(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Prince Bhanderi | 23010101022 | 26-11-2024 ¶

01) WAP to print “Hello World”

In [1]:
```
print("Hello World")
```

```
Hello World
```

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

In [4]:
```
a=10
b=20
print(a+b)
a=int(input("Enter number 1: "))
b=int(input("Enter number 2: "))
print(a+b)
```

```
30
Enter number 1: 66
Enter number 2: 56
122
```

03) WAP to check the type of the variable.

In [16]:
```
a=10
type(a)
```

Out[16]:
```
int
```

04) WAP to calculate simple interest.

In [23]:
```
p=float(input("Enter principle amount: "))
r=float(input("Enter rate of interest: "))
t=float(input("Enter amount of time: "))

SI=(p*r*t)/100

print("Simple Interest=",SI)
print("Total amount=",SI+p)
```

```
Enter principle amount: 500000
Enter rate of interest: 1000
Enter amount of time: 56
Simple Interest= 280000000.0
Total amount= 280500000.0
```

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

In [30]:
```
import math

r=float(input("Enter radious of circle: "))

c=2*math.pi*r
print("Perimeter of circle is " , "%.2f" %c)
a=math.pi*r*r
print("Area of circle is " , "%.2f" %a)
```

```
Enter radious of circle: 10
Perimeter of circle is  62.83
Area of circle is  314.16
```

06) WAP to calculate area of a triangle.

In [31]:
```
b=float(input("Enter base of triangle: "))
h=float(input("Enter height of triangle: "))

a=(b*h)/2
print("Area of circle is " , "%.2f" %a)
```

```
Enter base of triangle: 25
Enter height of triangle: 30
Area of circle is  375.00
```

07) WAP to compute quotient and remainder.

In [34]:
```
a=float(input("Enter dividend: "))
b=float(input("Enter divider: "))

q=a//b
#If you do single slash than answer come in point
r=a%b

print("Quotient=",q)
print("Reminder=",r)
```

```
Enter dividend: 5
Enter divider: 2
Quotient= 2.0
Reminder= 1.0
```

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

In [39]:
```
c=float(input("Enter Degree: "))
f=float(input("Enter Farenheit: "))

ctof=(c*(9/5))+32
ftoc=(f-32)*5/9

print("Celcius to Farenheit",ctof)
print("Farenheit to Celcius",ftoc)
```

```
Enter Degree: -40
Enter Farenheit: -40
Celcius to Farenheit -40.0
Farenheit to Celcius -40.0
```

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

In [42]:
```
x1=int(input("Enter x cordinate of point A: "))
y1=int(input("Enter y cordinate of point A: "))
x2=int(input("Enter x cordinate of point B: "))
y2=int(input("Enter y cordinate of point B: "))

distance=math.sqrt((x2-x1)**2+(y2-y1)**2)
print("Distance between 2 point is","%.4f"%distance)
```

```
Enter x cordinate of point A: 1
Enter y cordinate of point A: 1
Enter x cordinate of point B: 2
Enter y cordinate of point B: 2
Distance between 2 point is 1.4142
```

10) WAP to print sum of n natural numbers.

In [48]:
```
n=int(input("Enter number upto where you want to sum: "))
# sum=0
# for i in range(n+1):
#     sum+=i
sum=(n*(n+1))/2
print(sum)
```

```
Enter number upto where you want to sum: 10
55.0
```

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

In [50]:
```
n=int(input("Enter number upto where you want to sum: "))

sqr=0
for i in range(n+1):
    sqr+=i**2

print(sqr)
```

```
Enter number upto where you want to sum: 20
2870
```

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

In [1]:
```
first_name = input()
last_name = input()
print(first_name + " " + last_name)
```

```
Prince
Bhanderi
Prince Bhanderi
```

13) WAP to swap two numbers.

In [2]:
```
num1 = float(input())
num2 = float(input())
num1, num2 = num2, num1
print(num1, num2)
```

```
12
24
24.0 12.0
```

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

In [3]:
```
km = float(input())
print(km * 1000)
print(km * 3280.84)
print(km * 39370.1)
print(km * 100000)
```

```
1000
1000000.0
3280840.0
39370100.0
100000000.0
```

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

In [4]:
```
day = int(input())
month = int(input())
year = int(input())
print(f"{day:02d}-{month:02d}-{year % 100:02d}")
```

```
1
2
2004
01-02-04
```

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Prince Bhanderi | 23010101022 | 03-12-2024 ¶

if..else..

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

In [4]:
```
n = int(input("Enter Number: "))
if n>=0 :
    print('Positive Number')
else:
    print('Negative Number')
```

```
Enter Number: -11
Negative Number
```

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

In [3]:
```
n = int(input("Enter Number: "))
if n%2==0 :
    print('Even Number')
else:
    print('Odd Number')
```

```
Enter Number: 11
Odd Number
```

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

In [10]:
```
n = int(input("Enter 1st Number: "))
m = int(input("Enter 2nd Number: "))

#Simple if

# if n>m :
#     print(n,'is Largest Number')
# else:
#     print(m,'is Largest Number')

#Ternary Operator

largest = n if n>m else m
print(largest,'is Largest Number'+)
```

```
Enter 1st Number: 5
Enter 2nd Number: 6
6
```

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

In [19]:
```
n = int(input("Enter 1st Number: "))
m = int(input("Enter 2nd Number: "))
o = int(input("Enter 3rd Number: "))

#Ternary Operator

largest = n if (n>m and n>o) else (m if m>o else o)
print(largest,'is Largest Number')
```

```
Enter 1st Number: 2
Enter 2nd Number: 3
Enter 3rd Number: 1
3 is Largest Number
```

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 [26]:
```
year = int(input("Enter Year: "))

if (year%4==0 and year%100!=0):
    print("Entered year is a leap year.")
else:
    if year%400==0 :
        print("Entered year is a leap year.")
    else:
        print("Entered year is not a leap year.")
```

```
Enter Year: 1200
Entered year is a leap year.
```

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

In [33]:
```
n = int(input("Enter Day Index: "))
a = (n%7)

match a:
    case 1:
        print("Sunday")
    case 2:
        print("Monday")
    case 3:
        print("Tuesday")
    case 4:
        print("Wednesday")
    case 5:
        print("Thursday")
    case 6:
        print("Friday")
    case 7:
        print("Saturday")
```

```
Enter Day Index: 1185
Monday
```

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

In [38]:
```
n = float(input("Enter 1st Number: "))
m = float(input("Enter 2nd Number: "))
o = input("Enter Operator(+,-,*,/): ")

match o:
    case '+':
        print('Sum of 2 number is ',n+m)
    case '-':
        print('Substraction of 2 number is ',n-m)
    case '*':
        print('Multiplication of 2 number is ',n*m)
    case '/':
        print('Division of 2 number is ',n/m)
    case _:
        print('Invalid Operator')
```

```
Enter 1st Number: 56
Enter 2nd Number: 235
Enter Operator(+,-,*,/): /
Division of 2 number is  0.23829787234042554
```

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 [68]:
```
marks1 = float(input("Enter marks for Subject 1: "))
marks2 = float(input("Enter marks for Subject 2: "))
marks3 = float(input("Enter marks for Subject 3: "))
marks4 = float(input("Enter marks for Subject 4: "))
marks5 = float(input("Enter marks for Subject 5: "))

total_marks = marks1 + marks2 + marks3 + marks4 + marks5
percentage = total_marks / 5

print('percentage',percentage,'%')

if percentage > 70:
    print("Distinction")
elif percentage >= 60:
    print("First Class")
elif percentage >= 45:
    print("Second Class")
elif percentage >= 35:
    print("Pass Class")
else:
    print("Fail")
```

```
Enter marks for Subject 1: 25
Enter marks for Subject 2: 64
Enter marks for Subject 3: 51
Enter marks for Subject 4: 42
Enter marks for Subject 5: 45
percentage 45.4 %
Second Class
```

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 [71]:
```
a = float(input("Enter the length of side 1: "))
b = float(input("Enter the length of side 2: "))
c = float(input("Enter the length of side 3: "))

if a + b > c and b + c > a and c + a > b:

    if a == b == c:
        print("The triangle is Equilateral.")
    elif a == b or b == c or c == a:
        print("The triangle is Isosceles.")
    else:
        print("The triangle is Scalene.")

    if a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or c**2 + a**2 == b**2:
        print("The triangle is Right-Angled.")
else:
    print("The entered sides do not form a valid triangle.")
```

```
Enter the length of side 1: 15
Enter the length of side 2: 15
Enter the length of side 3: 18
The triangle is Isosceles.
```

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

In [73]:
```
n = int(input("Enter 1st Number: "))
m = int(input("Enter 2nd Number: "))
o = int(input("Enter 3rd Number: "))

#Ternary Operator

if (n>m and n>o):
    if m>o :
        print(m,'is 2nd Largest Number')
    else:
        print(o,'is 2nd Largest Number')
elif (m>o):
    if n>o :
        print(n,'is 2nd Largest Number')
    else:
        print(o,'is 2nd Largest Number')
else:
    if m>n :
        print(m,'is 2nd Largest Number')
    else:
        print(n,'is 2nd Largest Number')
```

```
Enter 1st Number: 3
Enter 2nd Number: 1
Enter 3rd Number: 2
2 is 2nd Largest Number
```

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 [74]:
```
units = float(input("Enter the number of units consumed: "))

if units <= 50:
    bill = units * 2.60
elif units <= 100:
    bill = 50 * 2.60 + (units - 50) * 3.25
elif units <= 200:
    bill = 50 * 2.60 + 50 * 3.25 + (units - 100) * 5.26
else:
    bill = 50 * 2.60 + 50 * 3.25 + 100 * 5.26 + (units - 200) * 8.45

print(f"Your electricity bill is: Rs. {bill:.2f}")
```

```
Enter the number of units consumed: 200
Your electricity bill is: Rs. 818.50
```

In [ ]:

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Prince Bhanderi | 23010101022 | 10-12-2024

for and while loop

01) WAP to print 1 to 10.

