(op==3) :
      x=int(input("Enter the element to insert : "))
      pos=int(input("Enter the position of element to insert : "))
      s.insertion_middle(x,pos)
   if(op==4) :
      pos=int(input("Enter the position of element to delete : "))
      s.deletion(pos)
   if(op==5) :
       s.display()
   op = int(input("Enter your choice : "))
```

OUTPUT :

```
1.insertion_front
2.insertion_end
3.insertion_middle
4.deletion
5.display
-1.exit
Enter your choice : 1
Enter the element to insert : 10
Enter your choice : 1
Enter the element to insert : 20
Enter your choice : 2
Enter the element to insert : 50
Enter your choice : 3
Enter the element to insert : 80
Enter the position of element to insert : 4
Enter your choice : 5
20
10
50
80
Enter your choice : -1
```

43.AIM : Write a program using double linked list

PROGRAM :

```
class node() :
   def __init__(self,data,prevnode,nextnode) :
      self.data = data
```

```
        self.prevnode = prevnode
        self.nextnode = nextnode
class DLL() :
    def __init__(self,head,tail) :
        self.head=head
        self.tail=tail
    def insertion_front(self,x) :
        p=node(x,None,self.head)
        if(self.head==None) :
            self.tail=self.head=p
        else :
            self.head.prevnode=p
            self.head=p
    def insertion_end(self,x) :
        p=node(x,self.tail,None)
        if(self.head==None) :
            self.tail=self.head=p
        else :
            self.tail.nextnode=p
            self.tail=p
    def insertion_middle(self,x,pos) :
        if(pos==1) :
            self.insertion_front(x)
        elif(pos==self.count()+1) :
            self.insertion_end(x)
        else :
            t=r=self.head
            for i in range(1,pos) :
                t = r
                r = r.nextnode
            p=node(x,t,r)
            t.nextnode = p
            r.prevnode = p
    def count(self) :
        r=self.head
        c=0
        while(r!=None) :
            c=c+1
            r=r.nextnode
        return c
    def deletion_front(self) :
        print("Deleted element is : ", self.head.data)
        self.head=self.head.nextnode
        self.head.prevnode=None
    def deletion_end(self) :
        print("Deleted element is : ", self.tail.data)
        self.tail=self.tail.prevnode
        self.tail.nextnode=None
    def deletion(self,pos) :
        if(self.head==None) :
            print("Underflow\n")
        elif(self.head==self.tail) :
            print("Deleted element is : ", self.head.data)
            self.head=self.tail=None
        else :
            if(pos==1) :
                self.deletion_front()
            elif(pos==self.count()) :
                self.deletion_end()
            else :
```

```
            t=r=self.head
            for i in range(1,pos) :
               t = r
               r = r.nextnode
            t.nextnode = r.nextnode
            r.nextnode.prevnode = t
            print("Deleted element is : ", r.data)
   def display(self) :
      r=self.head
      while(r!=None) :
         print(r.data)
         r=r.nextnode
s=DLL(None,None)
op = int(input("1.insertion_front\n2.insertion_end\n3.insertion_middle\n4.deletion\n5.display\n-1.exit\nEnter your choice
: "))
while(op!=-1) :
   if(op==1) :
      x=int(input("Enter the element to insert : "))
      s.insertion_front(x)
   if(op==2) :
      x=int(input("Enter the element to insert : "))
      s.insertion_end(x)
   if(op==3) :
      x=int(input("Enter the element to insert : "))
      pos=int(input("Enter the position of element to insert : "))
      s.insertion_middle(x,pos)
   if(op==4) :
      pos=int(input("Enter the position of element to delete : "))
      s.deletion(pos)
   if(op==5) :
       s.display()
   op = int(input("Enter your choice : "))
```

OUTPUT :

```
1.insertion_front
2.insertion_end
3.insertion_middle
4.deletion
5.display
-1.exit
Enter your choice : 1
Enter the element to insert : 10
Enter your choice : 1
Enter the element to insert : 20
Enter your choice : 2
Enter the element to insert : 30
Enter your choice : 3
Enter the element to insert : 40
Enter the position of element to insert : 2
Enter your choice : 5
20
40
10
30
Enter your choice : 4
Enter the position of element to delete : 3
Deleted element is :  10
Enter your choice : 5
20
40
30
```

Enter your choice : -1

44. AIM : Write a program using Circular linked list

PROGRAM:

