Python Programming - 2301CS404

Lab - 1

Dhol Namra

Enroll:23010101407

25-11-2024

01) WAP to print “Hello World”

In [12]:
```
print('Hello World')
```

```
Hello World
```

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

In [3]:
```
#without input
a,b=4,6
print(a+b)

#with input
a = int(input('Enter first number'))
b = int(input('Enter second number'))
print(a+b)
```

```
10
Enter first number5
Enter second number5
10
```

03) WAP to check the type of the variable.

In [11]:
```
print(type('abc'))
print(type(3))
print(type(3.745))
```

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

04) WAP to calculate simple interest.

In [18]:
```
p = 500
r = 0.04
t = 2

ans = (p*r*t)/100
print(ans)
```

```
0.4
```

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

In [17]:
```
r = float(input('Enter Radius Of Circle'))
print(3.14*r*r)
print(2*3.14*r)
```

```
Enter Radius Of Circle1
3.14
6.28
```

06) WAP to calculate area of a triangle.

In [16]:
```
b = float(input('Enter value of base'))
h = float(input('Enter value of height'))
print(0.5*b*h)
```

```
Enter value of base3
Enter value of height2
3.0
```

07) WAP to compute quotient and remainder.

In [1]:
```
a = int(input('Enter the number'))
print(a/10)
print(a%10)
```

```
1
0.1
```

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

In [14]:
```
fah=0;
cel = int(input("Enter celcius:"))
fah = (9/5)*cel +32;
print("Fahrenheit is:",fah)

fah = int(input("Enter Fahrenheit:"))
```

```
cel = (fah-32)*(5/9)
print("Celcius is:",cel)
```

```
Fahrenheit is: -0.3999999999999986
Celcius is: -1.1111111111111112
```

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

In [ ]:
```
import math


def calculate_distance(x1, y1, x2, y2):
    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
    return distance


try:
    print("Enter the coordinates of the first point:")
    x1 = float(input("x1: "))
    y1 = float(input("y1: "))

    print("Enter the coordinates of the second point:")
    x2 = float(input("x2: "))
    y2 = float(input("y2: "))

    distance = calculate_distance(x1, y1, x2, y2)
    print(f"The distance between ({x1}, {y1}) and ({x2}, {y2}) is: {distance:.2f
except ValueError:
    print("Please enter valid numeric inputs!")
```

10) WAP to print sum of n natural numbers.

In [2]:
```
num = int(input("Enter Range:"))
total=0;
for i in range(1,num+1):
    total=total+i
print(total)
```

```
15
```

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

In [3]:
```
num = int(input("Enter Range:"))
total=0;
for i in range(1,num+1):
    total=total+(i*i)
print(total)
```

```
14
```

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

In [19]:
```
firstname = input('Enter First name')
lastname = input('Enter Last name')
print(firstname+lastname)
```

```
Enter First namerutvik
Enter Last namebhagiya
rutvikbhagiya
```

13) WAP to swap two numbers.

In [15]:
```
a = input('Enter First Number')
b = input('Enter second Number')
print('before swapped',a,b)
temp = a
a = b
b = temp
print('After swapped',a,b)
```

```
Enter First Number5
Enter second Number2
before swapped 5 2
After swapped 2 5
```

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

In [3]:
```
kilometer = float(input("Enter Kilometer:"))
meter = kilometer * 1000
feet = kilometer * 3280.84
inch = kilometer * 39370.1
centimeter = kilometer * 100000

print("Meter: ",meter)
print("Feet: ",feet)
print("Inch: ",inch)
print("Centimeter: ",centimeter)
```

```
Meter:  10000.0
Feet:  32808.4
Inch:  393701.0
Centimeter:  1000000.0
```

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

In [6]:
```
day = input("Enter Day:")
month = input("Enter Month:")
year = input("Enter Year:")

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

```
10 - 10 - 10
```

Python Programming - 2301CS404

Lab - 2

Dhol Namra

23010101407

2-12-2024

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

In [2]:
```
num = int(input("Enter number: "))
if(num>0):
    print("Number is Positive")
else:
    print("Number is Negative")
```

```
Enter number: -6
Number is Negative
```

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

In [8]:
```
num = int(input("Enter number :"))
if(num%2==0):
    print("Number is Even")
else:
    print("Number is Odd")
```

```
Enter number :6
Number is Even
```

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

In [10]:
```
n1 = int(input("Enter num 1:"))
n2 = int(input("Enter num 2:"))
ans = (print("Number 1 is largest")) if (n1>n2) else (print("Number 2 is Largest
```

```
Enter num 1:10
Enter num 2:2
Number 1 is largest
```

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

In [12]:
```
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))
num3 = int(input("Enter num3: "))
if(num1>num2):
    if(num1>num3):
        print("Number ",num1," is largest")
    else:
        print("Number ",num3," is largest")
else:
    if(num2>num3):
        print("Number ",num2," is largest")
    else:
        print("Number ",num3," is largest")
```

```
Enter num1: 5
Enter num2: 6
Enter num3: 7
Number  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 [17]:
```
year = int(input("Enter year:"))
if((year%400==0) or (year%100!=0) and (year%4==0)):
    print(year,"is Leap year")
else:
    print(year,"is Not leap year")
```

```
Enter year:1900
1900 is Not leap year
```

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

In [19]:
```
num = int(input("Enter day num:"))
if(num==1):
    print("Sunday")
elif(num==2):
    print("Monday")
elif(num==3):
    print("Teuseday")
elif(num==4):
    print("Wednesday")
```

```
elif(num==5):
    print("Thursday")
elif(num==6):
    print("Friday")
elif(num==7):
    print("Saturday")
```

```
Enter day num:5
Thursday
```

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

In [20]:
```
num = int(input("Enter num for condition"))
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))
match(num):
    case 1:
        print(num1+num2)
    case 2:
        print(num1-num2)
    case 3:
        print(num1*num2)
    case 4:
        print(num1/num2)
```

```
Enter num for condition1
Enter num1: 2
Enter num2: 3
5
```

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 [21]:
```
mark = int(input("Enter marks"))
if(mark<35):
    print("Fail")
elif(mark>35 and mark<45):
    print("Pass")
elif(mark>45 and mark<60):
    print("Second")
elif(mark>60 and mark<70):
    print("First")
else:
    print("Distinction")
```

```
Enter marks50
Second
```

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 [3]:
```
a = float(input("Enter the length of the first side: "))
b = float(input("Enter the length of the second side: "))
c = float(input("Enter the length of the third side: "))

if a + b > c and a + c > b and b + c > a:
    if a == b == c:
        print("The triangle is equilateral.")
    elif a == b or b == c or a == c:
        print("The triangle is isosceles.")
    elif round(a**2, 5) == round(b**2 + c**2, 5) or round(b**2, 5) == round(a**2
        print("The triangle is right-angled.")
    else:
        print("The triangle is scalene.")
else:
    print("The given sides do not form a valid triangle.")
```

```
The given sides do not form a valid triangle.
```

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

In [29]:
```
num1 = int(input("Enter number1 :"))
num2 = int(input("Enter number2 :"))
num3 = int(input("Enter number3 :"))
if((num1>num2 and num1<num3) or (num1<num2 and num1>num3)):
    print(num1,"is second largest")
elif((num2>num1 and num2<num3) or (num2<num1 and num2>num3)):
    print(num2,"is second largest")
elif((num3>num1 and num3<num2) or (num3<num1 and num3>num2)):
    print(num3,"is second largest")
else:
     print("Not Find")
```

```
Enter number1 :3
Enter number2 :2
Enter number3 :1
2 is second 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 [6]:
```
units = float(input("Enter the number of units consumed: "))
bill = 0

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("Total electricity bill: Rs. ",bill)
```

```
Total electricity bill: Rs.  760.64
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 3

Dhol Namra

23010101407

9-12-2024

01) WAP to print 1 to 10.

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

```
0
1
2
3
4
5
6
7
8
9
10
```

02) WAP to print 1 to n.

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

```
1
2
3
4
5
```

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

In [6]:
```
num = int(input("Enter range Number:"))
for i in range(1,num+1,2):
    print(i)
```

```
1
3
5
```

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

In [7]:
```
num1 = int(input("Enter first Number:"))
num2 = int(input("Enter last Number:"))
for i in range(num1,num2+1):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
2
4
8
10
14
```

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

In [9]:
```
temp=0;
num = int(input("Enter range:"))
for i in range(1,num+1):
    temp = temp + i
print(temp)
```

```
21
```

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

In [16]:
```
temp = 0;
num = int(input("Enter range:"))
for i in range(1,num+1,1):
    temp = temp+(i*i)
print(temp)
```

```
14
```

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

In [13]:
```
temp1 = 0;
temp2 = 0;
```

```
ans = 0;
num = int(input("Enter range:"))
for i in range(1,num+1,2):
    temp1 = temp1+i
for i in range(2,num+1,2):
    temp2 = temp2+i
ans = temp1-temp2
print(ans)
```

```
-3
```

08) WAP to print multiplication table of given number.

In [18]:
```
num = int(input("Enter Number:"))
for i in range(1,11):
    print(num," * ",i," = ",(num*i))
```

```
6  *  1  =  6
6  *  2  =  12
6  *  3  =  18
6  *  4  =  24
6  *  5  =  30
6  *  6  =  36
6  *  7  =  42
6  *  8  =  48
6  *  9  =  54
6  *  10  =  60
```

09) WAP to find factorial of the given number.

In [21]:
```
temp = 1;
n = int(input("Enter number:"))
while(n>0):
    temp = temp*n;
    n=n-1;
print(temp)
```

```
120
```

10) WAP to find factors of the given number.