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

```
Enter Number: 15
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

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

In [5]:
```
n = int(input("Enter Number: "))
for i in range(1,n+1):
    if(i%2==1):
        print(i)
```

```
Enter Number: 10
1
3
5
7
9
```

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

In [9]:
```
n = int(input("Enter 1st Number: "))
m = int(input("Enter 2nd Number: "))

for i in range(n,m+1):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
Enter 1st Number: 1
Enter 2nd Number: 30
2
4
8
10
14
16
20
22
26
28
```

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

In [12]:
```
n = int(input("Enter Number: "))
sum=0
for i in range(1,n+1):
    sum+=i
print(sum)
```

```
Enter Number: 15
120
```

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

In [15]:
```
n = int(input("Enter Number: "))
sum=0
for i in range(1,n+1):
    sum+=(i*i)
print(sum)
```

```
Enter Number: 15
1240
```

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

In [16]:
```
n = int(input("Enter Number: "))
sum=0
for i in range(1,n+1):
    if(i%2==0):
        sum-=i
    else:
        sum+=i
print(sum)
```

```
Enter Number: 3
2
```

08) WAP to print multiplication table of given number.

In [19]:
```
n = int(input("Enter Number: "))
mult=0
for i in range(1,11):
    mult=n*i
    print(n,'x',i,'=',mult)
```

```
Enter Number: 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

09) WAP to find factorial of the given number.

In [20]:
```
n = int(input("Enter Number: "))
fac=1
for i in range(1,n):
    fac*=i

print('Factorial of',n,'is',fac)
```

```
Enter Number: 5
Factorial of 5 is 24
```

10) WAP to find factors of the given number.

In [3]:
```
n = int(input("Enter Number: "))

for i in range(1,n):
    if(n%i==0):
        print('Factor of',n,'is',i)
```

```
Enter Number: 30
Factor of 30 is 1
Factor of 30 is 2
Factor of 30 is 3
Factor of 30 is 5
Factor of 30 is 6
Factor of 30 is 10
Factor of 30 is 15
```

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

In [15]:
```
n = int(input("Enter Number: "))
flag=0
for i in range(2,int((n/2)+1)):
    if(n%i==0):
        flag+=1
        break
if(flag==0):
    print('prime')
else:
    print('not prime')
```

```
Enter Number: 10
not prime
```

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

In [12]:
```
n = int(input("Enter a number: "))

sum_digits = sum(int(digit) for digit in str(n))

print("Sum of digits:", sum_digits)
```

```
Enter a number: 123
Sum of digits: 6
```

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

In [35]:
```
n = int(input("Enter a number: "))
temp=n
digit=0
while(n>0):
    a=n%10
    digit=(digit*10)+a
    n//=10

if(temp==digit):
    print('Entered number is palindrome')
else:
    print('Entered number is not palindrome')
```

```
Enter a number: 123
321
Entered number is not palindrome
```

14) WAP to print GCD of given two numbers.

In [38]:
```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

while num2 != 0:
    num1, num2 = num2, num1 % num2

print("The GCD is:", num1)
```

```
Enter the first number: 10
Enter the second number: 5
The GCD is: 5
```

Pattern

In [40]:
```
size = 9

for i in range(size):
    row = ""
    for j in range(size):

        row += str(min(i, j, size - i - 1, size - j - 1) + 1)
    print(row)
```

```
111111111
122222221
123333321
123444321
123454321
123444321
123333321
122222221
111111111
```

In [ ]:
```

```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 4

Prince Bhanderi | 23010101022 | 17-12-2024

String

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

In [2]:
```
string=input("Enter a String:");
reverse=string[::-1];


if(string==reverse):

    print(f"{string} is palindrome!!");
else:
    print(f"{string} is not palindrome!!");
```

```
Enter a String:abccba
abccba is palindrome!!
```

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

In [4]:
```
string=input("Enter a String:");
reverse=string[::-1];

print(f"Reversed string:{reverse}");
```

```
Enter a String:lknkljn
Reversed string:njlknkl
```

03) WAP to remove ith character from given string.

In [5]:
```
string = input('Enter a string: ')
i = int(input('Enter the position to remove a character: '))

if i < 0 or i >= len(string):
    print('Index out of range');
else:
    string = string[:i] + string[i+1:]
    print('Modified string:', string);
```

```
Enter a string: jksfgkjldsn
Enter the position to remove a character: 2
Modified string: jkfgkjldsn
```

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

In [6]:
```
string = input('Enter the string: ')
length = 0

for i in string:
    length += 1

print(f'The length of the string is {length}')
```

```
Enter the string: Malay
The length of the string is 5
```

05) WAP to print even length word in string.

In [19]:
```
string = input('Enter the string: ')
words = string.split()

for word in words:
    if len(word) % 2 == 0:
        print(word);
```

```
Enter the string: guyihu
guyihu
```

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

In [24]:
```
string=input("Enter a string:");
string=string.lower();
count=0;

for i in string:
    if(i=='a'or i=='e' or i=='i' or i=='o' or i=='u'):
        count+=1;
print(f"The number of vowel is {count}");
```

```
Enter a string:AJPOOR
The number of vowel is 3
```

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

In [33]:
```
string = input("Enter a string: ")
words = string.split()
result = []

for word in words:
    if len(word) > 1:
        word = word[0].upper() + word[1:-1] + word[-1].upper()
    else:
        word = word.upper()
    result.append(word)

output = ' '.join(result)
print("Modified string:", output);
```

```
Enter a string: try
Modified string: TrY
```

08) WAP to convert given array to string.

In [30]:
```
array = ['Hello', 'World', 'From', 'Python']
result = ' '.join(array)

print(f"Converted string:{result}");
```

```
Converted string:Hello World From Python
```

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

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

In [37]:
```
password=input("Enter password:");
confirm_password=input("Enter confirm_password:");
if password==confirm_password:
    print("Password matched");
elif password.lower()==confirm_password.lower():
    print("Case's mistake")
else:
    print("Password unmatched");
```

```
Enter password:mnbvcxz
Enter confirm_password:mnbVcxz
Case's mistake
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [38]:
```
card_number = input("Enter the credit card number (with spaces): ")

if ''.join(card_number.split()).isnumeric() and len(card_number) == 19:
    parts = card_number.split()
    masked_parts = ['****'] * (len(parts) - 1) + [parts[-1]]
    masked_card = ' '.join(masked_parts)
    print("Masked card number:", masked_card);
else:
    print("Invalid card number");
```

```
Enter the credit card number (with spaces): 1234 5678 9012 3456
Masked card number: **** **** **** 3456
```

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

s1 = decimal and s2 = medical are Anagram

In [42]:
```
s1 = input("Enter the first string: ")
s2 = input("Enter the second string: ")

s1 = s1.replace(" ", "").lower()
s2 = s2.replace(" ", "").lower()

if sorted(s1) == sorted(s2):
    print(f'"{s1}" and "{s2}" are Anagrams.')
else:
    print(f'"{s1}" and "{s2}" are not Anagrams.')
```

```
Enter the first string: mug
Enter the second string: gum
"mug" and "gum" are Anagrams.
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [43]:
```
string = input("Enter the string: ")

lowercase = ''.join([char for char in string if char.islower()])
uppercase = ''.join([char for char in string if char.isupper()])
result = lowercase + uppercase

print("Rearranged string:", result);
```

```
Enter the string: EHlsarwiwhtwMV
Rearranged string: lsarwiwhtwEHMV
```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 5

Prince Bhanderi | 23010101022 | 24-12-2024 ¶

List

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

In [11]:
```
list = [10,20,30,40,50]


print(sum(list))
```

```
150
```

02) WAP to find largest element in a List.

In [12]:
```
list = [10,20,30,40,50]

print(max(list))
```

```
50
```

03) WAP to find the length of a List.