```
class Node():
   def __init__(self,data):
      self.data=data
      self.next=None
class CLL:
   def __init__(self):
      self.head=None
   def insertion_front(self,data):
      in_fr=Node(data)
      if self.head is None:
         self.head=in_fr
         self.head.next=self.head
      else:
         temp=self.head
         r=self.head
         in_fr.next=temp
         while r.next is not self.head:
            r=r.next
         r.next=in_fr
         self.head=in_fr
   def insertion_end(self,data):
      in_ed=Node(data)
      if self.head is None:
         self.head=in_fr
         temp.next=self.head
      else:
         temp=self.head
         r=self.head
         while r.next is not self.head:
            r=r.next
         r.next=in_ed
         in_ed.next=temp
   def insertion_middle(self,pos,data):
      in_md=Node(data)
      temp=self.head
      for i in range(1,pos-1):
         temp=temp.next
      in_md.next=temp.next
      temp.next=in_md
   def deletion_front(self):
      if self.head is None:
         print("Linked is Empty")
      elif self.head.next is self.head:
         self.next=None
      else:
         temp=self.head
         r=self.head.next
         while r.next is not self.head:
```

```
            r=r.next
        r.next=self.head.next
        self.head=temp.next
        temp.next=None
    def deletion_end(self):
        if self.head is None:
            print("Linked list is Empty")
        elif self.head.next is self.head:
            self.next=None
        else:
            temp=self.head
            r=self.head.next
            while r.next is not self.head:
                r=r.next
                temp=temp.next
            temp.next=self.head
            r.next=None
    def deletion_middle(self,pos):
        if self.head is None:
            print("Linked list is Empty")
        else:
            temp=self.head.next
            r=self.head
            for i in range(1,pos-1):
                temp=temp.next
                r=r.next
            r.next=temp.next
            temp.next=None
    def display(self):
        temp=self.head
        if self.head is None:
            print("Linked List is Empty")
        else:
            while temp:
                print(temp.data,'-->',end=" ")
                if temp.next==self.head:
                    break
                temp=temp.next
l=CLL()
while True:
    print("\n----MENU----\n1.Insertion at beginning\n2.Insertion at end\n3.Insertion at middle\n4.Deletion at
front\n5.Deletion at end\n6.Deletion at middle\n7.Display\n8.Exit")
    i=int(input("Enter your choice : "))
    if i==1:
        l.insertion_front(int(input("Enter element : ")))
    elif i==2:
        l.insertion_end(int(input("Enter element : ")))
    elif i==3:
        l.insertion_middle(int(input("Enter position : ")),int(input("Enter element : ")))
    elif i==4:
        l.deletion_front()
    elif i==5:
```

```
        l.deletion_end()
    elif i==6:
        l.deletion_middle(int(input("Enter position : ")))
    elif i==7:
        l.display()
    elif i==8:
        break
    else:
        print("INVALID ENTRY")
```

OUTPUT:

```
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 1
Enter element : 10
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 1
Enter element : 20
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 1
Enter element : 30
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
```

```
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 2
Enter element : 40
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 3
Enter position : 2
Enter element : 50
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 7
30 --> 50 --> 20 --> 10 --> 40 -->
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 4
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
```

```
8.Exit
Enter your choice : 6
Enter position : 3
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 7
50 --> 20 -->
CLL OPERATIONS:
1.Insertion at beginning
2.Insertion at end
3.Insertion at middle
4.Deletion at front
5.Deletion at end
6.Deletion at middle
7.Display
8.Exit
Enter your choice : 8
```

9.Implement Python Programs to perform operations on tree

45.AIM : Implement Python Programs to perform operations on tree

PROGRAM :

```
class node() :
    def __init__(self,data,prevnode,nextnode) :
        self.data = data
        self.prevnode = prevnode
        self.nextnode = nextnode
class root() :
     def __init__(self,t) :
        self.t=t
class BST() :
    def insertion(self,t,x) :
        if(t==None) :
            p=node(x,None,None)
            return p
        elif(t.data > x):
            t.prevnode=self.insertion(t.prevnode,x)
        elif(t.data < x) :
            t.nextnode=self.insertion(t.nextnode,x)
        return t
    def findmax(self,t) :
        if(t==None) :
            return None
        else :
            if(t.nextnode==None) :
                return t
            else :
                return self.findmax(t.nextnode)
    def deletion(self,t,x) :
        if(t==None) :
            return None
        else :
            if(t.data > x) :
                t.prevnode=self.deletion(t.prevnode,x)
            elif(t.data < x) :
                t.nextnode=self.deletion(t.nextnode,x)
            else :
                if(t.prevnode and t.nextnode) :
                    temp=self.findmax(t.prevnode)
                    t.data=temp.data
                    t.prevnode=self.deletion(t.prevnode,t.data)
                elif(t.prevnode == None) :
                    t=t.nextnode
                else :
                    t=t.prevnode
            return t
    def inorder(self,t) :
        if(t):
            self.inorder(t.prevnode)
            print(t.data)
            self.inorder(t.nextnode)
    def preorder(self,t) :
        if(t):
            print(t.data)
            self.preorder(t.prevnode)
```