In [23]:
```
num = int(input("Enter Number:"))
for i in range(1,num+1):
    if(num%i==0):
        print(i)
```

```
1
2
3
4
6
12
```

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

In [40]:
```
num = int(input("Enter Number:"))
for i in range(2,((num+1)//2)+1):
    if(num%i==0):
```

```
        print("Number is Not Prime")
        break;
else:
    print("Number is Prime")
```

```
Number is Not Prime
```

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

In [2]:
```
num = int(input("Enter Number:"))
total=0;
rem=0;
while(num>0):
    rem = num%10;
    total = total + rem;
    num = num//10;
print("Sum is:",total)
```

```
Sum is: 6
```

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

In [10]:
```
ans = 0;
rem = 0;
num = int(input("Enter Number:"))
temp = num;
while num>0:
    rem = num%10;
    ans = rem + (ans*10);
    num = num//10;
if ans==temp:
    print("Number ",temp," is Palindrome");
else:
    print("Number ",temp," is Not Palindrome");
```

```
Number  1010  is Not Palindrome
```

14) WAP to print GCD of given two numbers.

In [18]:
```
num1 = int(input("Enter First Number:"))
num2 = int(input("Enter Second Number:"))
for i in range(2,num1+1 or num2+1):
    if(num1%i==0 and num2%i==0):
        print("Gcd is: ",i)
```

```
Gcd is:  3
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 4

Dhol Namra

23010101407

16-12-2024

String

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

In [3]:
```
str = input("Enter String");
str2 = str[::-1]
if str2==str:
    print("String is Palindrome");
else:
    print("String is Not Palindrome");
```

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

In [2]:
```
str = input("Enter String:");
newstr = str.split()[::-1];
arr = []
for i in newstr:
    arr.append(i)
print(" ".join(arr));
```

```
asdugfsadiufgsjad
```

03) WAP to remove ith character from given string.

In [3]:
```
str = input("Enter String");
i = int(input("Enter index"));
result = str[:i] + str[i+1:]
print(result)
```

```
sd
```

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

In [4]:
```
str = input("Enter String");
count=0;
for i in str:
    if i==" ":
        break
    count+=1
print(count)
```

```
5
```

05) WAP to print even length word in string.

In [5]:
```
str = input("Enter String")
for word in str.split():
    if len(word) % 2 == 0:
        print(word)
```

```
sdfh
difh
```

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

In [23]:
```
str = input("Enter String:")
count = 0;
for i in str:
    if(i in "a" or i in "e" or i in "i" or i in "o" or i in "u"):
        count+=1;
print(count)
```

```
10
```

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

In [6]:
```
str = input("Enter a string: ")
result = []
for word in str.split():
    nword = word[0].upper() + word[1:-1] + word[-1].upper() if len(word) > 1 els
    result.append(nword)

outstr = " ".join(result)
print("Output string is:", outstr)
```

```
Output string is: DlsdiG SihgsI SiprG
```

08) WAP to convert given array to string.

In [11]:
```
num = int(input("Enter Array Size"))
arr=[];
for i in range(num):
    arr.append(input("Enter num"))
result="".join(arr)
print(result)
```

```
123
```

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

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

In [13]:
```
password = input("Enter Password:")
co_pass = input("Enter Confirm Password:")

if (password.lower() == co_pass.lower()):
    if(password == co_pass):
        print("Password Matches")
    else:
        print("Password Matches but case not mathces")
else:
    print("Wrong Password")
```

```
Password Matches but case not mathces
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [23]:
```
cnum = input("Enter Card Number:")
cardnum = "**** **** **** "+cnum[-4:]
print(cardnum)
```

```
**** **** **** 2131
```

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

s1 = decimal and s2 = medical are Anagram

In [1]:
```
s1 = input("Enter First String:")
s2 = input("Enter Second String:")

if (sorted(s1) == sorted(s2)):
    print("Both string are Anagram")
else:
    print("Both string are Not Anagram")
```

```
Both string are Anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [29]:
```
string = input("Enter String: ")
upper=""
lower=""
for i in string:
    if (i.isupper()):
        upper += i
    else:
        lower += i
print(lower+upper)
```

```
ksjdjdsfHJFHJFH
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 5

Dhol Namra

23010101407

23-12-2024

List

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

In [2]:
```
li = [1,2,3,4,5]
sum = 0;
for i in li:
    sum+=i
print(sum)
```

```
15
```

02) WAP to find largest element in a List.

In [14]:
```
li=[]
size = int(input("Enter List size:"))
for i in range(size):
    num = int(input("Enter Element"))
    li.append(num)
print(li)
max=0
for i in li:
    if(max<i):
```

```
        max = i
print(max)
```

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

03) WAP to find the length of a List.

In [16]:
```
li=[]
size = int(input("Enter List size:"))
count=0
for i in range(size):
    num = int(input("Enter Element"))
    li.append(num)
    count+=1
print("length: "count)
```

```
5
```

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

In [21]:
```
li=[1,2,3,4,5]
print(li)
temp=0
temp = li[0]
li[0] = li[-1]
li[-1] = temp
print(li)
```

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

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

In [32]:
```
li = [1,2,3,4,5,6]
l1=[]
for i in range(len(li)//2,len(li)):
    l1.append(li[i])
for i in range(len(li)//2):
    l1.append(li[i])
print(l1)
```

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

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

In [35]:
```
li = [1,2,3,4,5,6,7,8,9,10]
pos1 = int(input("Enter position1: "))
pos2 = int(input("Enter position2: "))
print(li)
temp=0;
temp = li[pos1-1]
li[pos1-1] = li[pos2-1]
li[pos2-1] = temp
print(li)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 8, 3, 4, 5, 6, 7, 2, 9, 10]
```

07) WAP to reverse the list entered by user.

In [40]:
```
li=[]
size = int(input("Enter List size:"))
count=0
for i in range(size):
    num = int(input("Enter Element"))
    li.append(num)
print(li)
li.reverse()
li
```

```
[1, 2, 3]
```

Out[40]: [3, 2, 1]

08) WAP to print even numbers in a list.

In [50]:
```
li = [1,2,3,4,5,6,7,8,9,10]
for i in li:
    if i%2==0:
        print(i)
```

```
2
4
6
8
10
```

09) WAP to count unique items in a list.

In [49]:
```
li=[1,1,2,2,3,3,4,4,5,5]
l1=[]
for i in li:
    if i not in l1:
        count+=1
        l1.append(i)
print(count)
```

```
5
```

10) WAP to copy a list.

In [52]:
```
l1 = [1,2,3,4,5]
l2 = l1.copy()
print(l2)
```

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

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

In [54]:
```
li=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
r1 = int(input("Enter Range start :"))
r2 = int(input("Enter Range start :"))
```

```
for i in range(r1,r2):
    if li[i]%2!=0:
        print(li[i])
```

```
5
7
```

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

In [55]:
```
li=[1,1,2,2,3,3,4,4,5,5]
num = int(input("Enter Num:"))
li.count(num)
```

Out[55]: 2

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

In [59]:
```
li=[12,42,1,78,23,8,23,567,74,9,23,65]
li.sort()
li[-2]
```

Out[59]: 78

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

In [9]:
```
li = [1,1,2,2,3,1,5,2,3,4,2,1,5,7,8,1,2,1,3,4,4,5,5,6,6,7]
result = []
k = int(input("Enter Key: "))
for i in li:
    if (li.count(i)>k and i not in result):
        result.append(i);
print("Element which have greater feq than ",k," are:",result)
```

```
Element which have greater feq than  4  are: [1, 2]
```

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

In [11]:
```
#with List Comprehension
li1 = [i**2 for i in range(0,10)]
print(li1)
#without List Comprehension
li2 = []
for i in range(0,10):
    li2.append(i**2)
print(li2)
```

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

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

In [14]:
```
fruits = []
b_fruits=[]
```

```
f=""
num = int(input("Enter range:"))
for i in range(num):
    f = input("Enter fruit");
    fruits.append(f)
for i in fruits:
    if (i.lower().startswith('b')):
        b_fruits.append(i)
print(b_fruits)
```

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

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

In [20]:
```
li1 = [1,2,3,4,5,6,7,8,9]
li2 = [2,3,4,5,23,43,45,56,65,6]
li = []
for i in li1:
    for j in li2:
        if i==j and i not in li:
            li.append(i)
print(li)
```

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

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 6

Dhol Namra

23010101407

30-12-2024

Tuple

01) WAP to find sum of tuple elements.

In [1]:
```
t1 = (1,2,3,4,5,6,7)
sum=0
for i in t1:
    sum+=i
print(sum)
```

```
28
```

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

In [21]:
```
t1 = (83,99,41,3,10,23,63,65)
k = int(input("Enter elements k value:"))
l1 = list(t1)
l1.sort()
t2 = tuple(l1)
print("Minimum elements:",t2[:k],"\nMaximun elements:",t2[-k:])
```

```
Minimum elements: (3, 10)
Maximun elements: (83, 99)
```

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

In [8]:
```
t1=[(2,3,4),(2,4,6),(3,6,9)]
k = int(input("Enter divisible num:"))
ans=[]
for tup in t1:
    count=0
    for i in tup:
        if(i%k!=0):
            count+=1
    if(count == 0):
        ans.append(tup)
print(ans)
```

```
[(2, 4, 6)]
```

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

In [32]:
```
li = [1,2,3,4,5,6,7,8,9]
t1 = [(i,i**3) for i in li]
print(t1)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216), (7, 343), (8, 512), (9, 729)]
```

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

In [10]:
```
t1 = [(-1, 2, 3), (4, 5, 6), (7, -8, 9), (10, 11, 12)]
pt = []
for tup in t1:
    count=0
    for i in tup:
        if(i<0):
            count+=1
    if(count==0):
        pt.append(tup)
print("Positive Elements List of Tuple:", pt)
```

```
Positive Elements Tuple: [(4, 5, 6), (10, 11, 12)]
```

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

In [34]:
```
l = [1, 2, 3]
t = (4, 5, 6)
l.append(t)
print("List after adding tuple:",l)

t2 = t + tuple(l)
print("Tuple after adding list:", t2)
```

```
List after adding tuple: [1, 2, 3, (4, 5, 6)]
Tuple after adding list: (4, 5, 6, 1, 2, 3, (4, 5, 6))
```

07) WAP to remove tuples of length K.

