Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 1

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

01) WAP to print "Hello World"

In [2]:
```
print("Hello World")
```

```
Hello World
```

02) WAP to print addition of two numbers with and without using input().

In [4]:
```
num1 = 10
num2 = 12
print(num1+num2)
num3 = int(input("enter number 1 : "))
num4 = int(input("enter number 2 : "))
print(num3+num4)
```

```
22
enter number 1 :  12
enter number 2 :  12
24
```

03) WAP to check the type of the variable.

In [27]:
```
a = True
print(type(a))
```

```
<class 'bool'>
```

04) WAP to calculate simple interest.

In [29]:
```
p = float(input("enter P : "))
r = float(input("enter R : "))
n = float(input("enter N : "))
print("Simple Intrest : ",(p*r*n)/100)
```

```
Simple Intrest :  60.0
```

05) WAP to calculate area and perimeter of a circle.

In [30]:
```
import math

radius = float(input("enter Radius : "))
print("Area of a circle = ",math.pi*radius**2 ,"\nPerimeter of a circle : ",2*math.
```

```
Area of a circle =  28.274333882308138
Perimeter of a circle :  18.84955592153876
```

06) WAP to calculate area of a triangle.

In [31]:
```
height = float(input("enter Height : "))
base = float(input("enter Base : "))
print("Area of triangle : ",(height*base)/2)
```

```
Area of triangle :  3.0
```

07) WAP to compute quotient and remainder.

In [32]:
```
num5 = int(input("enter number 1 : "))
num6 = int(input("enter number 2 : "))
print("Quotient : ",int(num5/num6),"\nRemainder : ",num5%num6)
```

```
Quotient :  4
Remainder :  4
```

08) WAP to convert degree into Fahrenheit and vice versa.

In [25]:
```
f = float(input("Enter tempreture in fahrenheit : "))
print("Tempreture in Celsius : ",(f - 32) * 5/9,"°C")
c = float(input("Enter tempreture in Celsius : "))
print("Tempreture in fahrenheit : ",(c * 9/5) + 32,"°F")
```

```
Tempreture in Celsius :  25.0 °C
Tempreture in fahrenheit :  77.0 °F
```

09) WAP to find the distance between two points in 2-D space.

In [19]:
```
x1 = int(input("enter x1 : "))
x2 = int(input("enter x2 : "))
```

```
y1 = int(input("enter y1 : "))
y2 = int(input("enter y2 : "))
print("The distance beetween two points = ",((x2-x1)**2 + (y2-y1)**2)**0.5)
```

```
The distance beetween two points =  5.0990195135927845
```

10) WAP to print sum of n natural numbers.

In [17]:
```
n1 = int(input("enter number : "))
print("The sum of n natural numbers : ",n1*(n1+1)/2)
```

```
The sum of n natural numbers :  200010000.0
```

11) WAP to print sum of square of n natural numbers.

In [ ]:
```
n2 = int(input("enter number : "))
print("The sum of square of n natural numbers : ",n2*(n2+1)*(2*n2+1)/6)
```

```
14.0
```

12) WAP to concate the first and last name of the student.

In [8]:
```
fname = input("Enter Fname : ")
lname = input("Enter Lname : ")
print(fname,lname,sep=" ")
```

```
vishal parmar
```

13) WAP to swap two numbers.

In [ ]:
```
num7 = int(input("enter number a : "))
num8 = int(input("enter number b : "))
print("before swaping : a =",num7," & b =",num8)
temp = num7 # a = a + b
num7 = num8 # b = a - b
num8 = temp # a = a - b
print("after swaping : a =",num7," & b =",num8)
```

```
before swaping : a = 2  & b = 3
after swaping : a = 3  & b = 2
```

14) WAP to get the distance from user into kilometer, and convert it into meter, feet, inches and centimeter.

In [6]:
```
km = float(input("Enter Distance in kilometere : "))
print(km,"km =",km*1000,"m")
print(km,"km =",km*3280.84,"feet")
print(km,"km =",km*39370.08,"inches")
print(km,"km =",km*100000.0032,"cm")
```

```
10.0 km = 10000.0 m
10.0 km = 32808.4 feet
10.0 km = 393700.80000000005 inches
10.0 km = 1000000.0320000001 cm
```

15) WAP to get day, month and year from the user and print the date in the given format: 23-11-2024.

In [34]:
```
day = int(input("Enter Day: "))
month = int(input("Enter Month : "))
year = int(input("Enter Year : "))
print(day,month,year,sep="-")
```

```
2-2-23
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 2

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

if..else..

01) WAP to check whether the given number is positive or negative.

In [4]:
```
n1 = int(input("Enter number to check positive or negative : "))
if n1>=0 :
    print("Positive")
else :
    print("Negative")
```

```
Negative
```

02) WAP to check whether the given number is odd or even.

In [10]:
```
n2 = int(input("Enter number to check odd or even : "))
if n2%2==0 :
    print("Even")
else :
    print("Odd")
```

```
Even
```

03) WAP to find out largest number from given two numbers using simple if and ternary operator.

In [ ]:
```
n3 = int(input("Enter number 1 : "))
n4 = int(input("Enter number 2 : "))
if n3>=n4 :
    if n3==n4 :
        print("Eaqul")
    else :
        print("Largest Number : ",n3)
else :
        print("Largest Number : ",n4)
```

```
Largest Number :  4
```

04) WAP to find out largest number from given three numbers.

In [ ]:
```
n5 = int(input("Enter number 1 : "))
n6 = int(input("Enter number 2 : "))
n7 = int(input("Enter number 3 : "))
if n5==n6==n7 :
    print("Eaqul")
elif n5>=n6 :
    if n5>=n7 :
        print("Largest Number : ",n5)
    else :
        print("Largest Number : ",n7)
else :
    if n6>=n7 :
        print("Largest Number : ",n6)
    else :
        print("Largest Number : ",n7)
```

```
Eaqul
```

05) WAP to check whether the given year is leap year or not.

[If a year can be divisible by 4 but not divisible by 100 then it is leap year but if it is divisible by 400 then it is leap year]

In [2]:
```
year = int(input("Enter Year to check leap or not : "))
if year%400==0 or (year%4==0 and year%100!=0 ):
    print("Leap year.")
else :
    print("Not Leap year.")
```

```
Leap year.
```

06) WAP in python to display the name of the day according to the number given by the user.

In [8]:
```
day = int(input("Enter the number : "))
if day%7==0 :
    print("Sunday.")
elif day%7==1 :
    print("Monday")
elif day%7==2 :
    print("Tuesday")
elif day%7==3 :
    print("Wensday")
elif day%7==4 :
    print("Thursday")
elif day%7==5 :
    print("Friday")
elif day%7==6 :
    print("saterday")
```

```
Thursday
```

07) WAP to implement simple calculator which performs (add,sub,mul,div) of two no. based on user input.

In [12]:
```
n8 = int(input("Enter number 1 : "))
n9 = int(input("Enter number 2 : "))
op = int(input("Enter\n1)Add\n2)Sub\n3)Mul\n4)div\n"))
if op==1 :
    print(n8,"+",n9,"=",n8+n9)
elif op==2 :
    print(n8,"-",n9,"=",n8-n9)
elif op==3 :
    print(n8,"X",n9,"=",n8*n9)
elif op==4 :
    print(n8,"/",n9,"=",n8/n9)
else :
    print("Wrong input.")
```

```
123456 / 234 = 527.5897435897435
```

08) WAP to read marks of five subjects. Calculate percentage and print class accordingly.

Fail below 35 Pass Class between 35 to 45 Second Class between 45 to 60 First Class between 60 to 70 Distinction if more than 70

In [32]:
```
m1 = int(input("Enter first subject mark out of 100 : "))
m2 = int(input("Enter second subject mark out of 100 : "))
m3 = int(input("Enter third subject mark out of 100 : "))
m4 = int(input("Enter fourth subject mark out of 100 : "))
m5 = int(input("Enter fifth subject mark out of 100 : "))
per = (m1+m2+m3+m4+m5)/5
if per<35 :
    print("Fail")
elif per>=35 and per<45 :
    print("Pass")
```

```
elif per>=45 and per<60 :
    print("Second class")
elif per>=60 and per<70 :
    print("First class")
else :
    print("Distinct")
```

```
Distinct
```

09) Three sides of a triangle are entered through the keyboard, WAP to check whether the triangle is isosceles, equilateral, scalene or right-angled triangle.

In [ ]:
```
x = int(input("Enter side 1 : "))
y = int(input("Enter side 2 : "))
z = int(input("Enter side 3 : "))
if x==y==z :
    print("Equilateral Triangle.")
elif x==y or y==x or z==x :
    print("Isosceles Triangle.")
else :
    print("Scalene Triangle.")
```

10) WAP to find the second largest number among three user input numbers.

In [ ]:
```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
c = int(input("Enter number 3 : "))

if a==b!=c or a==c!=b or b==c!=a :
    if a>c :
        print("Second Largest number : ",c)
    elif a>b :
        print("Second Largest number : ",b)
    else :
        print("Second Largest number : ",a)
else :
    if a>b :
        if a>c :
            if b>c :
                print("Second Largest number : ",b)
            else :
                print("Second Largest number : ",c)
        else :
            print("Second Largest number : ",a)
    else :
        if b>c :
            if a>c :
                print("Second Largest number : ",a)
            else :
                print("Second Largest number : ",c)
```

```
        else :
            print("Second Largest number : ",b)
```

```
Second Largest number :  4
```

11) WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

a. First 1 to 50 units – Rs. 2.60/unit b. Next 50 to 100 units – Rs. 3.25/unit c. Next 100 to 200 units – Rs. 5.26/unit d. above 200 units – Rs. 8.45/unit

In [13]:
```
unit = int(input("Enter total unit : "))
bill = 0
if unit<=50 :
    bill += unit*2.60
elif unit<=100 :
    bill += 50*2.60 + (unit-50)*3.25
elif unit<=200 :
    bill += 50*2.60 + 50*3.25 + (unit-100)*5.26
else :
    bill += 50*2.60 + 50*3.25 + 100*5.26 + (unit-200)*8.45

print("Your Bill is : ",bill,"Rs.")
```

```
Your Bill is :  292.5 Rs.
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 3

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

for and while loop

01) WAP to print 1 to 10.

In [1]:
```
for i in range(1,11) :
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

02) WAP to print 1 to n.

In [3]:
```
n = int(input("Enter number : "))
for i in range(1,n+1) :
    print(i)
```

```
Enter number : 5
1
2
3
4
5
```

03) WAP to print odd numbers between 1 to n.

In [4]:
```
n = int(input("Enter number : "))
for i in range(1,n+1) :
    if i%2==1 :
        print(i)
```

```
Enter number : 25
1
3
5
7
9
11
13
15
17
19
21
23
25
```

04) WAP to print numbers between two given numbers which is divisible by 2 but not divisible by 3.

In [7]:
```
n1 = int(input("Enter first number : "))
n2 = int(input("Enter second number : "))
for i in range(n1+1,n2) :
    if i%2==0 and i%3!=0 :
        print(i)
```

```
Enter first number : 10
Enter second number : 20
14
16
```

05) WAP to print sum of 1 to n numbers.

In [9]:
```
n = int(input("Enter number : "))
sum = 0
for i in range(1,n+1) :
    sum += i
print("Sum of first",n,"numbers :",sum)
```

```
Enter number : 5
Sum of first 5 numbers : 15
```

06) WAP to print sum of series 1 + 4 + 9 + 16 + 25 + 36 + ...n.

In [10]:
```
n = int(input("Enter number : "))
sum = 0
for i in range(1,n+1) :
    sum += i**2
print("Sum of first",n,"numbers squre :",sum)
```

```
Enter number : 5
Sum of first 5 numbers squre : 55
```

07) WAP to print sum of series 1 – 2 + 3 – 4 + 5 – 6 + 7 ... n.

In [11]:
```
n = int(input("Enter number : "))
sum = 0
for i in range(1,n+1) :
    if i%2==1 :
        sum += i
    else :
        sum -= i
print("sum of series 1 - 2 + 3 - 4 + 5 - 6 + 7 ... n :",sum)
```

```
Enter number : 5
sum of series 1 - 2 + 3 - 4 + 5 - 6 + 7 ... n : 3
```

08) WAP to print multiplication table of given number.

In [12]:
```
n = int(input("Enter number : "))
for i in range(1,11) :
    print(n,"X",i,"=",n*i)
```

```
Enter number : 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

09) WAP to find factorial of the given number.

