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

Python Programming - 2301CS404

Lab - 1

KACHA MIHIR | 23010101118 | 26/11/2024

01) WAP to print “Hello World”

In [1]:
```
print("Hello world...!")
```

```
Hello world...!
```

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

In [ ]:
```
a = 2
b = 5
print(a + b)

c = int(input("enter a number :"))
d = int(input("enter a number :"))

print(c + d)
```

```
7
enter a number4
enter a number5
9
```

03) WAP to check the type of the variable.

In [5]:
```
a = 11
b = "mihir"
c = 6.9
print(type(a))
print(type(b))
print(type(c))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
```

04) WAP to calculate simple interest.

In [7]:
```
p = int(input("enter principal amount :"))
r =  int(input("enter interest rate :"))
t = int(input("enter time in years:"))

si = (p*r*t)/100

print("interest :",si)
print("total amount",si+p)
```

```
enter principal amount :1000
enter interest rate :15
enter time in years:5
interest : 750.0
total amount 1750.0
```

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

In [11]:
```
import math
r=float(input("enter the radius od circel: "))

area = math.pi*r*r

perimeter = 2*math.pi*r

print("area =","%0.4f"%area)
print("perimeter =","%0.4f"%perimeter)
```

```
enter the radius od circel: 100
area = 31415.9265
perimeter = 628.3185
```

06) WAP to calculate area of a triangle.

In [12]:
```
base = int(input("enter the base :"))
height = int(input("enter the height :"))

area= (base*height)/2

print("area",area)
```

```
enter the base :4
enter the height :4
area 8.0
```

07) WAP to compute quotient and remainder.

In [14]:
```
a=7
b=3

quotient = a//b
remainder = a%b

print("quotient =",quotient)
print("remainder =",remainder)
```

```
quotient = 2
remainder = 1
```

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

In [16]:
```
degree = float (input("enter degree"))

Fahrenheit = (degree*9/5)+32

print("Fahrenheit",Fahrenheit)

Fahrenheit = float (input("enter Fahrenheit"))

degree = (Fahrenheit-32)*5/9

print("degree",degree)
```

```
enter degree0
Fahrenheit 32.0
enter Fahrenheit32
degree 0.0
```

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

In [21]:
```
import math
x1 = int(input("enter coordinate x1 :"))
y1 = int(input("enter coordinate y1 :"))

x2 = int(input("enter coordinate x2 :"))
y2 = int(input("enter coordinate y2 :"))

distance = math.sqrt((x2-x1)**2 + (y2-y1)**2)

print("distance",distance)
```

```
enter coordinate x1 :1
enter coordinate y1 :1
enter coordinate x2 :2
enter coordinate y2 :2
1.4142135623730951
```

10) WAP to print sum of n natural numbers.

In [23]:
```
n = int (input("enter the number upto which you want sum :"))

sum = (n*(n+1))/2

print("sum =",sum)
```

```
enter the number upto which you want sum5
sum = 15.0
```

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

In [26]:
```
n = int (input("enter the number upto which you want sum :"))

sum = (n*(n+1)*(2*n+1))/6

print("sum =",sum)
```

```
enter the number upto which you want sum :5
sum = 55.0
```

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

In [27]:
```
firstname = input("enter first name")
lastname = input("enter last name")

fullname = firstname+lastname

print("fullname",fullname)
```

```
enter first namemihir
enter last namekacha
fullname mihirkacha
```

13) WAP to swap two numbers.

In [1]:
```
a = int(input("enter number in a :"))
b = int(input("enter number in b :"))

a,b=b,a

print("a = ",a,"b = ",b)
```

```
a =  8 b =  7
```

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

In [ ]:
```
a = int(input("enter kilometer :"))

print("meter =",a*1000)
print("feet =",a*3280.84)
```

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

In [1]:
```
day = int(input("Enter day: "))
month = int(input("Enter month: "))
year = int(input("Enter year: "))

print(day, month, year, sep="-")
```

```
Enter day: 5
Enter month: 4
Enter year: 2005
5-4-2005
```

In [ ]:

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

Python Programming - 2301CS404

Lab - 2

KACHA MIHIR | 23010101118 | 03/12/2024

if..else..

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

In [2]:
```
n=int(input("enter a number: "))

if (n>=0):
    print(n,'is positive')
else:
    print(n,'is negative')
```

```
enter a number: -6
-6 is negative
```

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

In [4]:
```
n=int(input("enter a number: "))

if (n%2==0):
    print(n,'is even')
else:
    print(n,'is odd')
```

```
enter a number: 5
5 is odd
```

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

In [7]:
```
a=int(input("enter a number1: "))
b=int(input("enter a number2: "))

if (a>b):
    print(a,'is largest')
elif (b>a):
    print(b,'is largest')
else:
    print(a,"=",b,"both are equal")
```

```
enter a number1: 5
enter a number2: 5
5 = 5
```

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

In [12]:
```
a=int(input("enter a number1: "))
b=int(input("enter a number2: "))
c=int(input("enter a number3: "))

if(a>b):

    if(b>c):
        print(a,"is largest")
    else:
        print(c,"is largest")
else:
    if(b>c):
        print(b,'is largest')
    else:
        print(c,'is largest')
```

```
enter a number1: 4
enter a number2: 7
enter a number3: 2
7 is largest
```

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 [15]:
```
year = int(input("enter a year: "))

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

```
enter a year: 1600
1600 is a leap year
```

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

In [27]:
```
day = int(input('emter a day number: '))-1
day = day%7

match day:
    case (0):
        print('sunday')
    case (1):
        print('monday')
    case (2):
        print('tuesday')
    case (3):
        print('wednesday')
    case (4):
        print('thursday')
    case (5):
        print('friday')
    case (6):
        print('saturday')
    case _:
        print('invalid')
```

```
emter a day number: -1
friday
```

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

In [29]:
```
a=int(input("enter a number1: "))
choice = input("enter a operator: ")
b=int(input("enter a number2: "))



match (choice[0]):
    case ('+'):
        print(a,"+",b,"=",a+b)
    case ('-'):
        print(a,"-",b,"=",a-b)
    case ('*'):
        print(a,"*",b,"=",a*b)
    case ('/'):
        print(a,"/",b,"=",a/b)
```

```
enter a number1: 354
enter a operator: -/+
enter a number2: 345
354 - 345 = 9
```

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 [30]:
```
sub1=int(input("enter a sub1 mark: "))
sub2=int(input("enter a sub2 mark: "))
sub3=int(input("enter a sub3 mark: "))
sub4=int(input("enter a sub4 mark: "))
sub5=int(input("enter a sub5 mark: "))

percentage = (sub1 + sub2 + sub3 + sub4 + sub5)/5

if(percentage<35):
    print('fail')
elif(percentage>=35 and percentage<45):
    print('pass')
elif(percentage>=45 and percentage<60):
    print('Second Class')
elif(percentage>=60 and percentage<70):
    print('first Class')
elif(percentage>=70 and percentage<=100):
    print('Distinction')
else:
    print('enter proper mark')
```

```
enter a sub1 mark: 30
enter a sub2 mark: 34
enter a sub3 mark: 34
enter a sub4 mark: 34
enter a sub5 mark: 34
fail
```

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 [37]:
```
side1=int(input("enter a side1 : "))
side2=int(input("enter a side2 : "))
side3=int(input("enter a side3 : "))

if(side1==side2 and side2==side3):
    print('equilateral')
elif(side1==side2 or side1 == side2 or side2 == side3):
    print('isosceles')
elif(((side1**2)==(side2**2)+(side3**2))or((side2**2)==(side1**2)+(side3**2))or
    print("right-angled triangle")
else:
    print('triangle is simple')
```

```
enter a side1 : 1
enter a side2 : 1
enter a side3 : 2
isosceles
```

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

In [ ]:
```
a=int(input("enter a number1: "))
b=int(input("enter a number2: "))
c=int(input("enter a number3: "))

if(a>b):

    if(b>c):
        print(a,"is largest")
    else:
        print(c,"is largest")
else:
    if(b>c):
        print(b,'is largest')
    else:
        print(c,'is largest')
```

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 [39]:
```
unit =int(input("enter a units: "))
bill=0

if(unit<=50):
    bill=unit*2.60
elif(unit>50 and unit<=100):
    bill=(50*2.6)+((unit-50)*3.25)
elif(unit>100 and unit<=200):
     bill=(50*2.6)+(50*3.25)+((unit-100)*5.26)
else:
    bill=(50*2.6)+(50*3.25)+(100*5.26)+((unit-200)*8.45)

print("your bill is",bill)
```

```
enter a units: 259
your bill is 1317.05
```

In [ ]:

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

Python Programming - 2301CS404

Lab - 3

KACHA MIHIR | 23010101118 | 10/12/2024

for and while loop

01) WAP to print 1 to 10.

In [2]:
```
for i in range(1,11,1):
    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 a number: "))

for i in range(1,n+1):
    print(i)
```

```
enter a number: 7
1
2
3
4
5
6
7
```

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

In [6]:
```
n=int(input("enter a number: "))

for i in range(1,n+1,2):
    print(i)
```

```
enter a number: 9
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 [7]:
```
a=int(input("enter a number: "))
b=int(input("enter a number: "))

for i in range(a,b,1):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
enter a number: 5
enter a number: 23
8
10
14
16
20
22
```

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

In [11]:
```
a=int(input("enter a number: "))
sum=0
for i in range(1,a+1,1):
    sum+=i
print(sum)
```

```
enter a number: 10
55
```

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

In [6]:
```
a=int(input("enter a number: "))
sum=0
for i in range(1,a+1,1):
    sum+=i**2
print(sum)
```

```
enter a number: 10
385
```

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

In [12]:
```
a=int(input("enter a number: "))
sum=0
for i in range(1,a+1,1):
    if(i%2!=0):
        sum+=i
    else:
        sum-=i
print(sum)
```

```
enter a number: 7
4
```

08) WAP to print multiplication table of given number.

In [14]:
```
a=int(input("enter a number: "))
sum=0
for i in range(1,11,1):
    print(a,"X",i," =",a*i)
```

```
enter a number: 9
9 X 1  = 9
9 X 2  = 18
9 X 3  = 27
9 X 4  = 36
9 X 5  = 45
9 X 6  = 54
9 X 7  = 63
9 X 8  = 72
9 X 9  = 81
9 X 10  = 90
```

09) WAP to find factorial of the given number.

In [15]:
```
a=int(input("enter a number: "))
sum=1
for i in range(1,a+1,1):
    sum*=i
print(sum)
```

```
enter a number: 5
120
```

10) WAP to find factors of the given number.

In [17]:
```
a=int(input("enter a number: "))

for i in range(1,a+1,1):
    if(a%i==0):
        print(i)
```

```
enter a number: 6
1
2
3
6
```

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

In [6]:
```
import math
n=int(input("Enter a number "))
ans=True
for i in range(2,int(math.sqrt(n)+1)):
    if(n%i==0):
        ans=False
        break
if(ans):
    print(n,'is a prime number')
else:
    print(n,'is not a prime number')
```

```
Enter a number 9
9 is not a prime number
```

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

In [3]:
```
a=input("enter a number: ")
ans = 0
for i in a:
    ans+=int(i)
print(ans)
```

```
enter a number: 5687
26
```

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

In [8]:
```
a=int(input("enter a number: "))
temp = a
rev = 0
while(temp>0):
    rev = (rev*10) + temp%10
    temp = temp//10

print(rev)

if(a == rev):
    print(a,'is palindrome')
else:
    print(a,'is not palindrome')
```

```
enter a number: 123321
123321
123321 is palindrome
```

14) WAP to print GCD of given two numbers.