In [19]:
```
t1 = [(1, 2, 3), (4, 5), (6, 7, 8, 9), (10, 11)]
k = int(input("Length of tuple to remove: "))
li=[]
for i in t1:
    if len(i)!=k:
        li.append(i)
print("List after removing:", li)
```

```
List after removing: [(4, 5), (6, 7, 8, 9), (10, 11)]
```

08) WAP to remove duplicates from tuple.

In [24]:
```
t1 = (1,1,2,3,3,4,5,2,4,1,8,6,9,2,3,5,1,4)
li = []
for tup in t1:
    if tup not in li:
        li.append(tup)
t2 = tuple(li)
print(t2)
```

```
(1, 2, 3, 4, 5, 8, 6, 9)
```

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

In [28]:
```
t1 = (1,2,3,4,5,6)
li=[]
for i in range(0,len(t1)-1):
    li.append(t1[i]*t1[i+1])
t2 = tuple(li)
print(t2)
```

```
(2, 6, 12, 20, 30)
```

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

In [39]:
```
t1 = (1,6,3,9,2,5,0)
li = list(t1)
distinct = True
for i in range(0,len(t1)):
    for j in range(i+1,len(t1)):
        if(t1[i]==t1[j]):
            distinct = False
            break
if (distinct):
    print("Distinct")
else:
    print("Not Distinct")
```

```
Distinct
```

In [ ]:
```

```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

Dhol Namra

23010101407

06-01-1025

Set & Dictionary

01) WAP to iterate over a set.

In [10]:
```
size = int(input("Enter size of set"))
s1=set({})
for i in range(size):
    s1.add(input("Enter set Element"))
for i in s1:
    print(i)
```

```
2
3
1
```

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

In [19]:
```
size = int(input("Enter size of set"))
s1=set({})
for i in range(size):
    s1.add(input("Enter set Element"))
print(s1)
l1 = list(s1)
print(l1)
```

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

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

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

In [17]:
```
s1 = {22,34,12,98,34,88,67}
print("Maximum:",max(s1))
print("Minimum:",min(s1))
```

```
Maximum: 98
Minimum: 12
```

04) WAP to perform union of two sets.

In [20]:
```
s1={1,2,3,4,5}
s2={6,7,8,9,10}

ans = s1.union(s2)
print(ans)
```

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

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

In [24]:
```
l1=[1,2,3,4,5,6]
l2=[10,4,7,19,1,5]

s1 = set(l1)
s2 = set(l2)

ans = s1.intersection(s2)
print(list(ans))
```

```
[1, 4, 5]
```

06) WAP to remove duplicates from list.

In [28]:
```
l1 = [1,1,2,2,3,3,4,4,5,5,6,6,5]
s1 = set(l1)
print(set(l1))
```

```
{1, 2, 3, 4, 5, 6}
```

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

In [33]:
```
st = "Apple Is Red, Apple Is Red Fruit"
words = st.split()
unique_words = set(words)
print(unique_words)
```

```
{'Is', 'Red,', 'Apple', 'Fruit', 'Red'}
```

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

In [34]:
```
a={1,3,2,6,4,9,4}
b={5,2,8,5,1,7,2}
print(a-b)
```

```
{9, 3, 4, 6}
```

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

In [46]:
```
s1 = "apple"
s2 = "pale"
ans1=set({})
ans2=set({})
for i in s1:
    ans1.add(i)
for i in s2:
    ans2.add(i)
print(ans1,ans2)
if(ans1==ans2):
    print("Anagram")
else:
    print("Not Anagram")
```

```
{'p', 'l', 'a', 'e'} {'p', 'l', 'a', 'e'}
Anagram
```

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

In [38]:
```
l1 = [1,2,3,4,5,6]
l2 = [2,6,0,9,11,4]
l3 = [6,1,9,0,4,21,5]

s1 = set(l1)
s2 = set(l2)
s3 = set(l3)

ans1 = s1.intersection(s2)
ans2 = ans1.intersection(s3)
print(list(ans2))
```

```
[4, 6]
```

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

In [40]:
```
s1='Rutvik'
vowel = {'a','e','i','o','u','A','E','I','O','U'}
vcount = 0
for i in s1:
    if (i in vowel):
        vcount+=1
print(vcount)
```

```
2
```

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

In [49]:
```
st = '1010101011'
bi = {'0','1'}
isBinary = True
for i in st:
    if i not in bi:
        isBinary = False
if(isBinary):
    print("String is Binary")
else:
    print("String is Not Binary")
```

```
String is Binary
```

13) WAP to sort dictionary by key or value.

In [5]:
```
d1 = {"rutvik":21,"rishabh":73,"yash":2,"smit":54,"meet":92}
sorted_key = dict(sorted(d1.items())) #because it will gives output in list of t
sorted_value = dict(sorted(d1.items(),key=lambda x:x[1]))
print("Sorted by keys:",sorted_key)
print("Sorted by values:",sorted_value)
```

```
Sorted by keys: {'meet': 92, 'rishabh': 73, 'rutvik': 21, 'smit': 54, 'yash': 2}
Sorted by values: {'yash': 2, 'rutvik': 21, 'smit': 54, 'rishabh': 73, 'meet': 9
2}
```

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

In [7]:
```
d1 = {'a':1,'c':3,'f':6,'r':18,'j':10}
sum_values = sum(d1.values())
print(sum_values)
```

```
38
```

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 [9]:
```
d1 = {'k':12,'f':34,'s':56,'q':89,'r':87,'u':21,'v':9}
key = input("Enter your key:")
if key in d1:
    print("Key is ",key," and value is ",d1[key])
else:
    print("Key is not found")
```

```
Key is not found
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 8

Dhol Namra

Enroll:23010101407

11-01-2025

User Defined Function

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

In [1]:
```
def BMI (mass,h):
        return mass/(h**2)

BMI(50,5)
```

Out[1]: 2.0

02) Write a function that add first n numbers.

In [3]:
```
def sum(n):

    return (n*(n+1)/2)
sum(3)
```

Out[3]: 6.0

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

In [39]:
```
number = int(input("Enter a number to check: "))
def is_prime(number):
  if number >1:
    for i in range(2,number):
      if (number % i) == 0:
        return False
      else:
        return True
```

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

In [ ]:
```
number = int(input("Enter a number to check: "))
def is_prime(number):
  if number >1:
    for i in range(2,number):
      if (number % i) == 0:
        return False
      else:
        return True
```

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

In [7]:
```
number = input("Enter a number to check: ")
def is_palindrom(number):
    if (number == number[::-1]):
        print("The string is a palindrome.")
    else:
        print("The string is not a palindrome.")
is_palindrom(number)
```

```
The string is not a palindrome.
```

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

In [20]:
```
def sum_of_list(numbers):
    return sum(numbers)

numbers = [1, 2, 3, 4, 5]
print(f"The sum of the list is: {sum_of_list(numbers)}")
```

```
The sum of the list is: 15
```

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

In [26]:
```
def sum_of_first_elements(tuple_list):
    return sum(t[0] for t in tuple_list)
```

```
tuples = [(1, 2), (3, 4), (10, 6)]
print(f"The sum of the first elements is: {sum_of_first_elements(tuples)}")
```

```
The sum of the first elements is: 14
```

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

In [30]:
```
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

fibonacci(4)
```

Out[30]: 3

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 [34]:
```
def get_student_name(rollno, student_dict):
    return student_dict.get(rollno, "Roll number not found")

dict1 = {101: 'Ajay', 102: 'Rahul', 103: 'Jay', 104: 'Pooja'}
print(get_student_name(103, dict1))
```

```
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 [41]:
```
def sum_scores_ending_with_zero(scores):
    total = 0
    for score in scores:
        if score % 10 == 0:
            total += score
    return total
print(sum_scores_ending_with_zero([200, 456, 300, 100, 234, 678]))
```

```
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 [43]:
```
def invert_dictionary(d):
    """
    Invert keys and values in a dictionary.

    :param d: Dictionary to invert
    :return: Inverted dictionary
    """
    return {v: k for k, v in d.items()}
invert_dictionary({'a':10,'b':20})
```

Out[43]:
```
{10: 'a', 20: 'b'}
```

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 [ ]:
```
def is_pangram(s):

    return set("abcdefghijklmnopqrstuvwxyz").issubset(s.lower())
is_pangram('the quick brown fox jumps over the lazy dog')
```

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 [ ]:
```
s1  = AbcDEfgh
def
```

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

In [ ]:
```
x=2
y=5
smallest = lambda x, y: x if x < y else y

print(smallest(x,y))
```

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

In [ ]:
```
def filter_long_names(names):

    return list(filter(lambda name: len(name) > 7, names))
filter_long_names(['harry','sejal','marko jansen'])
```

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

In [1]:
```
students = ["alice", "bob", "charlie", "dave", "eve"]

capitalized_students = list(map(lambda name: name.capitalize(), students))

print(capitalized_students)
```

```
['Alice', 'Bob', 'Charlie', 'Dave', 'Eve']
```

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 [3]:
```
def pos_args(name, age):
    print(name, age)

pos_args("Alice", 25)


def kw_args(name, age):
    print(name, age)

kw_args(age=30, name="Bob")


def default_args(name, age=20):
    print(name, age)

default_args("Charlie")
default_args("David", 35)


def var_pos_args(*args):
    print(args)

var_pos_args(10, 20, 30)


def var_kw_args(**kwargs):
    print(kwargs)

var_kw_args(name="Eve", age=22)


def kw_only(*, name, age):
    print(name, age)

kw_only(name="Frank", age=28)
```

```
def pos_only(name, age, /):
    print(name, age)

pos_only("Grace", 24)
```

```
Alice 25
Bob 30
Charlie 20
David 35
(10, 20, 30)
{'name': 'Eve', 'age': 22}
Frank 28
Grace 24
```

In [ ]:

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 9

Dhol Namra

Enroll:23010101407

22-01-2025

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 [7]:
```
fp = open("abc.txt","r")
print(fp.read())
fp.close()
#Line by line
fp = open("abc.txt","r")
print(fp.readline())
fp.close()
# in the form of a list
fp = open("abc.txt","r")
print(fp.readlines())
fp.close()
```