In [13]:
```
n = int(input("Enter number : "))
fac = 1
for i in range(1,n+1) :
    fac *= i
print("factorial o given number =",fac)
```

```
Enter number : 5
factorial o given number = 120
```

10) WAP to find factors of the given number.

In [15]:
```
n = int(input("Enter number : "))
for i in range(1,n+1) :
    if n%i==0 :
        print(i)
```

```
Enter number : 15
1
3
5
15
```

11) WAP to find whether the given number is prime or not.

In [18]:
```
n = int(input("Enter number : "))
count = 0
for i in range(2,n) :
    if n%i==0 :
        count += 1
        break
if count==0 :
    print(n,"is prime number")
else :
    print(n,"is not prime number")
```

```
Enter number : 10
10 is not prime number
```

12) WAP to print sum of digits of given number.

In [21]:
```
n = int(input("Enter number : "))
sum = 0
while(n!=0):
    temp = n%10
    sum += temp
    n = int(n/10)
print("sum of digits of given number :",sum)
```

```
Enter number : 123
sum of digits of given number : 6
```

13) WAP to check whether the given number is palindrome or not

In [44]:
```
n = int(input("Enter number : "))
temp_n = n
sum = 0
while(n!=0):
    temp = int(n%10)
    sum = int(sum*10 + temp)
    n = int(n/10)
if temp_n==sum :
```

```
    print("Palindrone")
else :
    print("Not Palindrone")
```

```
Enter number : 12546
Not Palindrone
```

14) WAP to print GCD of given two numbers.

In [43]:
```
n1 = int(input("Enter first number : "))
n2 = int(input("Enter second number : "))
gcd = 0
temp = n2 if n1>n2 else n1
for i in range(1,temp+1) :
    if n1%i==0 and n2%i==0:
        gcd = i
print("GCD of",n1,"and",n2,"is :",gcd)
```

```
Enter first number : 12
Enter second number : 1
GCD of 12 and 1 is : 1
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 4

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

String

01) WAP to check whether the given string is palindrome or not.

In [3]:
```
s = input("Enter String to check palindrome or not : ")
if s == s[::-1] :
    print("Palindrome")
else :
    print("Not Palindrome")
```

```
Palindrome
```

02) WAP to reverse the words in the given string.

In [ ]:
```
s = input("Enter String : ")
ts = s.split()[::-1]
ans = ' '.join(ts)
print("old string :",s)
print("new string :",ans)
```

```
woh
old string : hi how
new string : how hi
```

03) WAP to remove ith character from given string.

In [35]:
```
s = input("Enter String : ")
index = int(input("Enter index to remove character at this index : "))
print("Original String :",s)
s = s[:index]+s[index+1:]
print("String after remove character :",s)
```

```
Original String : asdfhjdsjskjdkcdbvc njdvbjsjbjn jsdk
String after remove character : asdfhjdsjskjdkcdbvc ndvbjsjbjn jsdk
```

04) WAP to find length of string without using len function.

In [36]:
```
s = input("Enter String : ")
count = 0
for i in s :
    count += 1
print("Length of String :",count)
```

```
Length of String : 8
```

05) WAP to print even length word in string.

In [37]:
```
s = input("Enter String : ")
ts = s.split()
ans = ''
for i in range(len(ts)) :
    if len(ts[i]) %2 == 0 :
        if i == 0 :
            ans += ts[i]
        else :
            ans += ','+ts[i]
print(ans)
```

```
hi,yu
```

06) WAP to count numbers of vowels in given string.

In [39]:
```
s = input("Enter String : ")
count = 0
for i in range(len(s)) :
    if s[i].casefold()=='a' or s[i].casefold()=='e' or s[i].casefold()=='i' or s[i]
        count += 1
print("Your String :",s)
print("Number of vowels in s :",count)
```

```
Your String : vishal parmar
Number of vowels in s : 4
```

07) WAP to capitalize the first and last character of each word in a string.

In [17]:
```
s = input("Enter String : ")
ts = s.split()
ts[0][1].upper
for i in range(len(ts)) :
    ts[i] = ts[i][0].upper() + ts[i][1:len(ts[i])-1] + ts[i][-1].upper()
ans = ' '.join(ts)
print("old string :",s)
print("new string :",ans)
```

```
old string : hi how
new string : HI HoW
```

08) WAP to convert given array to string.

In [2]:
```
arr = ['hi','hello','how','are','you']
print(' '.join(arr))
```

```
hi hello how are you
```

09) Check if the password and confirm password is same or not.

In case of only case's mistake, show the error message.

In [21]:
```
ps = input("Enter password : ")
cps = input("Enter confirm password : ")
if ps==cps :
    print("same password.")
elif ps.upper() == cps.upper() :
    print("Error.....")
else :
    print("Not same.")
```

```
Not same.
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

display as : **** **** **** 3456

In [28]:
```
cn = input("Enter credit card number : ")
ans = ''
for i in range(len(cn)-4) :
    if cn[i].isnumeric() :
        ans += '*'
    else :
        ans += ' '
ans += cn[-4:]
print(ans)
```

```
**** **** **** 1234
```

11) : Checking if the two strings are Anagram or not.

s1 = decimal and s2 = medical are Anagram

In [38]:
```
s1 = input("Enter first string : ")
s2 = input("Enter second string : ")
if sorted(s1.replace(' ','').lower()) == sorted(s2.replace(' ','').lower()):
    print("Given String is Anagram")
else :
    print("Given String is not Anagram")
```

```
Given String is not Anagram
```

12) : Rearrange the given string. First lowercase then uppercase alphabets.

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [1]:
```
s = input("Enter String : ")
ans = ''
upper = ''
for i in s:
    if i.islower() :
        ans += i
    else :
        upper += i
ans += upper
print(ans)
```

```
absnxkjSB
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 5

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

List

01) WAP to find sum of all the elements in a List.

In [2]:
```
l1 = [2,3,4,5]
print('The sum of all elements in list :',sum(l1))
```

```
The sum of all elements in list : 14
```

02) WAP to find largest element in a List.

In [4]:
```
l2 = [1,2,4,7,54]
large = l2[0]
for i in l2:
    if i>large:
        large = i
print("larget number in list :",large)
```

```
larget number in list : -7
```

03) WAP to find the length of a List.

In [6]:
```
l3 = [2,345,6,3256,6,355,46,3525]
length = 0
```

```
for i in l3:
    length += 1
print('The length of list :',length)
```

```
The length of list : 8
```

04) WAP to interchange first and last elements in a list.

In [7]:
```
l4 = [3,4,23,35,3,2,2524,24,24]
print('Before swaping :',l4)
temp = l4[0]
l4[0] = l4[len(l4)-1]
l4[len(l4)-1] = temp
print('after swaping :',l4)
```

```
Before swaping : [3, 4, 23, 35, 3, 2, 2524, 24, 24]
after swaping : [24, 4, 23, 35, 3, 2, 2524, 24, 3]
```

05) WAP to split the List into two parts and append the first part to the end.

In [13]:
```
l5 = [1,2,3,4,5,6,7,8,9]
half1 = l5[:int(len(l5)/2)]
half2 = l5[int(len(l5)/2):]
half2.extend(half1)
print(half2)
```

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

06) WAP to interchange the elements on two positions entered by a user.

In [14]:
```
l6 = [1,2,3,4,5,6,7,8,9,0]
print('Before swaping :',l6)
p1 = int(input("Enter first position : "))
p2 = int(input("Enter second position : "))
temp = l6[p1]
l6[p1] = l6[p2]
l6[p2] = temp
print('After swaping :',l6)
```

```
Before swaping : [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
After swaping : [1, 2, 4, 3, 5, 6, 7, 8, 9, 0]
```

07) WAP to reverse the list entered by user.

In [15]:
```
l7 = [23,23,24,133,7,2,0,8,9,5]
l7.reverse()
print(l7)
```

```
[5, 9, 8, 0, 2, 7, 133, 24, 23, 23]
```

08) WAP to print even numbers in a list.

In [17]:
```
l8 = [1,2,3,4,5,6,7,8,9,0]
for i in l8:
    if i%2==0:
        print(i)
```

```
2
4
6
8
0
```

09) WAP to count unique items in a list.

In [24]:
```
l9 = [1,3,5,2,4,1,0]
count = 0
for i in l9:
    if l9.count(i)==1:
        count += 1
print('Number of Uniqe Elements :',count)
```

```
Number of Uniqe Elements : 5
```

10) WAP to copy a list.

In [28]:
```
l10 = [1,3,6,9,2,5,0,7]
tempList = l10.copy()
print('Copied List :',tempList)
```

```
Copied List : [1, 3, 6, 9, 2, 5, 0, 7]
```

11) WAP to print all odd numbers in a given range.

In [30]:
```
l11 = [1,2,3,4,5,6,7,8,9,0,12,3,564,5,2,6,24,62,48,9,7]
r1 = int(input("Enter first range : "))
r2 = int(input("Enter second second : "))
for i in range(r1,r2+1):
    if l11[i]%2==1:
        print(l11[i])
```

```
5
7
9
```

12) WAP to count occurrences of an element in a list.

In [1]:
```
l12 = [1,2,3,4,5,2,4,7,4,4,2,1,0]
temp = []
for i in l12:
    if temp.__contains__(i) == False:
        temp.append(i)
        c = l12.count(i)
        print("Occurrences of {0} in list : {1}".format(i,c))
```

```
Occurrences of 1 in list : 2
Occurrences of 2 in list : 3
Occurrences of 3 in list : 1
Occurrences of 4 in list : 4
Occurrences of 5 in list : 1
Occurrences of 7 in list : 1
Occurrences of 0 in list : 1
```

13) WAP to find second largest number in a list.

In [41]:
```
l13 = [0,1,2,3,4,5,6,7,8,9]
# large = l13[0]
# secondLarge = l13[1]
# for i in l13:
#     if large<i:
#         large = i
# for i in l13:
#     if i>secondLarge and i<large:
#         secondLarge = i
temp = l13.copy()
secondLarge = temp[len(temp)-2]
print('Second large number in list :',secondLarge)
```

```
Second large number in list : 8
```

14) WAP to extract elements with frequency greater than K.

In [43]:
```
l14 = [1,2,3,4,5,2,4,7,4,4,2,0]
k = int(input("Enter value of K : "))
temp = []
for i in l14:
    if temp.__contains__(i) == False:
        temp.append(i)
        c = l14.count(i)
        if c>k:
            print("Occurrences of {0} in list : {1}".format(i,c))
```

```
Occurrences of 2 in list : 3
Occurrences of 4 in list : 4
```

15) WAP to create a list of squared numbers from 0 to 9 with and without using List Comprehension.

In [45]:
```
l15_1 = []
for i in range(10):
    l15_1.append(i**2)
print(l15_1)
l15_2 = [i**2 for i in range(10)]
print(l15_2)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

16) WAP to create a new list (fruit whose name starts with 'b') from the list of fruits given by user.

In [46]:
```
l16 = ['banana','cherry','apple','Berry','Citrus', 'Plum', 'Mandarin','Blueberry']
fruits = []
for i in l16:
    if i[0].upper() == 'B':
        fruits.append(i)
print("Names of fruits start from B :",fruits)
```

```
Names of fruits start from B : ['banana', 'Berry', 'Blueberry']
```

17) WAP to create a list of common elements from given two lists.