```
            self.preorder(t.nextnode)
    def postorder(self,t) :
        if(t):
            self.postorder(t.prevnode)
            self.postorder(t.nextnode)
            print(t.data)
t=None
s=BST()
op = int(input("1.insertion\n2.deletion\n3.inorder\n4.preorder\n5.postorder\n-1.exit\nEnter your choice : "))
while(op!=-1) :
    if(op==1) :
        x=int(input("Enter the element to insert : "))
        t=s.insertion(t,x)
        root(t)
    if(op==2) :
        x=int(input("Enter the element to delete : "))
        t=s.deletion(t,x)
        root(t)
    if(op==3) :
        s.inorder(t)
    if(op==4) :
        s.preorder(t)
    if(op==5) :
        s.postorder(t)
    op = int(input("Enter your choice : "))
```

OUTPUT :

```
1.insertion
2.deletion
3.inorder
4.preorder
5.postorder
-1.exit
Enter your choice : 1
Enter the element to insert : 30
Enter your choice : 1
Enter the element to insert : 25
Enter your choice : 1
Enter the element to insert : 40
Enter your choice : 1
Enter the element to insert : 20
Enter your choice : 1
Enter the element to insert : 50
Enter your choice : 3
20
25
30
40
50
Enter your choice : 4
30
25
20
40
50
Enter your choice : 5
20
25
50
40
30
```

```
Enter your choice : 2
Enter the element to delete : 1 25
Enter your choice : 2
Enter the element to delete : 30
Enter your choice : 2
Enter the element to delete : 50
Enter your choice : 3
20
40
Enter your choice : 4
20
40
Enter your choice : 5
40
20
Enter your choice : -1
```

10.Develop an application using python packages

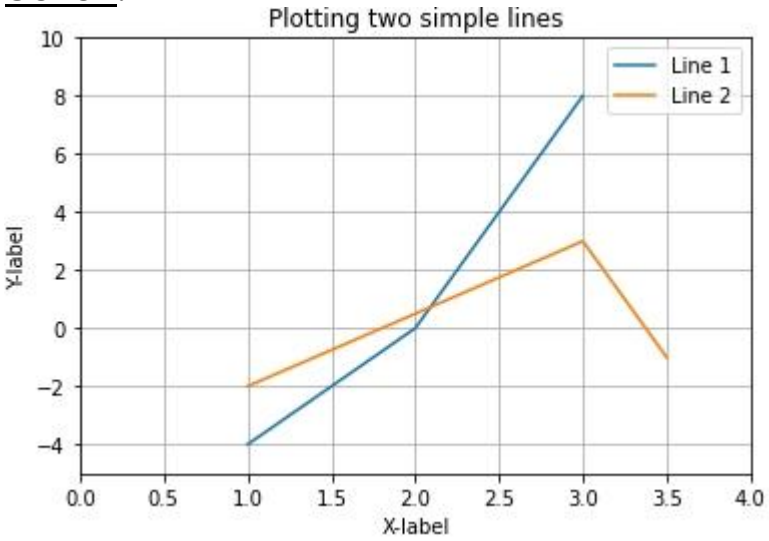
46.AIM : Develop an application using python packages.

PROGRAM :

1.PRELIMINARIES:

```
import matplotlib.pyplot as plt
x=[1,2,3]
x2=[1,3,3.5]
y=[-4,0,8]
y2=[-2,3,-1]
plt.plot(x,y,label="line1")
plt.plot(x2,y2,label="line2")
plt.title("plotting two simple lines")
plt.grid(True)
plt.xlabel("xlabel")
plt.ylabel("ylabel")
plt.xlim([0,4])
plt.ylim([-5,8])
plt.legend()
plt.show()
```

OUPUT:



2.SCATTER PLOTS:

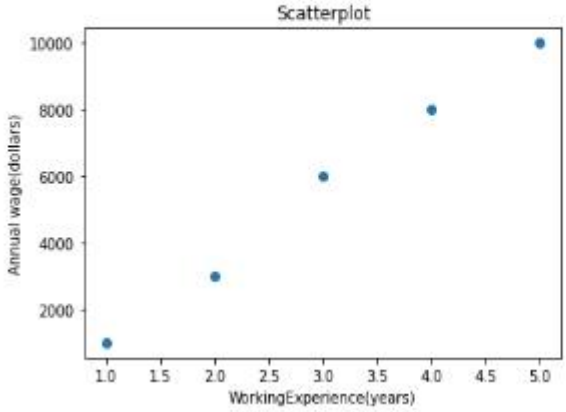
```
import matplotlib.pyplot as plt
import pandas as pd
from google.colab import files
uploaded=files.upload()
import io
data=pd.read_csv(io.BytesIO(uploaded['salary.csv']))
x=data["exp"]
y=data["salary"]
plt.scatter(x,y)
plt.title("Scatterplot")
plt.xlabel("WorkingExperience(years)")
plt.ylabel("Annual wage(dollars)")
plt.show()
```

OUTPUT:

Choose Files salary.csv

- **salary.csv**(application/vnd.ms-excel) - 53 bytes, last modified: 7/12/2021 - 100% done

```
Saving salary.csv to salary (1).csv
```

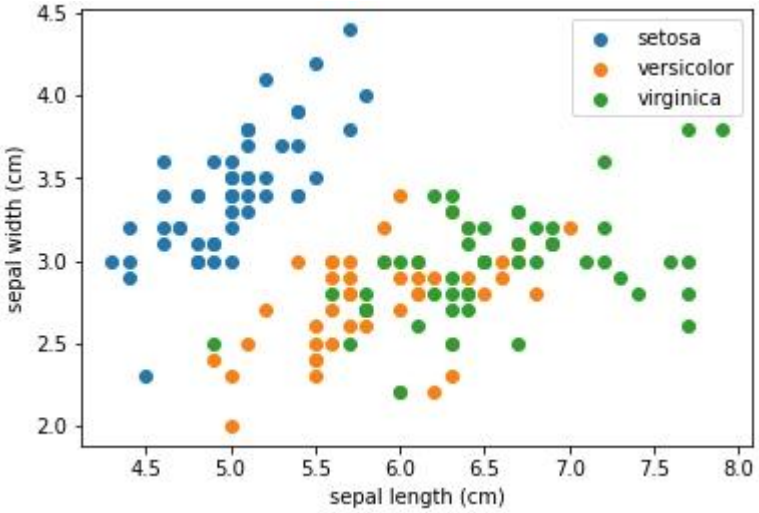


#Program to demonstrate scatterplot

```
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np
iris=datasets.load_iris()
x_iris=iris.data[:,:2]
y_iris=iris.target
n_classes=3
for i in range(n_classes):
  index=np.where(y_iris==i)
  plt.scatter(x_iris[index,0],x_iris[index,1])
  label=iris.target_names[i]
plt.legend()
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()
```

OUTPUT:

```
<function matplotlib.pyplot.show>
```

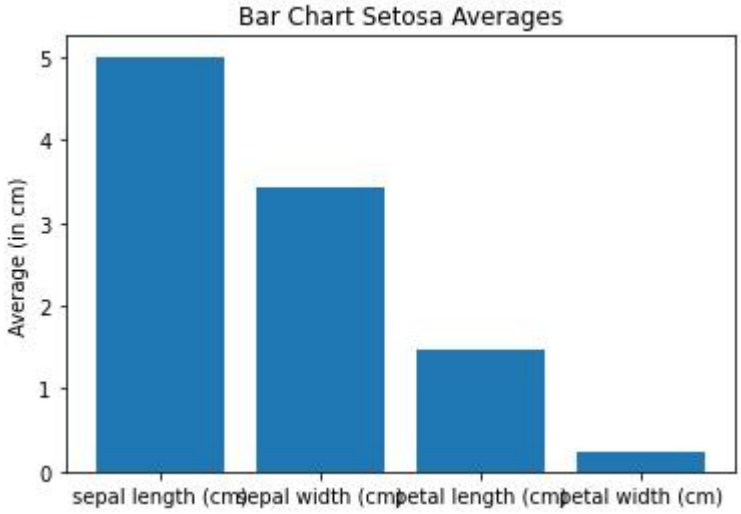


3.BAR CHARTS:

```
from sklearn import datasets
import matplotlib.pyplot as plt
iris=datasets.load_iris()
x_iris=iris.data
```

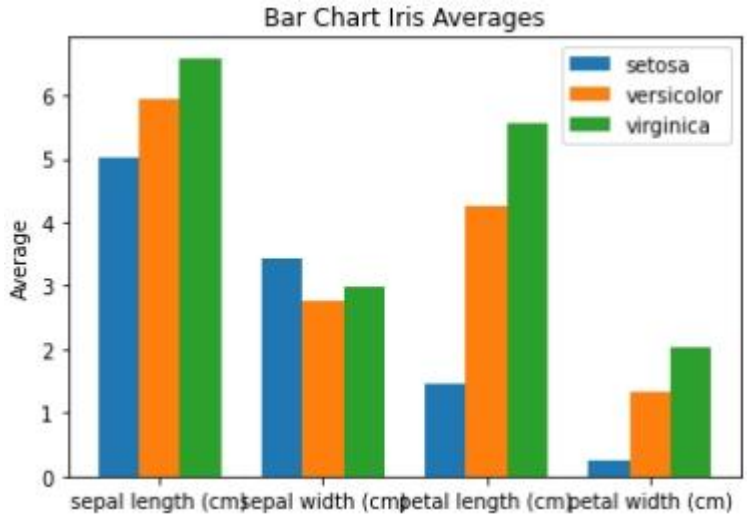
```
y_iris=iris.target
average=x_iris[y_iris==0].mean(axis=0)
plt.bar(iris.feature_names,average)
plt.title("Bar chart Setosa averages")
plt.ylabel("Avg in cm")
plt.show()
```

OUTPUT:




```
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np
iris=datasets.load_iris()
x_iris=iris.data
y_iris=iris.target
n_classes=3
averages=[x_iris[y_iris==i].mean(axis=0) for i in range(n_classes)]
x=np.arange(len(iris.feature_names))
fig=plt.figure()
ax=fig.add_subplot()
bar1=ax.bar(x-0.25,averages[0],0.25,label=iris.target_names[0])
bar2=ax.bar(x,averages[1],0.25,label=iris.target_names[1])
bar3=ax.bar(x+0.25,averages[2],0.25,label=iris.target_names[2])
ax.set_xticks(x)
ax.set_xticklabels(iris.feature_names)
#plt.xlabel(iris.feature_names)
plt.legend()
plt.title("Bar chart Iris Averages")
plt.ylabel("Average")
plt.show()
```

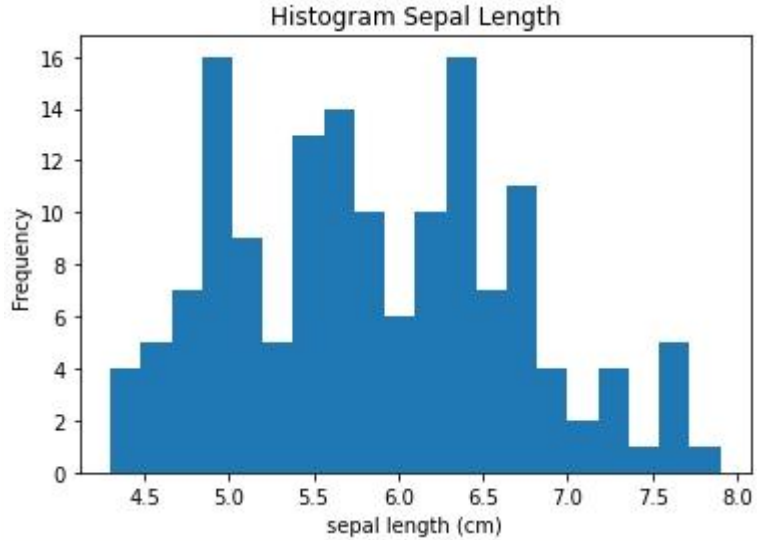
OUTPUT:



4.HISTOGRAMS:

```
from sklearn import datasets
import matplotlib.pyplot as plt
bins = 20
iris = datasets.load_iris()
X_iris = iris.data
X_sepal = X_iris[:, 0]
plt.hist(X_sepal, bins)
plt.title("Histogram Sepal Length")
plt.xlabel(iris.feature_names[0])
plt.ylabel("Frequency")
plt.show
```

OUTPUT:

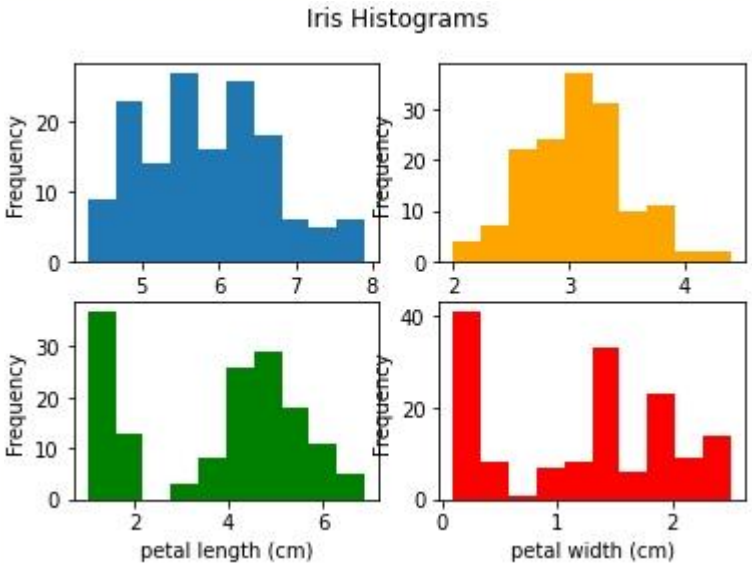



```
from sklearn import datasets
import matplotlib.pyplot as plt
bins = 20
iris = datasets.load_iris()
X_iris = iris.data
fig, axs = plt.subplots(2, 2)
axs[0, 0].hist(X_iris[:, 0])
axs[0, 1].hist(X_iris[:, 1], color='orange')
```

```
axs[1, 0].hist(X_iris[:, 2], color='green')
axs[1, 1].hist(X_iris[:, 3], color='red')
i = 0
for ax in axs.flat:
    ax.set(xlabel=iris.feature_names[i], ylabel='Frequency')
    i += 1
fig.suptitle("Iris Histograms")
```

OUTPUT:

```
Text(0.5, 0.98, 'Iris Histograms')
```

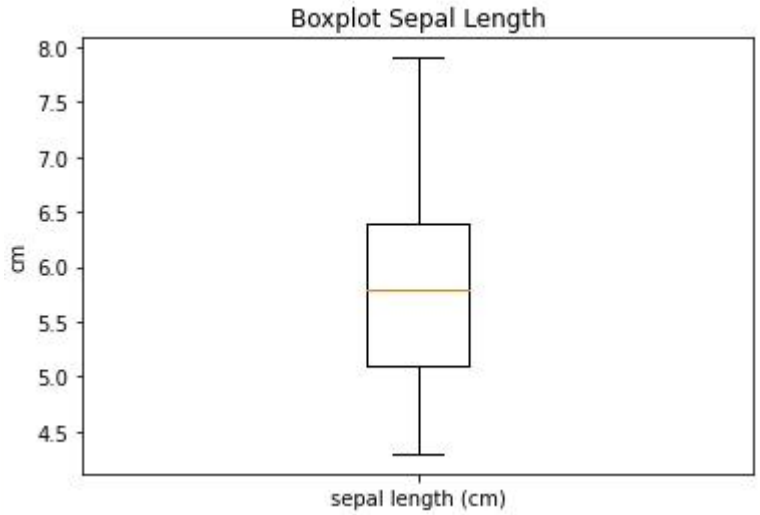


5.BOXPLOTS:

```
from sklearn import datasets
import matplotlib.pyplot as plt
iris = datasets.load_iris()
X_iris = iris.data
X_sepal = X_iris[:, 0]
plt.boxplot(X_sepal, labels=[iris.feature_names[0]])
plt.title("Boxplot Sepal Length")
plt.ylabel("cm")
plt.show
```

OUTPUT:

```
<function matplotlib.pyplot.show>
```




```
from sklearn import datasets
import matplotlib.pyplot as plt
iris = datasets.load_iris()
X_iris = iris.data
plt.boxplot(X_iris, labels=[iris.feature_names[0], iris.feature_names[1], iris.feature_names[2], iris.feature_names[3]])
plt.title("Boxplots Iris features")
plt.ylabel("cm")
plt.show
```

OUTPUT:

<function matplotlib.pyplot.show>