```
asdfghjklqwertyuioj
sdfghjkl
asdfghjklqwertyuioj

['asdfghjklqwertyuioj\n', 'sdfghjkl']
```

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

In [1]:
```python
import os

filename = "new.txt"

if not os.path.exists(filename):
    with open(filename, "w") as file:
        print(f"File '{filename}' created successfully.")
else:
    print(f"File '{filename}' already exists.")
```

```
File 'new.txt' created successfully.
```

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

In [21]:
```python
fp = open("new.txt","r")
for i in range(5):
        print(fp.readline())
fp.close()
```

```
sdfghj

asdfghj

sdfgh

sdfgh

aewertet
```

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

In [50]:
```python
f = open('new.txt','r')
s = f.read().split()
# st = list(s.split(' '))
# longest = ''
# for i in st:
#     if(len(longest) < len(i) ):
#         longest = i

# print(longest)
# fp.close()
longest=""
# print(s)
for i in s:
    if(len(longest) < len(i)):
        longest = i
print(longest)
```

```
aewertet
```

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

In [56]:
```
fp = open("abc.txt","rt")
lines=fp.readlines()
num_lines = len(lines)
num_words = sum(len(line.split()) for line in lines)
num_chars = sum(len(line) for line in lines)
print(num_lines)
print(num_words)
print(num_chars)
fp.close()
```

```
6
7
50
```

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

In [54]:
```
fp = open("new.txt","rt")
lines=fp.read()

fp = open("abc.txt","wt")
fp.write(lines)

fp.close()
fp.close()
```

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

In [82]:
```
fp = open("abc.txt","rb")
size=len(fp.read())
print(size)
fp.close()
```

```
55
```

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

In [62]:
```
def count_word_in_file(words, word):
    count = words.count(word)
    return count
fp = open("abc.txt","rt")
data = fp.read()
word = "sdfgh"
words = data.split()
fp.close()

print(count_word_in_file(words, word))
```

```
2
```

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 [88]:
```
fp = open("score.txt", "w")
for i in range(1, 6):
    scores = input(f"Enter the score for subject {i}: ")
    fp.write(scores)
fp.close()

fp = open("score.txt", "r")
fp.readlines()
scores = [int(score.strip()) for score in scores]
highest_score = max(scores)
print(highest_score)
fp.close()
```

```
5
```

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

(Note: each number should be in new line)

In [3]:
```
def is_prime(n):
    """Check if a number is prime."""
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def generate_primes(count):
    """Generate first 'count' prime numbers."""
    primes = []
    num = 2
    while len(primes) < count:
        if is_prime(num):
            primes.append(num)
        num += 1
    return primes


prime_numbers = generate_primes(100)

with open("primenumbers.txt", "w") as file:
    for prime in prime_numbers:
        file.write(str(prime) + "\n")

print("First 100 prime numbers have been written to 'primenumbers.txt'.")
```

```
First 100 prime numbers have been written to 'primenumbers.txt'.
```

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

In [7]:
```
def merge_files(new, file2, output_file):
    """Merge contents of file1 and file2 into output_file."""
    with open(output_file, "w") as outfile:
        for file in [new, file2]:
            with open(new, "r") as infile:
                outfile.write(infile.read() + "\n")

# File names
new = "new.txt"
file2 = "file2.txt"
output_file = "merged.txt"

# Merge the files
merge_files(new, file2, output_file)

print(f"Files '{new}' and '{file2}' have been merged into '{output_file}'.")
```

```
Files 'new.txt' and 'file2.txt' have been merged into 'merged.txt'.
```

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

In [11]:
```
def replace_word(new, output_file, word1, word2):
    """Replace all occurrences of word1 with word2 in input_file and save to out
    with open(input_file, "r") as infile:
        data = infile.read()

    updated_data = data.replace(word1, word2)

    with open(output_file, "w") as outfile:
        outfile.write(updated_data)

    print(f"Replaced '{word1}' with '{word2}' and saved to '{output_file}'.")


input_file = "new.txt"
output_file = "updated.txt"
word1 = "oldword"
word2 = "newword"

replace_word(input_file, output_file, word1, word2)
```

```
Replaced 'oldword' with 'newword' and saved to 'updated.txt'.
```

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

In [19]:
```
with open("new.txt", "w") as file:
    file.write("Hello, this is a test file.\nWelcome to file handling in Python.

with open("new.txt", "rb") as file:
    print(f"Initial Position: {file.tell()}")

    print("Reading 10 bytes:", file.read(10))
    print(f"Current Position: {file.tell()}")

    file.seek(0, 0)
```

```
    print(f"After seek(0, 0), Position: {file.tell()}")

    file.seek(5, 1)
    print(f"After seek(5, 1), Position: {file.tell()}")

    file.seek(-10, 2)
    print(f"After seek(-10, 2), Position: {file.tell()}")

    print("Remaining content:", file.read().decode("utf-8"))
```

```
Initial Position: 0
Reading 10 bytes: b'Hello, thi'
Current Position: 10
After seek(0, 0), Position: 0
After seek(5, 1), Position: 5
After seek(-10, 2), Position: 54
Remaining content: in Python.
```

In [ ]:

Python Programming - 2301CS404

Lab - 10

Dhol Namra

Enroll:23010101407

29-01-2025

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 [34]:
```
try:

    n1 = int(input("Enter the numerator: "))
    n2 = int(input("Enter the denominator: "))
    print(n1/n2)


    result_add = n1 + "5"
    print(f"Result of addition: {result_add}")

except ZeroDivisionError:
    print("ZeroDivisionError")
except ValueError:
```

```
        print("ValueError")
    except TypeError:
        print("TypeError")


    except (ZeroDivisionError, ValueError, TypeError) as e:
        print(f"An error occurred: {e}")
```

```
2.0
TypeError
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [51]:
```
try:
    list = [1,2,3,3,4,5,9]
    print(type(list))
    print(list[1])
    d1 = {1:"asdf",2:"asdfg"}
    print(d1[3])
# except IndexError:
#     print("IndexError")
# except KeyError:
#     print("KeyError")
except (IndexError, KeyError) as e:
        print(f"An error occurred: {e}")
```

```
<class 'list'>
2
An error occurred: 3
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [60]:
```
try:
    # fp = open("abc.txt","r")
    # print(fp.read())
    # fp.close()
    import asdfghjkl
except FileNotFoundError:
    print("fileNotFound")
except ModuleNotFoundError:
    print("module not found")
```

```
module not found
```

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

In [67]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
```

```
except Exception as err:
    print(err)
```

```
[Errno 2] No such file or directory: 'abc.txt'
```

05) WAP to demonstrate else and finally block.

In [87]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
except Exception as err:
    print(err)
else:
    print("else block excuted")
finally:
    print("finally block excuated")
```

```
else block excuted
finally block excuated
```

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 [107…
```
    grade_input = input("Enter a list of grades separated by commas: ")

    try:
        grade_strings = grade_input.split(",")

        grades = [int(grade.strip()) for grade in grade_strings]

        print("Grades entered:", grades)

    except ValueError:
        print("ValueError")
```

```
ValueError
```

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

In [99]:
```
def Divide(a,b):
    try:
        print(a/b)
    except Exception as error:
        print("Error accurd :",error)

Divide(5,0)
```

```
Error accurd : division 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 [142…
```
try:
    age = int(input("Enter a list of grades separated by commas: "))
    if(age > 18):
        print(age)
    else:
        raise ValueError
except ValueError:
            print("ValueError")
```

```
ValueError
```

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 [182…
```
try:
    s1 = input("enter a user name :;")
    if(len(s1) >= 5  and len(s1) <=15):
        print(s1)
    else:
        raise ValueError("abcd")
except ValueError as err:
    print(err)
```

```
abcd
```

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 [172…
```
import math

class NegativeNumberError(Exception):
    def __init__(self, message="Cannot calculate the square root of a negative n
        self.message = message
        super().__init__(self.message)

def calculate_square_root(number):
    if number < 0:
```

```
        raise NegativeNumberError()
    print(f"Square root: {math.sqrt(number)}")

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

```
Error: Cannot calculate the square root of a negative number
```

In [ ]:

Python Programming - 2301CS404

Lab - 10

Dhol Namra

Enroll:23010101407

29-01-2025

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 [34]:
```
try:

    n1 = int(input("Enter the numerator: "))
    n2 = int(input("Enter the denominator: "))
    print(n1/n2)


    result_add = n1 + "5"
    print(f"Result of addition: {result_add}")

except ZeroDivisionError:
    print("ZeroDivisionError")
except ValueError:
```

```
        print("ValueError")
    except TypeError:
        print("TypeError")


    except (ZeroDivisionError, ValueError, TypeError) as e:
        print(f"An error occurred: {e}")
```

```
2.0
TypeError
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [51]:
```
try:
    list = [1,2,3,3,4,5,9]
    print(type(list))
    print(list[1])
    d1 = {1:"asdf",2:"asdfg"}
    print(d1[3])
# except IndexError:
#     print("IndexError")
# except KeyError:
#     print("KeyError")
except (IndexError, KeyError) as e:
        print(f"An error occurred: {e}")
```

```
<class 'list'>
2
An error occurred: 3
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [60]:
```
try:
    # fp = open("abc.txt","r")
    # print(fp.read())
    # fp.close()
    import asdfghjkl
except FileNotFoundError:
    print("fileNotFound")
except ModuleNotFoundError:
    print("module not found")
```

```
module not found
```

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

In [67]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
```

```
except Exception as err:
    print(err)
```

```
[Errno 2] No such file or directory: 'abc.txt'
```

05) WAP to demonstrate else and finally block.