In [47]:
```
l17_1 = [1,2,3,4,5,3]
l17_2 = [2,3,4,6,3,8]
result = []
for i in l17_1:
    if i in l17_2 and i not in result:
        result.append(i)
print('List of common elements :',result)
```

```
List of common elements : [2, 3, 4]
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 6

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

Tuple

01) WAP to find sum of tuple elements.

In [2]:
```
t1 = (1,2,3,4,5)
print(f"The sum of tuple elements = {sum(t1)}")
```

```
The sum of tuple elements = 15
```

02) WAP to find Maximum and Minimum K elements in a given tuple.

In [ ]:
```
t2 = (1,2,3,4,5,6,7,8,9,0)
k = int(input("Enter value of k : "))
t2 = sorted(t3)
print(f"Maximum {k} elements : {t2[-(k):]}")
print(f"Minimum {k} elements : {t2[:k]}")
```

```
Maximum 3 elements : [7, 8, 9]
Minimum 3 elements : [0, 1, 2]
```

03) WAP to find tuples which have all elements divisible by K from a list of tuples.

In [3]:
```
t3 = (1,2,3,4,5,6,7,8,9,0)
k = int(input("Enter value of k : "))
for i in t3:
    if i%k==0:
        print(i)
```

```
3
6
9
0
```

04) WAP to create a list of tuples from given list having number and its cube in each tuple.

In [10]:
```
t4 = (1,2,3,4,5,6,7,8,9)
ans4 = tuple((i,i**2) for i in t4)
print(ans4)
```

```
((1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81))
```

05) WAP to find tuples with all positive elements from the given list of tuples.

In [11]:
```
t5 = (1,-2,-3,4,-5,-6,7,8,-9)
ans5 = tuple(i for i in t5 if i>0)
print(ans5)
```

```
(1, 4, 7, 8)
```

06) WAP to add tuple to list and vice – versa.

In [36]:
```
t6 = (1,2,3,4,5)
l6 = [6,7,8,9,0]
l6 += t6
print(f"Add list to tuple : {l6}")
t6 = (1,2,3,4,5)
l6 = [6,7,8,9,0]
t6 = tuple(list(t6)+l6)
print(f"Add tuple to list : {t6}")
```

```
Add list to tuple : [6, 7, 8, 9, 0, 1, 2, 3, 4, 5]
Add tuple to list : (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
```

07) WAP to remove tuples of length K.

In [37]:
```
t7 = (1,2,3,4,5,6,7,8,9,0)
k = int(input("Enter value of k : "))
t7 = tuple(list(t7)[:len(t7)-k])
print(t7)
```

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

08) WAP to remove duplicates from tuple.

In [27]:
```
t8 = (1,2,3,4,5,2,3,5,7)
t8 = list(t8)
for i in t8:
    if t8.count(i)>1:
        t8.remove(i)
t8 = tuple(t8)
print(t8)
```

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

09) WAP to multiply adjacent elements of a tuple and print that resultant tuple.

In [23]:
```
t9 = (1,2,3,4,5)
ans9 = []
ans9.append(t9[0]*t9[1])
ans9.extend(t9[i]*t9[i+1]*t9[i-1] for i in range(1,len(t9)-1))
ans9.append(t9[len(t9)-2]*t9[len(t9)-1])
print(ans9)
```

```
[2, 6, 24, 60, 20]
```

10) WAP to test if the given tuple is distinct or not.

In [25]:
```
t10 = (1,2,3,4,5,1)
flag = True
for i in t10:
    if t10.count(i)>1:
        flag = False
print('Distinct' if flag else 'Not Distinct')
```

```
Not Distinct
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

Set & Dictionary

01) WAP to iterate over a set.

In [1]:
```
s1={1,2,3,4}
for i in s1:
    print(i)
s1
```

```
1
2
3
4
```

Out[1]: {1, 2, 3, 4}

02) WAP to convert set into list, string and tuple.

In [6]:
```
s1={1,2,3,4}
s2={'1','2','3','4'}
l1=list(s1)
l1
s3=' '.join(s2)
s3
```

```
t1=tuple(s1)
t1
```

Out[6]: (1, 2, 3, 4)

03) WAP to find Maximum and Minimum from a set.

In [11]:
```
s1={1,2,3,4}
print(min(s1))
print(max(s1))
```

```
1
4
```

04) WAP to perform union of two sets.

In [15]:
```
s1={1,2,3,4}
s2={5,6,7,8}
s3=s1|s2
s3
```

Out[15]: {1, 2, 3, 4, 5, 6, 7, 8}

05) WAP to check if two lists have at-least one element common.

In [18]:
```
l1=[1,2,3,4]
l2=[3,1,5,6]
l3=set(l1)&set(l2)
l4=list(l3)
l4
```

Out[18]: [1, 3]

06) WAP to remove duplicates from list.

In [20]:
```
l1=[1,1,2,2,3,3,4,5]
s1=set(l1)
l2=list(s1)
l2
```

Out[20]: [1, 2, 3, 4, 5]

07) WAP to find unique words in the given string.

In [29]:
```
str1='abc abcfdg'
s1=set(str1.split())
s1
```

Out[29]: {'abc', 'abcfdg'}

08) WAP to remove common elements of set A & B from set A.

In [27]:
```
a={1,2,3,4}
b={3,1,7,8}
s3=a&b
a-s3
```

Out[27]: {2, 4}

09) WAP to check whether two given strings are anagram or not using set.

In [28]:
```
str1='decimal'
str2='medical'
if(set(str1)==set(str2)):
    print("anagram")
else:
    print("not anagram")
```

anagram

10) WAP to find common elements in three lists using set.

In [32]:
```
l1=[1,2,3,4]
l2=[5,7,3,9]
l3=[3,1,5,6]
s1=set(l1)&set(l2)&set(l3)
list(s1)
```

Out[32]: [3]

11) WAP to count number of vowels in given string using set.

In [35]:
```
str1='abcyagid'
l1=list(str1)
l2={'a','e','i','o','u'}
count=0
for i in l1:
    if(i in l2):
        count+=1;
count
```

Out[35]: 3

12) WAP to check if a given string is binary string or not.

In [37]:
```
str1 = '0101010101'
s1 = set(str1)
s2 = {'0','1'}
```

```
if(s1.issubset(s2)):
    print("binary")
else:
    print("not binary")
```

```
binary
```

13) WAP to sort dictionary by key or value.

In [36]:
```
d1 = {
    'a':1,'c':3,'b':2,'d':4,
}
lkeys = list(d1.keys())
lkeys.sort()
lvalues = list(d1.values())
lvalues.sort()

d2 = {i: d1[i] for i in lkeys}
d2
```

Out[36]:
```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

14) WAP to find the sum of all items (values) in a dictionary given by user. (Assume: values are numeric)

In [38]:
```
d1 = {
    'a':1,'c':3,'b':2,'d':4,
}
lvalues = list(d1.values())
sum = 0
for i in lvalues:
    sum += i

print(sum)
```

```
10
```

15) WAP to handle missing keys in dictionaries.

Example : Given, dict1 = {'a': 5, 'c': 8, 'e': 2}

if you look for key = 'd', the message given should be 'Key Not Found', otherwise print the value of 'd' in dict1.

In [39]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2,'d':1}
char = input("enter key")
l1 = list(dict1.keys())
l2 = list(dict1.values())
for i in l1:
    if(i==char):
        print (l2[l1[i]])
```

```
enter key2
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 8

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

User Defined Function

01) Write a function to calculate BMI given mass and height. (BMI = mass/h**2)

In [31]:
```
def BMI(weight,height):
    bmi=weight/(height**2);
    return bmi

BMI(56,1.80)
```

Out[31]:
```
17.28395061728395
```

02) Write a function that add first n numbers.

In [41]:
```
def add(n):
    sum=0
    for i in range(1,n+1):
        sum=sum+i
    return sum

add(5)
```

Out[41]:
```
15
```

03) Write a function that returns 1 if the given number is Prime or 0 otherwise.

In [61]:
```
def prime(a):
    if(a==0 or a==1):
        return 0;
    elif(a>1):
        for i in range(2,a):
            if(a%i==0):
                return 0
        else:
            return 1

prime(43)
```

Out[61]: 1

04) Write a function that returns the list of Prime numbers between given two numbers.

In [68]:
```
li=[]
def Primeno(a,b):
    for i in range(a,b):
        if(prime(i)):
            li.append(i)
Primeno(1,10)
print(li)
```

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

05) Write a function that returns True if the given string is Palindrome or False otherwise.

In [91]:
```
st=input("Enter a string:")
def palindrome(st):
    if(st[::-1]==st):
        return True
    else:
        return False
palindrome(st)
```

```
Enter a string: lol
```

Out[91]: True

06) Write a function that returns the sum of all the elements of the list.

In [93]:
```
li=[1,2,3,4,5,6,7,8,9]
def ans():
    Sum=sum(li)
```

```
    return Sum
ans()
```

Out[93]: 45

07) Write a function to calculate the sum of the first element of each tuples inside the list.

In [107…

```
li=[(0,1,2),(4,5,6),(7,8,9)]
def listsum():
    sum=0
    for i,j,k in li:
            sum=sum+i
    print(sum)
listsum()
```

```
11
```

08) Write a recursive function to find nth term of Fibonacci Series.

In [157…

```
def fibonacci(a,b,n):
    if n==1:
        return a
    return fibonacci(b,a+b,n-1)

print(fibonacci(0,1,12))
```

```
89
```

09) Write a function to get the name of the student based on the given rollno.

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

In [31]:

```
dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
def nameofstudent(n):
    a=dict1.get(n)
    return a

result=nameofstudent(103)
print(result)
```

```
Jay
```

10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

In [17]:
```
scores = [200, 456, 300, 100, 234, 678]
def sumofScores(scores):
    total=0
    for i in scores:
        if i%10==0:
            total+=i
    return total

result=sumofScores(scores)
print(result)
```

```
600
```

11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

Before : {'a': 10, 'b':20, 'c':30, 'd':40}

After : {10:'a', 20:'b', 30:'c', 40:'d'}

In [39]:
```
ini_dict={'a': 10, 'b':20, 'c':30, 'd':40}
def invertDictonary(ini_dict):
    inv_dict = dict(zip(ini_dict.values(), ini_dict.keys()))
    return inv_dict

invertDictonary(ini_dict)
```

Out[39]:
```
{10: 'a', 20: 'b', 30: 'c', 40: 'd'}
```

12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

In [53]:
```
string="the quick brown fox jumps over the lazy dog"

def pangram(string):
    string=string.replace(" ","")
    string=string.lower()
    x=list(set(string))
    x.sort()
    x="".join(x)
    alphabets="abcdefghijklmnopqrstuvwxyz"
    if(x==alphabets):
        print("The string is a pangram")
    else:
        print("The string is not a pangram")

pangram(string)
```

```
The string is a pangram
```

13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

In [65]:
```
s1 = "AbcDEfgh"
def countUPLW(s1):
    lower=0
    upper=0
    for i in s1:
        if i.islower():
            lower+=1
        else:
            upper+=1
    print(f"Lower case are:{lower}")
    print(f"Upper case are:{upper}")

countUPLW(s1)
```

```
Lower case are:5
Upper case are:3
```

14) Write a lambda function to get smallest number from the given two numbers.

In [67]:
```
min_number = lambda a, b : min(a,b)

print(min_number(5, 8))
```

```
5
```

15) For the given list of names of students, extract the names having more that 7 characters. Use filter().

In [3]:
```
students=['Alexander','Benjamin','Jonathan','Malay','Meet','Dhairya']
def myFunc(x):
  if len(x) > 7:
    return True
  else:
    return False

names = filter(myFunc, students)

for x in names:
  print(x)
```

```
Alexander
Benjamin
Jonathan
```

16) For the given list of names of students, convert the first letter of all the names into uppercase. use map().

In [62]:
```
students = ['alexander', 'benjamin', 'jonathan', 'malay', 'meet', 'dhairya']

def uppercaseusingMap(names):
    return list(map(str.capitalize, names))

result = uppercaseusingMap(students)
print(result)
```

```
['Alexander', 'Benjamin', 'Jonathan', 'Malay', 'Meet', 'Dhairya']
```

17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & variable length Keyword Arguments (*kwargs)
5. Keyword-Only & Positional Only Arguments

In [92]:
```
# Positional Arguments
print("Positional Arguments::")
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)

nameAge(name="Prince", age=20)
print()