In [14]:
```
list = [10,20,30,40,50]
print(len(list))
```

```
5
```

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

In [20]:
```
list=[]

while(True):
    a=int(input("Enter an element : "))
    if(a==-1):
        break
    list.append(a)

list[0],list[-1]=list[-1],list[0]
print(list)
```

```
Enter an element : 1
Enter an element : 2
Enter an element : 3
Enter an element : 4
Enter an element : 5
Enter an element : -1
[5, 2, 3, 4, 1]
```

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

In [23]:
```
first = list[:len(list)//2]
second = list[len(list)//2:]

third=second+first
print(third)
#second method

#second.extend(first)
#print(second)
```

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

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

In [36]:
```
list1=[]
while(True):
    a=int(input("Enter an element : "))
    if(a==-1):
        break
    list1.append(a)
a1=int(input("Enter your index1 : "))
a2=int(input("Enter your index2 : "))

if((a1>=len(list1) or a1<(-(len(list1)))) and (a2>=len(list1) or a2<(-(len(list
    print("index is out of range")
list1[a1],list[a2]=list1[a2],list1[a1]
print[list1]
```

```
Enter an element : 1
Enter an element : 2
Enter an element : 3
Enter an element : -1
Enter your index1 : -3
Enter your index2 : -3
```

07) WAP to reverse the list entered by user.

In [39]:
```
list2=[]
while(True):
    a=int(input("Enter an element : "))
    if(a==-1):
        break
    list2.append(a)
list2.reverse()
print(list2)
```

```
Enter an element : 1
Enter an element : 2
Enter an element : 3
Enter an element : 4
Enter an element : -1
[4, 3, 2, 1]
<list_reverseiterator object at 0x0000021B0D470A90>
```

08) WAP to print even numbers in a list.

In [38]:
```
list4=[]
while(True):
    a=int(input("Enter an element : "))
    if(a==-1):
        break
    list4.append(a)
for i in list4 :
    if(i%2==0):
        print(i)
```

```
Enter an element : 1
Enter an element : 2
Enter an element : 3
Enter an element : -1
2
```

09) WAP to count unique items in a list.

In [8]:
```
lst = [1, 2, 3, 1, 2, 4, 5]
unique_items = []
for item in lst:
    if item not in unique_items:
        unique_items.append(item)
count = len(unique_items)
print(count)
```

```
5
```

10) WAP to copy a list.

In [7]:
```
list3 = [10,20,30,40,50]
mylist=list3.copy()
print(mylist)
```

```
[10, 20, 30, 40, 50]
```

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

In [6]:
```
list5=[]
while(True):
    a=int(input("Enter an element : "))
    if(a==-1):
        break
    list5.append(a)
a1=int(input("Enter your index1 : "))
a2=int(input("Enter your index2 : "))

for i in range(a1,a2+1):
    if(list5[i]%2==1):
        print(list5[i])
```

```
Enter an element : 12
Enter an element : 10
Enter an element : 11
Enter an element : 32
Enter an element : 31
Enter an element : 11
Enter an element : 11
Enter an element : 12
Enter an element : 10
Enter an element : 58
Enter an element : 95
Enter an element : -1
Enter your index1 : 1
Enter your index2 : 0
```

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

In [5]:
```
my_list = [1, 2, 3, 2, 4, 2, 5]
element = 2
count = my_list.count(element)
print(count)
```

```
3
```

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

In [4]:
```
list6 = [10, 20, 30, 40, 50]
list6.sort()
second_largest = list6[-2]
print(second_largest)
```

```
40
```

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

In [3]:
```
from collections import Counter
my_list = [1, 2, 2, 3, 3, 3, 4]
k = 2
freq = Counter(my_list)
result = [key for key, value in freq.items() if value > k]
print(result)
```

```
[3]
```

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

In [54]:
```
#without using Comprehension
list7=[]
ans=0
for i in range(1,10):
    ans=i*i
    list7.append(ans)
print(list7)

#with using Comprehension
list8=[i*i for i in range(1,10)]
print(list8)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
[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 [2]:
```
fruits = ['banana', 'apple', 'blueberry', 'cherry', 'berry']
b_fruits = [fruit for fruit in fruits if fruit.startswith('b')]
print(b_fruits)
```

```
['banana', 'blueberry', 'berry']
```

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

In [1]:
```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
common_elements = [element for element in list1 if element in list2]
print(common_elements)
```

```
[4, 5]
```

In [ ]:
```

```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 6

Prince Bhanderi | 23010101022 | 31-12-2024

Tuple

01) WAP to find sum of tuple elements.

In [5]:
```
a=(1,2)
sum=0
for i in range(len(a)):
    sum=sum+a[i]
print(sum)
```

```
3
```

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

In [17]:
```
a=(1,2,3,4,1,2,3,4)
b=sorted(set(a))
print(b)
k=int(input("enter elememt"))
max_ele = b[-k:]
min_ele = b[:k]
print(max_ele)
print(min_ele)
```

```
[1, 2, 3, 4]
enter elememt2
[3, 4]
[1, 2]
```

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

In [21]:
```
List1 = [(1, 2, 3), (4, 8, 10)]
k = 2

for i in List1:

    if all(x % k == 0 for x in i):
        print(i)
```

```
(4, 8, 10)
```

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

In [29]:
```
t1 = (1, 2, 3)
List3=[]
for i in range(len(t1)):
     List3.append(tuple([t1[i],t1[i]**3]))
print(List3)

#comprehention
l = [(i,i**3) for i in t1]
print(l)
```

```
[(1, 1), (2, 8), (3, 27)]
[(1, 1), (2, 8), (3, 27)]
```

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

In [31]:
```
List1 = [(1, 2, 3), (4, 8, 10),(-1,-2,3)]

for i in List1:

    if all(x > 0 for x in i):
        print(i)
```

```
(1, 2, 3)
(4, 8, 10)
```

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

In [46]:
```
l1=[1,2,3,4]
t2=(1,2,3,4,5,6)

l1.extend(t2)
print(l1)

t3 = tuple(l1)+t2
print(t3)
```

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

07) WAP to remove tuples of length K.

In [48]:
```
l4=[(1,2,3),(1,2,3,4,5,6),(1,2)]
k=2
l5=[i for i in l4 if len(i)!=k]
print(l5)
```

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

08) WAP to remove duplicates from tuple.

In [49]:
```
t6=(1,1,1,2,2,2,3,3,3)
b=set(t6)
t6=tuple(b)
print(t6)
```

```
(1, 2, 3)
```

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

In [50]:
```
t7 = (1, 2, 3, 4)
t8 = tuple(t7[i] * t7[i+1] for i in range(len(t7)-1))
print(t8)
```

```
(2, 6, 12)
```

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

In [53]:
```
t10 = (1, 2, 3, 1)
if len(t10) == len(set(t10)):
    print("The tuple is distinct")
else:
    print("The tuple is not distinct")
```

```
The tuple is not distinct
```

In [ ]:
```

```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 7

Prince Bhanderi | 23010101022 | 21-01-2024

Set & Dictionary

01) WAP to iterate over a set.

In [5]:
```
my_set = {1, 2, 3, 6, 4, 5, 1, 2 ,6}

for element in my_set:
    print(element)
```

```
1
2
3
4
5
6
```

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

In [4]:
```
my_set = {1, 2, 3, 4, 5,1, 2 ,6}
List = list(my_set)
string = str(my_set)
my_tuple = tuple(my_set)
print(List,string,my_tuple)
```

```
[1, 2, 3, 4, 5, 6] {1, 2, 3, 4, 5, 6} (1, 2, 3, 4, 5, 6)
```

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

In [10]:
```
my_set = {1, 2, 3, 4, 5, 1, 2 ,6}
print("Minimum number in set is",min(my_set))
print("Maximum number in set is",max(my_set))
```

```
Minimum number in set is 1
Maximum number in set is 6
```

04) WAP to perform union of two sets.

In [11]:
```
my_set1 = {1,2,5,3,4,6,2,5,3}
my_set2 = {6,7,8,9,5,10,15,20}

print(my_set1.union(my_set2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20}
```

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

In [13]:
```
my_set1 = {1,2,5,3,4,6,2,5,3}
my_set2 = {6,7,8,9,5,10,15,20}

print(my_set1.intersection(my_set2))
```

```
{5, 6}
```

06) WAP to remove duplicates from list.

In [14]:
```
my_list = [1,2,3,5,6,4,5,6,3,21,8]
my_set = set(my_list)
my_list = list(my_set)
print(my_list)
```

```
[1, 2, 3, 4, 5, 6, 8, 21]
```

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

In [21]:
```
string = "Prince Bhanderi prince"
words = string.split()
words = [word.lower() for word in words]

my_set = set(words)

print("Unique Word is")
for word in my_set:
    print(word)
```

```
Unique Word is
prince
bhanderi
```

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

In [24]:
```
my_set1 = {1,2,5,3,4,6,2,5,3}
my_set2 = {6,7,8,9,5,10,15,20}
my_set1 = my_set1 - my_set2

print(my_set1)
```

```
{1, 2, 3, 4}
```

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

In [36]:
```
str1 = input("Enter the First String: ").lower()
str2 = input("Enter the Second String: ").lower()

set1 = set(str1)
set2 = set(str2)

if (set1==set2):
    print("The strings are anagrams.")
else:
    print("The strings are not anagrams.")
```