In [87]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
except Exception as err:
    print(err)
else:
    print("else block excuted")
finally:
    print("finally block excuated")
```

```
else block excuted
finally block excuated
```

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 [107…
```
    grade_input = input("Enter a list of grades separated by commas: ")

    try:
        grade_strings = grade_input.split(",")

        grades = [int(grade.strip()) for grade in grade_strings]

        print("Grades entered:", grades)

    except ValueError:
        print("ValueError")
```

```
ValueError
```

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

In [99]:
```
def Divide(a,b):
    try:
        print(a/b)
    except Exception as error:
        print("Error accurd :",error)

Divide(5,0)
```

```
Error accurd : division 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 [142…
```
try:
    age = int(input("Enter a list of grades separated by commas: "))
    if(age > 18):
        print(age)
    else:
        raise ValueError
except ValueError:
            print("ValueError")
```

```
ValueError
```

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 [182…
```
try:
    s1 = input("enter a user name :;")
    if(len(s1) >= 5  and len(s1) <=15):
        print(s1)
    else:
        raise ValueError("abcd")
except ValueError as err:
    print(err)
```

```
abcd
```

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 [172…
```
import math

class NegativeNumberError(Exception):
    def __init__(self, message="Cannot calculate the square root of a negative n
        self.message = message
        super().__init__(self.message)

def calculate_square_root(number):
    if number < 0:
```

```
        raise NegativeNumberError()
    print(f"Square root: {math.sqrt(number)}")

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

```
Error: Cannot calculate the square root of a negative number
```

In [ ]:

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 12

Dhol Namra

Enroll:23010101407

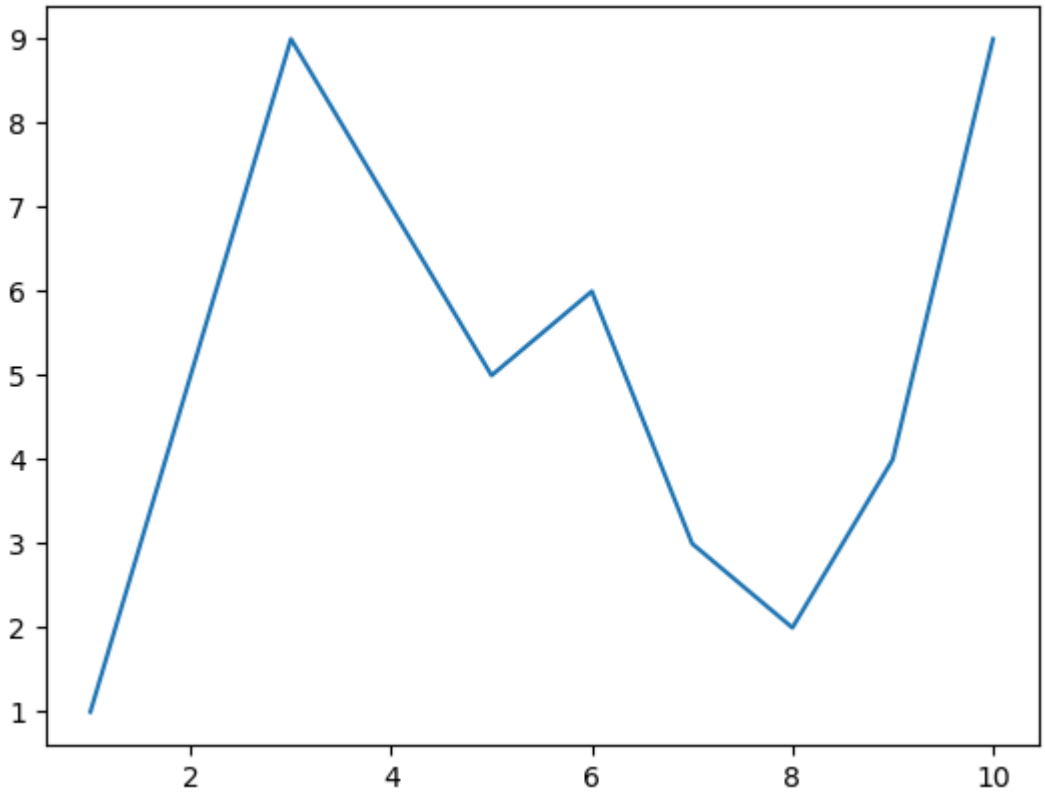
17-02-2025

In [14]:
```
#import matplotlib below
import matplotlib.pyplot as plt
```

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

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

In [6]:
```
plt.plot(x,y)
```

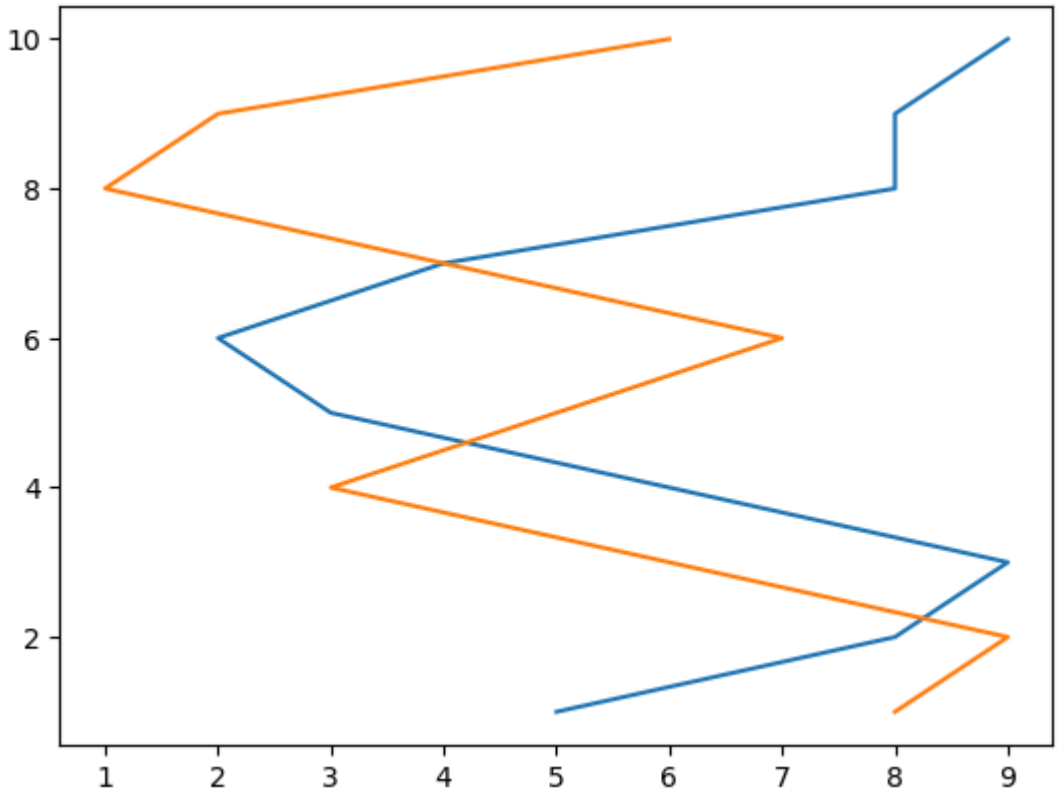
Out[6]:
```
[<matplotlib.lines.Line2D at 0x20bdc66d1f0>]
```



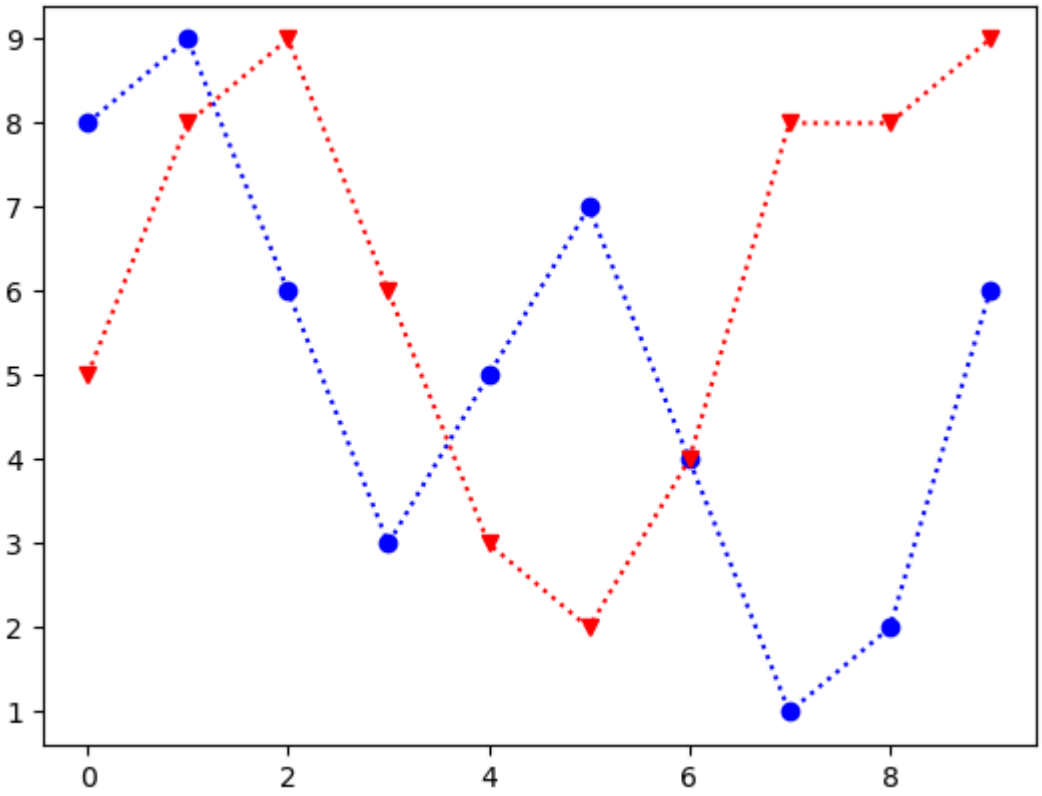
In [8]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

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

In [12]:
```
plt.plot(cxMarks,x)
plt.plot(cyMarks,x)
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x20bdc8db260>]
```

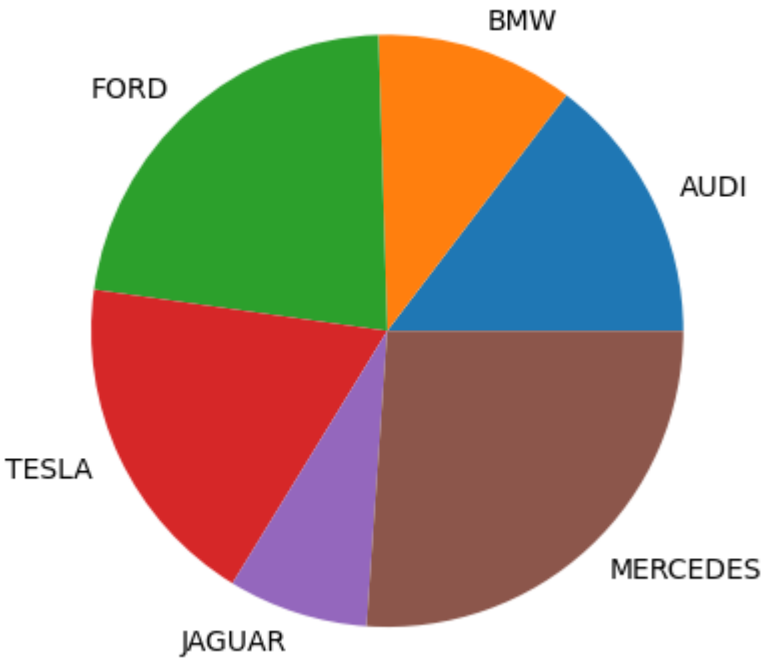


In [18]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

plt.plot(cxMarks,'o:b')
plt.plot(cyMarks,'v:r')

plt.show()
# write a code to generate below graph
```