# Keyword Arguments
print("Keyword Arguments::")
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "alexander", child2 = "benjamin", child3 = "Jonathan")
print()

# Default Arguments
print("Default Arguments::")
def my_function(country = "India"):
  print("I am from " + country)

my_function()
my_function("Australia")
print()

# Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)
print("Variable Legngth Positional(*args) & variable length Keyword Arguments (**kw
# *args
def my_function(*kids):
```

```
  print("*args:The youngest child is " + kids[2])

my_function("alexander", "benjamin", "Jonathan")
#**kwargs
def my_function(**kid):
  print("**kargs:His last name is " + kid["lname"])

my_function(fname = "alexander", lname = "benjamin")
print()

# Keyword-Only & Positional Only Arguments
print("Keyword-Only & Positional Only Arguments::")
```

```
Positional Arguments::
Hi, I am Prince
My age is  20

Keyword Arguments::
The youngest child is Jonathan

Default Arguments::
I am from India
I am from Australia

Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)::
*args:The youngest child is Jonathan
**kargs:His last name is benjamin

Keyword-Only & Positional Only Arguments::
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 9

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

File I/O

01) WAP to read and display the contents of a text file. (also try to open the file in some other directory)

- in the form of a string

- line by line

- in the form of a list

In [11]:
```
f = open("newFile.txt", "r")
print(f.read())
f.close()

f = open("newFile.txt", "r")
print(f.readline())
f.close()

f = open("newFile.txt", "r")
print(f.readlines())
f.close()
```

```
Hello World!!!!!!!!!!
Good Mornings........
Hello World!!!!!!!!!!

['Hello World!!!!!!!!!!\n', 'Good Mornings........']
```

02) WAP to create file named "new.txt" only if it doesn't exist.

In [49]:
```
f = open("new.txt","w")
f.write("Hi Hello from python")
f.close()
```

03) WAP to read first 5 lines from the text file.

In [31]:
```
f = open("newFile.txt", "r")
for i in range(5):
    print(f.readline())
f.close()
```

```
Hello World!!!!!!!!!!

Good Mornings........

1

2

3
```

04) WAP to find the longest word(s) in a file

In [55]:
```
f = open("newFile.txt", "r")
words = f.read().split()
print (max(words, key=len))
f.close()
```

```
Mornings........
```

05) WAP to count the no. of lines, words and characters in a given text file.

In [51]:
```
#No.of lines
f = open("newFile.txt","r")
lines=len(f.readlines())
print(lines)
f.close()

#No. of words
f = open("newFile.txt","r")
lines=len(f.readline())
print(lines)
f.close()
```

```
#No. of characters
f = open("newFile.txt","r")
lines=len(f.read())
print(lines)
f.close()
```

```
9
22
57
```

06) WAP to copy the content of a file to the another file.

In [59]:
```
f1 = open("newFile.txt","r")
f2 = open("newFile2.text","w")
for i in f1:
    f2.write(i)
f1.close()
f2.close()
```

07) WAP to find the size of the text file.

In [67]:
```
import os
sz = os.path.getsize("newFile.txt")
print(sz)
```

```
57
```

08) WAP to create an UDF named frequency to count occurances of the specific word in a given text file.

In [71]:
```
def frequency(file_content,specific_word):
    # Method-1
    count = file_content.count(specific_word)
    # Method-2
    # count = 0
    # for i in file_content:
    #     if i == specific_word:
    #         count += 1
    return count

fp = open("newFile.txt","r")
data = fp.read()
print(frequency(data.split(),'Hello'))
fp.close()
```

```
1
```

09) WAP to get the score of five subjects from the user, store them in a file. Fetch those marks and find the highest score.

In [5]:
```
fp = open("newFile.txt", "w+")
for i in range(1, 6):
```

```
    mark = input(f"Enter marks of subject {i}: ")
    fp.write(mark + "\n")
fp.seek(0)
l1 = [int(mark.strip()) for mark in fp.readlines()]
print(f"The highest score is: {max(l1)}")
fp.close()
```

```
Enter marks of subject 1:  5
Enter marks of subject 2:  10
Enter marks of subject 3:  15
Enter marks of subject 4:  20
Enter marks of subject 5:  25
The highest score is: 25
```

10) WAP to write first 100 prime numbers to a file named primenumbers.txt

(Note: each number should be in new line)

In [3]:
```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_numbers(n, filename):
    count = 0
    num = 2
    with open(filename, 'w') as fp:
        while count < n:
            if is_prime(num):
                fp.write(f"{num}\n")
                count += 1
            num += 1

prime_numbers(100, "primenumbers.txt")
```

11) WAP to merge two files and write it in a new file.

In [13]:
```
f1 = open("newFile.txt", "r")
f2 = open("new.txt", "r")
fp = open("mergedFile.txt", "w")
fp.write(f1.read())
fp.write(f2.read())
f1.close()
f2.close()
fp.close()

print("Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.txt
```

```
Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.txt'
```

12) WAP to replace word1 by word2 of a text file. Write the updated data to new file.

In [15]:
```
f1 = open("mergedFile.txt", "r")
content = f1.read()
updated_content = content.replace('Hi', 'bye')
new_file = open("updated_content.txt", "w")
new_file.write(updated_content)
f1.close()
new_file.close()

print("The word replacement has been done and saved to 'updated_content.txt'")
```

```
The word replacement has been done and saved to 'updated_content.txt'
```

13) Demonstrate tell() and seek() for all the cases(seek from beginning-end-current position) taking a suitable example of your choice.

In [3]:
```
with open('newFile.txt', 'w') as fp:
    fp.write("Hello, this is a sample file for demonstrating tell() and seek().")

with open('newFile.txt', 'r') as fp:
    # Case 1: Tell the current position from the beginning
    print("Case 1: Current position from beginning (before reading):", fp.tell())
    print("Reading first 5 characters:")
    print(fp.read(5))  # Read first 5 characters
    print("Position after reading 5 characters:", fp.tell())

    # Case 2: Seek from the beginning (SEEK_SET)
    fp.seek(0, 0)  # Move the pointer to the beginning
    print("\nCase 2: Seek from the beginning to position 0:", fp.tell())
```

```
Case 1: Current position from beginning (before reading): 0
Reading first 5 characters:
Hello
Position after reading 5 characters: 5

Case 2: Seek from the beginning to position 0: 0
```

In [ ]:
```

```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 10

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

Exception Handling

01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError

Note: handle them using separate except blocks and also using single except block too.

In [14]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # ValueError
    value = int("not_a_number")

    # TypeError
    result = "string" + 10

#handling using seperate except block
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
except ValueError:
```

```
    print("Error: Invalid value provided.")
except TypeError:
    print("Error: Type mismatch encountered.")

try:
    # ZeroDivisionError
    result = 10 / 0

    # ValueError
    value = int("not_a_number")

    # TypeError
    result = "string" + 10

#handling using single except block
except (ZeroDivisionError, ValueError, TypeError) as e:
    print(f"Error: {str(e)}")
```

```
Error: Cannot divide by zero.
Error: division by zero
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [42]:
```
#Index Error
try:
    l1=[1,2,3,4,5,6]
    print(l1[6])
except IndexError as e:
    print(f'Index Error:{str(e)}')

#KeyError
try:
    d1={'a':1,'b':2,'c':3}
    print(d1['d'])
except KeyError as e:
    print(f'KeyError: {str(e)}')
```

```
Index Error:list index out of range
KeyError: 'd'
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [50]:
```
#FileNotFoundError
try:
    fp=open("a.txt", "r")
    fp.read()
except FileNotFoundError as e:
    print(f'FileNotFoundError:{str(e)}')
```

```
#ModuleNotFoundError
try:
    import a

except ModuleNotFoundError as e:
    print(f'ModuleNotFoundError:{str(e)}')
```

```
FileNotFoundError:[Errno 2] No such file or directory: 'a.txt'
ModuleNotFoundError:No module named 'a'
```

04) WAP that catches all type of exceptions in a single except block.

In [63]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # IndexError
    # my_list = [1, 2, 3]
    # print(my_list[5])

    # ValueError
    # value = int("not_a_number")

    # KeyError
    # my_dict = {'a': 1}
    # print(my_dict['b'])

    # FileNotFoundError
    # with open("non_existent_file.txt", "r") as file:
    #     content = file.read()

    # ModuleNotFoundError
    # import non_existent_module

except Exception as e:
    print(f"An error occurred: {str(e)}")
```

```
An error occurred: division by zero
```

05) WAP to demonstrate else and finally block.

In [57]:
```
try:
    result = 10 / 2
    print(result)

except ZeroDivisionError as e:
    print(f"An error occurred: {str(e)}")

else:
    print("he try block executed successfully.")
```

```
finally:
    print("This block is always executed")
```

```
5.0
he try block executed successfully.
This block is always executed
```

06) Create a short program that prompts the user for a list of grades separated by commas.

Split the string into individual grades and use a list comprehension to convert each string to an integer.

You should use a try statement to inform the user when the values they entered cannot be converted.

In [75]:
```
grades_input = input("Enter a list of grades separated by commas: ")

try:
    grades = [int(grade.strip()) for grade in grades_input.split(',')]
    print("Grades:", grades)

except ValueError:
    print("Error: Some of the values you entered cannot be converted to integers. P
```

```
Enter a list of grades separated by commas:  10,20.3.50,60
Error: Some of the values you entered cannot be converted to integers. Please enter
valid numbers.
```

07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

In [65]:
```
def divide(a, b):
    try:
        result = a / b

    except ZeroDivisionError:
        return "Error: Cannot divide by zero."

    else:
        return result

a = float(input("Enter the a: "))
b = float(input("Enter the b: "))

result = divide(a, b)
print(f"Result: {result}")
```

```
Enter the a:  5
Enter the b:  0
Result: Error: Cannot divide by zero.
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

In [71]:
```
def ageInvalid():
    try:
        age = int(input("Enter your age: "))
        if age < 18:
            raise ValueError("Enter Valid Age")
        else:
            print(f"Your age is {age}")
    except ValueError as e:
        print(f"Error: {str(e)}")

ageInvalid()
```

```
Enter your age:  5
Error: Enter Valid Age
```

09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

In [81]:
```
class InvalidUsername(Exception):
    pass
def custom_exception():
    try:
        name=input("Enter your name::")
        if(len(name)<5 or len(name)>15):
            raise InvalidUsername("Username must be between 5 and 15 characters lon
    except InvalidUsername as e:
        print(f"Error:{str(e)}")
custom_exception()
```

```
Enter your name:: raj
Error:Username must be between 5 and 15 characters long
```

10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

In [79]:
```
import math
class NegativeNumberError(Exception):
    pass
def calculate_square_root():
    try:
        number = float(input("Enter a number to calculate its square root: "))
        if number < 0:
            raise NegativeNumberError("Cannot calculate the square root of a negati
        sqrt_result = math.sqrt(number)
        print(f"The square root of {number} is: {sqrt_result}")
    except NegativeNumberError as e:
        print(f"Error: {str(e)}")
    except ValueError:
        print("Error: Please enter a valid number.")
calculate_square_root()
```

```
Enter a number to calculate its square root:  25
The square root of 25.0 is: 5.0
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 11

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

Modules

In [51]:
```
import calculator
import random as rand
from datetime import datetime, timedelta, date
import math
```

01) WAP to create Calculator module which defines functions like add, sub,mul and div.

Create another .py file that uses the functions available in Calculator module.

In [36]:
```
print(f"add = {calculator.add(10,7)}")
print(f"sub = {calculator.sub(10,7)}")
print(f"mult = {calculator.mult(10,7)}")
print(f"div = {calculator.div(10,7)}")
print(f"fdiv = {calculator.fdiv(10,7)}")
print(f"rem = {calculator.rem(10,7)}")
```

```
add = 17
sub = 3
mult = 70
div = 1.4285714285714286
fdiv = 1
rem = 3
```

02) WAP to pick a random character from a given String.