In [13]:
```
a=int(input("enter a number: "))
b=int(input("enter a number: "))

print('gcd of',a,",",b,"is",end=" ")

while(b!=0):
    a,b=b,a%b

print(a)
```

```
enter a number: 5
enter a number: 3245645
gcd of 5 , 3245645 is 5
```

In [ ]:

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

Python Programming - 2301CS404

Lab - 4

KACHA MIHIR | 23010101118 | 17/12/2024

String

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

In [1]:
```
s1 = input("Enter string: ")

if s1 == s1[::-1]:
    print(s1, "is palindrome")
else:
    print(s1, "is not palindrome")
```

```
Enter string: mihir
mihir is not palindrome
```

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

In [2]:
```
s1 = input("Enter string: ")

rev = s1[::-1]
rev = rev.split()
rev.reverse()
rev = " ".join(rev)
print(rev)

rev = s1.split()
rev.reverse()
rev = " ".join(rev)
print(rev)
```

```
Enter string: mihir
rihim
mihir
```

03) WAP to remove ith character from given string.

In [5]:
```
str=input("Enter the String:")
print(str)

i = int(input("ith Character which you want to remove:"))
str2 = str[:i]+str[i+1:]

print(str2)
```

```
Enter the String:mihir
mihir
ith Character which you want to remove:1
mhir
```

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

In [6]:
```
s1 = input("Enter string: ")
l = 0

for i in s1:
    l += 1
print(l)
```

```
Enter string: mihir
5
```

05) WAP to print even length word in string.

In [9]:
```
s1 = input("Enter string: ")
li = s1.split()
li = [i for i in li if len(i) % 2 == 0]

str = " ".join(li)
print(str)
```

```
Enter string: mihir kachaa
kachaa
```

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

In [10]:
```
s1 = input("Enter string: ")
c = 0

for i in s1:
    if "aeiouAEIOU".find(i) != -1:
        c += 1
print(c)
```

```
Enter string: mihir
2
```

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

In [11]:
```
s1 = input("Enter string: ")
s1 = s1.title()
li = s1.split()
s1 = ""

for i in li:
    s1 += i[: len(i) - 1 :] + i[-1].upper() + " "


print(s1)
```

```
Enter string: hello there mihir
HellO TherE MihiR
```

08) WAP to convert given array to string.

In [4]:
```
arr = [1, 2, 3, 4, 5, 6, 7]
s1 = " ".join(str(i) for i in arr)
print(s1)
```

```
1 2 3 4 5 6 7
```

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

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

In [6]:
```
password = input("Enter password: ")
c_password = input("Enter confirm password: ")

if password == c_password:
    print("Password confirmed successfully!")
elif password.lower() == c_password.lower():
    print("Error: Password and confirm password do not match due to case sensit
else:
    print("Error: Password and confirm password do not match.")
```

```
Error: Password and confirm password do not match due to case sensitivity.
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [9]:
```
cno = input("Enter card number: ")

maskno = "*" * (len(cno) - 4) + cno[len(cno) - 4 : :]
print("Masked card number:", maskno)
```

```
Masked card number: ********9123
```

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

s1 = decimal and s2 = medical are Anagram

In [14]:
```
s1 = "decimal"
s2 = "medical"
print(s1, s2, sep="\n")

s1 = sorted(s1.lower())
s2 = sorted(s2.lower())

if s1 == s2:
    print("Both are Anagrams.")
else:
    print("Both are not Anagrams.")
```

```
decimal
medical
Both are Anagrams.
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [16]:
```
s1 = input("Enter string")
print("Input string", s1)
lower = []
upper = []

for i in s1:
    if i.islower():
        lower.append(i)
    else:
        upper.append(i)
s1 = "".join(lower) + "".join(upper)

print("First lowercase then uppercase alphabets")
print(s1)
```

```
Input string EHlsarwiwhtwMV
First lowercase then uppercase alphabets
lsarwiwhtwEHMV
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

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

Python Programming - 2301CS404

Lab - 5

KACHA MIHIR | 23010101118 | 24/12/2024

List

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

In [3]:
```
li = [1, 2, 3, 4, 5, 6, 7]
sum = 0

for i in li:
    sum += i

print("Sum:", sum)
```

```
Sum: 28
```

02) WAP to find largest element in a List.

In [17]:
```
li = [-1, 2, -3, 4, -5, 6, 7]
# li.sort()  --mutable
li = sorted(li)  # immutable
largest = li[-1]

print("Largest:", largest)
```

```
Largest: 7
```

03) WAP to find the length of a List.

In [20]:
```
li = [-1, 2, -3, 4, -5, 6, 7]

print("Length of list:", len(li))
```

```
Length of list:  7
```

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

In [21]:
```
li = [-1, 2, -3, 4, -5, 6, 7]
print("Before:", li)

li[0], li[-1] = li[-1], li[0]

print("After:", li)
```

```
Before: [-1, 2, -3, 4, -5, 6, 7]
After: [7, 2, -3, 4, -5, 6, -1]
```

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

In [36]:
```
li = [-1, 2, -3, 4, -5, 6, 7]

li = li[len(li) // 2 : :] + li[0 : len(li) // 2 :]
print(li)
```

```
[4, -5, 6, 77, -1, 2, -3]
```

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

In [56]:
```
li = [-1, 2, -3, 4, -5, 6, 7]
print("Before:", li)

p1 = int(input("Enter one position: "))
p2 = int(input("Enter second position: "))

if p1 >= len(li) or p2 >= len(li):
    print("Invalid position")
else:
    li[p1], li[p2] = li[p2], li[p1]
print("After:", li)
```

```
Before: [-1, 2, -3, 4, -5, 6, 7]
Enter one position: 1
Enter second position: 5
After: [-1, 6, -3, 4, -5, 2, 7]
```

07) WAP to reverse the list entered by user.

In [41]:
```
li = [-1, 2, -3, 4, -5, 6, 7]
li.reverse()
print(li)
```

```
<class 'NoneType'>
```

08) WAP to print even numbers in a list.

In [43]:
```
li = [-1, -2, -3, 4, -5, 6, 7]

for i in li:
    if i % 2 == 0:
        print(i)
```

```
-2
4
6
```

09) WAP to count unique items in a list.

In [55]:
```
li = [-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 6, 8, 7]

print(len(set(li)))
```

```
12
```

10) WAP to copy a list.

In [51]:
```
li = [-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 6, 7]
li2 = li.copy()
print(li2)
```

```
[-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 6, 7]
```

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

In [58]:
```
a = int(input("Enter starting range: "))
b = int(input("Enter ending range: "))

for i in range(a, b):
    if i % 2 == 1:
        print(i)
```

```
Enter starting range: 10
Enter ending range: 20
11
13
15
17
19
```

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

In [72]:
```
li = [-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 6, 7]
di = dict().fromkeys(li, 0)

for i in li:
    di[i] = di.get(i) + 1

print(di)
```

```
{-1: 1, -2: 1, -3: 1, 4: 2, -5: 1, 6: 2, 7: 2, 1: 1, 2: 1, 3: 1, 5: 1}
```

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

In [66]:
```
li = [-1, 2, -3, 4, -5, 6, 7]
# li.sort()
slargest = li[-2]

print("Second Largest:", slargest)
```

```
Second Largest: 6
```

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

In [ ]:
```
li = [-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 4, 6, 7]
di = dict().fromkeys(li, 0)

for i in li:
    di[i] = di.get(i) + 1

k = int(input("Enter frequency: "))
ans = []
for i in di:
    if di.get(i) > k:
        ans.append(i)
print(di)
print(ans)
```

```
{-1: 1, -2: 1, -3: 1, 4: 2, -5: 1, 6: 2, 7: 2, 1: 1, 2: 1, 3: 1, 5: 1}
[4, 6, 7]
```

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

In [63]:
```
li = [i**2 for i in range(10)]
print(li)
```

```
[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 [65]:
```
li = ["apple", "banana", "papaya", "cherry", "Banana"]
li_b = []
for i in li:
    if i[0] == "b" or i[0] == "B":
        li_b.append(i)

print(li_b)
```

```
['banana', 'Banana']
```

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

In [11]:
```
li1 = [-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 4, 6, 7]
li2 = [6, 7, 1, 2, 3, 4, 5, 4, 6, 7]
li_ans = list(set(li1) & set(li2))

print(li1)
print(li2)
print("Common elements: ", li_ans)
```

```
[-1, -2, -3, 4, -5, 6, 7, 1, 2, 3, 4, 5, 4, 6, 7]
[6, 7, 1, 2, 3, 4, 5, 4, 6, 7]
Common elements:  [1, 2, 3, 4, 5, 6, 7]
```

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

Python Programming - 2301CS404

Lab - 6 ¶

KACHA MIHIR | 23010101118 | 26/11/2024

Tuple

In [6]:
```
def createTuple():
    n=int(input("enter the size of tuple: "))
    li=[]

    for i in range(n):
        element = int(input("enter element: "))
        li.append(element)

    return tuple(li)
```

01) WAP to find sum of tuple elements.

In [7]:
```
t1 = createTuple()

sum=sum(t1)
print(sum)
```

```
enter the size of tuple: 3
enter element: 12
enter element: 34
enter element: 45
91
```

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

In [8]:
```
t1 = createTuple()

li=sorted(t1)

k=int(input("enter value of k"))

print("max k element: ",li[-k::])
print("min k element: ",li[:k:])
```

```
enter the size of tuple: 5
enter element: 23
enter element: 54
enter element: 65
enter element: 7682
enter element: 2
enter value of k3
max k element:  [54, 65, 7682]
min k element:  [2, 23, 54]
```

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

In [12]:
```
n=int(input("enter the size of list"))
li=[]

for i in  range(n):
    t1 = createTuple()
    li.append(t1)

k=int(input("enter the value of k: "))
ans = [i for i in li if all(x%k==0 for x in i)]

print(ans)
```

```
enter the size of list5
enter the size of tuple: 3
enter element: 23
enter element: 23
enter element: 23
enter the size of tuple: 3
enter element: 45
enter element: 324
enter element: 34
enter the size of tuple: 3
enter element: 213
enter element: 12
enter element: 21
enter the size of tuple: 3
enter element: 432
enter element: 23
enter element: 12
enter the size of tuple: 3
enter element: 12
enter element: 34
enter element: 23
enter the value of k: 23
[(23, 23, 23)]
```

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

In [1]:
```
li=[1,2,3,4,5]

ans=[(i,i**3) for i in li]
print(ans)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
```

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

In [13]:
```
n=int(input("enter the size of list"))
li=[]

for i in  range(n):
    t1 = createTuple()
    li.append(t1)


ans = [i for i in li if all(x>0 for x in i)]

print(ans)
```

```
enter the size of list2
enter the size of tuple: 3
enter element: 23
enter element: 243
enter element: 45
enter the size of tuple: 3
enter element: 3
enter element: 45
enter element: -34
enter the value of k: 34
[(23, 243, 45)]
```

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

In [14]:
```
n=int(input("enter the size of list"))
li=[]

for i in  range(n):
    t1 = createTuple()
    li.append(t1)


print(li)
```

```
enter the size of list3
enter the size of tuple: 3
enter element: 23
enter element: 23
enter element: 23
enter the size of tuple: 3
enter element: 54
enter element: 45
enter element: 45
enter the size of tuple: 3
enter element: 65
enter element: 6667
enter element: 76
[(23, 23, 23), (54, 45, 45), (65, 6667, 76)]
```

In [18]:
```
n=int(input("enter the size of tuple: "))
tol=[]

for i in range(n):
    m=int(input("enter the size of list:"))
    li=[]

    for i in range(m):
        x=int(input("enter the element: "))
        li.append(x)
    tol.append(li)

tup=tuple(tol)
print(tup)
```

```
enter the size of tuple: 2
enter the size of list:2
enter the element: 23
enter the element: 23
enter the size of list:2
enter the element: 23
enter the element: 23
([23, 23], [23, 23])
```

07) WAP to remove tuples of length K.