```
Enter the First String: prince
Enter the Second String: bhanderi
The strings are not anagrams.
```

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

In [34]:
```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7]
list3 = [5, 8, 9, 4]

common_elements = set(list1) & set(list2) & set(list3)
print(common_elements)
```

```
{4, 5}
```

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

In [33]:
```
string = "Hello World"
vowels = set("aeiouAEIOU")
count = sum(1 for char in string if char in vowels)
print(count)
```

```
3
```

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

In [32]:
```
string = input("Enter a string: ")
if all(char in "01" for char in string):
    print("The string is a binary string.")
else:
    print("The string is not a binary string.")
```

```
Enter a string: 1
The string is a binary string.
```

13) WAP to sort dictionary by key or value.

In [30]:
```
dict1 = {'a': 5, 'c': 8, 'b': 2}
sorted_by_key = dict(sorted(dict1.items()))
sorted_by_value = dict(sorted(dict1.items(), key=lambda item: item[1]))

print("Sorted by key:", sorted_by_key)
print("Sorted by value:", sorted_by_value)
```

```
Sorted by key: {'a': 5, 'b': 2, 'c': 8}
Sorted by value: {'b': 2, 'a': 5, 'c': 8}
```

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

In [29]:
```
dict1 = {'a': 5, 'b': 3, 'c': 7}
total_sum = sum(dict1.values())
print(total_sum)
```

```
15
```

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 [28]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = 'd'

if key in dict1:
    print(dict1[key])
else:
    print("Key Not Found")
```

```
8
```

In [ ]:
```

```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Prince Bhanderi | 23010101022 | 21-01-2025

Lab - 8

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 [7]:
```
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

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="".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 [1]:

```
print("Positional Arguments::")
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)

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


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

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

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

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

print("Variable Legngth Positional(*args) & variable length Keyword Arguments (

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

my_function("alexander", "benjamin", "Jonathan")

def my_function(**kid):
  print("**kargs:His last name is " + kid["lname"])

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

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 (**kwa
rgs)::
*args:The youngest child is Jonathan
**kargs:His last name is benjamin

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

In [ ]:

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 9 ¶

Prince Bhanderi | 23010101022 | 28-01-2025

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 [4]:
```
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

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
```

```
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 [ ]:
```
```

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 10

Prince Bhanderi | 23010101022 | 04-02-2025

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 [2]:
```
try:
    x = int(input("Enter a number: "))
    y = int(input("Enter another number: "))
    result = x / y
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero!")
except ValueError:
    print("Error: Invalid input, please enter integers.")
except TypeError:
    print("Error: Type mismatch!")
```

```
Enter a number: 1
Enter another number: 2
Error: Type mismatch!
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [12]:
```
list1=[1,2,3]
try:
    print(list1[4])
except IndexError as e:
    print("Erorr:",str(e))
dic1={1:'hello',2:'hiiii'}
try:
    print(dic1[4])
except KeyError as a:
    print("Erorr:",str(a))
```

```
Erorr: list index out of range
Erorr: 4
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [17]:
```
try:
    f=open('text1.txt')
except FileNotFoundError as a:
    print("Erorr:",str(a))
try:
    import non_existent_module

except ModuleNotFoundError as a:
    print("Error:",str(a))
```

```
Erorr: [Errno 2] No such file or directory: 'text1.txt'
Error: No module named 'non_existent_module'
```

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

In [19]:
```
try:
    x = int(input("Enter a number: "))
    y = int(input("Enter another number: "))
    result = x / y
    print("Result:", result)
except Exception as e:
    print(f"Error: {e}")
```

```
Enter a number: 1
Enter another number: raj
Error: invalid literal for int() with base 10: 'raj'
```

05) WAP to demonstrate else and finally block.

In [20]:
```
try:
    x = int(input("Enter a number: "))
    y = int(input("Enter another number: "))
    result = x / y
except Exception as e:
    print(f"Error: {e}")
else:
    print("Result:", result)
finally:
    print("done!!!!")
```

```
Enter a number: 1
Enter another number: 2
Result: 0.5
done!!!!
```

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 [26]:
```
try:
    grades_input = input("Enter grades separated by commas: ")
    print(grades_input.split(','))
    grades = [int(grade.strip()) for grade in grades_input.split(',')]
    print("Grades:", grades)
except ValueError:
    print("Error: One or more values could not be converted to an integer.")
```

```
Enter grades separated by commas: 1   ,2
['1   ', '2']
Grades: [1, 2]
```

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

In [3]:
```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")


result = divide(10, 0)
```

```
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 [2]:
```
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: {e}")
```

```
Enter your age: 123
Your age is: 123
```

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 [30]:
```
class InvalidUsernameError(Exception):
    def __init__(self,msg):
        self.msg = msg

def validate_username(username):
    if len(username) < 5 or len(username) > 15:
        raise InvalidUsernameError("Username must be between 5 and 15 character
    else:
        print(f"Username is valid: {username}")


try:
    username = input("Enter a username: ")
    validate_username(username)
except InvalidUsernameError as e:
    print(f"Error: {e}")
```

```
Enter a username: 123
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 [1]:

```
import math

class NegativeNumberError(Exception):
    pass

def calculate_square_root(number):
    if number < 0:
        raise NegativeNumberError("Cannot calculate the square root of a negati
    else:
        print(f"The square root of {number} is: {math.sqrt(number)}")

# Example usage
try:
    number = float(input("Enter a number: "))
    calculate_square_root(number)
except NegativeNumberError as e:
    print(f"Error: {e}")
```

```
Enter a number: 12
The square root of 12.0 is: 3.4641016151377544
```

In [ ]:

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 11

Prince Bhanderi | 23010101022 | 11-02-2025

Modules

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 [2]:
```
import calc

a = 10
b = 5

print("Addition:", calc.add(a, b))
print("Subtraction:", calc.sub(a, b))
print("Multiplication:", calc.mul(a, b))
print("Division:", calc.div(a, b))
```

```
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
```

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

In [5]:
```
import random

m = "Hello, World!"
r= random.choice(m)
print("Random character:", r)
```

```
Random character: l
```

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

In [11]:
```
import random

m = [1,2,3,4,5]
r= random.choice(m)
print("Random character:", r)
```

```
Random character: 5
```

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

In [31]:
```
import random

m = [1,2,3,4,5,6]
r=1
p=0
while(p!= 1):
    p=random.choice(m)
print("Random character:", r,p)
```

```
Random character: 1 1
```

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

In [34]:
```
import random
ans=[]
j=0
p=0
while(len(ans)!=3):
    r = random.randrange(100,999)
    if(r % 5 == 0):
        ans.append(r)
ans
```

Out[34]:
```
[125, 580, 490]
```

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 [38]:
```
import random
tickets = random.sample(range(100,999),100)
print(tickets)
for i in range(1,3):
    r=random.choice(tickets)
    print("you are winnnerrrrr!!! ticket no:",r)
```

```
[767, 650, 975, 310, 147, 679, 825, 997, 384, 133, 393, 472, 993, 464, 438, 7
45, 846, 578, 830, 219, 764, 186, 270, 637, 445, 216, 106, 671, 955, 187, 94
4, 531, 306, 762, 307, 880, 126, 868, 662, 829, 615, 917, 976, 760, 694, 667,
341, 899, 214, 796, 250, 919, 544, 579, 362, 816, 775, 227, 167, 140, 103, 13
6, 715, 952, 283, 479, 632, 371, 807, 841, 906, 487, 773, 682, 888, 973, 315,
843, 486, 342, 252, 399, 968, 757, 538, 707, 234, 301, 419, 215, 848, 949, 97
7, 866, 291, 302, 748, 320, 209, 561]
you are winnnerrrrr!!! ticket no: 106
you are winnnerrrrr!!! ticket no: 531
```

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

In [8]:
```
from datetime import datetime
datetime.now()
```

Out[8]:
```
datetime.datetime(2025, 2, 10, 10, 56, 32, 916237)
```

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

In [14]:
```
from datetime import datetime, timedelta

given_date = datetime(2025, 2, 10)  # Example: 10th Feb 2025
new_date = given_date - timedelta(days=7)
print("New Date after subtracting a week:", new_date)
```

```
New Date after subtracting a week: 2025-02-03 00:00:00
```

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

In [15]:
```
from datetime import datetime

date1 = datetime(2025, 1, 1)
date2 = datetime(2025, 2, 10)

difference = date2 - date1
print("Number of days between dates:", difference.days)
```

```
Number of days between dates: 40
```

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

In [17]:
```
from datetime import datetime

given_date = datetime(2025, 7, 10)
day_of_week = given_date.strftime("%A")
print("Day of the week:", day_of_week)
```

```
Day of the week: Thursday
```

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

In [18]:
```
from datetime import datetime

now = datetime.now()
print("Current Date and Time:", now)

formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print("Formatted Date and Time:", formatted_date)
```