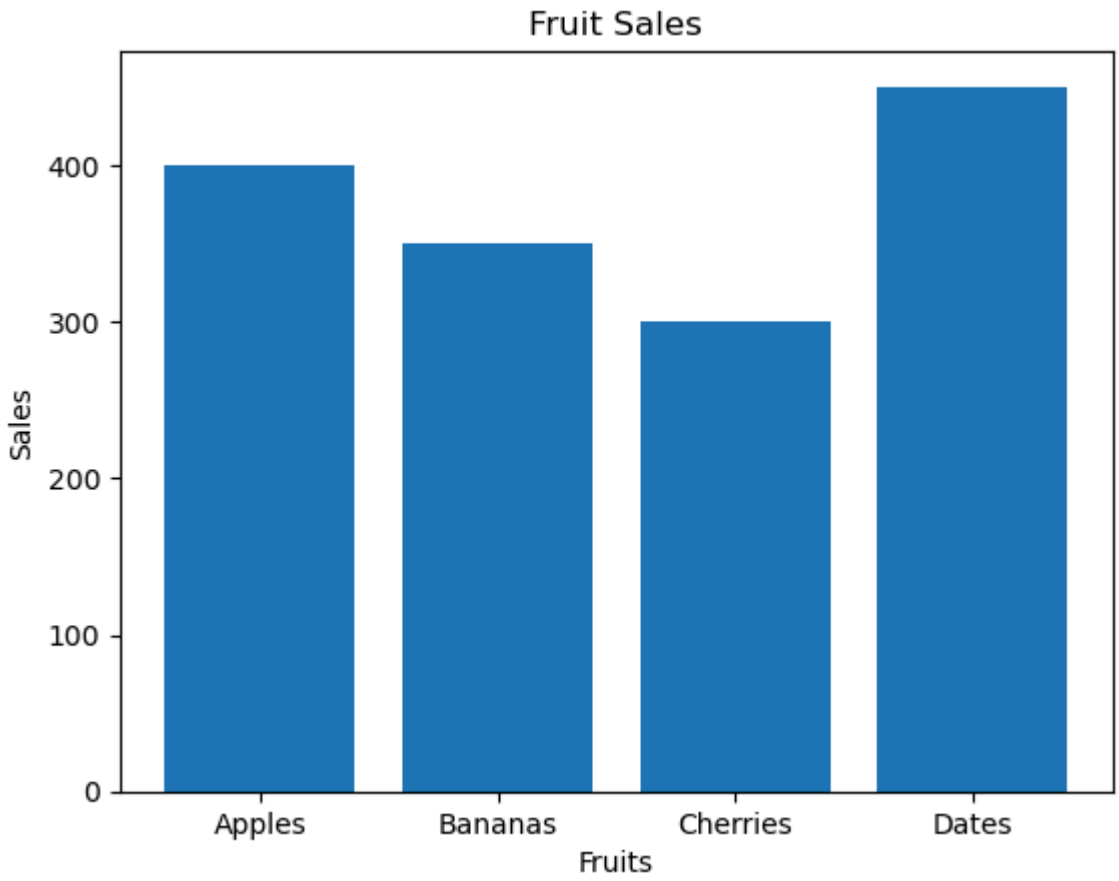
04) WAP to demonstrate the use of Pie chart.

In [25]:
```
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

plt.pie(data, labels=cars)


plt.show()
```



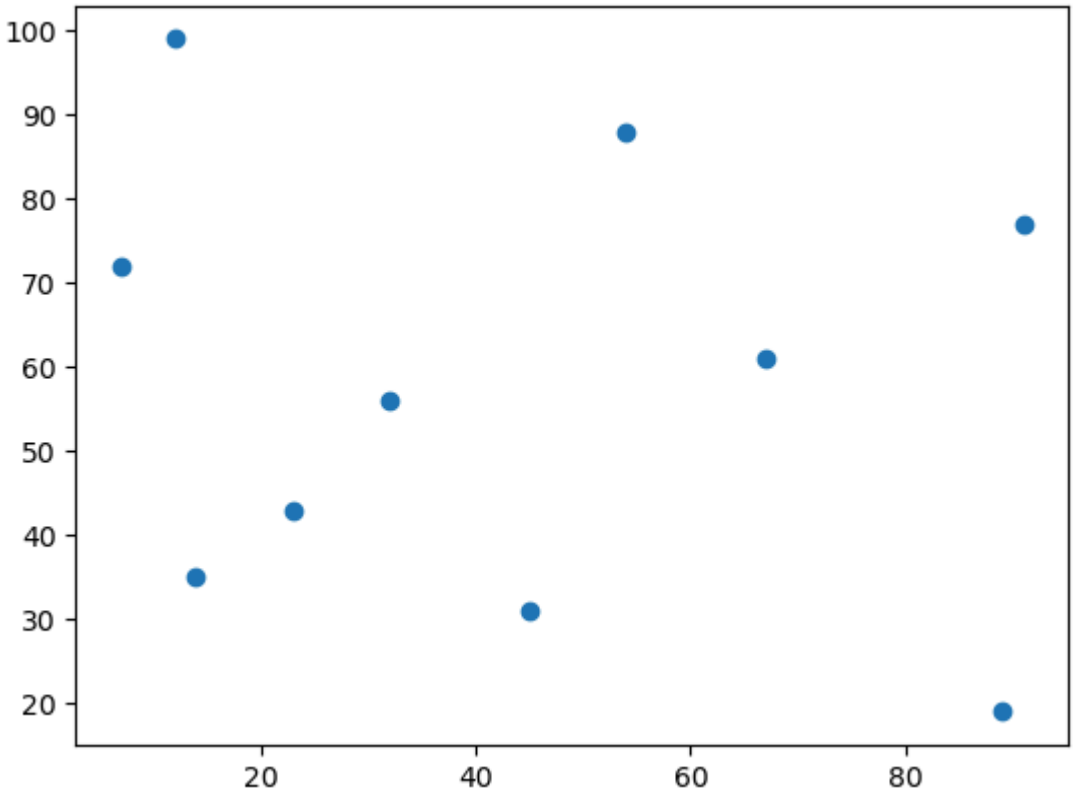
05) WAP to demonstrate the use of Bar chart.