In [3]:
```
n=int(input("enter the size of list"))
li=[]

for i in  range(n):
    t1 = createTuple()
    li.append(t1)


k=int(input("enter value of k"))

for i in li:
    if(len(i)==k):
        li.remove(i)

print(li)
```

```
enter the size of list2
enter the size of tuple: 2
enter element: 23
enter element: 23
enter the size of tuple: 4
enter element: 23
enter element: 23
enter element: 23
enter element: 23
enter value of k4
[(23, 23)]
```

08) WAP to remove duplicates from tuple.

In [5]:
```
 t1 = createTuple()
t1=tuple(set(t1))
print(t1)
```

```
enter the size of tuple: 4
enter element: 23
enter element: 23
enter element: 45
enter element: 76
(76, 45, 23)
```

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

In [4]:
```
n=int(input("enter the size of tuple"))
li=[]

for i in  range(n):
    x=int(input("enter the element: "))
    li.append(x)

print(li)

for i in range(n-1):
    li[i]=li[i]*li[i+1]

li.pop()
tup=tuple(li)

print(tup)
```

```
enter the size of tuple4
enter the element: 2
enter the element: 4
enter the element: 6
enter the element: 8
[2, 4, 6, 8]
(8, 24, 48)
```

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

In [12]:
```
t1 = createTuple()

flag = True
d1 = dict()
d1=d1.fromkeys(t1,0)

for i in t1:
    d1[i]+=1
    if(d1[i]>1):
        flag = False
        break

if(flag):
    print("distinct")
else:
    print("not  distinct")
```

```
enter the size of tuple: 3
enter element: 123
enter element: 345
enter element: 21
distinct
```

In [ ]:

In [ ]:

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

Python Programming - 2301CS404

Lab - 7

KACHA MIHIR | 23010101118 | 21/1/2025

Set & Dictionary

01) WAP to iterate over a set.

In [2]:
```
s = {1,22,"naam ekadu"}

for i in s:
    print(i)

# output because set is inodered
```

```
1
naam ekadu
22
```

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

In [5]:
```
s = {1,22,"naam ekadu",34,65,78,69}

li = list(s)
print(li)

string = str(s)
print(string)
print(type(string))

tp = tuple(s)
print(tp)
```

```
[65, 'naam ekadu', 34, 1, 69, 22, 78]
{65, 'naam ekadu', 34, 1, 69, 22, 78}
<class 'str'>
(65, 'naam ekadu', 34, 1, 69, 22, 78)
```

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

In [9]:
```
s = {1,22,34,65,78,69}

print(max(s))

print(min(s))
```

```
78
1
```

04) WAP to perform union of two sets.

In [10]:
```
s = {1,22,34,65,78,69}
s1 = {1,22,"naam ekadu"}

print(s.union(s1))
```

```
{65, 34, 1, 'naam ekadu', 69, 78, 22}
```

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

In [12]:
```
s = {1,22,34,65,78,69}
s1 = {1,22,"naam ekadu"}

print(s.intersection(s1))
```

```
{1, 22}
```

06) WAP to remove duplicates from list.

In [14]:
```
li = [1,22,34,65,78,69,34,56,33,22,1,43]

li = list(set(li))

print(li)
```

```
[65, 1, 34, 33, 69, 43, 78, 22, 56]
```

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

In [2]:
```
string = "i am mihir i am studing python"

li = string.split(" ")

s = set(li)

print(s)
```

```
{'studing', 'python', 'am', 'mihir', 'i'}
```

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

In [3]:
```
a = {1,22,34,65,78,69}
b = {1,22,"naam ekadu"}

a = a-b

print(a)
```

```
{65, 34, 69, 78}
```

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

In [19]:
```
str1 = "fuul"
str2 = "full"

if len(str1) != len(str2) or set(str1) != set(str2):
    print("The strings not are anagrams.")
else:
    if set(str1) == set(str2):
        for i in set(str1):
            if str1.count(i) != str2.count(i):
                print("The strings are not anagrams.")
                break
        else:
            print("The strings not are anagrams.")
```

```
The strings are not anagrams.
```

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

In [4]:
```
li1 = [1, 2, 3, 4, 5, 6, 7]
li2 = [7, 8, 9, 0, -1, -2, -3]
li3 = [10, 11, 12, 7, 13, 14]

print(set(li1) & set(li2) & set(li3))
```

```
{7}
```

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

In [6]:
```
str = "mihir"

vowels = set("aeiou")
count = 0

for i in str.lower():
    if i in vowels:
        count += 1

print(count)
```

```
2
```

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

In [11]:
```
str = "01011010101110"

if set(str) <= set("01"):
    print("String is binary.")
else:
    print("String is not binary.")
```

```
String is binary.
```

13) WAP to sort dictionary by key or value.

In [3]:
```
di = dict()
di[10] = 1000
di[2] = 100
di[3] = 10


di = sorted(di.values())

print(di)
```

```
[10, 100, 1000]
```

In [5]:
```
di = dict()
di[10] = 1000
di[2] = 100
di[3] = 10


di = sorted(di.items())

print(di)
```

```
[(2, 100), (3, 10), (10, 1000)]
```

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

In [13]:
```
di = dict()
di[0] = 1000
di[1] = 100
di[2] = 10
di[3] = 1


di = sum(di.values())

print(di)
```

```
1110
```

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 [15]:
```
di = dict()
di["a"] = 1000
di["b"] = 100
di["c"] = 10
di["d"] = 1

key = "a"

if key in di:
    print(di[key])
else:
    print("key not found")
```

```
1000
```

In [ ]:
```

```

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

Python Programming - 2301CS404

Lab - 8

KACHA MIHIR | 23010101118 | 28/1/2025

User Defined Function

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

In [6]:
```
m=float(input("enter weight: "))
h=float(input("enter height: "))

def bmi(m,h):
    return (m/(h**2));

print(bmi(m,h))
```

```
enter weight: 80
enter height: 1.86
23.12406058503873
```

02) Write a function that add first n numbers.

In [8]:
```
n=int(input("enter a number: "))

def sumofn(n):
    return (n*(n+1))/2;


print(sumofn(n))
```

```
enter a number: 2
3.0
```

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

In [4]:
```
n=int(input("enter a number: "))


def primeornot(n):
    for i in range(2,(n//2)+1,1):
        if(n%i==0):
            return 0;
    return 1;

print(primeornot(n))
```

```
enter a number: 11
1
```

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

In [3]:
```
import math
n=int(input("enter a starting number: "))
m=int(input("enter a ending number: "))

list=[]

def primelist(n,m):
    for i in range(n,m):
        if(primeornot(i)==1):
            list.append(i);


primelist(n,m)
print(list)
```

```
enter a starting number: 5
enter a ending number: 11
[5, 7]
```

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

In [30]:
```
def palindromeornot(str):
    if(str == str[::-1]):
        return True
    else:
        return False

palindromeornot(input("enter a string: "))
```

```
enter a string: 1122
```

Out[30]:
```
False
```

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

In [12]:
```
list=[1,24,45,43,23,56]



def sumoflist(list):
    sum=0
    for i in list:
        sum+=i
    return sum

print(sumoflist(list))
```

```
192
```

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

In [3]:
```
li=[(12,23),(45,44),(56,67)]

def sumoftp(li):
    sum = 0
    for i in li:
        sum+=i[0]
    return sum

print(sumoftp(li))
```

```
113
```

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

In [17]:
```
def fibo(n):
    if(n==0):
        return 0
    if(n==1):
        return 1
    return fibo(n-1)+fibo(n-2)

term = int(input('enter nth term'))
print(fibo(term-1))
```

```
enter nth term3
1
```

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

rollno.

In [4]:
```
def getstu(di):
    return  di.get(rollno)

di={101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}

rollno = 103

print(getstu(di))
```

```
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 [6]:
```
scores = [200, 456, 300, 100, 234, 678]

def sumofscore(li):
    sum=0
    for i in li:
        if(i%10==0):
            sum+=i
    return sum
print(sumofscore(scores))
```

```
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 [8]:
```
def invertdictionary(di):
    return{v:k for k,v in di.items()}

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

print(invertdictionary(di))
```

```
{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 [10]:
```
def check(str):
    str==str.lower()
    alphabet = 'qwertyuioplkjhgfdsazxcvbnm'
    return all(char in str for char in alphabet)

str='the quick brown fox jumps over the lazy dog'
print(check(str))
```

```
True
```

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 [3]:
```
def countUpperLower(str):
    upper = sum(1 for char in str if char.isupper())
    lower = sum(1 for char in str if char.islower())
    return upper,lower

str='efgwedfHBGLHG'

print(countUpperLower(str))
```

```
(6, 7)
```

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

In [1]:
```
ans = lambda a,b: a if a<b else b

print(ans(5,9))
```

```
5
```

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

In [2]:
```
name=['ronit', 'shatrunjay', 'mihir', 'prem', 'vatsal', 'rudraraj', 'yashraja']

selected_name=filter(lambda x: len(x)>7, name)
print(list(selected_name))
```

```
['shatrunjay', 'rudraraj', 'yashraja']
```

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

In [3]:
```
name=map(lambda x: x[0].upper()+x[1:], name)

print(list(name))
```

```
['Ronit', 'Shatrunjay', 'Mihir', 'Prem', 'Vatsal', 'Rudraraj', 'Yashraja']
```

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 [4]:
```python
# 1. Positional Arguments
def positional_args(a, b):
    return f"Positional Args: a = {a}, b = {b}"

# 2. Keyword Arguments
def keyword_args(a, b):
    return f"Keyword Args: a = {a}, b = {b}"

# 3. Default Arguments
def default_args(a, b=10):
    return f"Default Args: a = {a}, b = {b}"

# 4. Variable Length Positional (*args) & Variable Length Keyword Arguments (**
def var_length_args(*args, **kwargs):
    return f"Var Length Positional Args: {args}, Var Length Keyword Args: {kwar

# 5. Keyword-Only & Positional-Only Arguments
def special_args(a, /, b, *, c):
    return f"Positional-Only a = {a}, Normal b = {b}, Keyword-Only c = {c}"