```
Current Date and Time: 2025-02-10 11:04:11.978997
Formatted Date and Time: 2025-02-10 11:04:11
```

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

In [19]:
```
import math

print("Pi:", math.pi)
print("Square Root of 16:", math.sqrt(16))
print("Factorial of 5:", math.factorial(5))
print("Cosine of 0 degrees:", math.cos(0))
```

```
Pi: 3.141592653589793
Square Root of 16: 4.0
Factorial of 5: 120
Cosine of 0 degrees: 1.0
```

In [ ]:
```

```

(https://www.darshan.ac.in/)

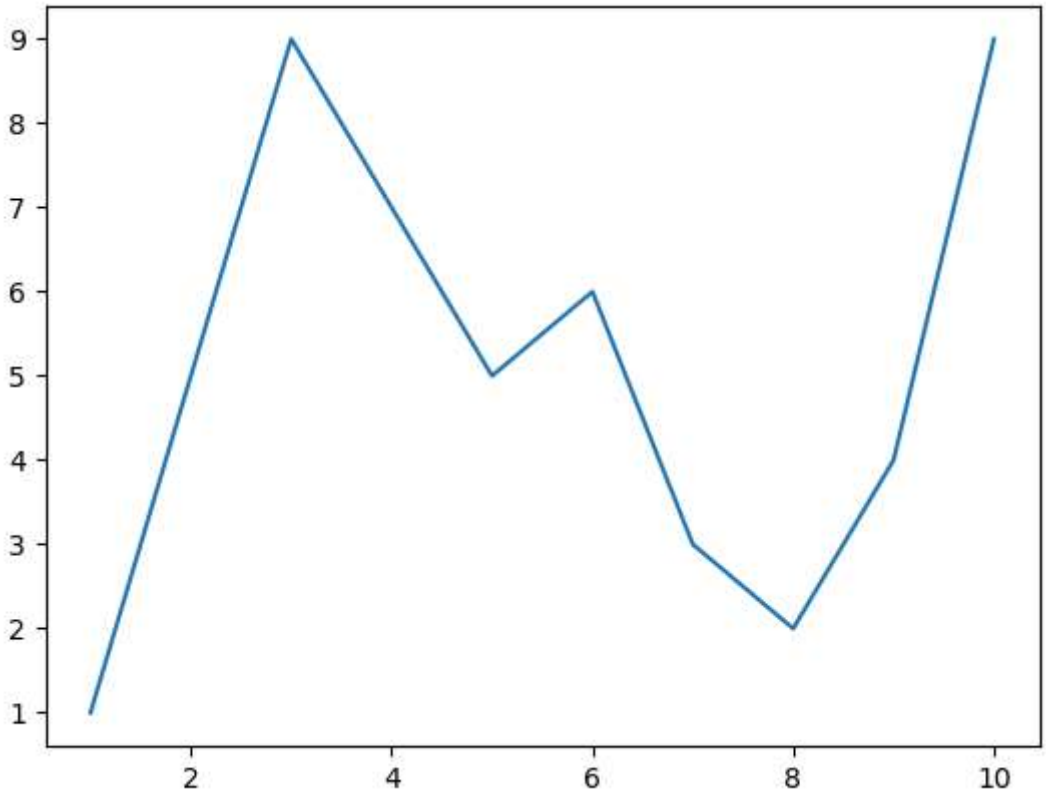
Python Programming - 2301CS404

Lab - 12

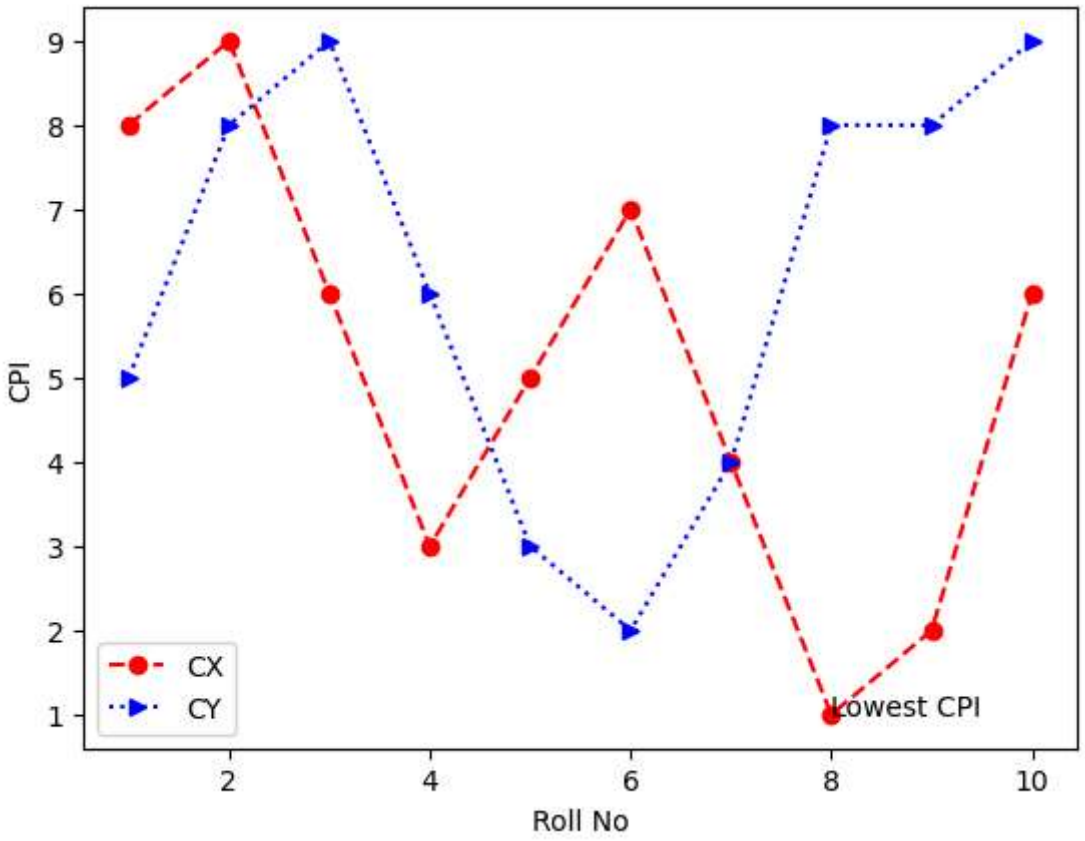
Prince Bhanderi | 23010101022 | 18-02-2025

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

In [4]:
```
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()
# write a code to display the line chart of above x & y
```



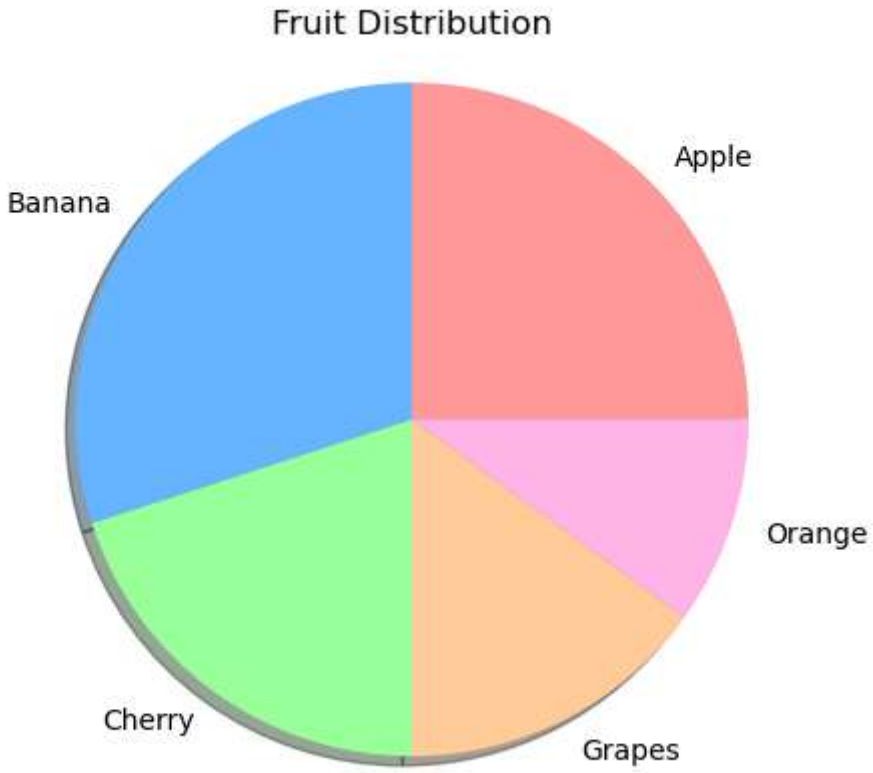
In [19]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks,label="CX",color="r",marker="o",linestyle="--")
plt.plot(x,cyMarks,label="CY",color="b",marker=">",linestyle="dotted")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.annotate('Lowest CPI',xy=[8,1])
plt.show()
```



04) WAP to demonstrate the use of Pie chart.