In [37]:
```
st = input("Enter a string : ")
print(st[rand.randrange(len(st))])
```

```
Enter a string : My name Jevin Morad
y
```

03) WAP to pick a random element from a given list.

In [38]:
```
li = input("Enter anything with space : ").split()
print(li[rand.randrange(len(li))])
```

```
Enter numbers with space : 1 2 3 4 5 6 7 8 9 10 11
3
```

04) WAP to roll a dice in such a way that every time you get the same number.

In [43]:
```
rand.seed(4)
print( rand.randint(100,1000))

rand.seed(2)
print( rand.randint(100,200))

rand.seed(2)
print( rand.randint(100,1000))
```

```
341
107
983
```

05) WAP to generate 3 random integers between 100 and 999 which is divisible by 5.

In [40]:
```
li = []
while(len(li)<3):
    num = rand.randint(100,1000)
    if(num%5==0):
        li.append(num)
print(li)
```

```
[820, 845, 955]
```

06) WAP to generate 100 random lottery tickets and pick two lucky tickets from it and announce them as Winner and

Runner up respectively.

In [49]:
```
li = rand.sample(range(1, 1000), 100)
winner = li[rand.randrange(len(li))]
print(f"Winner is {winner}")
li.remove(winner)
print(f"Runner up is {li[rand.randrange(len(li))]}")
```

```
Winner is 294
Runner up is 501
```

07) WAP to print current date and time in Python.

In [14]:
```
print(datetime.now())
```

```
2025-02-10 10:12:34.915800
```

08) Subtract a week (7 days) from a given date in Python.

In [13]:
```
print("Current time: ", datetime.now())
print(datetime.now() - timedelta(days=7))
```

```
Current time:  2025-02-10 10:12:30.132256
2025-02-03 10:12:30.133078
```

09) WAP to Calculate number of days between two given dates.

In [31]:
```
date1 = date(2006,1,29)
date2 = datetime.now().date()
print(f"days = {(date2-date1).days}")
```

```
days = 6952
```

10) WAP to Find the day of the week of a given date.(i.e. wether it is sunday/monday/tuesday/etc.)

In [22]:
```
days=['Monday','Tuesdya','Wednesday','Thursday','Friday','Saturday','Sunday']
print(days[datetime.now().weekday()])
```

```
Monday
```

11) WAP to demonstrate the use of date time module.

In [24]:
```
print(datetime.now())

date1 = date(2006,1,29)
date2 = date(2025,2,10)
print(f"days = {(date2-date1).days}")

days=['Monday','Tuesdya','Wednesday','Thursday','Friday','Saturday','Sunday']
print(days[datetime.now().weekday()])
```

```
2025-02-10 10:22:33.795948
days = 6952
Monday
```

12) WAP to demonstrate the use of the math module.

In [53]:
```
print(math.pow(2,2))
print(math.sqrt(7))
```

```
4.0
2.6457513110645907
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 12

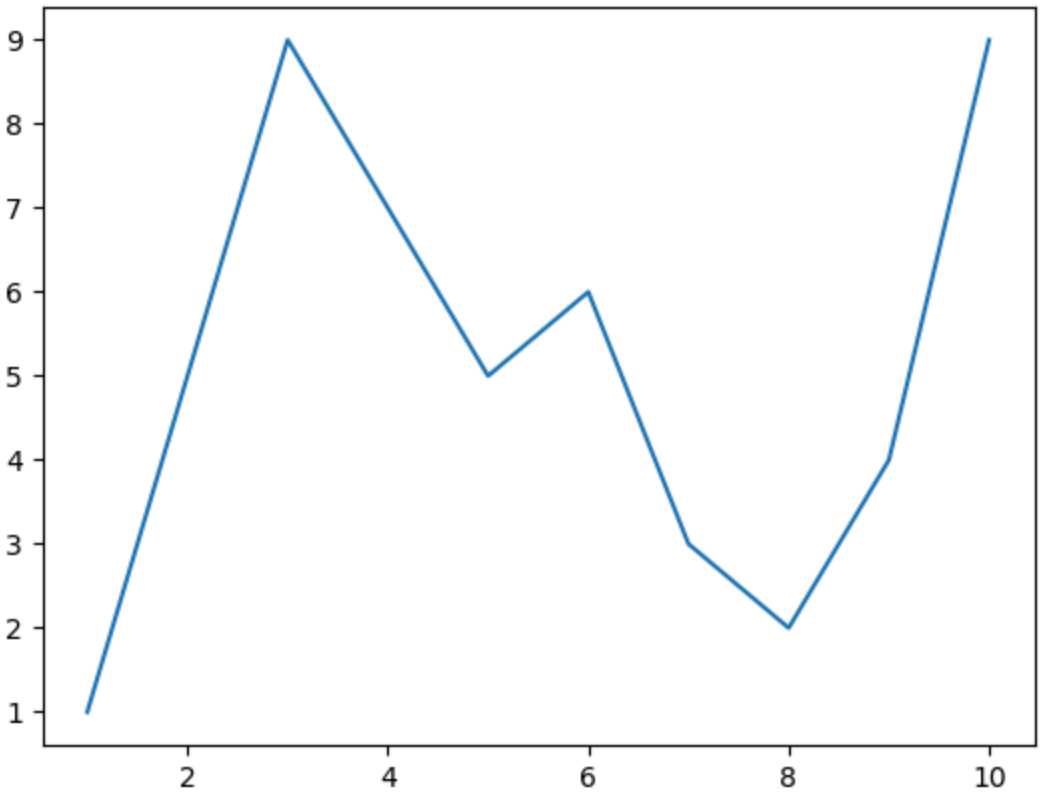
Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

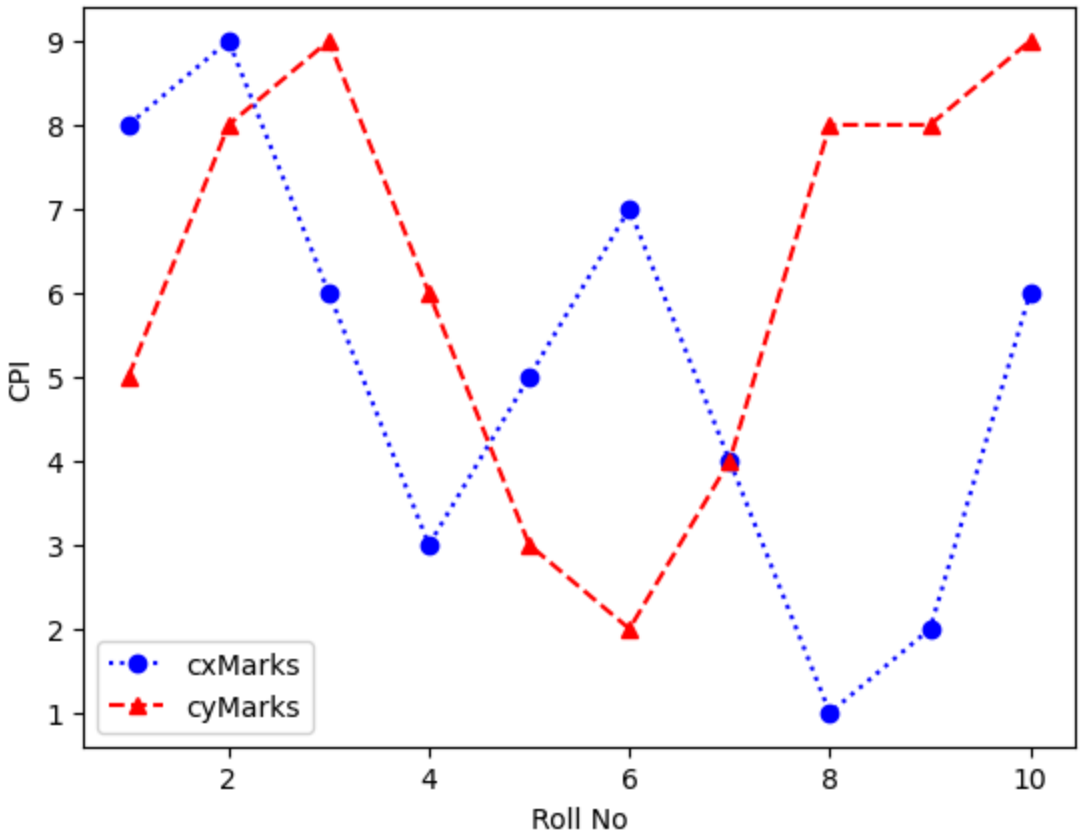
In [ ]:
```
#import matplotlib below
```

In [10]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

# write a code to display the line chart of above x & y

import matplotlib.pyplot as plt
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
plt.plot(x,y)
plt.show()
```

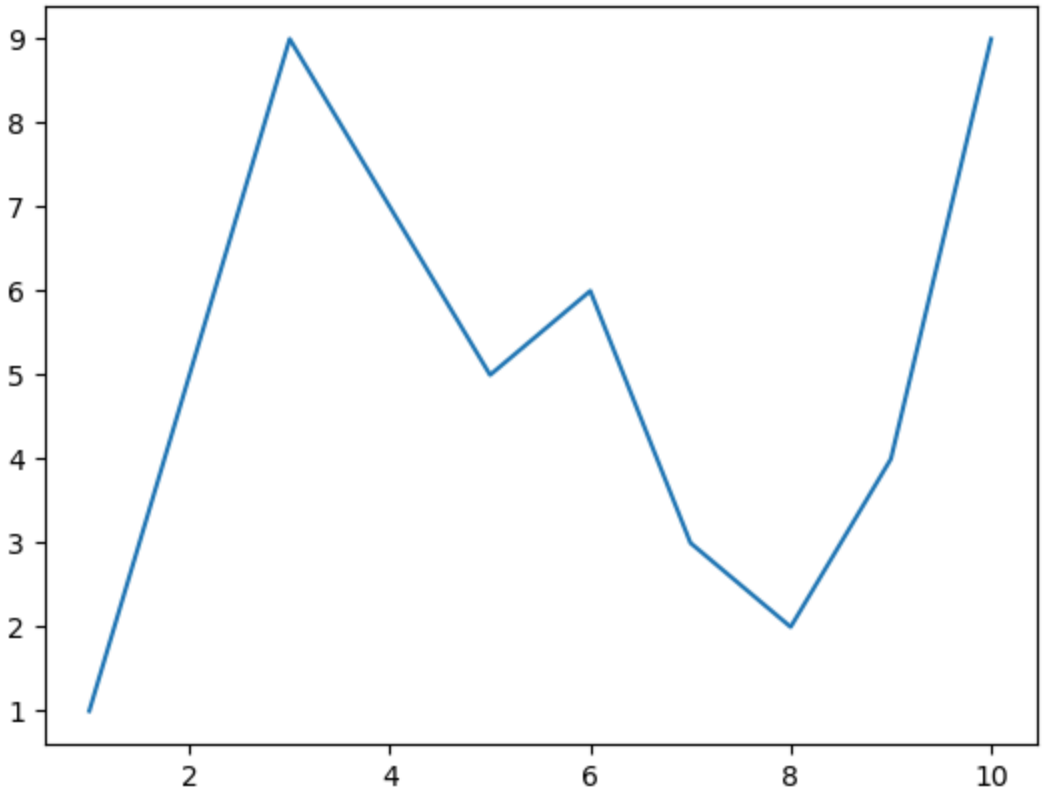


In [11]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

# write a code to display two lines in a line chart (data given above)
```

In [12]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks,label="cxMarks",color="b",marker="o",linestyle="dotted")
plt.plot(x,cyMarks,label="cyMarks",color="r",marker="^",linestyle="dashed")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.show()
# write a code to generate below graph
```