# Calling the UDFs
print(positional_args(1, 2))
print(keyword_args(a=3, b=4))
print(default_args(5))
print(var_length_args(1, 2, 3, x=10, y=20))
print(special_args(1, b=2, c=3))
```

```
Positional Args: a = 1, b = 2
Keyword Args: a = 3, b = 4
Default Args: a = 5, b = 10
Var Length Positional Args: (1, 2, 3), Var Length Keyword Args: {'x': 10,
'y': 20}
Positional-Only a = 1, Normal b = 2, Keyword-Only c = 3
```

In [ ]:

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

Python Programming - 2301CS404

Lab - 9

KACHA MIHIR | 23010101118 | 4/2/2025

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 [14]:
```
fp = open('../headder.txt','r')
print(fp.read())
fp.seek(0)
print('after this readline')
print(fp.readline())
fp.seek(0)
li = fp.readlines()
print(li[1])
fp.close
```

```
this is header text
this is second line
after this readline
this is header text

this is second line
```

Out[14]:
```
<function TextIOWrapper.close()>
```

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

In [30]:
```
fp = open('new.txt','a')
fp.write("this is new txt file\n")
fp.close()
```

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

In [34]:
```
fp = open('new.txt')
for i in range(5):
    print(fp.readline())
fp.close()
```

```
this is new txt file

this is new txt file

this is new txt file

this is new txt file

this is new txt file
```

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

In [2]:
```
with open('new.txt', 'r') as file:
    words = file.read().split()
    max_length = max(len(word) for word in words)
    longest_words = [word for word in words if len(word) == max_length]
    print("Longest word(s):", longest_words)
```

```
Longest word(s): ['this', 'file', 'this', 'file', 'this', 'file', 'this', 'fi
le', 'this', 'file', 'this', 'file']
```

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

In [38]:
```
fp = open('new.txt')
line = 0
for i in fp:
    line+=1
print(line)
fp.close()

fp = open('new.txt')
word=0
char=0
str = fp.read()
char = len(str)
li = str.split()
word = len(li)
print(word)
print(char)
fp.close
```

```
6
30
126
```

Out[38]:
```
<function TextIOWrapper.close()>
```

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

In [1]:
```
fpr = open('new.txt','r')
fpw = open('copy.txt','w')

fpw.write(fpr.read())
fpr.close()
fpw.close()
```

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

In [40]:
```
fp = open('new.txt')
char=0
str = fp.read()
char = len(str)
print(char)
fp.close
print('size = ',char)
```

```
125
size =  125
```

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

In [44]:
```
def frequency(fp,word):
        count=0
        str=fp.read()
        li=str.lower().split()

        for i in li:
            if(i==word.lower()):
                count+=1
        print(count)

fp=open('copy.txt')
frequency(fp,'is')
```

```
6
```

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 [3]:
```
file_path = "marks.txt"

scores = []
for i in range(1, 6):
    score = int(input(f"Enter marks for subject {i}: "))
    scores.append(score)

with open(file_path, 'w') as file:
    file.write(" ".join(map(str, scores)))

with open(file_path, 'r') as file:
    fetched_scores = list(map(int, file.read().split()))

highest_score = max(fetched_scores)

print("\nStored Scores:", fetched_scores)
print("Highest Score:", highest_score)
```

```
Enter marks for subject 1: 3
Enter marks for subject 2: 4
Enter marks for subject 3: 5
Enter marks for subject 4: 6
Enter marks for subject 5: 7

Stored Scores: [3, 4, 5, 6, 7]
Highest Score: 7
```

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

(Note: each number should be in new line)

In [2]:
```
import math
def write_primenumbers(fp):
    primes=[]
    n=2
    while len(primes)<100:
        if isPrime(n):
            primes.append(n)
            fp.write(str(n)+'\n')
        n+=1

def isPrime(n):
    for i in range(2,int(math.sqrt(n))+1):
        if n%i==0:
            return False
    return True

fp=open('primenumbers.txt', 'a+')
write_primenumbers(fp)
print(fp.read())
fp.close()
```

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

In [3]:
```
fp1=open('new.txt')
fp2=open('copy.txt')
fp3=open('mearge.txt','a')
fp3.write(fp1.read())
fp3.write(fp2.read())
fp1.close()
fp2.close()
fp3.close()
```

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

In [4]:
```
fp=open("new.txt", "r+")

new_content=fp.read().replace('this', 'that')

fp=open('new.txt', 'w')

fp.write(new_content)

fp.close()
```

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

In [5]:
```
fp=open('mearge.txt')

print ('Initial position:', fp.tell())

fp.seek(0,0)

print ('Position after seeking to beginning:', fp.tell())

fp.seek(5,0)

print ('Position after seeking 5 bytes from current position:', fp.tell())

fp.seek(0,2)

print ('Position after seeking to end:', fp.tell())



fp.close()
```

```
Initial position: 0
Position after seeking to beginning: 0
Position after seeking 5 bytes from current position: 5
Position after seeking to end: 528
```

In [ ]:
```

```

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

Python Programming - 2301CS404

Lab - 10

KACHA MIHIR | 23010101118 | 11/2/2025

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 [4]:
```
try:
    a = 10
    b = 0
    c = a /  b
    print(c)
except ZeroDivisionError:
    print("Error: Division by Zero")


try:
    a = 10
    b = 'v'
    c = a /  b
    print(c)
except TypeError:
    print("Error: Invalid Data Type")


try:
    a = int('frds')
    print(a)
except ValueError:
    print("Error: Invalid Input")
```

```
Error: Invalid Data Type
Error: Invalid Input
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [6]:
```
try:
    lst = [1, 2, 3, 4]
    print(lst[5])

except IndexError:
    print("Error: Index out of range")

try:
    dict = {"1": 1, "2": 2, "3": 3}
    print(dict["5"])

except KeyError:
    print("Error: Key not found")
```

```
Error: Index out of range
Error: Key not found
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [7]:
```
try:
    file = open('python.txt', 'r')
    print(file.read())
except FileNotFoundError:
    print("The file does not exist.")


try:
    import pythonmodule
    print("Module imported successfully.")

except ModuleNotFoundError:
    print("The module does not exist.")
```

```
The file does not exist.
The module does not exist.
```

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

In [8]:
```
try:
    a = 10
    b = 'v'
    c = a / b
    print(c)
    print("Division completed successfully.")
    a = 10
    b = 0
    c = a /  b
    print(c)
    print("Division completed successfully.")
    a = int('frds')
    print(a)
    lst = [1, 2, 3, 4]
    print(lst[5])
    dict = {"1": 1, "2": 2, "3": 3}
    print(dict["5"])
    file = open('python.txt', 'r')
    print(file.read())
    import pythonmodule
    print("Module imported successfully.")

    file = open('python.txt', 'r')
    print(file.read())
    print("File opened successfully.")



except Exception as e:
    print("An error occurred:", e)
```

```
An error occurred: unsupported operand type(s) for /: 'int' and 'str'
```

05) WAP to demonstrate else and finally block.

In [12]:
```
try:
    a = 10
    b = 0
    c = a / b
    print(c)
    print("Division completed successfully.")

except ZeroDivisionError:
    print("Error: Division by Zero")
else:
    print(c)
finally:
    print("Finally block executed.")
```

```
Error: Division by Zero
Finally block 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 [20]:
```
print("enter a list of grades separated by commas")
grades = input()

grades = grades.split(',')
print(grades)
try:
    grades = [int(grade) for grade in grades]
    print(grades)
    total = sum(grades)
    print(total)

except ValueError:
    print("Error: Invalid Input")
```

```
enter a list of grades separated by commas
['324', 'fgdh', '43']
Error: Invalid Input
```

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

In [17]:
```
def divide(a,b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed"


print(divide(4,2))

print(divide(2,0))
```

```
2.0
Error: Division by zero is not allowed
```

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 [24]:
```
try:
    def getage(a):
        if a < 18:
            raise ValueError("Enter Valid Age")
        else:
            print("Age:", a)

finally:
    print("Finally block executed.")

getage(34)
getage(17)
```

```
Finally block executed.
Age: 34

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[24], line 12
      9     print("Finally block executed.")
     11 getage(34)
---> 12 getage(17)

Cell In[24], line 4, in getage(a)
      2 def getage(a):
      3     if a < 18:
----> 4         raise ValueError("Enter Valid Age")
      5     else:
      6         print("Age:", a)

ValueError: 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 [1]:
```
class InvalidUsernameError(Exception):
    def __init__(self, message):
        super().__init__(message)

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

validate_username("mihir")
```

```
Username: mihir
```

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 [29]:
```
class NegativeNumberError(Exception):
    def __init__(self, message):
        super().__init__(message)

def root(number):
    if number < 0:
        raise NegativeNumberError("Cannot calculate the square root of a negati
    else:
        return number ** 0.5

root(3)
```

Out[29]:
```
1.7320508075688772
```

In [ ]:
```

```

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

Python Programming - 2301CS404

Lab - 11

KACHA MIHIR | 23010101118 | 18/2/2025

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 [1]:
```
import module as mk
print(mk.add(3,4))
print(mk.multy(2,3))
```

```
7
6
```

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

In [1]:
```
import random as r;
str = "mihr kacha"
print(r.choice(str))
```

```
k
```

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

In [2]:
```
li=[234,453,3455,4675,43532,45,54765,4345,6324235,3467457,68,764567345,324,2346
print(r.choice(li))
```

```
4675
```

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

In [12]:
```
import random as r;
r.seed(7)
roll=r.randint(1,6)
print(roll)
```

```
3
```

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

In [14]:
```
nums = [r.choice(range(100,1000,5)) for i in range(3)]
print(nums)
```

```
[160, 190, 785]
```

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 [15]:
```
lotteryTickets = [r.choice(range(100000,999999)) for i in range(100)]
winners = [r.choice(lotteryTickets) for i in range(2)]
print("winner = ",winners[0],"runner up = ",winners[1])
```

```
winner =  955770 runner up =  208061
```

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

In [7]:
```
from datetime import  datetime as dt
print(dt.now())
```

```
2025-02-18 10:49:48.925828
```

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

In [8]:
```
from datetime import  timedelta as td
today = dt.now()
newday = today -td(days=7)
print(newday)
```

```
2025-02-11 10:50:42.023240
```

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

In [15]:
```
date1  = dt.now()
date2 = dt.now() - td(weeks=4)
diff =  (date1 -date2).days

print(diff)
```

```
28
```

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

In [22]:
```
date = dt.now() - td(weeks=4)
day=date.strftime("%A")
print(day)
```

```
Tuesday
```

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

In [23]:
```
from datetime import time

Time = time(11, 34, 56)

print("hour =", Time.hour)
print("minute =", Time.minute)
print("second =", Time.second)
print("microsecond =", Time.microsecond)
```

```
hour = 11
minute = 34
second = 56
microsecond = 0
```

In [25]:
```
from datetime import datetime
from pytz import timezone

format = "%Y-%m-%d %H:%M:%S %Z%z"

# Current time in UTC
now_utc = datetime.now(timezone('UTC'))
print(now_utc.strftime(format))

timezones = ['Asia/Kolkata', 'Europe/Kiev', 'America/New_York']

for tzone in timezones:

    # Convert to Asia/Kolkata time zone
    now_asia = now_utc.astimezone(timezone(tzone))
    print(now_asia.strftime(format))
```

```
2025-02-18 05:36:43 UTC+0000
2025-02-18 11:06:43 IST+0530
2025-02-18 07:36:43 EET+0200
2025-02-18 00:36:43 EST-0500
```

In [ ]:
```
```

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

In [28]:
```
import math
print(math.pi)

print (math.isnan(math.pi))
print (math.isnan(math.e))
print (math.isnan(float('nan')))


print ("The value of log 2 with base 3 is : ", end="")
print (math.log(2,3))
print ("The value of log2 of 16 is : ", end="")
print (math.log2(16))
print ("The value of log10 of 10000 is : ", end="")
print (math.log10(10000))
```

```
3.141592653589793
False
False
True
The value of log 2 with base 3 is : 0.6309297535714574
The value of log2 of 16 is : 4.0
The value of log10 of 10000 is : 4.0
```

In [ ]:

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

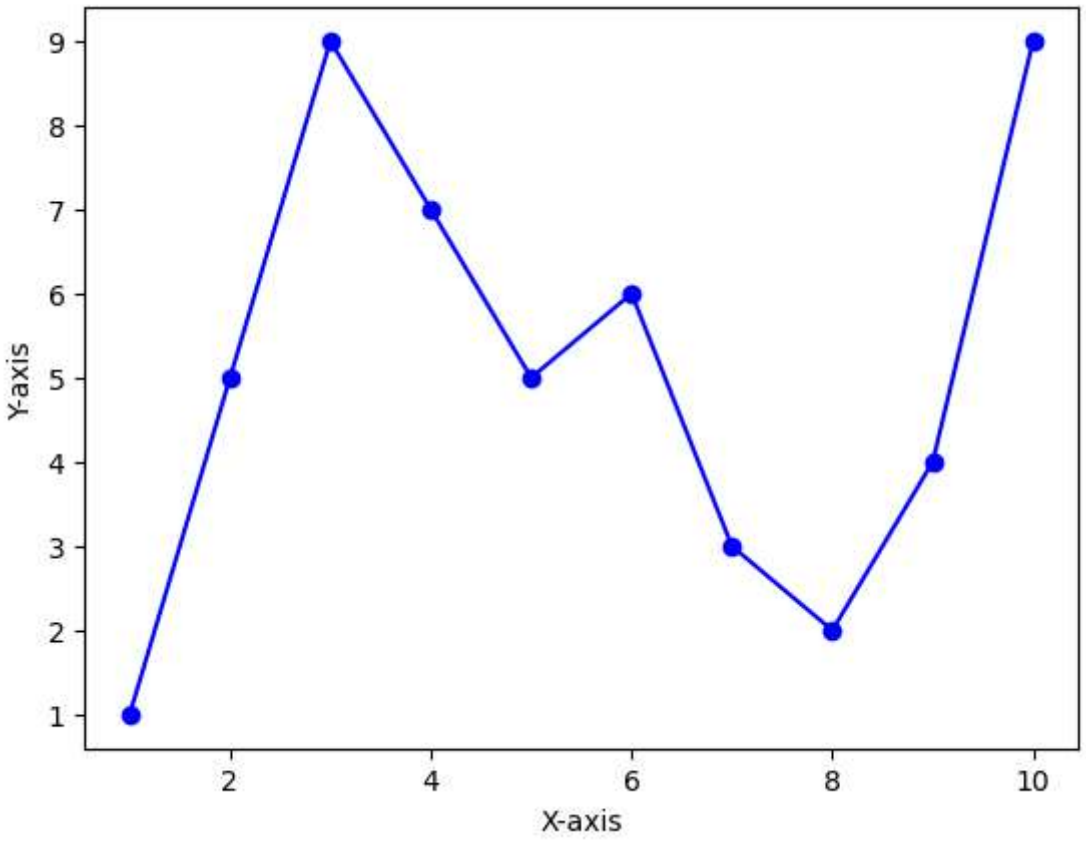
Python Programming - 2301CS404

Lab - 12

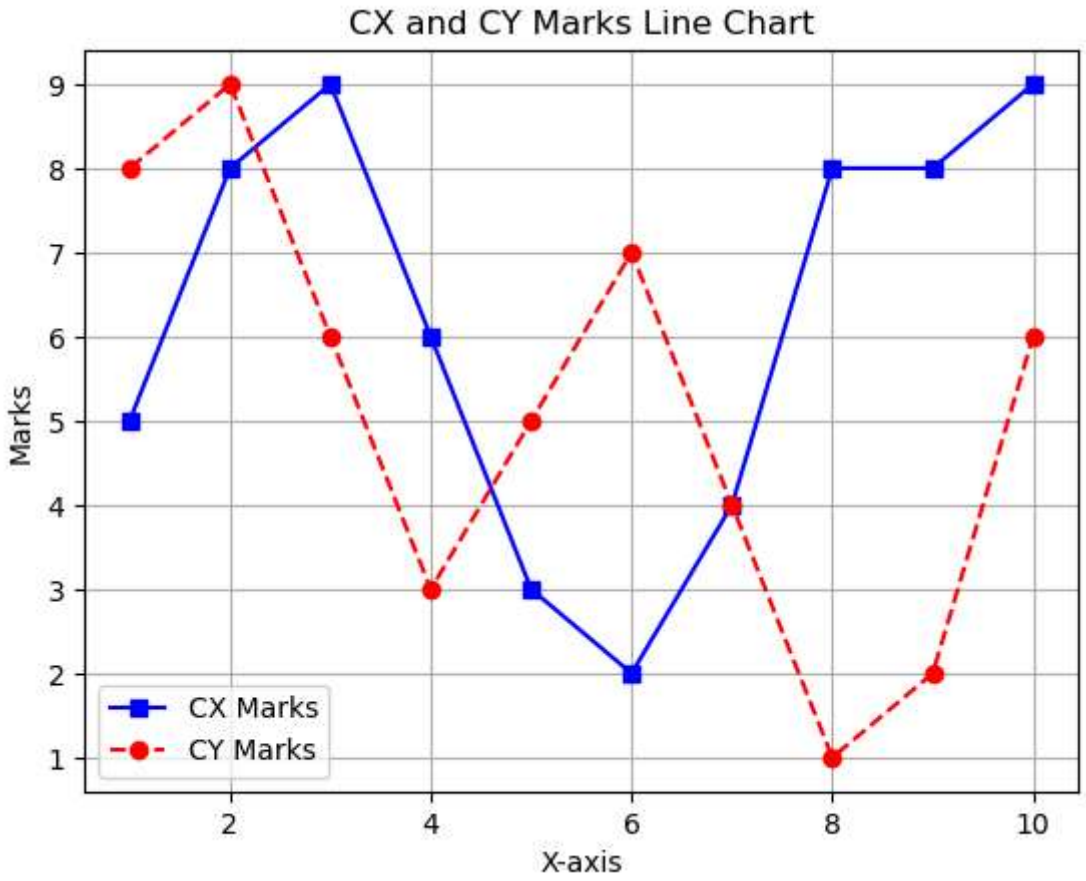
KACHA MIHIR | 23010101118 | 25/2/2025

In [5]:
```
#import matplotlib below
import matplotlib.pyplot as plt
```

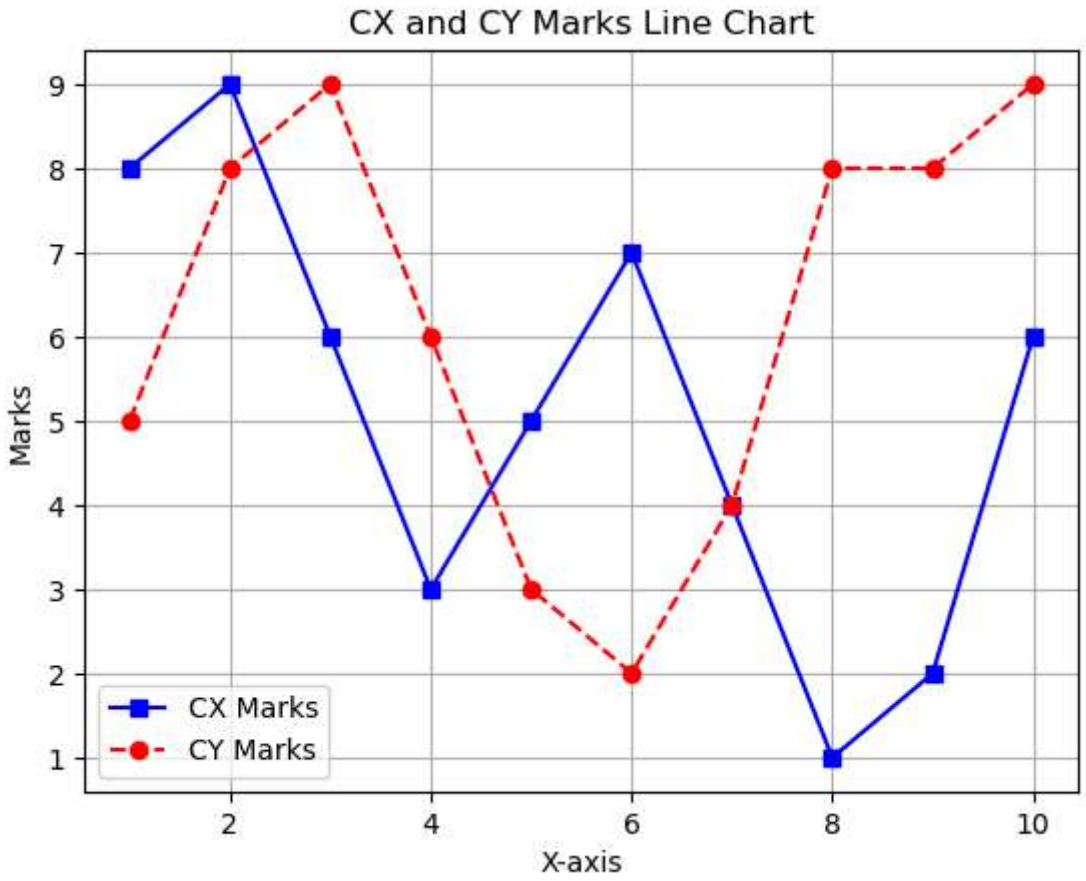
In [6]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

plt.plot(x, y, marker='o', linestyle='-', color='b')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

plt.show()
# write a code to display the line chart of above x & y
```