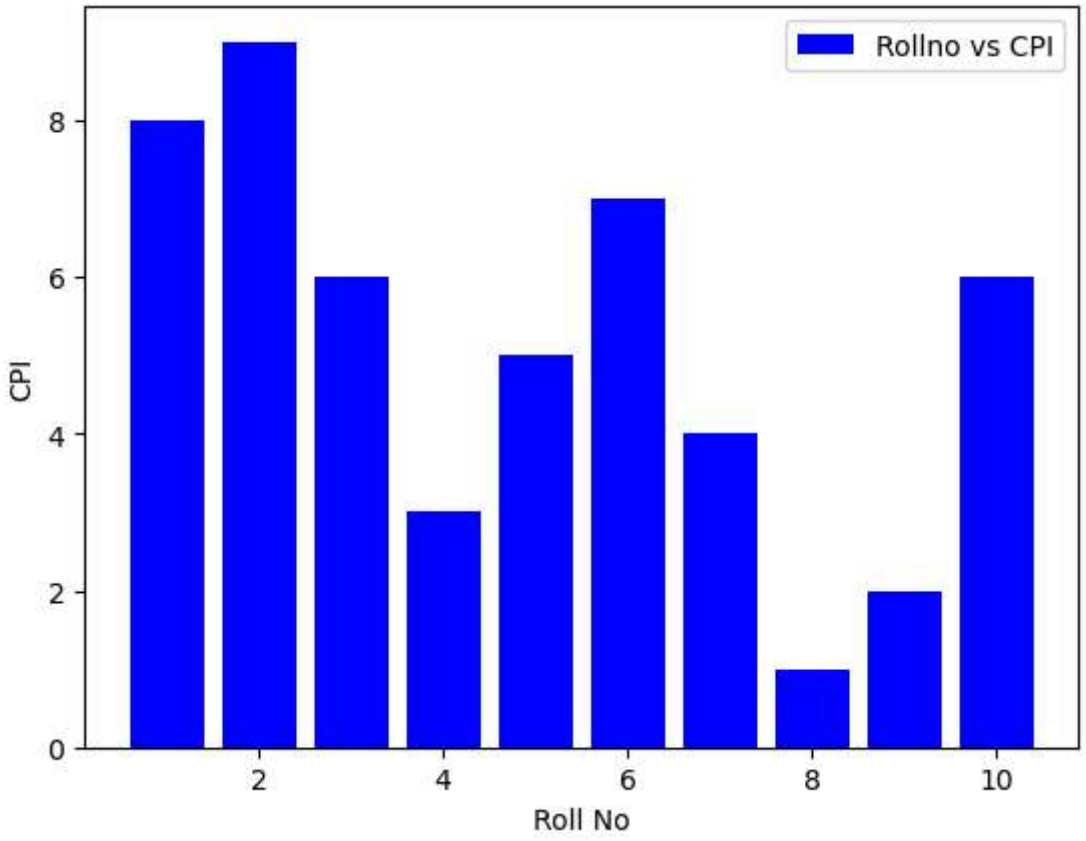
In [114]:
```
labels = ['Apple', 'Banana', 'Cherry', 'Grapes', 'Orange']
sizes = [25, 30, 20, 15, 10]
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#ffb3e6']

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



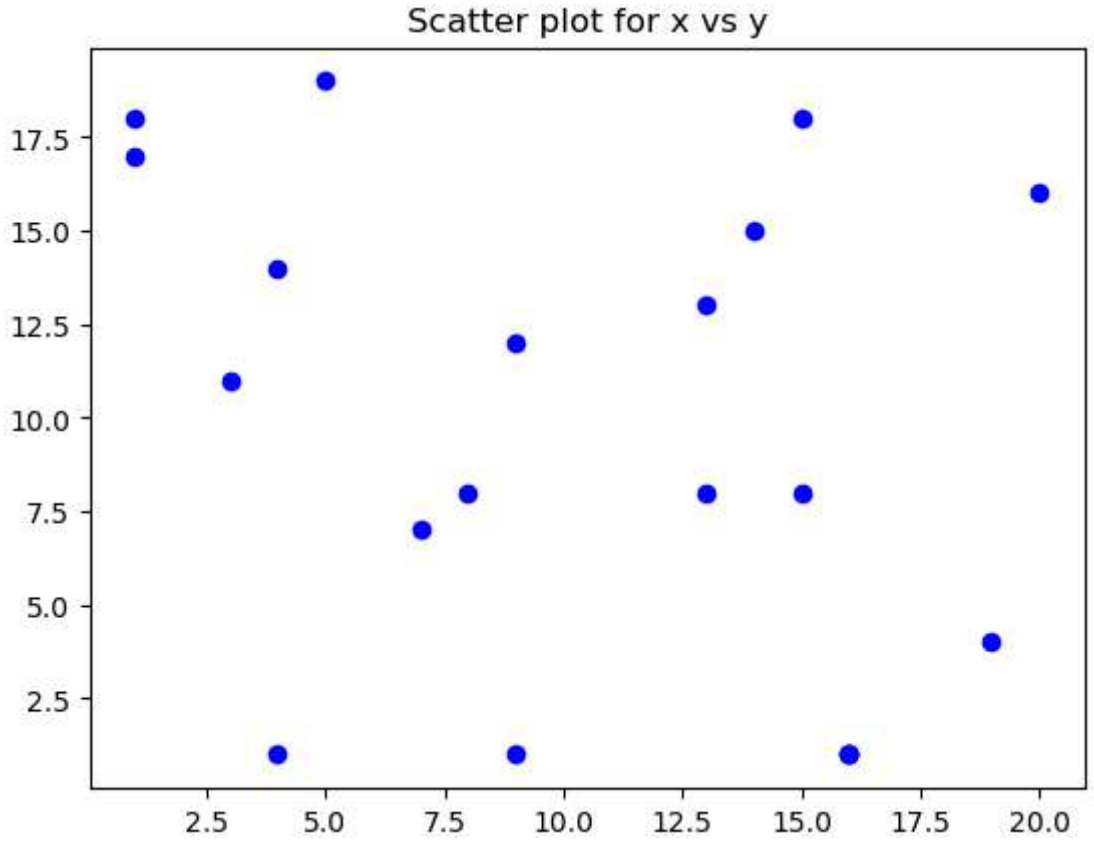
05) WAP to demonstrate the use of Bar chart.

In [58]:
```
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()
```



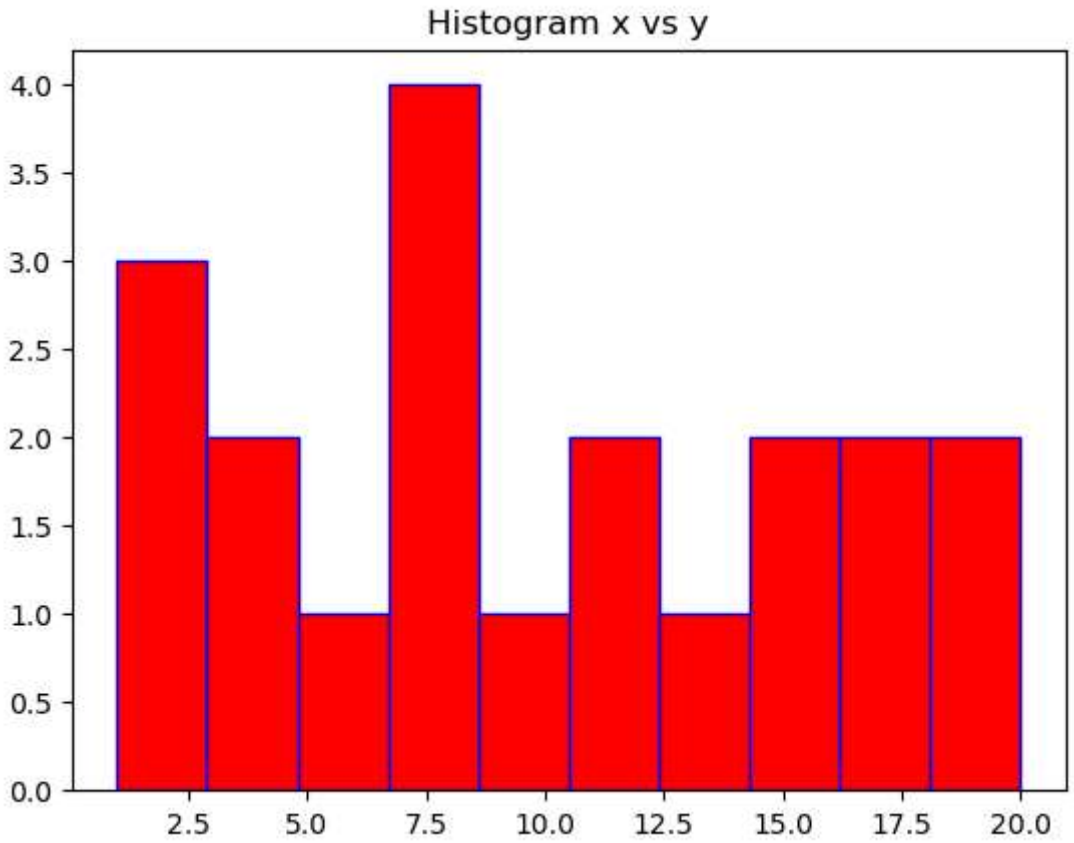
06) WAP to demonstrate the use of Scatter Plot.

In [82]:
```
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()
```



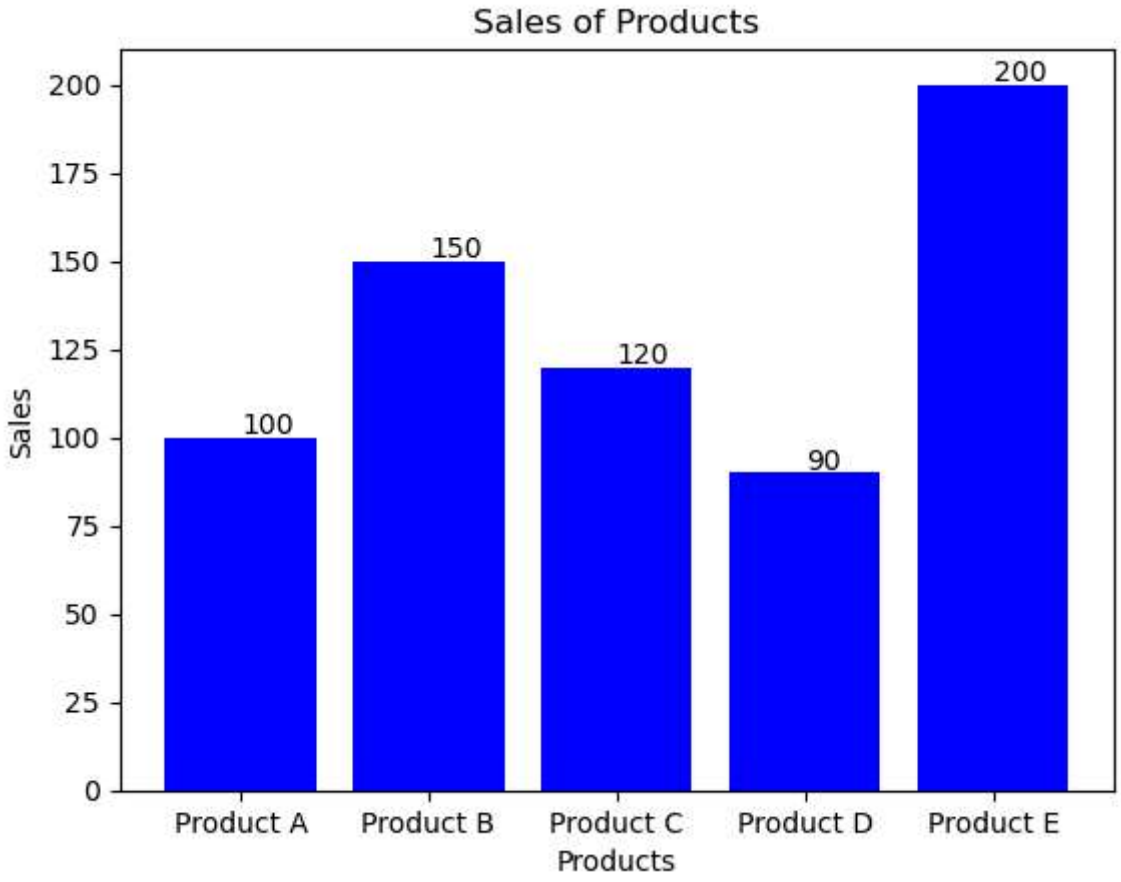
07) WAP to demonstrate the use of Histogram.

In [108]:
```
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.