In [27]:
```
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]

plt.bar(fruits, sales)
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```



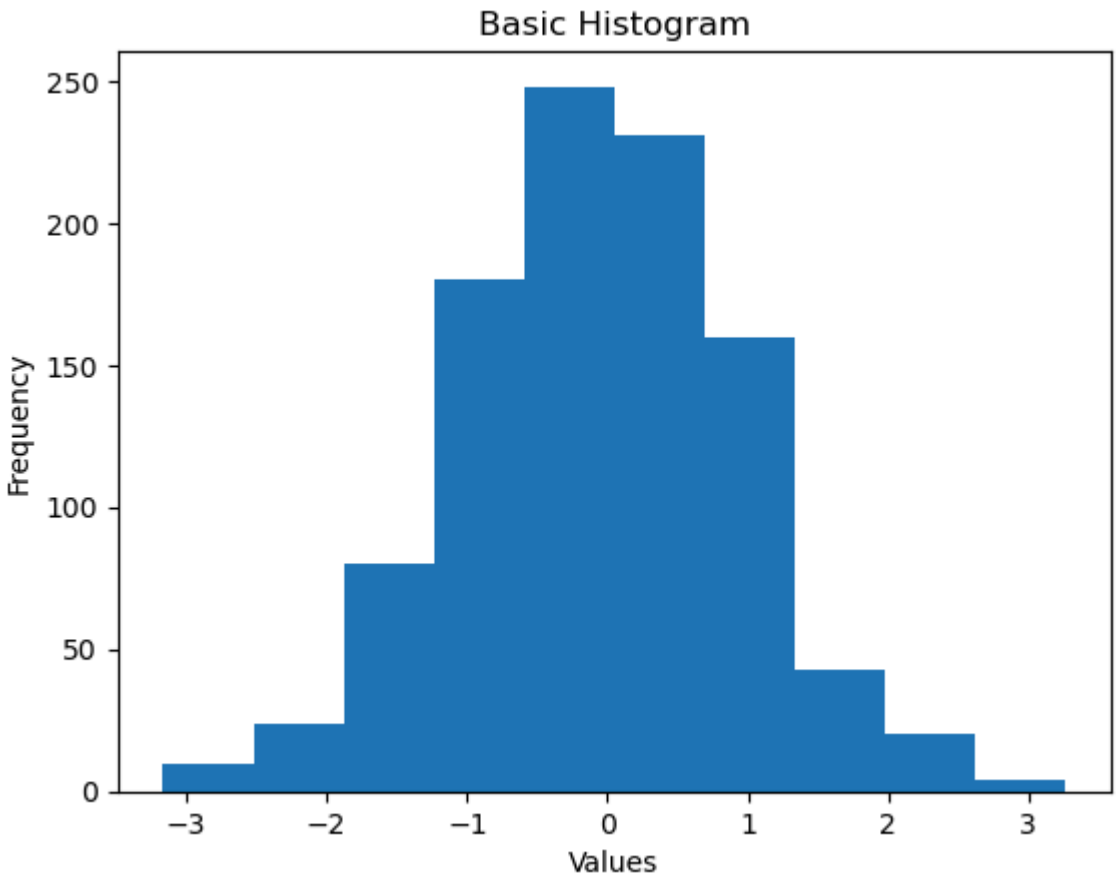
06) WAP to demonstrate the use of Scatter Plot.

In [31]:
```
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()
```



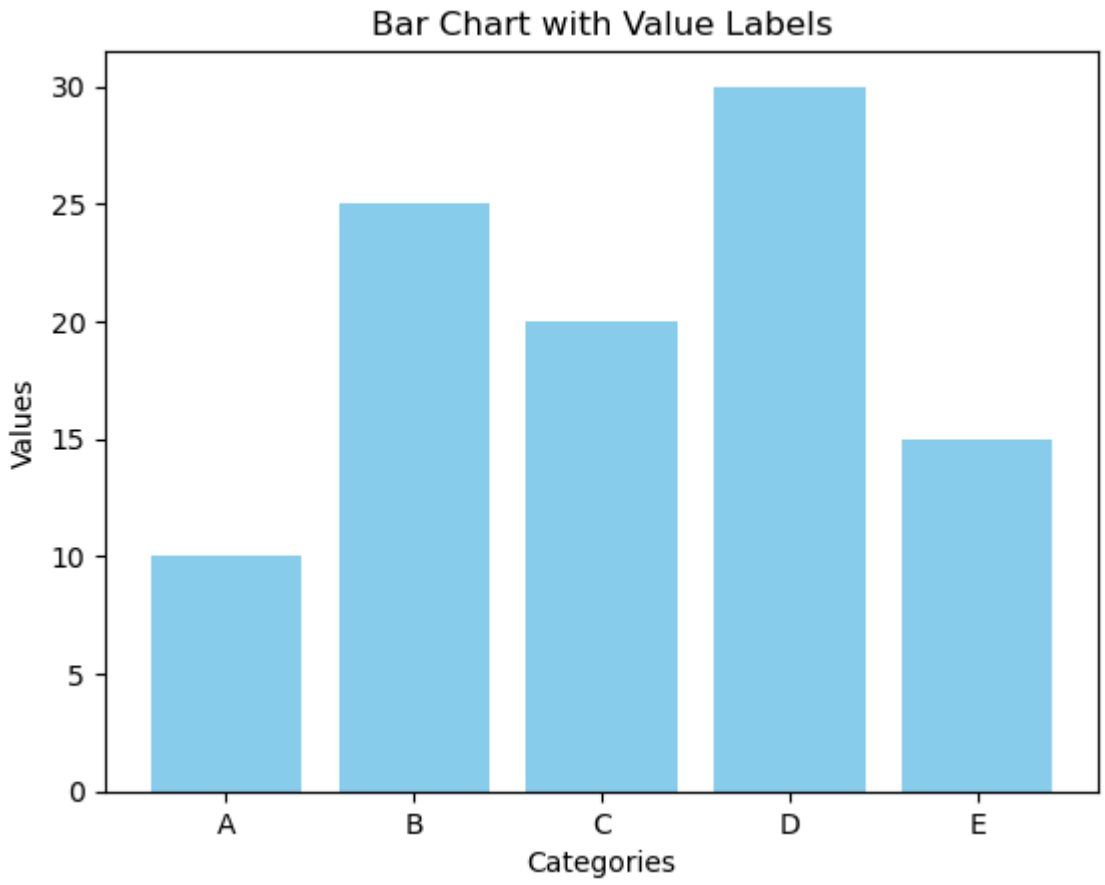
07) WAP to demonstrate the use of Histogram.

In [40]:
```
plt.hist(data)


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.

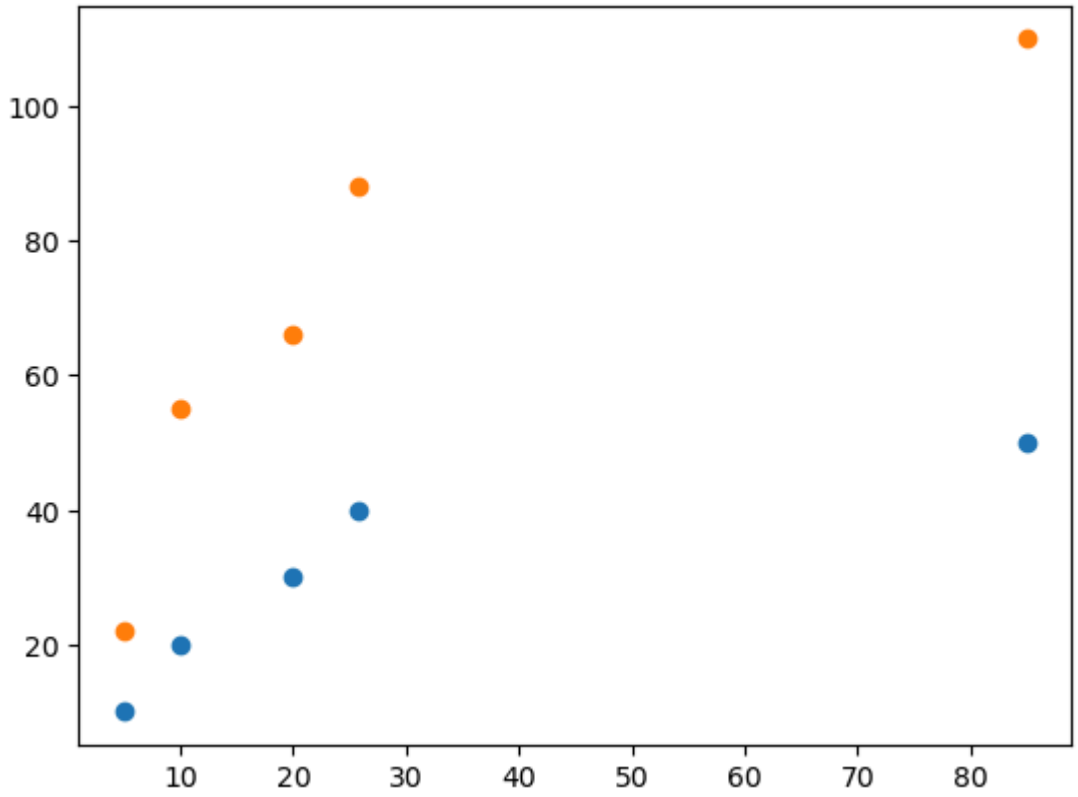
In [54]:
```
categories = ['A', 'B', 'C', 'D', 'E']
values = [10, 25, 20, 30, 15]


plt.bar(categories, values, color='skyblue')


plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Bar Chart with Value Labels")


plt.show()
```