In [40]:
```
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]

plt.plot(x, cxMarks, marker='s', linestyle='-', color='b', label='CX Marks')
plt.plot(x, cyMarks, marker='o', linestyle='--', color='r', label='CY Marks')
plt.xlabel('X-axis')
plt.ylabel('Marks')
plt.title('CX and CY Marks Line Chart')
plt.legend()
plt.grid(True)


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



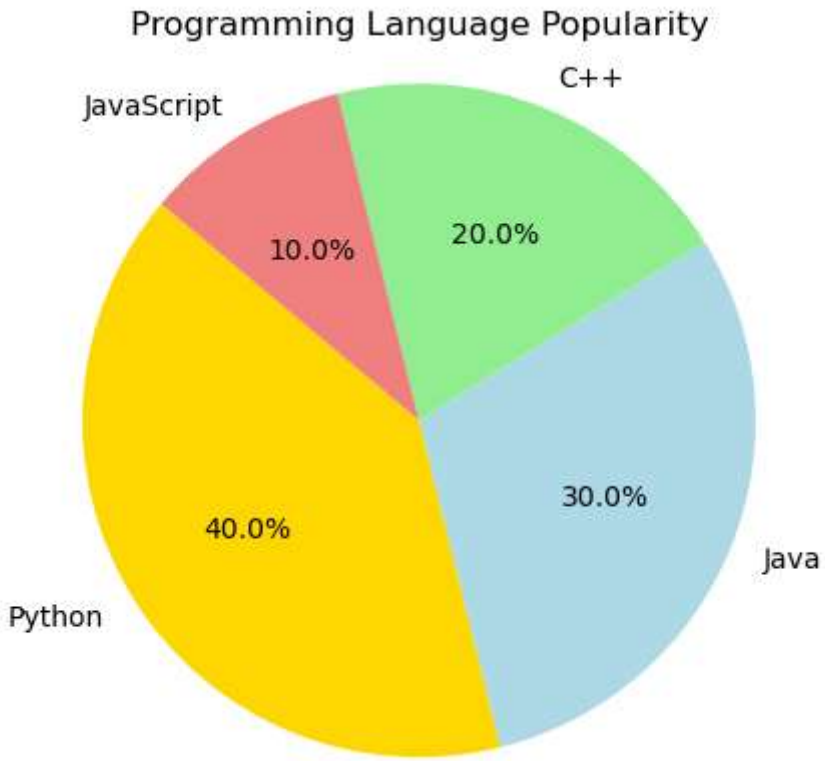
In [17]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x, cxMarks, marker='s', linestyle='-', color='b', label='CX Marks')
plt.plot(x, cyMarks, marker='o', linestyle='--', color='r', label='CY Marks')
plt.xlabel('X-axis')
plt.ylabel('Marks')
plt.title('CX and CY Marks Line Chart')
plt.legend()
plt.grid(True)


plt.show()

# write a code to generate below graph
```



In [ ]:

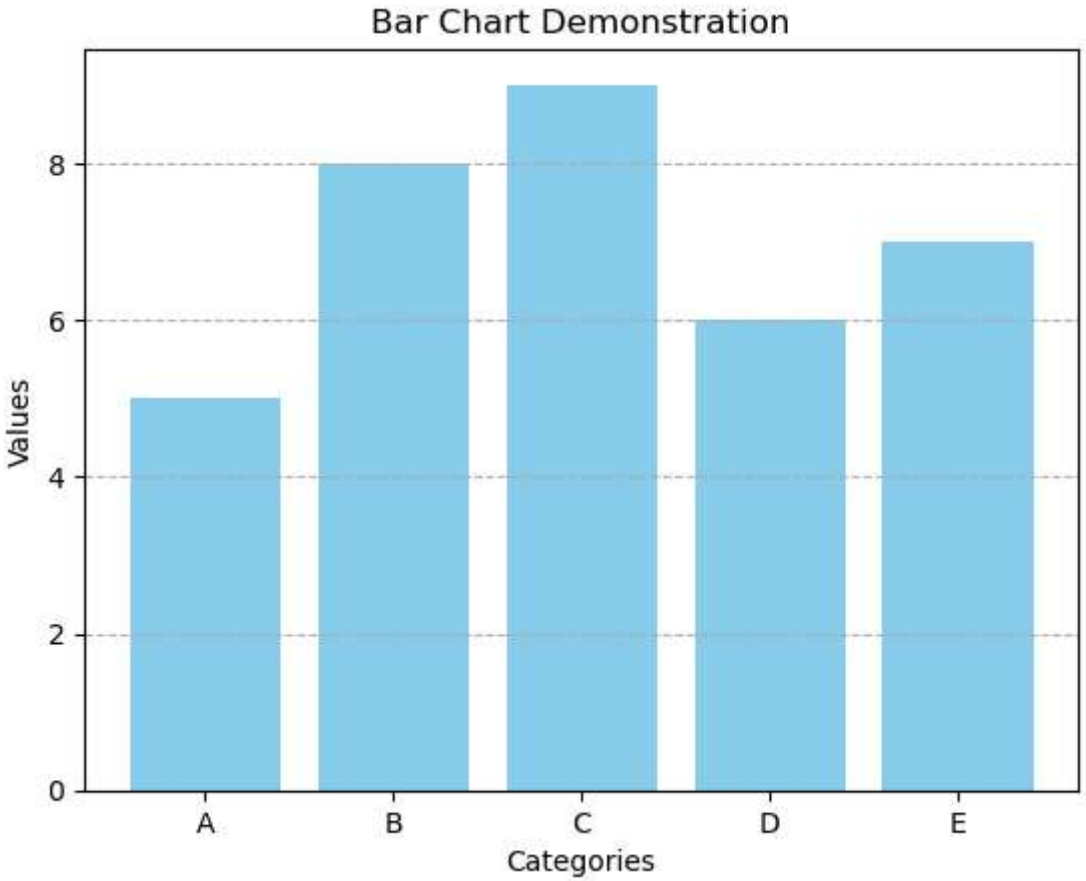
04) WAP to demonstrate the use of Pie chart.

In [37]:
```
labels = ['Python', 'Java', 'C++', 'JavaScript']
sizes = [40, 30, 20, 10]
colors = ['gold', 'lightblue', 'lightgreen', 'lightcoral']

plt.pie(sizes, labels=labels, colors=colors,  autopct='%1.1f%%', startangle=140
plt.axis('equal')
plt.title('Programming Language Popularity')

plt.show()
```



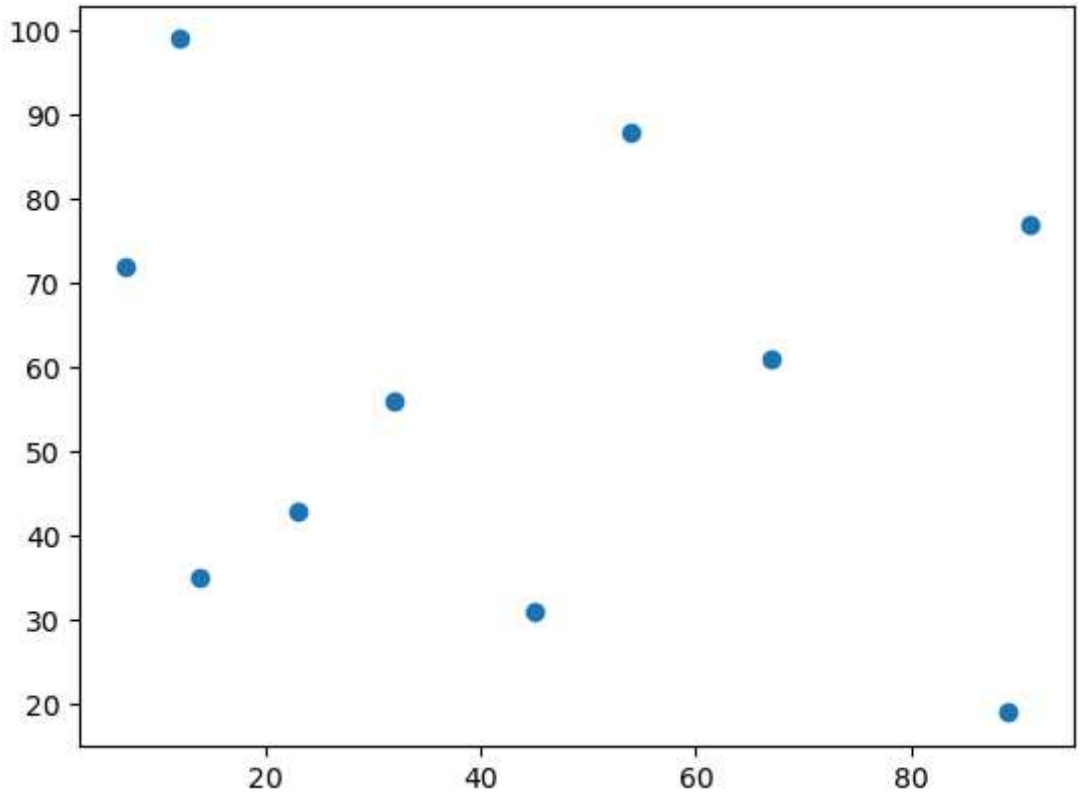
05) WAP to demonstrate the use of Bar chart.

In [44]:
```
categories = ['A', 'B', 'C', 'D', 'E']
values = [5, 8, 9, 6, 7]

plt.bar(categories, values, color='skyblue')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart Demonstration')
plt.grid(axis='y', linestyle='--')

plt.show()
```



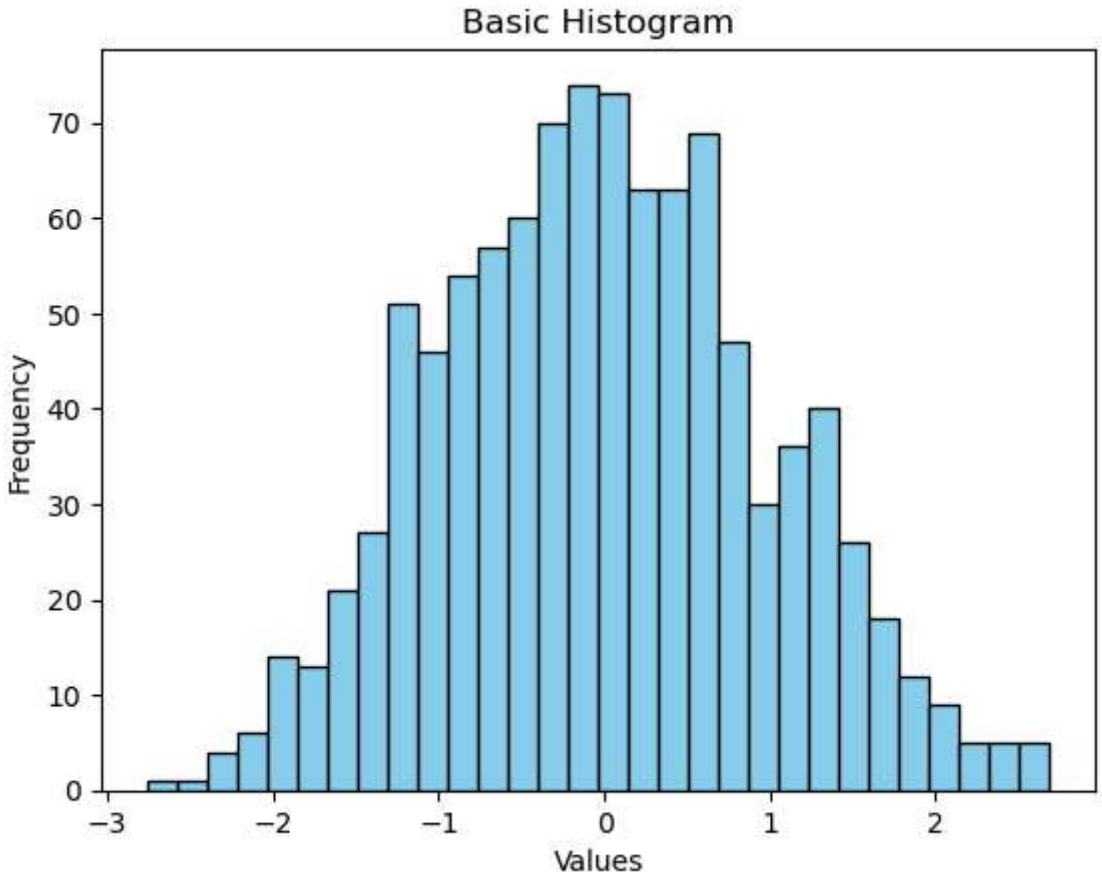
06) WAP to demonstrate the use of Scatter Plot.

In [25]:
```
import numpy as np

x = np.array([12, 45, 7, 32, 89, 54, 23, 67, 14, 91])
y = np.array([99, 31, 72, 56, 19, 88, 43, 61, 35, 77])

plt.scatter(x, y)
plt.show()
```



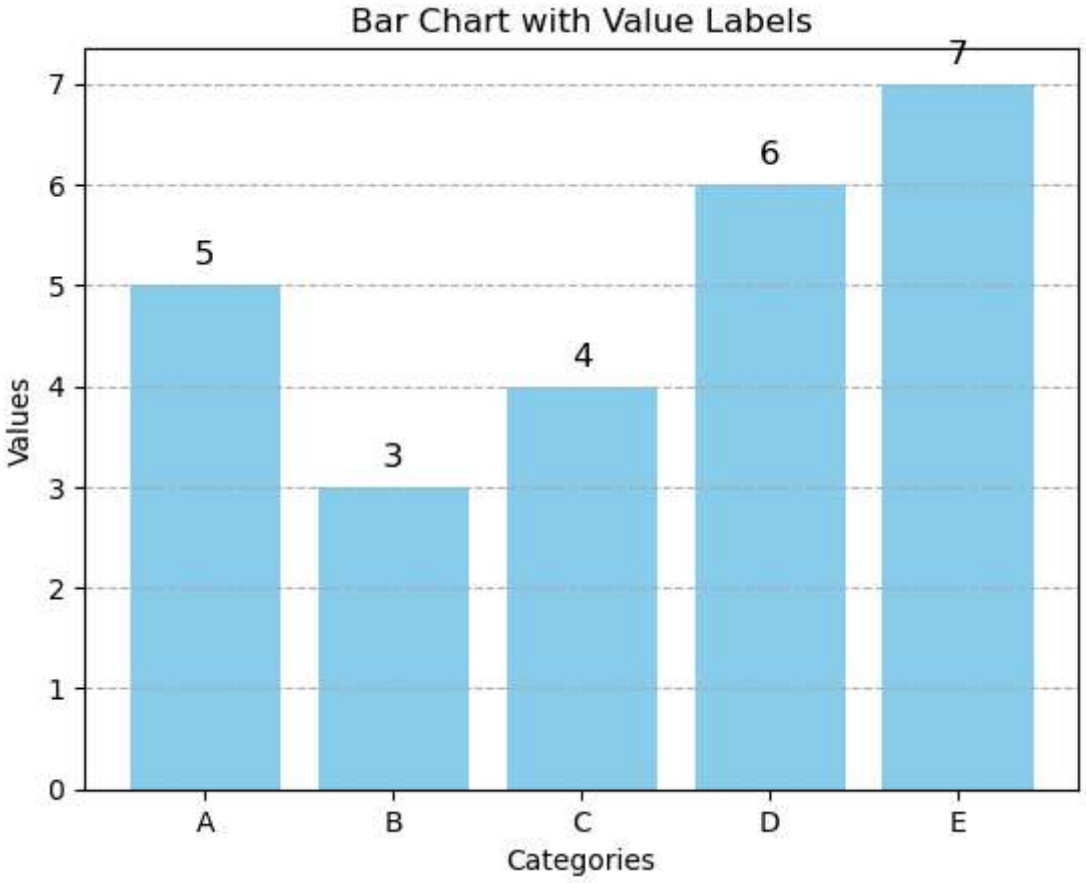
07) WAP to demonstrate the use of Histogram.

In [54]:
```
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='skyblue', edgecolor='black')

plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')

plt.show()
```



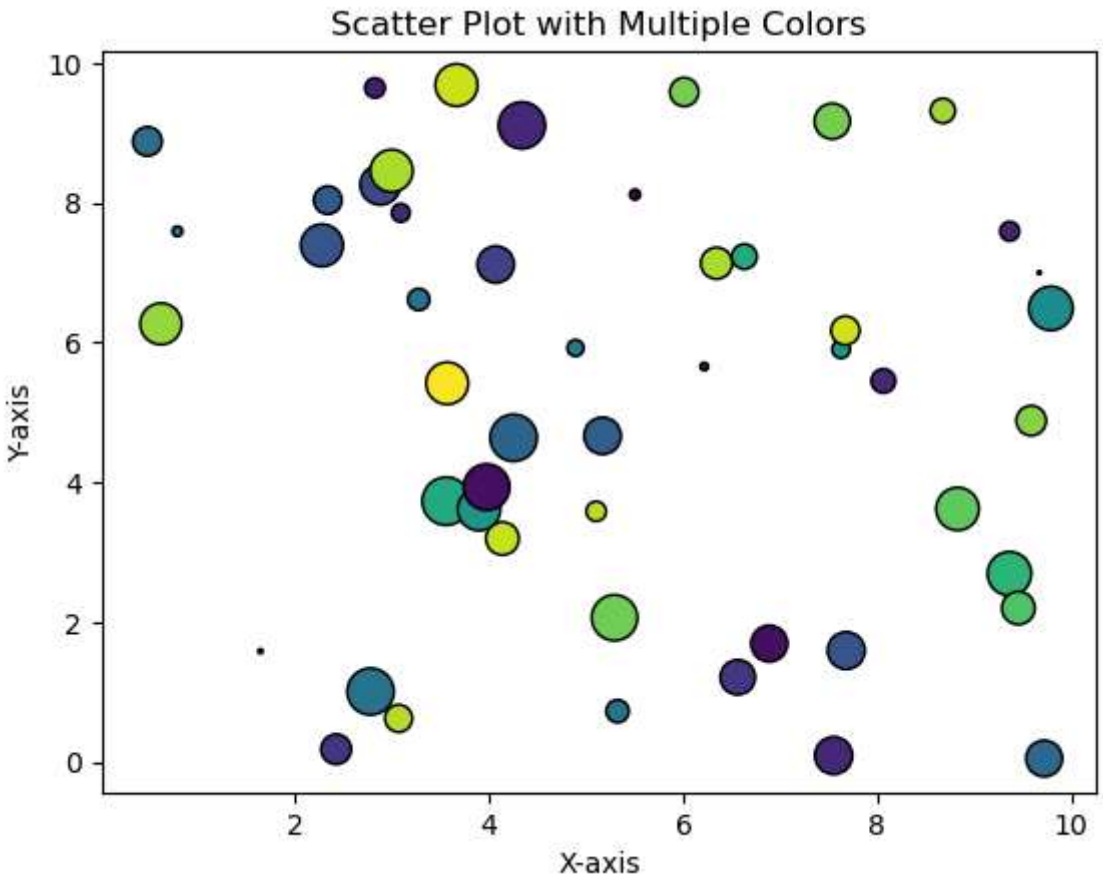
08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [29]:
```
categories = ['A', 'B', 'C', 'D', 'E']
values = [5, 3, 4, 6, 7]


plt.bar(categories, values, color='skyblue')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart with Value Labels')
plt.grid(axis='y', linestyle='--')


for i in range(len(values)):
    plt.text(i, values[i] + 0.2, str(values[i]), ha='center', fontsize=12)



plt.show()
```



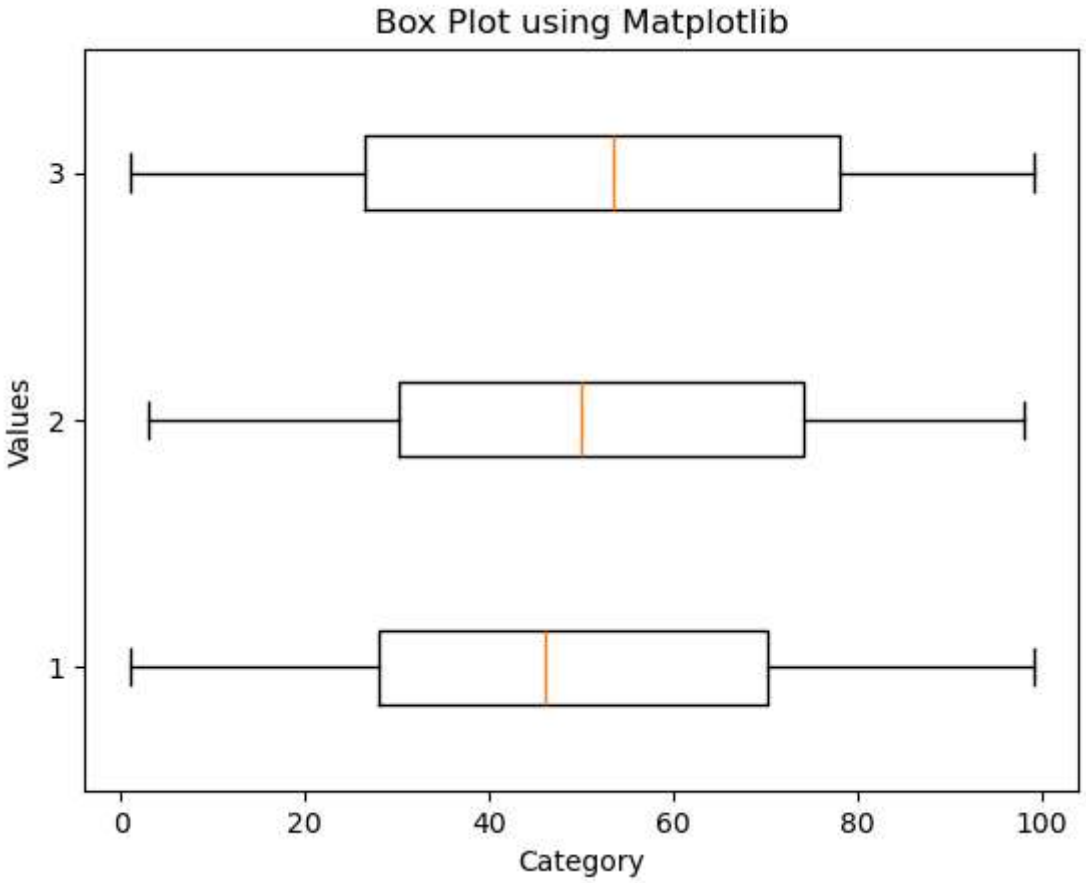
09) WAP create a Scatter Plot with several colors in Matplotlib?

In [31]:
```
import numpy as np


x = np.random.rand(50) * 10
y = np.random.rand(50) * 10
colors = np.random.rand(50)
sizes = np.random.rand(50) * 300


plt.scatter(x, y, c=colors, s=sizes,  edgecolors='black')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot with Multiple Colors')



# Show the chart
plt.show()
```



10) WAP to create a Box Plot.