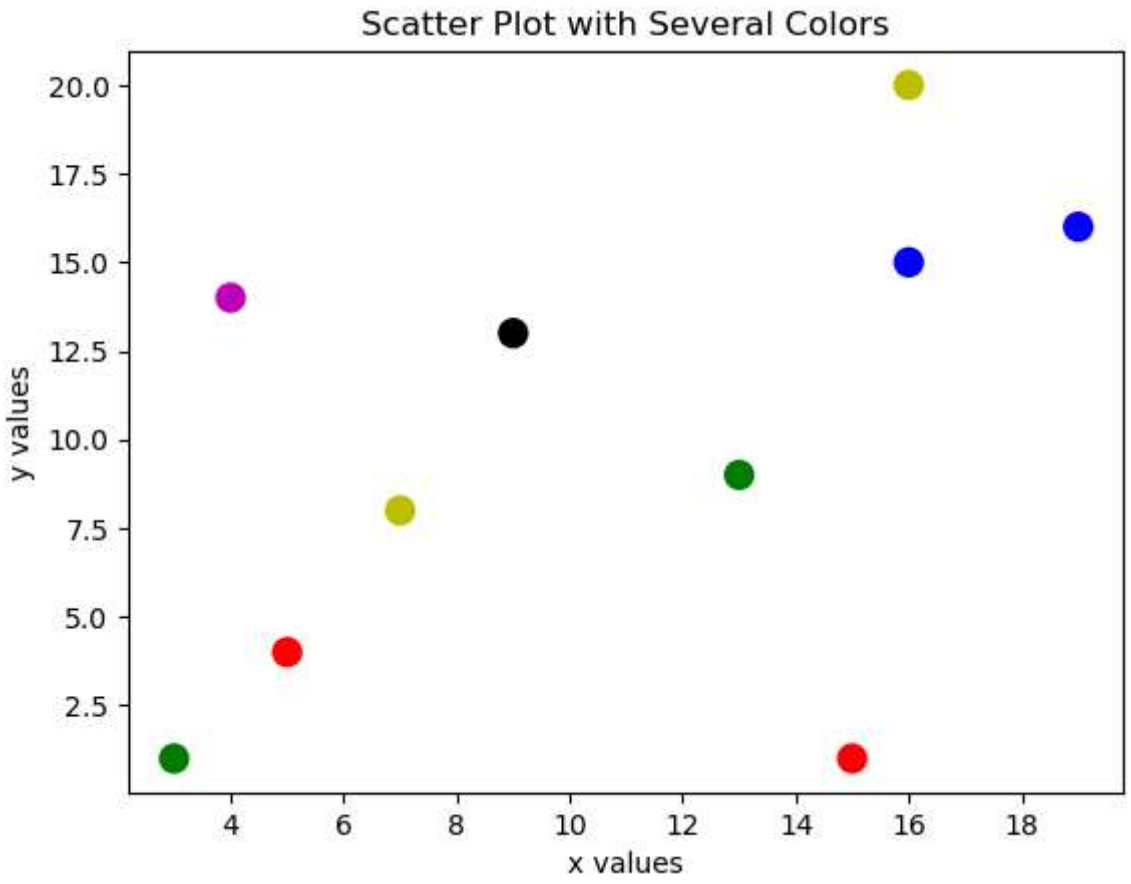
In [128]:
```
products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
sales = [100, 150, 120, 90, 200]
plt.bar(products, sales, color='b')
for i in range(len(products)):
    plt.text(i, sales[i]+1,str(sales[i]))
plt.title('Sales of Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()
```