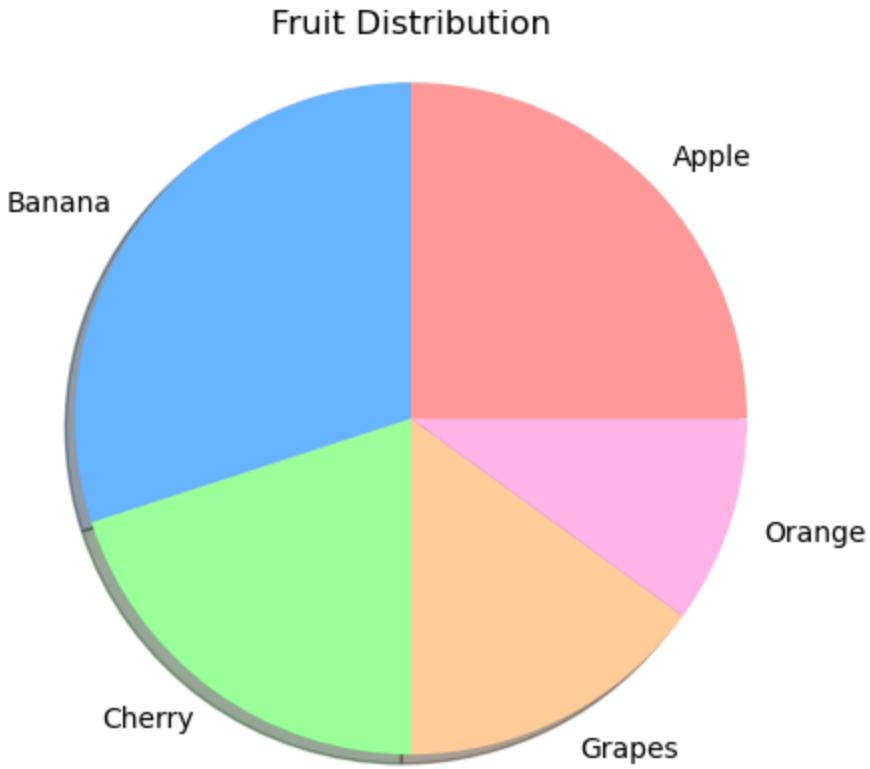


In [7]:



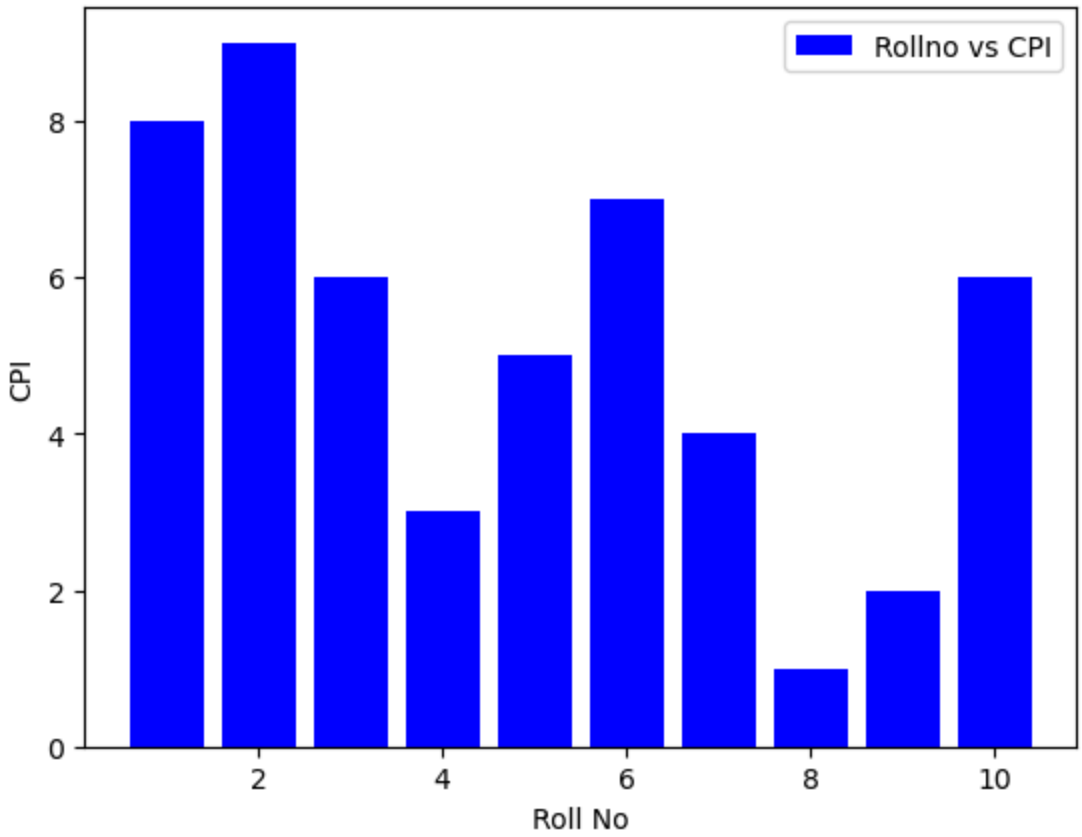
04) WAP to demonstrate the use of Pie chart.

In [13]:
```
labels = ['Apple', 'Banana', 'Cherry', 'Grapes', 'Orange']
sizes = [25, 30, 20, 15, 10]
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#ffb3e6']

plt.pie(sizes, labels=labels, colors=colors, shadow=True)
plt.axis('equal')
plt.title('Fruit Distribution')
plt.show()
```



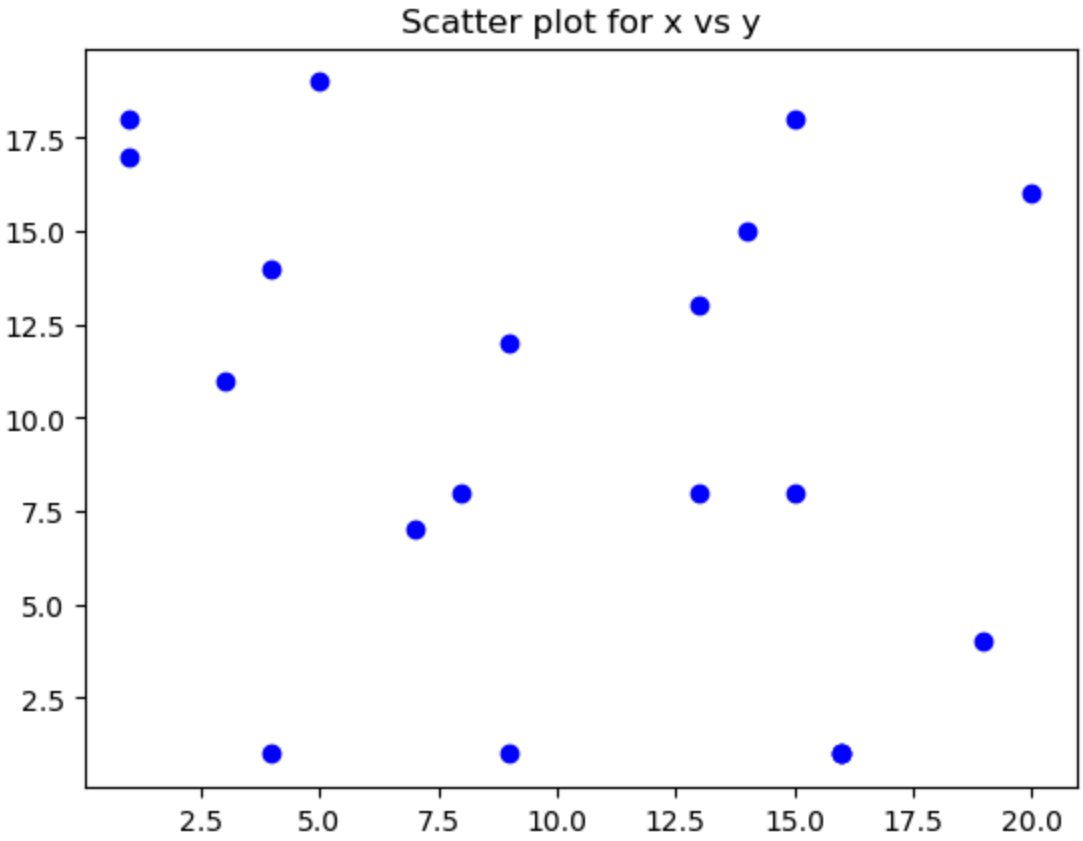
05) WAP to demonstrate the use of Bar chart.

In [14]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
plt.bar(x,cxMarks,label="Rollno vs CPI",color="b")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.show()
```



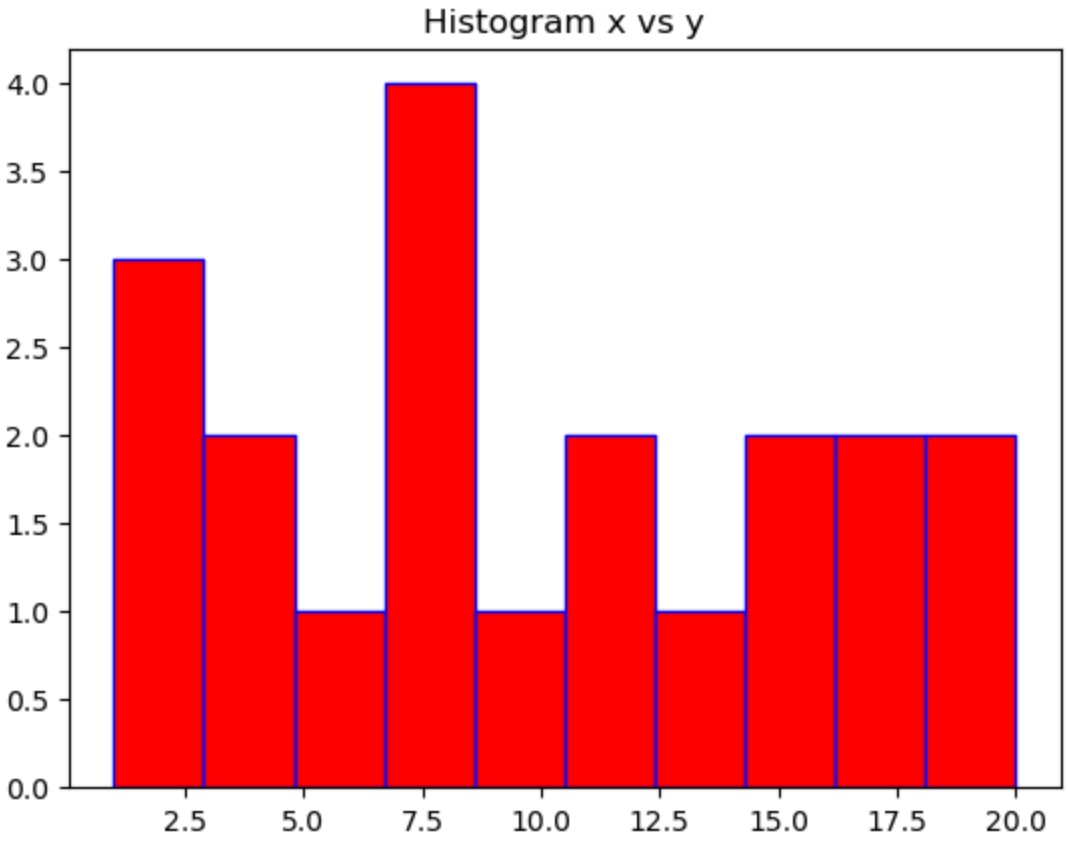
06) WAP to demonstrate the use of Scatter Plot.

In [15]:
```
import random as r
r.seed(1)
x = [r.randint(1,20) for i in range(20)]
y = [r.randint(1,20) for i in range(20)]
plt.scatter(x,y,color="b")
plt.title("Scatter plot for x vs y")
plt.show()
```



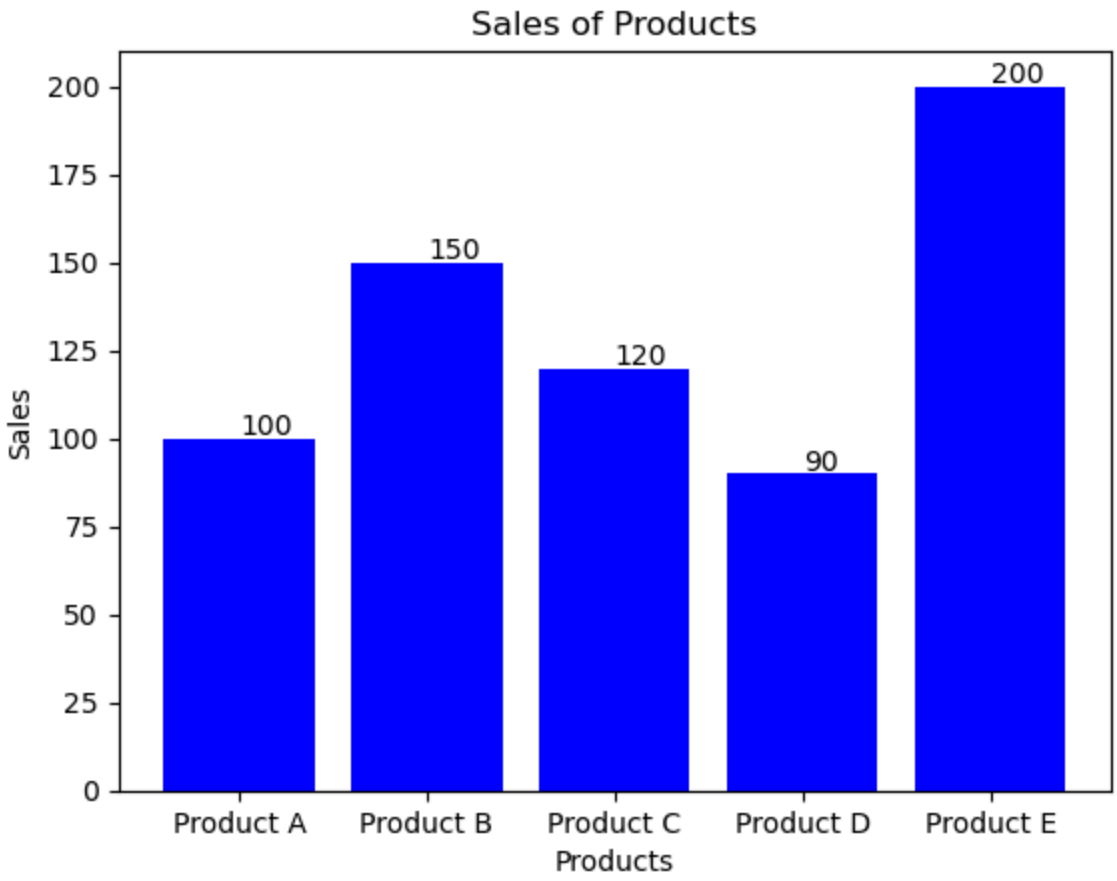
07) WAP to demonstrate the use of Histogram.

In [16]:
```
r.seed(5)
data = [r.randint(1,20) for i in range(20)]
plt.hist(data,edgecolor="b",color="r")
plt.title("Histogram x vs y")
plt.show()
```



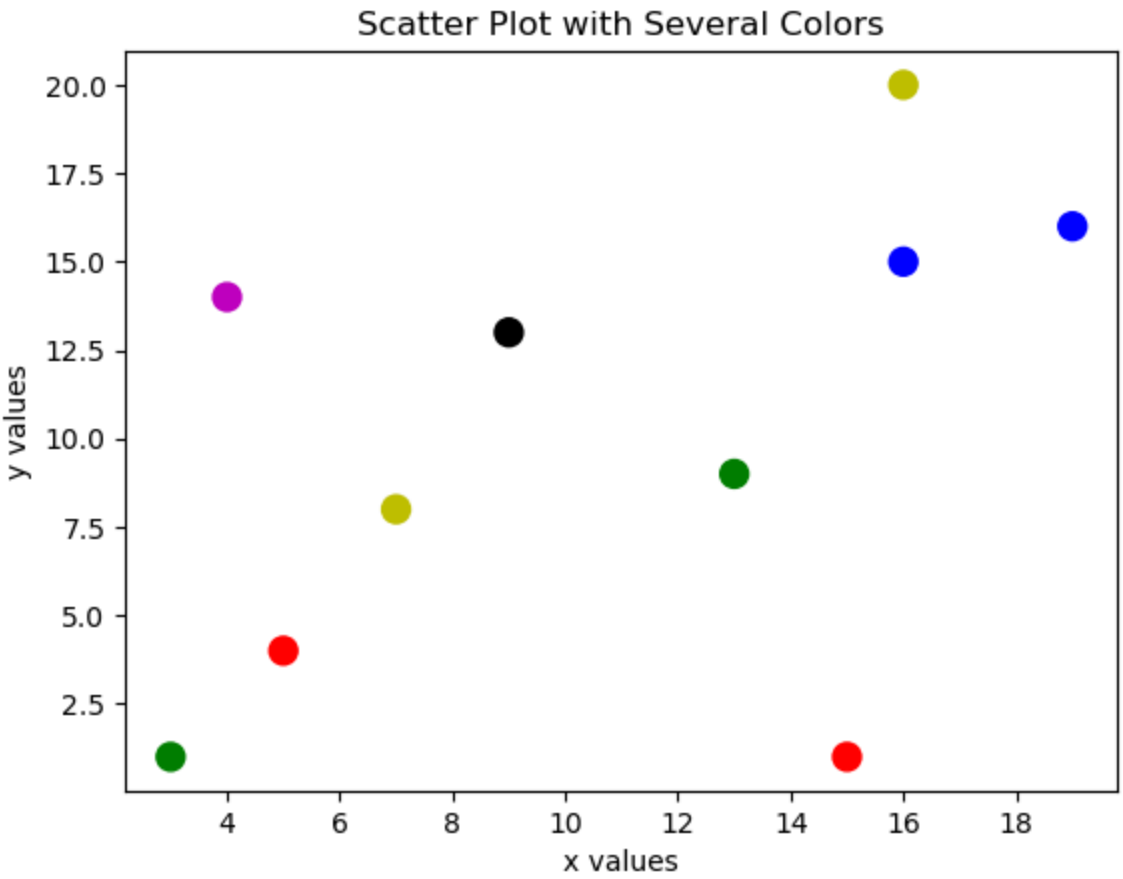
08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [17]:
```
products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
sales = [100, 150, 120, 90, 200]
plt.bar(products, sales, color='b')
for i in range(len(products)):
    plt.text(i, sales[i]+1,str(sales[i]))
plt.title('Sales of Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()
```



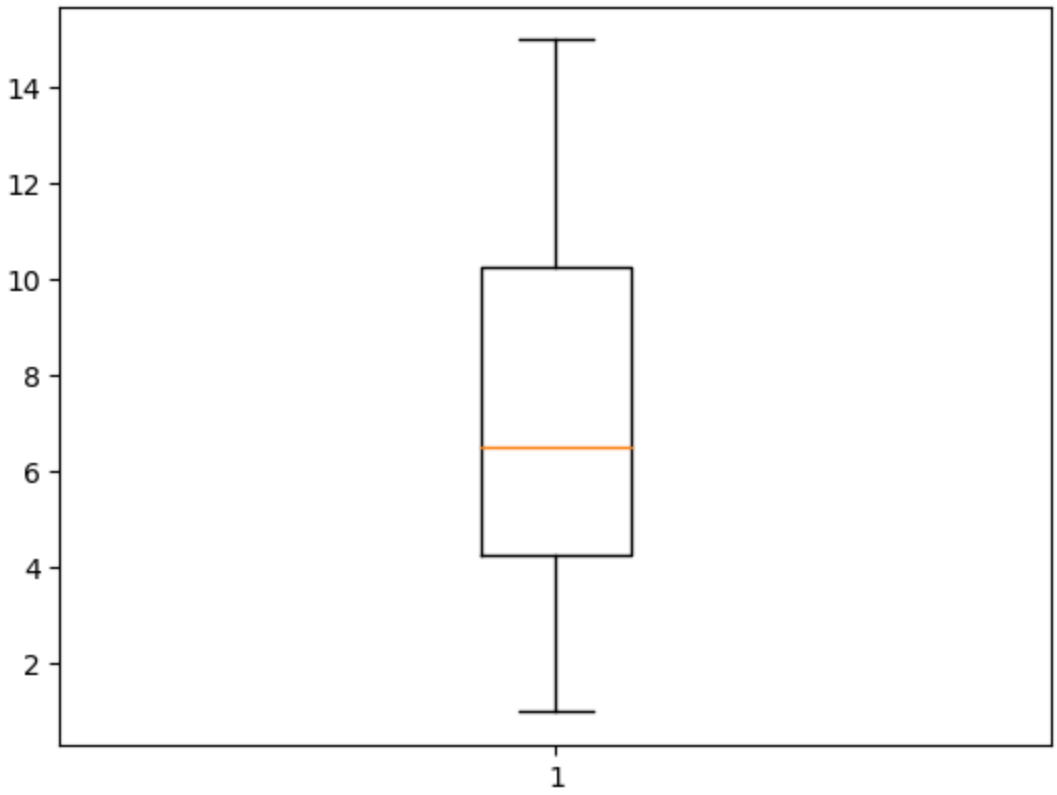
09) WAP create a Scatter Plot with several colors in Matplotlib?

In [18]:
```
r.seed(1)
x = [r.randint(1,20) for i in range(10)]
y = [r.randint(1,20) for i in range(10)]
colors = ['r','b','g','k','m','y','r','b','g','y']

plt.scatter(x, y, color=colors, s=100)
plt.title('Scatter Plot with Several Colors')
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```



10) WAP to create a Box Plot.

In [21]:
```
data=[r.randint(1,15) for i in range(10)]
plt.boxplot(data)
plt.show()
```


14
12
10
8
6
4
2
1

Python Programming - 2301CS404

Lab - 13_1

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

OOP

01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

In [7]:
```
class Student:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade

obj=Student("Malay",19,"A")

print(obj.name)
print(obj.age)
print(obj.grade)
```

```
Malay
19
A
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and

DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

In [23]:
```
class Bank_Account:
    def __init__(self):
        pass

    def GetAccountDetails(self):
        self.Account_no=int(input("Enter Account_no:"))
        self.User_Name=input("Enter User_Name:")
        self.Email=input("Enter Email:")
        self.Account_Type=input("Enter Account_Type:")
        self.Account_Balance=int(input("Enter Account_Balance:"))

    def DisplayAccountDetails(self):
        print(self.Account_no)
        print(self.User_Name)
        print(self.Email)
        print(self.Account_Type)
        print(self.Account_Balance)

obj=Bank_Account()
obj.GetAccountDetails()
obj.DisplayAccountDetails()
```

```
Enter Account_no: 213210
Enter User_Name: Malay
Enter Email: panaramalay@gmail.com
Enter Account_Type: Savings
Enter Account_Balance: 1000000000000000
213210
Malay
panaramalay@gmail.com
Savings
1000000000000000
```

03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

In [36]:
```
import math
class Circle:
    def area(self,r):
        print(f"Area:{math.pi*r*r}")

    def perimeter(self,r):
        print(f"Perimeter:{2*math.pi*r}")

obj=Circle()
obj.area(7)
obj.perimeter(7)
```

```
Area:153.93804002589985
Perimeter:43.982297150257104
```

04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

In [15]:
```
class Employee:
    def __init__(self,name,age,salary):
        self.name=name
        self.age=age
        self.salary=salary

    def updatedetails(self,name,age,salary):
        if name:
            self.name=name
        if age:
            self.age=age
        if salary:
            self.salary=salary

    def displaydetails(self):
        print("Employee Information:")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Salary: {self.salary}")

obj=Employee("Malay",20,555500000)
obj.displaydetails()
obj.updatedetails("Malay",19,1000000)
obj.displaydetails()
```

```
Employee Information:
Name: Malay
Age: 20
Salary: 555500000
Employee Information:
Name: Malay
Age: 19
Salary: 1000000
```

05) Create a bank account class with methods to deposit, withdraw, and check balance.

In [28]:
```
class Bank_Account:
    def __init__(self,acc_no,balance):
        self.acc_no=acc_no
        self.balance=balance

    def deposit(self,amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited Rs.{amount}. Current balance: Rs.{self.balance}")
        else:
            print("Deposit amount must be positive.")
```

```
    def withdraw(self,amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew Rs.{amount}. Current balance: Rs.{self.balance}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def check_balance(self):
        print(f"Balance:Rs.{self.balance}")

obj=Bank_Account(151322,10000000)
obj.deposit(1000)
obj.withdraw(123)
obj.check_balance()
```

```
Deposited Rs.1000. Current balance: Rs.10001000
Withdrew Rs.123. Current balance: Rs.10000877
Balance:Rs.10000877
```

06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

In [38]:

```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} {item_name}(s) to inventory.")