In [52]:
```
import numpy as np

data = [np.random.randint(1, 100, 100) for i in range(3)]

plt.boxplot(data,vert=False)

plt.title('Box Plot using Matplotlib')
plt.xlabel('Category')
plt.ylabel('Values')

plt.show()
```



In [ ]:
```

```

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

Python Programming - 2301CS404

Lab - 13

KACHA MIHIR | 23010101121 | 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

    def display_info(self):
        print("Name:",self.name)
        print("Age:",self.age)
        print("Grade:",self.grade)

student1 = Student("Mihir", 20, "O")
student1.display_info()  # Output: Name: John Doe, Age: 18, Grade: A
```

```
Name: Mihir
Age: 20
Grade: O
```

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

In [7]:
```
class BankAccount:
    def __init__(self, Account_No, User_Name, Email, Account_Type, Account_Bala
        self.Account_No = Account_No
        self.User_Name = User_Name
        self.Email = Email
        self.Account_Type = Account_Type
        self.Account_Balance = Account_Balance

    def GetAccountDetails(self):
        return self.Account_No, self.User_Name, self.Email, self.Account_Type,

    def DisplayAccountDetails(self):
        print("Account Number:", self.Account_No)
        print("User Name:", self.User_Name)
        print("Email:", self.Email)
        print("Account Type:", self.Account_Type)
        print("Account Balance:", self.Account_Balance)

account1 = BankAccount(1234567890, "Mihir", "mihir11@gmail.com", "Savings", 700

account_details = account1.GetAccountDetails()

account1.DisplayAccountDetails()
```

```
Account Number: 1234567890
User Name: Mihir
Email: mihir11@gmail.com
Account Type: Savings
Account Balance: 70000000
```

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

In [8]:
```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi* (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

circle1 = Circle(10)

print("Area:", circle1.area())
print("Perimeter:", circle1.perimeter())
```

```
Area: 314.1592653589793
Perimeter: 62.83185307179586
```

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

In [9]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_info(self, name=None, age=None, salary=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if salary:
            self.salary = salary

    def display_info(self):
        print("Name:", self.name)
        print("Age:", self.age)
        print("Salary:", self.salary)

employee1 = Employee("Mihir Kacha", 20, 70000)

employee1.display_info()

employee1.update_info(salary=100000)

employee1.display_info()

employee1.update_info(age=21)

employee1.display_info()
```

```
Name: Mihir Kacha
Age: 20
Salary: 70000
Name: Mihir Kacha
Age: 20
Salary: 100000
Name: Mihir Kacha
Age: 21
Salary: 100000
```

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

In [2]:
```
class BankAccount:
    def __init__(self, account_number, initial_balance):
        self.account_number = account_number
        self.balance = initial_balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)
        self.check_balance()

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
            self.check_balance()
        else:
            print("Insufficient balance.")

    def check_balance(self):
        print("Current Balance:", self.balance)

account1 = BankAccount(1234567890, 10000)

account1.deposit(5000)

account1.withdraw(2000)

account1.withdraw(3000)

account1.check_balance()
```

```
Deposited: 5000
Current Balance: 15000
Withdrawn: 2000
Current Balance: 13000
Withdrawn: 3000
Current Balance: 10000
Current Balance: 10000
```

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 [3]:
```
class Product:
    def __init__(self, item_name, price, quantity):
        self.item_name = item_name
        self.price = price
        self.quantity = quantity

    def add_item(self, quantity):
        self.quantity += quantity
        print("Item added:", self.item_name)
        self.display_info()

    def remove_item(self, quantity):
        if quantity <= self.quantity:
            self.quantity -= quantity
            print("Item removed:", self.item_name)
            self.display_info()
        else:
            print("Not enough items in stock.")

    def update_price(self, new_price):
        self.price = new_price
        print("Price updated:", self.item_name)
        self.display_info()

    def display_info(self):
        print("Item Name:", self.item_name)
        print("Price:", self.price)
        print("Quantity:", self.quantity)

product1 = Product("Laptop", 10000, 5)

product1.add_item(3)

product1.remove_item(2)

product1.update_price(12000)

product1.display_info()
```

```
Item added: Laptop
Item Name: Laptop
Price: 10000
Quantity: 8
Item removed: Laptop
Item Name: Laptop
Price: 10000
Quantity: 6
Price updated: Laptop
Item Name: Laptop
Price: 12000
Quantity: 6
Item Name: Laptop
Price: 12000
Quantity: 6
```

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

In [12]:
```python
class MyClass:
    def __init__(self, attribute1, attribute2):
        self.attribute1 = attribute1
        self.attribute2 = attribute2

    def display_attributes(self):
        print(f'Attribute 1: {self.attribute1}')
        print(f'Attribute 2: {self.attribute2}')

my_object = MyClass('Hello', 'World')

my_object.display_attributes()
```

```
Attribute 1: Hello
Attribute 2: World
```

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 [10]:
```
class StudentKit:
    PrincipalName='Mr.ABC'
    def __init__(self,name):
        self.name=name
        self.attendance=0
        self.certificate=0

    def attendance_method(self,days):
        self.attendance=days
        print(f'{self.name} has attended {self.attendance} days in class.')

    def certificate_method(self,days):
        self.certificate=days
        print(f'{self.name} has obtained {self.certificate} days of certificate

    def display_info(self):
        print(f'Student Name: {self.name}\nPrincipal Name: {self.PrincipalName}

student1=StudentKit('Mihir Kacha')

student1.attendance_method(25)

student1.certificate_method(30)

student1.display_info()
```

```
Mihir Kacha has attended 25 days in class.
Mihir Kacha has obtained 30 days of certificate.
Student Name: Mihir Kacha
Principal Name: Mr.ABC
Attendance: 25 days
Certificate: 30 days
```

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

In [11]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add_time(self, other_time):
        new_minute = self.minute + other_time.minute
        new_hour = self.hour + other_time.hour + new_minute // 60
        new_minute = new_minute % 60
        return Time(new_hour, new_minute)

    def display_time(self):
        print(f"{self.hour:02d}:{self.minute:02d}")

time1 = Time(10, 30)

time2 = Time(5, 45)

sum_time = time1.add_time(time2)

sum_time.display_time()
```

```
16:15
```

In [ ]:
```

```

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

Python Programming - 2301CS404

Lab - 13

KACHA MIHIR | 23010101121 | 11/03/2025

Continued..

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

In [1]:
```
class rectangle:
    def __init__(self,len,breadth):
        self.len=len
        self.breadth=breadth

def  area(obj):
    area=obj.len*obj.breadth
    print(area)

r1 = rectangle(len=10,breadth=10)

area(r1)
```

```
100
```

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 [6]:
```
class square:
    def __init__(self,len,breadth):
        self.len=len
        self.breadth=breadth

def output(printable):
    print(printable)

def area(obj):
    area=obj.len*obj.breadth
    output(area)

s1=square(20,20)

area(s1)
```

```
400
```

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 [17]:
```
class rectangle:
    def __init__(self,len,breadth):
        self.len=len
        self.breadth=breadth

    @classmethod
    def createrectangle(rec,len,breadth):
        if(len==breadth):
            print('this is saquare')
            return None
        else:
            return rec(len,breadth)


def output(printable):
    print(printable)

def area(obj):
    area=obj.len*obj.breadth
    output(area)


r1 = rectangle.createrectangle(20,20)

if r1:
    area(r1)
```

```
this is saquare
```

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 [19]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

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

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


sq = Square(8)
print(f"Side of square: {sq.get_side()}")
print(f"Area of square: {sq.calculate_area()}")

sq.set_side(11)
print(f"New side of square: {sq.get_side()}")
print(f"New area of square: {sq.calculate_area()}")
```

```
Side of square: 8
Area of square: 64
New side of square: 11
New area of square: 121
```

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 [20]:
```
class Profit:
    def __init__(self):
        self.profit = 0

    def getProfit(self):
        self.profit = float(input("Enter profit amount: "))
        return self.profit

class Loss:
    def __init__(self):
        self.loss = 0

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

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

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

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


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

```
Enter profit amount: 10000
Enter loss amount: 2343
Profit: $10000.0
Loss: $2343.0
Balance: $7657.0
```

15) WAP to demonstrate all types of inheritance.

In [21]:
```
class Parent:
    def __init__(self):
        self.parent_attribute = "This is from parent"

    def parent_method(self):
        print("This is parent method")

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.child_attribute = "This is from child"

    def child_method(self):
        print("This is child method")

class Father:
    def father_method(self):
        print("This is father method")

class Mother:
    def mother_method(self):
        print("This is mother method")

class Child2(Father, Mother):
    def child_method(self):
        print("This is child2 method")

class Grandparent:
    def grandparent_method(self):
        print("This is grandparent method")

class Parent2(Grandparent):
    def parent_method(self):
        print("This is parent2 method")

class Child3(Parent2):
    def child_method(self):
        print("This is child3 method")

class Parent3:
    def parent_method(self):
        print("This is parent3 method")

class ChildA(Parent3):
    def child_a_method(self):
        print("This is childA method")

class ChildB(Parent3):
    def child_b_method(self):
        print("This is childB method")


class Base:
    def base_method(self):
        print("This is base method")

class Derived1(Base):
```

```
    def derived1_method(self):
        print("This is derived1 method")

class Derived2(Base):
    def derived2_method(self):
        print("This is derived2 method")

class DerivedOfDerived(Derived1, Derived2):
    def derived_of_derived_method(self):
        print("This is derived of derived method")


print("\nSingle Inheritance:")
child = Child()
child.parent_method()
child.child_method()

print("\nMultiple Inheritance:")
child2 = Child2()
child2.father_method()
child2.mother_method()
child2.child_method()

print("\nMultilevel Inheritance:")
child3 = Child3()
child3.grandparent_method()
child3.parent_method()
child3.child_method()

print("\nHierarchical Inheritance:")
childA = ChildA()
childB = ChildB()
childA.parent_method()
childA.child_a_method()
childB.parent_method()
childB.child_b_method()

print("\nHybrid Inheritance:")
derived_of_derived = DerivedOfDerived()
derived_of_derived.base_method()
derived_of_derived.derived1_method()
derived_of_derived.derived2_method()
derived_of_derived.derived_of_derived_method()
```

```
Single Inheritance:
This is parent method
This is child method

Multiple Inheritance:
This is father method
This is mother method
This is child2 method

Multilevel Inheritance:
This is grandparent method
This is parent2 method
This is child3 method

Hierarchical Inheritance:
This is parent3 method
This is childA method
This is parent3 method
This is childB method

Hybrid Inheritance:
This is base method
This is derived1 method
This is derived2 method
This is derived of derived method
```

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 [23]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

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

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

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

# Create an employee
employee = Employee("Mihir Kacha", 21, 5000000)
employee.display_info()
```

```
Name: Mihir Kacha, Age: 21
Salary: $5000000
```

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 [27]:
```
class Shape:
    def draw(self):
        raise NotImplementedError("Subclass must implement abstract method")

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()

#run this for erorr
# s1= Shape()
# s1.draw()
```

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

In [ ]:
```

```