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

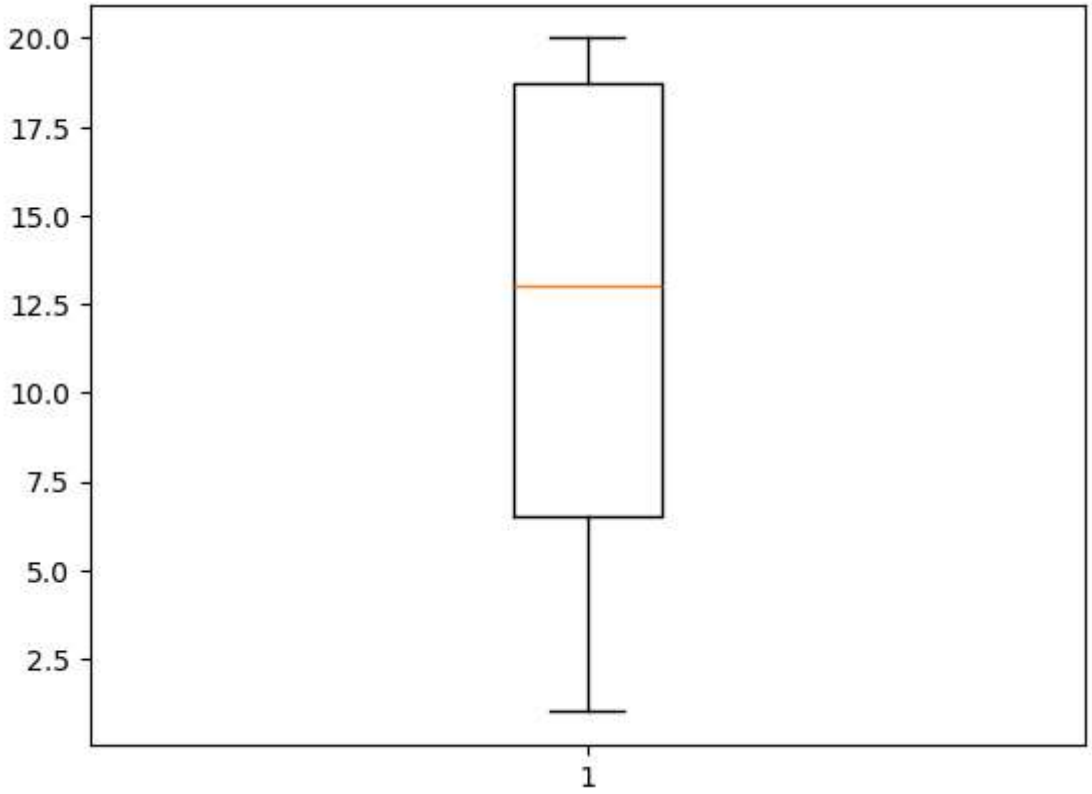
In [134]:
```
r.seed(1)
x = [r.randint(1,20) for i in range(10)]
y = [r.randint(1,20) for i in range(10)]
colors = ['r','b','g','k','m','y','r','b','g','y']

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 [158]:
```
data=[r.randint(1,20) for i in range(10)]
plt.boxplot(data)
plt.show()
```



(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 13

Prince Bhanderi | 23010101022 | 04-03-2025

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 [6]:
```
class Student:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade

obj=Student("Prince",19,"A+")

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

```
Prince
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 [5]:
```
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:230151116541
Enter User_Name:Prince Bhanderi
Enter Email:23010101022@da.com
Enter Account_Type:Saving
Enter Account_Balance:100000
230151116541
Prince Bhanderi
23010101022@da.com
Saving
100000
```

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 [4]:
```
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("Prince",20,555500000)
obj.displaydetails()
obj.updatedetails("Prince",19,1000000)
obj.displaydetails()
```

```
Employee Information:
Name: Prince
Age: 20
Salary: 555500000
Employee Information:
Name: Prince
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 - {d

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]:
```python
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 [1]:
```
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: Prince
Enter the number of days present in class: 220
Attendance recorded: 220 days
Certificate of Attendance

Principal: Mr ABC
Student: Prince
Days Present: 220
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]:
```
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 + oth
        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)
```

In [ ]:

(https://www.darshan.ac.in/)

Python Programming - 2301CS404

Lab - 13

Prince Bhanderi | 23010101022 | 11-03-2025 ¶

Continued..

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

In [20]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def calculate_area(self):
        return self.length * self.width

rect = Rectangle(10, 7)

def get_area(rectangle):
    return rectangle.calculate_area()

area = get_area(rect)
print("Area of rectangle:", area)
```

```
Area of rectangle: 70
```

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 [7]:
```
class Square:
    def __init__(self, length):
        self.length = length

    def area(self):
        area=self.length**2
        self.output(area)

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

square=Square(10)
square.area()
```

```
The area of squuare is:100
```

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 [21]:
```
class Rectangle:
    def __init__(self, length, width):
        if length == width:
            print("THIS IS SQUARE")
            self.length = None
            self.width = None
        else:
            self.length = length
            self.width = width

    def area(self):
        if self.length and self.width:
            area = self.length * self.width
            self.output(area)
        else:
            print("Cannot calculate area for a square object.")

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

    @classmethod
    def compare_sides(cls, length, width):
        if length == width:
            print("THIS IS SQUARE!!")
        else:
            print("This is a rectangle!!")

rect1 = Rectangle(10, 5)

if rect1.length and rect1.width:
    rect1.area()

Rectangle.compare_sides(10, 5)
Rectangle.compare_sides(5, 5)
```

```
The area of the rectangle is: 50
This is a rectangle!!
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 [25]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

    def set_side(self, side):
        if side > 0:
            self.__side = side
        else:
            print("Side length must be a positive number!")

    def area(self):
        return self.__side ** 2

square = Square(5)
print("Side of square:", square.get_side())
square.set_side(10)
print("New side of square:", square.get_side())
print("Area of square:", square.area())
```

```
Side of square: 5
New side of square: 10
Area of square: 100
```

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 [55]:
```
class Profit:
    def __init__(self):
        self.profit=0
    def getProfit(self):
        self.profit=float(input("Enter profit:"))

class Loss:
    def __init__(self):
        self.loss=0
    def getLoss(self):
        self.loss=float(input("Enter loss:"))

class Balance_sheet(Profit,Loss):
    def __init__(self):
        Profit.__init__(self)
        Loss.__init__(self)
        self.balance = 0

    def getBalance(self):
        self.balance = self.profit - self.loss

    def printBalance(self):
        print(f"Profit: {self.profit}")
        print(f"Loss: {self.loss}")
        print(f"Balance: {self.balance}")

balance_sheet=Balance_sheet()
balance_sheet.getProfit()
balance_sheet.getLoss()
balance_sheet.getBalance()
balance_sheet.printBalance()
```

```
Enter profit: 100000000
Enter loss: 5

Profit: 100000000.0
Loss: 5.0
Balance: 99999995.0
```

15) WAP to demonstrate all types of inheritance.

In [69]:
```python
# Single Inheritance
class Animal:
    def speak(self):
        print("Animal speaks")

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

# Multiple Inheritance
class Person:
    def __init__(self, name):
        self.name = name

    def introduce(self):
        print(f"Hello, my name is {self.name}")

class Employee:
    def __init__(self, employee_id):
        self.employee_id = employee_id

    def show_id(self):
        print(f"My employee ID is {self.employee_id}")

class Manager(Person, Employee):
    def __init__(self, name, employee_id):
        Person.__init__(self, name)
        Employee.__init__(self, employee_id)

    def work(self):
        print(f"{self.name} is working as a Manager.")

# Multilevel Inheritance
class Vehicle:
    def start_engine(self):
        print("Vehicle engine started")

class Car(Vehicle):
    def drive(self):
        print("Car is driving")

class SportsCar(Car):
    def speed(self):
        print("Sports car is speeding")

# Hierarchical Inheritance
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def area(self, radius):
        return 3.14 * radius * radius

class Rectangle(Shape):
    def area(self, length, width):
        return length * width
```

```python
# Hybrid Inheritance (Combination of Multiple and Multilevel Inheritance)
class School:
    def __init__(self, name):
        self.name = name

    def show_name(self):
        print(f"School Name: {self.name}")

class Teacher(School):
    def __init__(self, name, subject):
        School.__init__(self, name)
        self.subject = subject

    def teach(self):
        print(f"Teaching {self.subject}")

class HeadTeacher(Teacher):
    def __init__(self, name, subject, head_teacher_name):
        Teacher.__init__(self, name, subject)
        self.head_teacher_name = head_teacher_name

    def manage(self):
        print(f"{self.head_teacher_name} manages the teaching process.")


# 1. Single Inheritance
print("Single Inheritance:")
dog = Dog()
dog.speak()
dog.bark()
print()

# 2. Multiple Inheritance
print("Multiple Inheritance:")
manager = Manager("ABC", 101)
manager.introduce()
manager.show_id()
manager.work()
print()

# 3. Multilevel Inheritance
print("Multilevel Inheritance:")
sports_car = SportsCar()
sports_car.start_engine()
sports_car.drive()
sports_car.speed()
print()

# 4. Hierarchical Inheritance
print("Hierarchical Inheritance:")
circle = Circle()
print(f"Circle Area: {circle.area(5)}")
rectangle = Rectangle()
print(f"Rectangle Area: {rectangle.area(10, 5)}")
print()
```

```
# 5. Hybrid Inheritance
print("Hybrid Inheritance:")
head_teacher = HeadTeacher("DEF", "Maths", "ABC")
head_teacher.show_name()
head_teacher.teach()
head_teacher.manage()
```

```
Single Inheritance:
Animal speaks
Dog barks

Multiple Inheritance:
Hello, my name is ABC
My employee ID is 101
ABC is working as a Manager.

Multilevel Inheritance:
Vehicle engine started
Car is driving
Sports car is speeding

Hierarchical Inheritance:
Circle Area: 78.5
Rectangle Area: 50

Hybrid Inheritance:
School Name: DEF
Teaching Maths
ABC manages the teaching process.
```

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 [71]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")

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

    def display_employee_info(self):
        self.display_info()
        print(f"Salary: {self.salary}")

emp = Employee("ABC", 30, 50000)
emp.display_employee_info()
```

```
Name: ABC
Age: 30
Salary: 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 [73]:
```
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")

shapes = [Rectangle(), Circle(), Triangle()]

for shape in shapes:
    shape.draw()
```

```
Drawing a rectangle
Drawing a circle
Drawing a triangle
```

In [ ]:
```

```