    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if self.items[item_name]['quantity'] >= quantity:
                self.items[item_name]['quantity'] -= quantity
                print(f"Removed {quantity} {item_name}(s) from inventory.")
            else:
                print(f"Not enough {item_name} in inventory to remove.")
        else:
            print(f"{item_name} not found in inventory.")

    def update_price(self, item_name, new_price):
        if item_name in self.items:
            self.items[item_name]['price'] = new_price
            print(f"Updated price of {item_name} to ${new_price}.")
        else:
            print(f"{item_name} not found in inventory.")

    def display_inventory(self):
```

```
        if not self.items:
            print("Inventory is empty.")
        else:
            print("Inventory Details:")
            for item_name, details in self.items.items():
                print(f"{item_name}: Price - ${details['price']}, Quantity - {detai

inventory = Inventory()
inventory.add_item("Laptop", 1200, 10)
inventory.add_item("Phone", 800, 15)
inventory.display_inventory()
inventory.remove_item("Laptop", 5)
inventory.update_price("Phone", 850)
inventory.display_inventory()
inventory.remove_item("Laptop", 6)
inventory.update_price("Tablet", 300)
```

```
Added 10 Laptop(s) to inventory.
Added 15 Phone(s) to inventory.
Inventory Details:
Laptop: Price - $1200, Quantity - 10
Phone: Price - $800, Quantity - 15
Removed 5 Laptop(s) from inventory.
Updated price of Phone to $850.
Inventory Details:
Laptop: Price - $1200, Quantity - 5
Phone: Price - $850, Quantity - 15
Not enough Laptop in inventory to remove.
Tablet not found in inventory.
```

07) Create a Class with instance attributes of your choice.

In [36]:

```
class Car:
    def __init__(self, make, model, year, color):
        self.make = make
        self.model = model
        self.year = year
        self.color = color

    def display_car_info(self):
        print(f"Car Information:")
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
        print(f"Color: {self.color}")

car1 = Car("Toyota", "Century", 2021, "Blue")
car1.display_car_info()
car2 = Car("Honda", "Civic", 2020, "Red")
car2.display_car_info()
```

```
Car Information:
Make: Toyota
Model: Century
Year: 2021
Color: Blue
Car Information:
Make: Honda
Model: Civic
Year: 2020
Color: Red
```

08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

In [40]:
```python
class StudentKit:
    principal = "Mr ABC"

    def __init__(self, student_name):
        self.student_name = student_name
        self.attendance = 0

    def record_attendance(self, days_present):
        self.attendance = days_present
        print(f"Attendance recorded: {self.attendance} days")

    def generate_certificate(self):
        if self.attendance >= 75:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Passed (Attendance is sufficient)")
        else:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Failed (Attendance is insufficient)")

student_name = input("Enter the student's name: ")
student = StudentKit(student_name)
days_present = int(input("Enter the number of days present in class: "))
student.record_attendance(days_present)
student.generate_certificate()
```

```
Enter the student's name:  Malay
Enter the number of days present in class:  262
Attendance recorded: 262 days
Certificate of Attendance

Principal: Mr ABC
Student: Malay
Days Present: 262
Status: Passed (Attendance is sufficient)
```

09) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

In [42]:
```python
class Time:
    def __init__(self, hour=0, minute=0):
        self.hour = hour
        self.minute = minute

    def add_time(self, other):
        total_minutes = (self.hour * 60 + self.minute) + (other.hour * 60 + other.m
        total_hour = total_minutes // 60
        total_minute = total_minutes % 60
        return Time(total_hour, total_minute)

    def display_time(self):
        print(f"{self.hour} hour(s) and {self.minute} minute(s)")

time1 = Time(2, 45)
time2 = Time(3, 30)
total_time = time1.add_time(time2)
print("Time 1:")
time1.display_time()
print("Time 2:")
time2.display_time()
print("Total Time after adding:")
total_time.display_time()
```

```
Time 1:
2 hour(s) and 45 minute(s)
Time 2:
3 hour(s) and 30 minute(s)
Total Time after adding:
6 hour(s) and 15 minute(s)
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13_2

Name : PARMAR VISHAL

Roll no. : 139

Enrollment no. : 23010101197

Continued..

10) Calculate area of a ractangle using object as an argument to a method.

In [6]:
```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def calculate_area(rectangle):
        return rectangle.width * rectangle.height


rect = Rectangle(5, 10)
area = calculate_area(rect)
print(f"The area of the rectangle is: {area}")
```

```
The area of the rectangle is: 50
```

11) Calculate the area of a square.

Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

In [5]:
```
class Square:
    def __init__(self, side_length):
        self.side_length = side_length

    def area(self):
        area_value = self.side_length ** 2
        self.output(area_value)

    def output(self, area_value):
        print(f"The area of the square with side length {self.side_length} is {area

# Example usage
square = Square(5)
square.area()
```

```
The area of the square with side length 5 is 25
```

12) Calculate the area of a rectangle.

Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

Also define a class method that compares the two sides of reactangle. An object is instantiated only if the two sides are different; otherwise a message should be displayed : THIS IS SQUARE.

In [7]:
```
class Rectangle:
    def __init__(self, length, width):
        if length == width:
            print("THIS IS SQUARE.")
        else:
            self.length = length
            self.width = width

    def area(self):
        if hasattr(self, 'length') and hasattr(self, 'width'):
            area = self.length * self.width
            self.output(area)

    def output(self, calculated_area):
        print(f"The area of the rectangle is: {calculated_area}")

    @classmethod
```

```
    def compare_sides(cls, length, width):
        if length == width:
            return "THIS IS SQUARE."
        else:
            return "Length and width are different."

# Example usage
rect1 = Rectangle(10, 5)
rect1.area()

rect2 = Rectangle(4, 4)  # This will print "THIS IS SQUARE."
```

```
The area of the rectangle is: 50
THIS IS SQUARE.
```

13) Define a class Square having a private attribute "side".

Implement get_side and set_side methods to accees the private attribute from outside of the class.

In [12]:
```
class Square:
    def __init__(self, side):
        self.__side = side  # private attribute

    def get_side(self):
        return self.__side

    def set_side(self, side):
        self.__side = side

# Example usage:
square = Square(5)
print(square.get_side())  # Output: 5
square.set_side(10)
print(square.get_side())  # Output: 10
```

```
5
10
```

14) Create a class Profit that has a method named getProfit that accepts profit from the user.

Create a class Loss that has a method named getLoss that accepts loss from the user.

Create a class BalanceSheet that inherits from both classes Profit and Loss and calculates the balanace. It has two methods getBalance() and printBalance().

In [13]:
```
class Profit:
    def getProfit(self):
        self.profit = float(input("Enter profit amount: "))
```

```
        return self.profit


class Loss:
    def getLoss(self):
        self.loss = float(input("Enter loss amount: "))
        return self.loss


class BalanceSheet(Profit, Loss):
    def getBalance(self):
        self.total_profit = self.getProfit()
        self.total_loss = self.getLoss()
        self.balance = self.total_profit - self.total_loss
        return self.balance

    def printBalance(self):
        print(f"Total Profit: {self.total_profit}")
        print(f"Total Loss: {self.total_loss}")
        print(f"Balance: {self.balance}")


# Example usage
if __name__ == "__main__":
    balance_sheet = BalanceSheet()
    balance_sheet.getBalance()
    balance_sheet.printBalance()
```

```
Enter profit amount: 10000
Enter loss amount: 2000
Total Profit: 10000.0
Total Loss: 2000.0
Balance: 8000.0
```

15) WAP to demonstrate all types of inheritance.

In [14]:

```
# Single Inheritance
class Animal:
    def speak(self):
        return "Animal speaks"

class Dog(Animal):
    def bark(self):
        return "Dog barks"

# Multiple Inheritance
class Father:
    def skills(self):
        return "Gardening, Painting"

class Mother:
    def skills(self):
        return "Cooking, Dancing"

class Child(Father, Mother):
```

```
    def talents(self):
        return "Singing"

# Multilevel Inheritance
class Grandparent:
    def wisdom(self):
        return "Wisdom from Grandparent"

class Parent(Grandparent):
    def experience(self):
        return "Experience from Parent"

class ChildMultilevel(Parent):
    def youthfulness(self):
        return "Youthfulness from Child"

# Hierarchical Inheritance
class Base:
    def base_method(self):
        return "Base method called"

class Derived1(Base):
    def derived1_method(self):
        return "Derived1 method called"

class Derived2(Base):
    def derived2_method(self):
        return "Derived2 method called"

# Hybrid Inheritance
class BaseHybrid:
    def base_hybrid_method(self):
        return "Base method in Hybrid"

class DerivedA(BaseHybrid):
    pass

class DerivedB(BaseHybrid):
    pass

class MoreComplex(DerivedA, DerivedB):
    pass

# Demonstration
# Single Inheritance
dog = Dog()
print(dog.speak())
print(dog.bark())

# Multiple Inheritance
child = Child()
print(child.skills())  # From Father
print(child.talents())
print(child.skills())  # From Mother

# Multilevel Inheritance
```

```
child_ml = ChildMultilevel()
print(child_ml.wisdom())
print(child_ml.experience())
print(child_ml.youthfulness())

# Hierarchical Inheritance
derived1 = Derived1()
derived2 = Derived2()
print(derived1.base_method())
print(derived1.derived1_method())
print(derived2.base_method())
print(derived2.derived2_method())

# Hybrid Inheritance
hybrid_obj = MoreComplex()
print(hybrid_obj.base_hybrid_method())
```

```
Animal speaks
Dog barks
Gardening, Painting
Singing
Gardening, Painting
Wisdom from Grandparent
Experience from Parent
Youthfulness from Child
Base method called
Derived1 method called
Base method called
Derived2 method called
Base method in Hybrid
```

16) Create a Person class with a constructor that takes two arguments name and age.

Create a child class Employee that inherits from Person and adds a new attribute salary.

Override the **init** method in Employee to call the parent class's **init** method using the super() and then initialize the salary attribute.

In [15]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Employee(Person):
    def __init__(self, name, age, salary):
        super().__init__(name, age)
        self.salary = salary

# Example of creating an instance of Employee
employee = Employee("Alice", 30, 50000)
```

```
print(employee.name)  # Output: Alice
print(employee.age)   # Output: 30
print(employee.salary)  # Output: 50000
```

```
Alice
30
50000
```

17) Create a Shape class with a draw method that is not implemented.

Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors.

Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

In [16]:
```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def draw(self):
        pass

class Rectangle(Shape):
    def draw(self):
        print("Drawing a Rectangle")

class Circle(Shape):
    def draw(self):
        print("Drawing a Circle")

class Triangle(Shape):
    def draw(self):
        print("Drawing a Triangle")

# Creating a list of Shape objects
shapes = [Rectangle(), Circle(), Triangle()]

# Iterating through the list and calling draw method on each object
for shape in shapes:
    shape.draw()
```

```
Drawing a Rectangle
Drawing a Circle
Drawing a Triangle
```