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

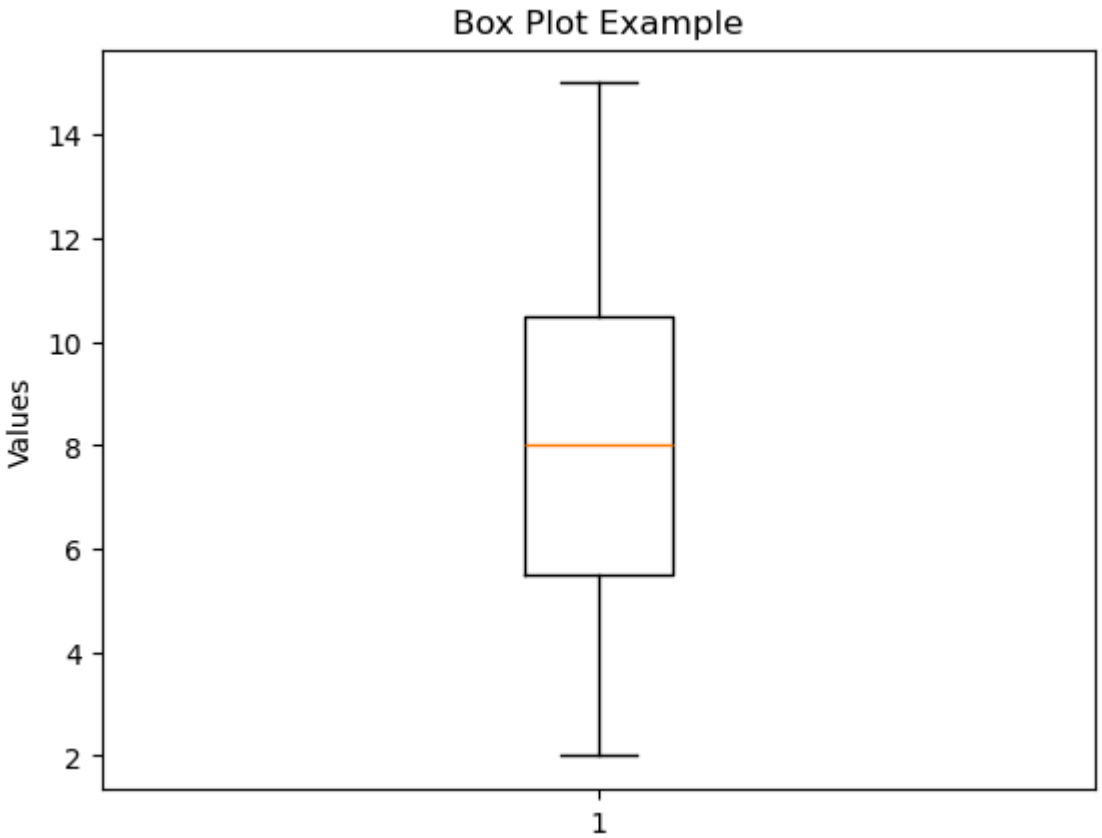
In [48]:
```
demo = [5,10,20,25.75,85]
x = [22,55,66,88,110]
y = [10,20,30,40,50]
plt.scatter(demo,y)
plt.scatter(demo,x)
```

Out[48]:
```
<matplotlib.collections.PathCollection at 0x14d14c994c0>
```



10) WAP to create a Box Plot.

In [46]:
```
data = [7, 8, 5, 6, 9, 15, 3, 4, 12, 10, 8, 6, 11, 13, 2]
plt.boxplot(data)
plt.title("Box Plot Example")
plt.ylabel("Values")
plt.show()
```



In [ ]:

Python Programming - 2301CS404

Lab - 13

Dhol Namra

Enroll:23010101407

03-03-2025

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 [2]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

stu = Students("Yash",19,"A++")

print("Name =",stu.name)
print("Age =",stu.age)
print("Grade =",stu.grade)
```

```
Name = Yash
Age = 19
Grade = 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 [6]:
```
class Bank_Account:
    def __init__(self, Account_No, User_Name, Email, Account_Type, Account_Balan
        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 {
            "Account_No" : self.Account_No,
            "User_Name" : self.User_Name,
            "Email" : self.Email,
            "Account_Type" : self.Account_Type,
            "Account_Balance" : self.Account_Balance
        }

    def DisplayAccountDetails(self):
        Account_Details = self.GetAccountDetails()
        for key, value in Account_Details.items():
            print(f"{key} : {value}")

Account = Bank_Account(23010101169, "Yash", "yash@gmail.com", "Saving",3005000)
Account.DisplayAccountDetails()
```

```
Account_No : 23010101169
User_Name : Yash
Email : yash@gmail.com
Account_Type : Saving
Account_Balance : 3005000
```

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

In [15]:
```
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

circle = circle(5)
print("Area :",circle.area())
print("Perimeter :",circle.perimeter())
```

```
Area : 78.53981633974483
Perimeter : 31.41592653589793
```

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

In [21]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_salary(self,new_salary):
        self.salary = new_salary

    def update_age(self, new_age):
        self.age = new_age

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

employee = Employee("Yash",19,250000)
employee.display_information()
print("----------------------")

employee.update_salary(150000)
employee.update_age(22)
employee.display_information()
```

```
Name = Yash
Age = 19
Salary = 250000
----------------------
Name = Yash
Age = 22
Salary = 150000
```

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

In [25]:
```
class BankAccount:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance is {self.balance}.")
        else:
            print("Enter Positive Deposit Amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance is {self.balance}.")
        else:
            print("Invalid withdrawal amount or insufficient balance.")
```

```
    def check_balance(self):
        print(f"Your current balance is {self.balance}.")
```

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 [26]:
```
class InventoryItem:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def add(self, amount):
        self.quantity += amount

    def remove(self, amount):
        if amount <= self.quantity:
            self.quantity -= amount
        else:
            print("Not enough items to remove.")

    def update_price(self, new_price):
        self.price = new_price

    def __str__(self):
        return f"Item: {self.name}, Price: {self.price}, Quantity: {self.quantit
```

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

In [27]:
```
class Dog:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age

my_dog = Dog("Lucky", "Golden Retriever", 3)

print(my_dog.name)
print(my_dog.breed)
print(my_dog.age)
```

```
Lucky
Golden Retriever
3
```

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 [2]:
```
class StudentKit:
    # Class attribute
    principal_name = "Mr ABC"

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

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

    def generate_certificate(self):
        certificate = f"""

        Certificate of Attendance

        This is to certify that {self.student_name} has attended {self.days_pres

        Principal: {StudentKit.principal_name}

        """
        print(certificate)

# Taking input for student name
student_name = input("Enter student name: ")
student = StudentKit(student_name)

# Taking input for attendance days
days_present = int(input("Enter number of days present: "))
student.attendance(days_present)

# Generating certificate
student.generate_certificate()
```

```
Attendance recorded: demo attended 50 days.


        Certificate of Attendance

        This is to certify that demo has attended 50 days of classes.

        Principal: Mr ABC
```

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

In [4]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add(self, other):
        total_minutes = self.minute + other.minute
```

```
        total_hours = self.hour + other.hour + (total_minutes // 60)
        total_minutes %= 60
        total_hours %= 24

        return Time(total_hours, total_minutes)

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

# Example usage
t1 = Time(10, 45)
t2 = Time(2, 30)

result = t1.add(t2)
print("Resultant Time:", end=" ")
result.display()
```

```
Resultant Time: 13:15
```

In [ ]:

Python Programming - 2301CS404

Lab - 13

Dhol Namra

23010101407

10-03-2025

Continued..

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

In [3]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

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

def display_area(rect):
    print(f"Area of the rectangle: {rect.calculate_area()}")

my_rect = Rectangle(10, 5)

display_area(my_rect)
```

```
Area of the rectangle: 50
```

11) Calculate the area of a square.

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

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

    def area(self):
        area = self.side * self.side
        self.output(area)

    def output(self, area):
        print(f"Area of the square: {area}")

square = Square(6)

square.area()
```

```
Area of the square: 36
```

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 [9]:
```
class Rectangle:
    def __init__(self, length, width):
        if self.is_square(length, width):
            print("THIS IS SQUARE")
            self.length = self.width = None
        else:
            self.length = length
            self.width = width

    def area(self):
        if self.length is not None and self.width is not None:
            area = self.length * self.width
            self.output(area)
            return area

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

    @staticmethod
    def is_square(length, width):
        return length == width


rect1 = Rectangle(5, 10)
```

```
if rect1.length is not None:
    rect1.area()

rect2 = Rectangle(8, 8)
```

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

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

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

In [11]:
```
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 positive.")

sq = Square(5)
print("Side:", sq.get_side())
sq.set_side(10)
print("Updated Side:", sq.get_side())

sq.set_side(-3)
```

```
Side: 5
Updated Side: 10
Side length must be positive.
```

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 [15]:
```
class Profit:
    def __init__(self):
        self.profit = 0


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

class Loss:
```

```
    def __init__(self):
        self.loss = 0


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


    def printBalance(self):
        print(f"The balance is: {self.balance}")


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

```
The balance is: 950000.0
```

15) WAP to demonstrate all types of inheritance.

In [13]:

```
class Fruit:
    def taste(self):
        print("Fruits have different tastes.")

class Mango(Fruit):
    def color(self):
        print("Mango is yellow.")


class Alphonso(Mango):
    def size(self):
        print("Alphonso mango is small.")


class Citrus:
    def vitamin_c(self):
        print("Citrus fruits are rich in Vitamin C.")

class Orange(Fruit, Citrus):
    def peel(self):
        print("Orange has a thick peel.")

class Apple(Fruit):
    def shape(self):
        print("Apple is round.")
```

```
class Banana(Fruit, Citrus):
    def energy(self):
        print("Banana gives instant energy.")


mango = Mango()
mango.taste()
mango.color()

alphonso = Alphonso()
alphonso.taste()
alphonso.color()
alphonso.size()

orange = Orange()
orange.taste()
orange.vitamin_c()
orange.peel()

apple = Apple()
apple.taste()
apple.shape()

banana = Banana()
banana.taste()
banana.vitamin_c()
banana.energy()
```

```
Fruits have different tastes.
Mango is yellow.
Fruits have different tastes.
Mango is yellow.
Alphonso mango is small.
Fruits have different tastes.
Citrus fruits are rich in Vitamin C.
Orange has a thick peel.
Fruits have different tastes.
Apple is round.
Fruits have different tastes.
Citrus fruits are rich in Vitamin C.
Banana gives instant energy.
```

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(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(self):
        super().display()
        print(f"Salary: {self.salary}")

emp = Employee("namra", 30, 50000)

emp.display()
```

```
Name: namra, 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 [19]:
```
from abc import ABC, abstractmethod

class Shape(ABC):
    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 [ ]: