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

Python Programming - 2301CS404

Lab - 1

223 | Vishal Baraiya | 23010101014

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 [3]: a = 23
        b = 67
        print(a,"+",b,"=",a+b)
        a = int(input("Enter the first Number:"))
        b = int(input("Enter the Second Number:"))
        print(a,"+",b,"=",a+b)
```

```
23 + 67 = 90

Enter the first Number: 89
Enter the Second Number: 09

89 + 9 = 98
```

03) WAP to check the type of the variable.

In [9]:
```
a = 89
b = 8.9
str = "tempString"
print("type of a",type(a))
print("type of b",type(b))
print("type of str",type(str))
```

```
type of a <class 'int'>
type of b <class 'float'>
type of str <class 'str'>
```

04) WAP to calculate simple interest.

In [13]:
```
p = int(input("Enter the total Amount:"))
n = int(input("Enter the time Period:"))
r = int(input("Enter the rate of Interest:"))
i = float((p*n*r)/100)
print("simple interest:",i)
```

```
Enter the total Amount: 1000
Enter the time Period: 10
Enter the rate of Interest: 10

simple interest: 1000.0
```

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

In [24]:
```
import math
r = int(input("Enter the radius of the Circle:"))
perimeter = 2*math.pi*r
area = math.pi*r*r
print("area =",perimeter)
print("area =",area)
```

```
Enter the radius of the Circle: 7

area = 43.982297150257104
area = 153.93804002589985
```

06) WAP to calculate area of a triangle.

In [26]:
```
b = int(input("Enter the base:"))
h = int(input("Enter the height:"))
area = b*h*0.5
print("area =",area)
```

```
Enter the base: 3
Enter the height: 6

area = 9.0
```

07) WAP to compute quotient and remainder.

In [31]:
```
a = int(input("Enter the number:"))
b = int(input("the number is divide by:"))
quotient = a/b
remainder = a%b
print("quotient =",quotient)
print("remainder =",remainder)
```

```
Enter the number: 10
the number is divide by: 8

quotient = 1.25
remainder = 2
```

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

In [36]:
```
f = float(input("Enter the temprature in Fahrenheit:"))
c = ((f-32)*5)/9
print("temprature in degree:",c)

c = float(input("Enter the temprature in degree:"))
f = ((c*9)/5)+32
print("temprature in Fahrenheit:",f)
```

```
Enter the temprature in Fahrenheit: 32

temprature in degree: 0.0

Enter the temprature in degree: 0

temprature in Fahrenheit: 32.0
```

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

In [42]:
```
import math
x1,y1 = 5,6
x2,y2 = 8,9
distance = math.sqrt(math.pow((x1-x2),2)+math.pow((y1-y2),2))
print("distance :",distance)
```

```
distance : 4.242640687119285
```

10) WAP to print sum of n natural numbers.

In [60]:
```
n = int(input("Enter the value of n :"))
sum = (n*(n+1))/2
print("sum :",sum)
```

```
Enter the value of n : 10

sum : 55.0
```

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

In [56]:
```
n = int(input("Enter the value of n :"))
sum = (n*(n+1)*(2*n+1))/6
print("sum :",sum)
```

```
Enter the value of n : 3

sum : 14.0
```

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

In [66]:
```
fn = "Vishal"
ln = "Baraiya"
res = fn+" "+ln
print(res)
```

```
Vishal Baraiya
```

13) WAP to swap two numbers.

In [68]:
```
a = 23
b = 87
temp = a
a = b
b = temp
print("a =",a)
print("b =",b)
```

```
a = 87
b = 23
```

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

In [70]:
```
d = int(input("Enter the distence in km :"))
print("in meters :",d*1000)
print("in feet :",d*3281)
print("in inches :",d*39370.1)
print("in cm",d*1000*100)
```

```
Enter the distence in km : 10

in meters : 10000
in feet : 32810
in inches : 393701.0
in cm 1000000
```

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

In [78]:
```
d = int(input("Enter the day :"))
m = int(input("Enter the month :"))
y = int(input("Enter the year :"))
print(d,"-",m,"-",y,sep="",end="\n")
```

```
Enter the day : 8
Enter the month : 9
Enter the year : 2024

8-9-2024
```

In [ ]:
```
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 2

223 | Vishal Baraiya | 23010101014

if..else..

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

In [2]:
```
n = int(input("Enter the number : "))
if n>0 :
    print("n is Positive")
elif n==0:
    print ("n is Zero")
else:
    print("n is negative")
```

```
n is negative
```

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

In [6]:
```
n = int(input("Enter the number :"))
if n%2==0 :
    print("N is even")
else :
    print("N is odd")
```

```
Enter the number :33
N is odd
```

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

In [1]:
```
n1 = int(input("Enter the first Number : "))
n2 = int(input("Enter the second Number : "))

#ernary operator
largest = n1 if n1 > n2 else n2
print("Largest is :",largest)

#if
if n1>n2 :
    largest = n1
else:
    largest = n2
print("Largest is :",largest)
```

```
Largest is : 45
Largest is : 45
```

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

In [14]:
```
n1 = int(input("Enter the first Number : "))
n2 = int(input("Enter the second Number : "))
n3 = int(input("Enter the third Number : "))

if n1>n2:
    if n1>n3:
        largest = n1
    else:
        largest = n3
else:
    if n2>n3:
        largest = n2
    else:
        largest = n3
print("Largest is :",largest)
```

```
Enter the first Number : 45
Enter the second Number : 32
Enter the third Number : 12
Largest is : 45
```

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 [20]:
```
year = int(input("Enter the year : "))

if (year%400 == 0) and (year%100 == 0):
    print(year,"is a Leap Year.")
elif (year%4 == 0) and (year%100 != 0):
    print(year,"is a Leap Year.")
else:
    print(year,"is not a Leap Year.")
```

```
Enter the year : 2024
2024 is a Leap Year.
```

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

In [14]:
```
n = int(input("Enter the 1 to 7 number :"))
n = n - 1
mylist = ["Sunday","Monday","Tuesday","Wendesday","Thurseday","Friday","Satureda
#match case

match n:
    case 0:
        print(mylist[n])
    case 1:
        print(mylist[n])
    case 2:
        print(mylist[n])
    case 3:
        print(mylist[n])
    case 4:
        print(mylist[n])
    case 5:
        print(mylist[n])
    case 6:
        print(mylist[n])
    case _:
        print("Wrong Input!")
```

```
Enter the 1 to 7 number :5
Thurseday
```

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

In [5]:
```
a = int(input("Enetr the first number : "))
b = int(input("Enter the second number : "))
op = input("Enter the Opration : ")

if op=='+':
    print("a+b = ",a+b)
elif op=='-':
    print("a-b = ",a-b)
elif op=='*':
    print("a*b = ",a*b)
elif op=='/':
    print("a/b = ",a/b)
else:
    print("Wrong Input!")
```

```
Enetr the first number : 90
Enter the second number : 89
Enter the Opration : -
a-b =  1
```

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 [10]:
```python
s1 = int(input("Enter the marks of subject-1 : "))
s2 = int(input("Enter the marks of subject-2 : "))
s3 = int(input("Enter the marks of subject-3 : "))
s4 = int(input("Enter the marks of subject-4 : "))
s5 = int(input("Enter the marks of subject-5 : "))

per = ((s1+s2+s3+s4+s5)/500)*100;

if per>=70:
    print("Distinction")
elif per>=60 and per<70:
    print("First class")
elif per>=45 and per<60:
    print("Second class")
elif per>=35 and per<45:
    print("Third class")
else:
    print("fail")
```

```
Enter the marks of subject-1 : 56
Enter the marks of subject-2 : 78
Enter the marks of subject-3 : 90
Enter the marks of subject-4 : 76
Enter the marks of subject-5 : 45
First class
```

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

In [2]:
```python
a = int(input("Enter the first side : "));
b = int(input("Enter the second side : "));
c = int(input("Enter the third side : "));

if (a==b==c):
    print("Equilateral Triangle.");
elif (a==b) or (a==c) or (b==c):
    print("Isosceles Triangle.")
else:
    print("Scalene Triangle.")

if ((a*a) == (b*b) + (c*c)) or ((b*b) == (a*a)+(c*c)) or ((c*c) == (a*a)+(b*b)):
    print("Right-Angled Triangle.")
```

```
Scalene Triangle.
Right-Angled Triangle.
```

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

In [12]:
```python
a = int(input("Enter the first Number : "))
b = int(input("Enter the second Number : "))
c = int(input("Enter the third Number : "))
```

```
if a>b and a>c:
    if b>c:
        print(b,"is second Largest.")
    else:
        print(c,"is second Largest.")

if b>a and b>c:
    if a>c:
        print(a,"is second Largest.")
    else:
        print(c,"is second Largest.")

if c>a and c>b:
    if a>b:
        print(a,"is second Largest.")
    else:
        print(b,"is second Largest.")
```

```
50 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 [15]:

```
n = int(input("Enter the Electricity Unit : "))
totalAmount = 0

if n>=1 and n<50:
    totalAmount = 2.6 * n
elif n>=50 and n<100:
    totalAmount = 3.25 * n
elif n>=100 and n<200:
    totalAmount = 5.26 * n
elif n>=200:
    totalAmount =  8.45 * n

print("Total Amount : ",totalAmount)
```

```
Total Amount :  920.5
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 3

223 | Vishal Baraiya | 23010101014

for and while loop

01) WAP to print 1 to 10.

In [4]:
```
# for loop
for i in range(1,11):
    print(i)

# while loop
# i=1
# while i<=10:
#     print(i)
#     i += 1
```

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

02) WAP to print 1 to n.

In [6]:
```
n = int(input("Enter the number : "))
for i in range(1,n+1):
    print(i)
```

```
1
2
3
4
5
```

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

In [8]:
```
n = int(input("Enter the number : "))
for i in range(1,n+1,2):
    print(i)
```

```
1
3
5
7
9
11
13
15
17
```

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

In [9]:
```
start = int(input("Enter the starting number : "))
end = int(input("Enter the ending number : "))
for i in range(start,end):
    if (i % 2 == 0) and (i % 3 != 0) :
        print(i)
```

```
26
28
32
34
38
40
44
```

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

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

```
55
```

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

In [16]:
```
n = int(input("Enter the number : "))
sum = 0
for i in range(1,n+1):
    sum += i**2
print(sum)
```

```
385
```

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

In [20]:
```
n = int(input("Enter the number : "))
sum = 0
for i in range(1,n+1):
    if (i%2==0):
        sum -= i
    else:
        sum += i
print(sum)
```

```
-5
```

08) WAP to print multiplication table of given number.

In [22]:
```
n = int(input("Enter the number : "))
for i in range(1,11):
    print(f"{n} * {i} = {n*i}")
```

```
21 * 1 = 21
21 * 2 = 42
21 * 3 = 63
21 * 4 = 84
21 * 5 = 105
21 * 6 = 126
21 * 7 = 147
21 * 8 = 168
21 * 9 = 189
21 * 10 = 210
```

09) WAP to find factorial of the given number.

In [24]:
```
n = int(input("Enter the number : "))
fac = 1
for i in range(1,n+1):
    fac *= i
print(f"factorial of {n} : {fac}")
```

```
factorial of 5 : 120
```

10) WAP to find factors of the given number.

In [25]:
```
n = int(input("Enter the number : "))
for i in range(1,n+1):
    if (n%i==0):
        print(i)
```

```
1
2
5
10
```

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

In [69]:
```
n = int(input("Enter the number : "))
for i in range(2,(n//2)+1):
    if (n%i==0):
        print(f"{n} is not a Prime Number.")
        break
else:
    print(f"{n} is a Prime Number.")
```

```
4 is not a Prime Number.
```

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

In [38]:
```
n = int(input("Enter the number : "))
sum = 0
while (n!=0):
    sum += int(n%10)
    n=n/10
print(f"sum : {sum}")
```

```
sum : 15
```

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

In [59]:
```
n = int(input("Enter the number : "))
temp = n
reverse = 0
while temp != 0:
    reverse = reverse * 10 + temp % 10
    temp = int(temp / 10) # n//
if (reverse == n):
    print(f"{n} is a Palindrome Number.")
else:
    print(f"{n} is Not a Palindrome Number.")
```

```
121 is a Palindrome Number.
```

14) WAP to print GCD of given two numbers.

In [6]:
```
n1 = int(input("Enter the first number : "))
n2 = int(input("Enter the second number : "))
gcd = 0
for i in range (1,(min(n1,n2))+1):
    if (n1%i==0) and (n2%i==0):
        gcd = i
print(f"GCD of {n1} and {n2} is = {gcd}")
```

```
GCD of 15 and 45 is = 15
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 4

223 | Vishal Baraiya | 23010101014

String

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

In [71]:
```
s = input("Enter the String : ")
if (s.lower() == s[::-1].lower()):
    print(f"{s} is a Palindrome String.")
else:
    print(f"{s} is Not a Palindrome String.")
```

```
Nayan is a Palindrome String.
```

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

In [2]:
```
s = input("Enter the String : ")
l = s.split(" ")
l1 = []
for i in l:
    l1.append(i[::-1])
s  = " ".join(l1)
print(s)
```

```
lahsiV ayiaraB
```

03) WAP to remove ith character from given string.

In [73]:
```
s = input("Enter the String : ")
i = int(input("Enter the i : "))
```

```
if 0 < i <= len(s):
    words = s[:i-1]+s[i:]

print(f"{s} is after removeing {i}th Character is = {words}")
```

```
Karan is after removeing 4th Character is = Karn
```

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

In [74]:
```
s = input("Enter the String : ")
count = 0
for i in s:
    count += 1

print(f"Length of {s} = {count}")
```

```
Length of Vishal = 6
```

05) WAP to print even length word in string.

In [75]:
```
s = input("Enter the String : ")
words = s.split()
for i in words:
    if (len(i) % 2 == 0):
        print(i)
    else:
        pass
```

```
Vishal
```

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

In [77]:
```
s = input("Enter the String : ")
s = s.lower()
vowels = ['a','e','i','o','u']
count = 0
for i in s:
    if i in vowels:
        count += 1
    else:
        pass
print(f"Total Vowels in {s} = {count}")

# if s.count('a'):
#     print("Count Of A = ",s.count('a'))
# if s.count('e'):
#     print("Count Of E = ",s.count('e'))
# if s.count('i'):
#     print("Count Of I = ",s.count('i'))
# if s.count('o'):
#     print("Count Of O = ",s.count('o'))
# if s.count('u'):
#     print("Count Of U = ",s.count('u'))
```

```
Total Vowels in vishal = 2
```

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

In [12]:
```
s = input("Enter the String : ")
list = s.split()
res = []
for i in list:
    res.append(i[0].upper()+i[1:-1].lower()+i[-1].upper())

res = " ".join(res)
print(res)
```

```
WelcomE TO HomE
```

08) WAP to convert given array to string.

In [79]:
```
del str
```

In [80]:
```
num_list = [1, 2, 3, 4, 5]
result = ",".join(str(num) for num in num_list)
print(result)
```

```
1,2,3,4,5
```

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

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

In [41]:
```
password = input("Enter the Password : ")
confirmPassword = input("Enter the Password : ")

if(password == confirmPassword):
    print("Successfully Matched.")
elif (password.lower() == confirmPassword.lower()):
    print("Check in your case.")
else:
    print("Password Does Not Matched")
```

```
Successfully Matched.
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [57]:
```
card_no = "1234 5678 9012 3456"
first = card_no[0:len(card_no)-4:]
last = card_no[len(card_no)-4::]
res = " "
for i in first:
    if(i.isdigit()):
        res += "*"
    else:
        res += " "
res = res + last
print(res)
```

```
**** **** **** 3456
```

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

s1 = decimal and s2 = medical are Anagram

In [69]:
```
string1 = input("Enter the str1 : ") # "decimal"
string2 = input("Enter the str2 : ") # "medical"
string1 = string1.replace(" ", "").lower()
string2 = string2.replace(" ", "").lower()
if sorted(string1) == sorted(string2):
    print(f"{s1} and {s2} is Anagram")
else:
    print(f"{s1} and {s2} is Not a Anagram")
```

```
Silent and Lisent is Anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [83]:
```
s = input("Enter the String : ")
lower = ""
upper = ""
for i in s:
    if i.isupper():
        upper = upper + i
    elif i.islower():
        lower = lower + i
s = lower + upper
print(s)
```

```
ishalaraiyaotadVBB
```

Python Programming - 2301CS404

Lab - 5

223 | Vishal Baraiya | 23010101014

List

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

In [5]:
```
l = [12,34,21,13,45]
sum = 0
for i in l:
    sum += i
print("Sum : ",sum)
```

```
Sum :  125
```

02) WAP to find largest element in a List.

In [6]:
```
l = [12,34,21,13,45]
max = l[0]
for i in l:
    if (max < i):
        max = i
print("Max : ",max)
```

```
Max :  45
```

03) WAP to find the length of a List.

In [8]:
```
l = [12,34,21,13,45]
length = 0
for i in l:
    length += 1
print("Length Of List : ",length)
```

```
Length Of List :  0
```

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

In [22]:
```
l = [12,34,56,78,90]
if (len(l) == 0 or len(l) == 1):
    print("List is Empty or contains only one elemant.")
else:
    temp = l[0]
    l[0] = l[-1]
    l[-1] = temp
print(l)
```

```
[90, 34, 56, 78, 12]
```

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

In [20]:
```
l = [12,1,2,3,5,6,7,9]
l1 = l[0:int(len(l)/2)]
l2 = l[int(len(l)/2):len(l)]
l2.extend(l1)
print(l2)
```

```
[5, 6, 7, 9, 12, 1, 2, 3]
```

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

In [23]:
```
l = [12,1,2,3,5,6,7,9]
print("Length of the List : ",len(l))
p1 = int(input("Enter the Position : "))
p2 = int(input("Enter the Position : "))
if (len(l)==0 or p1>len(l) or p2>len(l)):
    print("Length of the list is small then position.")
temp = l[p1]
l[p1] = l[p2]
l[p2] = temp
print(l)
```

```
[12, 1, 3, 2, 5, 6, 7, 9]
```

07) WAP to reverse the list entered by user.

In [25]:
```
n = int(input("Enter the Total Numbers you want to add: "))
l1 = list()
for i in range(0,n):
    l1.append(int(input("Enter the Number : ")))
print(l1)
```

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

08) WAP to print even numbers in a list.

In [26]:
```
n = int(input("Enter the size : "))
l1 = list()
```

```
for i in range(0,n):
    l1.append(int(input("Enter the Number : ")))
even = []
for i in l1:
    if (i % 2 == 0):
        even.append(i)
print(even)
```

```
[2, 4]
```

09) WAP to count unique items in a list.

In [38]:
```
l = [1,2,3,4,5,2,3,4]
print(l)
l2 = set(l)
print(l2)
```

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

10) WAP to copy a list.

In [39]:
```
l = [1,2,3,4,5]
l2 = l.copy()
print(l2)
```

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

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

In [42]:
```
start = int(input("Enter the Starting Point : "))
end = int(input("Enter the Ending Point : "))
for i in range(start,end+1):
    if (i % 2 != 0):
        print(i)
```

```
1
3
5
7
9
```

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

In [46]:
```
l1 = [12,34,5,6,5,6,12,78,90,90]
l2 = list(set(l1))
for i in l2:
    print(i,":",l1.count(i))
```

```
34 : 1
5 : 2
6 : 2
12 : 2
78 : 1
90 : 2
```

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

In [49]:
```
l = [1,2,3,4,5]
max1 = l[0]
for i in l:
    if (max1 < i):
        max1 = i
l.remove(max1)
max2 = l[0]
for i in l:
    if (max2 < i):
        max2 = i
print("Second Largest : ",max2)
```

```
Second Largest :  4
```

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

In [57]:
```
l1 = [12,34,5,6,5,6,12,90,90,90]
k = int(input("Enter the frequency : "))
l2 = list()
for i in l1:
    if (l1.count(i) > k):
        l2.append(i)
l2 = list(set(l2))
print(l2)
```

```
[90, 12, 5, 6]
```

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

In [61]:
```
l1 = []
for i in range(0,10):
    l1.append(i ** 2)
print (l1)

l2 = [i ** 2 for i in range(0,10)]
print(l2)
```

```
[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 [67]:
```
n = int(input("Enter the size : "))
l = list()
fruits = list()
for i in range(0,n):
    fruits.append(input("Enter the Fruit Name : "))

for fruit in fruits:
    if(fruit[0] == 'b'):
        l.append(fruit)
print(l)
```

```
['banana', 'blue bary']
```

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

In [3]:
```
l1 = [1,2,3,4,5]
l2 = [1,3,5,7,9]
newList = []
for i in l1:
    if(i in l2):
        newList.append(i)
print(newList)
```

```
[1, 3, 5]
```

Python Programming - 2301CS404

Lab - 6

223 | Vishal Baraiya | 23010101014

Tuple

01) WAP to find sum of tuple elements.

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

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

```
Sum =  15
```

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

In [6]:
```python
t1 = (1,2,3,4,5,6,7,8,9,10)
l = list(t1)
l.sort()
t1 = tuple(l)
k = int(input("Enter the Number : "))
print(f"{k} Minimum Numbers is : ")
for i in range(0,k):
        print(t1[i])

print(f"{k} Maximum Numbers is : ")
for i in range(-1,-k-1,-1):
        print(t1[i])
```

```
2 Minimum Numbers is :
1
2
2 Maximum Numbers is :
10
9
```

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

In [10]:
```python
def allAreDevisible(t,k):
    for i in t:
        if (i % k != 0):
            return False
    else:
        return True

t1 = (6,7,8,9,10)
t2 = (2,4,6,8,10)
t3 = (3,6,9)
t4 = (4,8)

l = [t1,t2,t3,t4]
k = int(input("Enter the Number : "))
for i in t:
    if( allAreDevisible(i,k)):
        print(i)
```

```
(2, 4, 6, 8, 10)
(4, 8)
```

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

In [16]:
```python
l = [1,2,3,4,5]

# First Way
l1 = []
for i in l:
    l1.append((i,i**3))
print(l1)

# Another Way
l2 = [(i,i**3) for i in l]
print(l2)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
[(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 [20]:
```python
def allArePositive(t):
    for i in t:
        if (i < 0):
            return False
    else:
        return True

t1 = (-6,7,-8,9,10)
t2 = (2,4,6,8,10)
t3 = (3,6,-9)
t4 = (4,8)
l = [t1,t2,t3,t4]

for i in l:
    if(allArePositive(i)):
        print(i)
```

```
(2, 4, 6, 8, 10)
(4, 8)
```

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

In [23]:
```
t1 = (1,2,3)
t2 = (4,5,6,7,8)
t3 = (3,6,-9)
l = [t1,t2,t3]
print(l)

# vice-verasa
l1 = [1,2,3]
l2 = [4,5,6,7,8]
l3 = [3,6,-9]
t = (l1,l2,l3)
print(t)
```

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

07) WAP to remove tuples of length K.

In [24]:
```
t1 = (-6,7,-8,9,10)
t2 = (2,4,6,8,10)
t3 = (3,6,-9)
t4 = (4,8)
l = [t1,t2,t3,t4]
k = int(input("Enter the length : "))
for i in l:
    if(len(i) == k):
        l.remove(i)

print(l)
```

```
[(-6, 7, -8, 9, 10), (2, 4, 6, 8, 10), (4, 8)]
```

08) WAP to remove duplicates from tuple.

In [26]:
```
t = (1,2,3,1,2,4,5,6,2,3,5)
s = set(t)
t = tuple(s)
print(t)
```

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

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

In [37]:
```
t = (1, 2, 3, 4, 5, 6)
l1 = []
l = list(t)
prev = l[0]
for i in range(1,len(l)):
    l1.append(l[i]*prev)
    prev = l[i]

t = tuple(l1)
print(t)
```

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

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

In [43]:
```
t1 = (1,2,3,4,5)
t2 = (1,2,3,4,5)

if (len(t1) != len(t2)):
    print("tuples is Not distinct.")

else:
    for i in range(0,len(t1)):
        if (t1[i] != t2[i]):
            print("tuples is Not distinct.")
            break
    else :
        print("tuples is distinct.")
```

```
tuples is distinct.
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

223 | Vishal Baraiya | 23010101014

Set & Dictionary

01) WAP to iterate over a set.

In [1]:
```
set1 = {1,2,3,4,5}
for i in set1:
    print(i)
```
```
1
2
3
4
5
```

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

In [12]:
```
set1 = {1,2,3,4,5,6,7,8,9}
l = list(set1)
s = ''.join([str(i) for i in l])
t = tuple(set1)

print(type(l)," : ",l)
print(type(s)," : ",s)
print(type(t)," : ",t)
```
```
<class 'list'>  :  [1, 2, 3, 4, 5, 6, 7, 8, 9]
<class 'str'>  :  123456789
<class 'tuple'>  :  (1, 2, 3, 4, 5, 6, 7, 8, 9)
```

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

```
In [20]: set1 = {1,2,3,15,-12,4,5,6,7,8,9}
         # print("max = ",max(set1))
         # print("max = ",min(set1))
         l = list(set1)
         minnum = l[0]
         maxnum = l[0]
         for i in l:
             if(maxnum < i):
                 maxnum = i
             if(minnum > i):
                 minnum = i
         print("max = ",maxnum)
         print("min = ",minnum)
```

```
max =  15
min =  -12
```

04) WAP to perform union of two sets.

```
In [23]: set1 = {1,2,3,4,5}
         set2 = {2,4,6,8,10}

         print(set1.union(set2))
```

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

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

```
In [28]: l1 = [12,3,5,4,5]
         l2 = [1,2,3,4]
         set1 = set(l1)
         set2 = set(l2)
         if len(set1.intersection(set2)) >= 1:
             print(set1.intersection(set2))
         else:
             print("list have not common elemants")
```

```
{3, 4}
```

06) WAP to remove duplicates from list.

```
In [42]: l = [1,2,3,4,5,3,4,2,6,6,6,6]
         l = list(set(l))
         print(l)
```

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

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

```
In [47]: s = "My Name is Tony Stark Tony Stark "
         set1 = set(s.split(" "))
         print(set1)
```

```
{'My', 'Stark', '', 'is', 'Name', 'Tony'}
```

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

In [3]:
```python
a = {1,2,3,4,5}
b = {2,4,6,8,10}
a.difference_update(b)
print(a)
```

```
{1, 3, 5}
```

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

In [63]:
```python
s1 = "abcd"
s2 = "dabc"
set1 = set(s1)
set2 = set(s2)
if (len(s1) == len(s2)) and (set1 == (set1 & set2)):
    print(f"{s1} and {s2} is anagram.")
else:
    print(f"{s1} and {s2} is Not anagram.")
```

```
abcd and dabc is anagram.
```

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

In [64]:
```python
l1 = [1,3,5,7,9]
l2 = [2,3,5,7]
l3 = [1,2,3,4,5,6,7,8]

set1 = set(l1)
set2 = set(l2)
set3 = set(l3)

print(set1 & set2 & set3)
```

```
{3, 5, 7}
```

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

In [72]:
```python
set1 = {'a','e','i','o','u'}
s = input("Enter the String : ")
count_vowels = 0
for i in s.lower():
    if i in set1:
        count_vowels+=1;
    else:
        pass
print(f"Number of Vowels in {s} : {count_vowels}")
```

```
Number of Vowels in aEiod : 4
```

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

In [82]:
```
set1 = {'1','0'}
s = input("Enter the String : ")
count_vowels = 0
for i in s:
    if i not in set1:
        print(f"{s} is Not a Binary String.")
        break
else:
    print(f"{s} is a Binary String.")
```

```
fcgv is Not a Binary String.
```

13) WAP to sort dictionary by key or value.

In [16]:
```
d = {'d': 1, 'b': 2, 'c': 3, 'a' : 4}
d1 = dict(sorted(d.items()))
print(d1)
d2 = dict(sorted(d.items(), key=lambda item: item[1]))
print(d2)
# l1 = list(d.keys())
# l1.sort()
# for i in l1:
#     print(f"{i} : {d[i]}")
```

```
{'a': 4, 'b': 2, 'c': 3, 'd': 1}
{'d': 1, 'b': 2, 'c': 3, 'a': 4}
```

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

In [91]:
```
d = {}
for i in range(1,6):
    n = int(input("Enter the Number"))
    d[i] = n
ans = 0
for i in d.keys():
    ans += d[i]
print("ans = ",ans)
```

```
ans =  270
```

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 [19]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = input("Enter the Key : ")
if key in dict1.keys():
    print(f"{key} : {dict1[key]}")
else:
    print("Key Not Found")
```

```
c : 8
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 8

223 | Vishal Baraiya | 23010101014

User Defined Function

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

In [3]:
```
def calBMI(mass,height):
    return mass/height**2

mass = int(input("Enter the Mass : "))
height = int(input("Enter the Height : "))
print(f"BMI = {calBMI(mass,height)}")
```

```
BMI = 2.0
```

02) Write a function that add first n numbers.

In [3]:
```
def sumOfFirstN(n):
    sum = 0
    for i in range(0,n+1):
        sum += i
    return sum

n = int(input("Enter the Number : "))
print(f"Sum of first {n} numbers is = {sumOfFirstN(n)}")
```

```
Sum of first 10 numbers is = 55
```

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

In [7]:
```
def isPrime(n):
    count = 0
    for i in range(1,n+1):
        if(n % i == 0):
            count += 1
        else:
            pass
    if count == 2 :
        return 1
    else:
        return 0

n = int(input("Enter the Number : "))
if isPrime(n):
    print(f"{n} is a Prime Number.")
else:
    print(f"{n} is Not a Prime Number.")
```

```
4 is Not a Prime Number.
```

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

In [9]:
```
def isPrime(n):
    count = 0
    for i in range(1,n+1):
        if(n % i == 0):
            count += 1
        else:
            pass
    if count == 2 :
        return True
    else:
        return False

def findPrimeInRange (start,end):
    l = []
    for i in range(start,end):
        if isPrime(i):
            l.append(i)
        else:
            pass
    return l
start = int(input("Enter the Start : "))
end = int(input("Enter the end : "))
l = findPrimeInRange(start,end)
print(l)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

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

In [17]:
```
def isPalindrome(s):
    s = s.lower()
    return s == s[::-1]

s = input("Enter the String : ")
```

```
if isPalindrome(s):
    print(f"{s} is a Palindrome string.")
else:
    print(f"{s} is Not a Palindrome string.")
```

```
nayan is a Palindrome string.
```

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

In [19]:
```
def sumOfAllElemantsOfList(l):
    ans = 0
    for i in l:
        ans += i
    return ans

l = [1,2,34,56,2,312,32]
print(f"Sum Of all Elemants Of List is : {sumOfAllElemantsOfList(l)}")
```

```
Sum Of all Elemants Of List is : 439
```

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

In [8]:
```
def sumOfFirstElemantOfListOfTuples(l):
    ans = 0
    for i in l:
        ans += i[0]
    return ans

l = [(1,2),(12,56),(23,45),(45,34),(5,78)]
print("Sum = ",sumOfFirstElemantOfListOfTuples(l))
```

```
Sum =  86
```

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

In [36]:
```
def findNthTermOfFibbonacci(n):
    t1 = 0
    t2 = 1
    if (n == 0):
        return 0
    s = 0
    for i in range(2,n+1):
        t1 = t2
        t2 = s
        s= t1 + t2
    return s

for i in range(1,10):
    print(findNthTermOfFibbonacci(i))
```

```
0
1
1
2
3
5
8
13
21
```

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 [37]:
```
d = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}

def findNameFromDict(rollNo):
    global d
    return d[rollNo]

print(f"103 : {findNameFromDict(103)}")
```

```
103 : 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 getScoreEndWithZero(l):
    sum = 0
    for i in l:
        if i % 10 == 0:
            sum += i
        else:
            pass
    return sum
scores = [200, 456, 300, 100, 234, 678,20]
print(f"Sum = {getScoreEndWithZero(scores)}")
```

```
Sum = 620
```

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 [52]:
```
def invertDictionary(d):
    di = {}
    for i in d.keys():
        di[d[i]] = i
    return di

d = {'a': 10, 'b':20, 'c':30, 'd':40}
print("Before : ",d)
d = invertDictionary(d)
print("After : ",d)
```

```
Before :  {'a': 10, 'b': 20, 'c': 30, 'd': 40}
After :  {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 [62]:
```
def isPanagram(s):
    s=s.replace(" ","")
    s=s.lower()
    s=list(set(s))
    s.sort()
    s="".join(s)
    alphabets = "abcdefghijklmnopqrstuvwxyz"
    return s == alphabets
s = "the quick brown fox jumps over the lazy dog"
print(f"{isPanagram(s)}")
```

```
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 [67]:
```
def count_lower(s : str):
    countLower = 0
    for i in s:
        if i.islower():
            countLower += 1
    return countLower
def count_upper(s : str):
    countUpper = 0
    for i in s:
        if i.isupper():
            countUpper += 1
    return countUpper

s1 = "AbcDEfgh"
print(f"Count Of Lower : {count_lower(s1)}")
print(f"Count Of Upper : {count_upper(s1)}")
```

```
Count Of Lower : 5
Count Of Upper : 3
```

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

In [72]:
```
largestOfTwo = lambda a,b : a if (a > b) else b
print(largestOfTwo(12,34))
```

```
34
```

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

In [74]:
```
name = ["abcdefgh","Vishal Baraiya","Rahim Roda","asdf","qwerty"]
l = filter(lambda i : len(i) > 7, name)
print(list(l))
```

```
['abcdefgh', 'Vishal Baraiya', 'Rahim Roda']
```

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

In [77]:
```
name = ["abcdefgh","Vishal Baraiya","Rahim Roda","asdf","qwerty"]
l = map(lambda i : i.capitalize(), name)
print(list(l))
```

```
['Abcdefgh', 'Vishal baraiya', 'Rahim roda', 'Asdf', 'Qwerty']
```

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 [83]:
```
def printNameAge(name = "Vishal", age = 18):
    print("Hi, I am", name)
    print("My age is ", age)

printNameAge("Prince", 20) # Positional Arguments , Positional Only Arguments
printNameAge(age=20, name="Prince") # Keyword Arguments , Keyword-Only
printNameAge() # Default Arguments

def sum_all(*args):
    result = 0
    for num in args:
        result += num
    return result

print(sum_all(1, 2, 3, 4, 5)) # Variable Legngth Positional(*args)

def mul_all(*args):
    result = 1
    for num in args:
        result *= num
    return result

print(mul_all(1, 2, 3, 4, 5)) # variable length Keyword Arguments (**kwargs)
```

```
Hi, I am Prince
My age is  20
Hi, I am Prince
My age is  20
Hi, I am Vishal
My age is  18
15
120
```

In [84]:

```
from functools import reduce

# Function to add two numbers
def add(x, y):
    return x + y

a = [1, 2, 3, 4, 5]
res = reduce(add, a)

print(res)  # Output: 15
```

```
15
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 9

223 | Vishal Baraiya | 23010101014

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]:
```
# Read by File Pointer in another directory
# fp = open("D:\\Media\\demo.txt","r")
# for i in fp:
#     print(i,end="")
# fp.close()

# Read by File Pointer
fp = open("Demo.txt","r")
for i in fp:
    print(i,end="")
fp.close()

# Read by Read()
print("\n")
fp = open("Demo.txt","r")
print(fp.read())
fp.close()

# Read by Readline()
print("\n")
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
fp = open("Demo.txt","r")
while True:
    line = fp.readline()  # Read a single line
    if not line:            # Break the Loop if EOF is reached
        break
    print(line,end="")
fp.close()

# Read by Readlines()
print("\n")
fp = open("Demo.txt","r")
l = fp.readlines()
print(l)
fp.close()
```

```
I am Python.
This is Demo Text File.

I am Python.
This is Demo Text File.


I am Python.
This is Demo Text File.

['I am Python.\n', 'This is Demo Text File.']
```

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

In [17]:
```
with open("new.txt","x") as fp:
    fp.write("Hello World!")

with open("new.txt","r") as fp:
    print(fp.read())
```

```
Hello World!
```

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

In [19]:
```
# Open the file
with open('demo.txt', 'r') as fp:
    for i in range(5):
        line = file.readline()
        if not line:  # Break if the file has less than 5 lines
            break
        print(line.strip())  # Print the line without extra whitespace
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor in
cididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliqui
p ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu f
ugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserun
t mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor in
cididunt ut labore et dolore magna aliqua.
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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

In [33]:
```
with open("demo.txt","r") as fp:
    l = fp.read().split()
    maxLengthS = " "
    for i in l:
        if (len(i) > len(maxLengthS)):
            maxLengthS = i
    print("Highest Length String :",maxLengthS)
```

```
Highest Length String : reprehenderit
```

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

In [45]:
```
with open("demo.txt","r") as fp:
    print("No. Of Lines : ",len(fp.readlines()))
    fp.seek(0,0)
    print("No. Of Words : ",len(fp.read().split()))
    fp.seek(0,0)
    print("No. Of Charcters : ",len(fp.read().replace("\n","")))
```

```
No. Of Lines :  8
No. Of Words :  138
No. Of Charcters :  884
```

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

In [46]:
```
s = " "
with open("demo.txt","r") as fp:
    s = fp.read()
with open("demo-copy.txt","w") as fp:
    fp.write(s)
```

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

In [47]:
```
import os

# Specify the file path
file_path = "demo.txt"

# Check if the file exists
if os.path.exists(file_path):
    # Get the size of the file in bytes
    file_size = os.path.getsize(file_path)
    print(f"The size of the file '{file_path}' is {file_size} bytes.")
else:
    print(f"The file '{file_path}' does not exist.")
```

```
The size of the file 'demo.txt' is 891 bytes.
```

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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [52]:
```python
def frequency(file_path,word):
    with open(file_path,'r')as fp:
        l = fp.read().split()
        count = 0
        for i in l:
            if word.lower() == i.lower():
                count += 1
    print(f"Count Of {word} is = {count}")

frequency("demo.txt","Lorem")
```

```
Count Of Lorem is = 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 [7]:
```python
with open("scores.txt","w") as fp:
    scores = []
    n = int(input("Enter the Length of List : "))
    for i in range(0,n,1):
        temp = int(input(f"Enter the Score of {i}th subject : "))
        scores.append(temp)
    for i in scores:
        fp.write(str(i))
        fp.write("\n")

with open("scores.txt","r") as fp:
    l = fp.read().split()
    print(f"Max Score : {max(l)}")
```

```
Max Score : 89
```

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

(Note: each number should be in new line)

In [11]:
```python
def isPrime(n = int):
    for i in range(2,n//2+2):
        if(n % i == 0):
            return False
    else:
        return True
with open("primenumbers.txt","w") as fp:
    i,count = 2,0
    while(count != 100):
        if isPrime(i):
            fp.write(str(i))
            fp.write("\n")
            count += 1
        i += 1
print("File Write Successfully!")
```

```
File Write Successfully!
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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

In [13]:
```
fp1 = open("scores.txt","r")
fp2 = open("primenumbers.txt")
fp3 = open("merged.txt","w")
fp3.write(fp1.read())
fp3.write(fp2.read())
fp1.close()
fp2.close()
fp3.close()
print("File Merged Successfully!")
```

```
File Merged Successfully!
```

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

In [27]:
```
word1 = input("Enter th word1 : ")
word2 = input("Enter th word2 : ")
s = " "
fp1 = open("demo.txt","r")
fp2 = open("new.txt","w")
s = fp1.read()
fp2.write(s.replace(word1,word2))
fp1.close()
fp2.close()
print("Successfully complated!")
```

```
Successfully complated!
```

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

In [31]:
```
with open("demo.txt","rb") as fp:
    print(fp.read(10))
    print("Current Position : ",fp.tell())
    fp.seek(3,0) # 0 is use for from begining to 3 offset
    print("Current Position : ",fp.tell())
    fp.seek(5,1) # 1 is use for from Current Position to 5 offset
    print("Current Position : ",fp.tell())
    fp.seek(-3,2) # 2 is use for from ending to -3 offset
    print("Current Position : ",fp.tell())

# seek() second args 1,2 use in only binary files
```

```
b'Lorem ipsu'
Current Position :  10
Current Position :  3
Current Position :  8
Current Position :  918
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Python Programming - 2301CS404

Lab - 10

223 | Vishal Baraiya | 23010101014

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 = int(input("Enter the Number : "))
    b = int(input("Enter the Number : ")) # Value Error a = "str"
    c = a / b
    print (f"{a}/{b} = {c}") # Zero Division Error
    print('a'+1) # Type Error
except ZeroDivisionError :
    print("Zero Division Error")
except ValueError :
    print("Value Error")
except TypeError :
    print("Type Error")
```

```
12/3 = 4.0
Type Error
```

02) WAP to handle following exceptions:

1. IndexError

2. KeyError

In [14]:
```
try:
    # a = [1, 2, 3]
    # print(a[5])  # IndexError

    d = {'a':1,'b':2,'c':3}
    print(d[2])  # KeyError

except IndexError as ie:
    print(type(ie).__name__,":",ie)
    print("Index Error is Occured.")
except KeyError as ke:
    print(type(ke).__name__,":",ke)
    print("Key Error is Occured.")
```

```
1
KeyError : 2
Key Error is Occured.
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [17]:
```
try:
    import index

    # with open('demo.txt') as fp:
    #     print(fp.read())
except FileNotFoundError as e:
    print(type(e).__name__,e)
except ModuleNotFoundError as e:
    print(type(e).__name__,e)
```

```
ModuleNotFoundError No module named 'index'
```

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

In [20]:
```
try :
    print(1/0)
except Exception as e:
    print(type(e).__name__,e)
```

```
ZeroDivisionError division by zero
```

05) WAP to demonstrate else and finally block.

In [23]:
```
try:
    a = int(input("Enter the Number : "))
    b = int(input("Enter the Number : "))
    ans = a / b
    print(f"{a} / {b} = {ans}")
except Exception as e:
    print(type(e).__name__,e)
else:
    print("Else block Executed!") # It will Execute when Error Will not Occured
finally:
    print("Finally block Executed!") # It will Execute Every Time
```

```
10 / 2 = 5.0
Else block Executed!
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 [29]:
```
try:
    s = input("Enter the Grades using ( , ) Seprated : ")
    l = s.split(',')
    l = [int(i) for i in l]
    print(l)
except Exception as e:
    print(type(e).__name__,e)
```

```
ValueError invalid literal for int() with base 10: 'rt'
```

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

In [36]:
```
def divide(a,b):
    try:
        return a/b
    except Exception as e:
        print(type(e).__name__,e)

divide(12,0)
```

```
ZeroDivisionError 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 [47]:
```
try:
    age = int(input("Enter the Age : "))
    if (age < 18):
        raise ValueError("This is Value Error Genrated By me.")
    else:
        print(f"Age = {age}")

except ValueError as e:
    print(type(e).__name__,e)
```

```
Age = 18
```

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 [58]:
```python
class InvalidUsernameError(Exception) :
    def __init__(self,msg):
        self.msg = msg
try:
    name = input("Enter the User Name : ")
    if (len(name) >= 5) and (len(name) <= 15):
        print(f"UserName : {name}")
    else:
        raise InvalidUsernameError("Username must be between 5 and 15 characters
except Exception as e:
    print(type(e).__name__,e)
```

```
InvalidUsernameError Username must be between 5 and 15 characters long
```

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

if the given number is negative.

otherwise print the square root of the given number.

In [65]:
```python
import math
class NegativeNumberError(Exception) :
    def __init__(self,msg):
        self.msg = msg
try:
    n = int(input("Enter the Number : "))
    if (n < 0):
        raise NegativeNumberError("Cannot calculate the square root of a negativ
    else:
        print(f"sqrt Of {n} = {math.sqrt(n)}")
except Exception as e:
    print(type(e).__name__,e)
```

```
NegativeNumberError Cannot calculate the square root of a negative number.
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 11

223 | Vishal Baraiya | 23010101014

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 Calculator
n1 = int(input("Enter the First Number : "))
n2 = int(input("Enter the Second Number : "))

print(f"{n1} + {n2} = {Calculator.add(n1,n2)}")
print(f"{n1} - {n2} = {Calculator.sub(n1,n2)}")
print(f"{n1} * {n2} = {Calculator.mul(n1,n2)}")
print(f"{n1} / {n2} = {Calculator.div(n1,n2)}")
```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
```

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

In [3]:
```
import random
s = input("Enter String : ")
ch = random.choice(s)
print(ch)
```

```
i
```

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

In [7]:
```
l = [1,2,3,4,5,6,7,8,9,10]
n = random.choice(l)
print(n)
```

```
2
```

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

In [22]:
```
import random

def roll_dice():
    random.seed(1)
    return random.randint(1,6)

print(roll_dice())
print(roll_dice())
print(roll_dice())
```

```
2
2
2
```

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

In [27]:
```
import random
for i in range(1,4):
    n = random.randrange(100,999,5)
    print(n)
```

```
730
675
700
```

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 [42]:
```
import random
l = [i for i in range(1,101)]

print(f"Winner is : {random.choice(l)}")
print(f"Runner up is : {random.choice(l)}")
```

```
Winner is : 49
Runner up is : 88
```

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

In [46]:
```
import datetime

print(f"Current Time : {datetime.datetime.now()}")
```

```
Current Time : 2025-02-10 13:03:56.890116
```

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

In [48]:
```
import datetime

d = datetime.datetime.now()
df = d - datetime.timedelta(days=7)
print(df)
```

```
2025-02-03 13:10:03.039109
```

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

In [53]:
```
s1 = input("Enter the Date (dd-mm-yyyy) : ")
s2 = input("Enter the Date (dd-mm-yyyy) : ")

d1 = datetime.datetime.strptime(s1,"%d-%m-%Y")
d2 = datetime.datetime.strptime(s2,"%d-%m-%Y")

print(abs(d1-d2).days)
```

```
6889
```

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

In [56]:
```
s = input("Enter the Date (dd-mm-yyyy) : ")
d = datetime.datetime.strptime(s,"%d-%m-%Y")

s = d.strftime("%A")
print(s)
```

```
Monday
```

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

In [57]:
```
import datetime

# Get the current date and time
current_datetime = datetime.datetime.now()
print("Current Date and Time:", current_datetime)

# Get the current date
current_date = datetime.date.today()
print("Current Date:", current_date)

# Create a specific date
specific_date = datetime.date(2025, 2, 10)
print("Specific Date:", specific_date)

# Get individual components
print("Year:", current_date.year)
print("Month:", current_date.month)
print("Day:", current_date.day)
```

```
# Time delta example (difference between dates)
delta = datetime.timedelta(days=10)
future_date = current_date + delta
print("Date after 10 days:", future_date)

# Formatting date and time
formatted_datetime = current_datetime.strftime("%Y-%m-%d %H:%M:%S")
print("Formatted Date and Time:", formatted_datetime)

# Parsing a date string
date_string = "2025-02-10 15:30:00"
parsed_date = datetime.datetime.strptime(date_string, "%Y-%m-%d %H:%M:%S")
print("Parsed Date and Time:", parsed_date)
```

```
Current Date and Time: 2025-02-10 13:30:23.860156
Current Date: 2025-02-10
Specific Date: 2025-02-10
Year: 2025
Month: 2
Day: 10
Date after 10 days: 2025-02-20
Formatted Date and Time: 2025-02-10 13:30:23
Parsed Date and Time: 2025-02-10 15:30:00
```

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

In [58]:
```
import math

# Square root
num = 16
print("Square root of", num, "is", math.sqrt(num))

# Factorial
num = 5
print("Factorial of", num, "is", math.factorial(num))

# Power
base = 2
exp = 3
print(base, "raised to the power of", exp, "is", math.pow(base, exp))

# Trigonometric functions
angle = math.radians(30)  # Convert degrees to radians
print("Sine of 30 degrees:", math.sin(angle))
print("Cosine of 30 degrees:", math.cos(angle))
print("Tangent of 30 degrees:", math.tan(angle))

# Logarithm
num = 100
print("Natural logarithm of", num, "is", math.log(num))
print("Base-10 logarithm of", num, "is", math.log10(num))

# Constants
print("Value of Pi:", math.pi)
print("Value of Euler's number (e):", math.e)
```

```
Square root of 16 is 4.0
Factorial of 5 is 120
2 raised to the power of 3 is 8.0
Sine of 30 degrees: 0.49999999999999994
Cosine of 30 degrees: 0.8660254037844387
Tangent of 30 degrees: 0.5773502691896257
Natural logarithm of 100 is 4.605170185988092
Base-10 logarithm of 100 is 2.0
Value of Pi: 3.141592653589793
Value of Euler's number (e): 2.718281828459045
```

```
In [ ]:
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

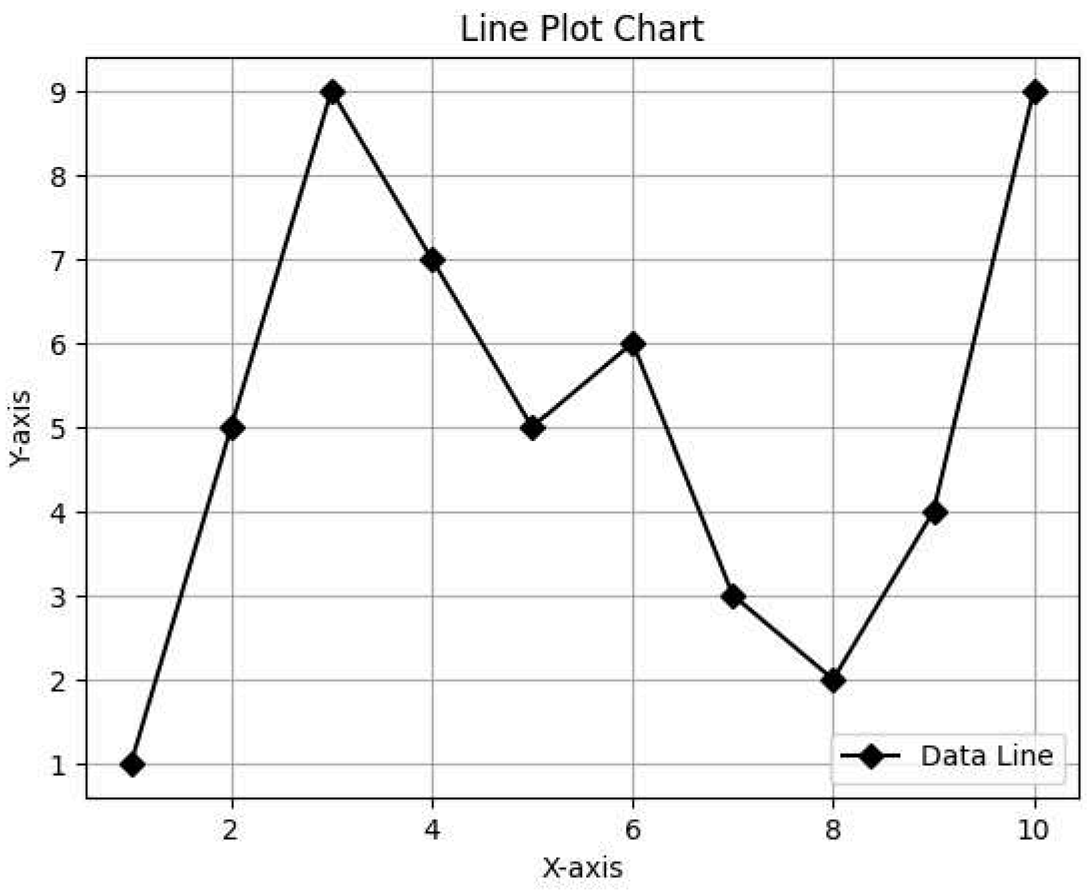
Python Programming - 2301CS404

Lab - 12

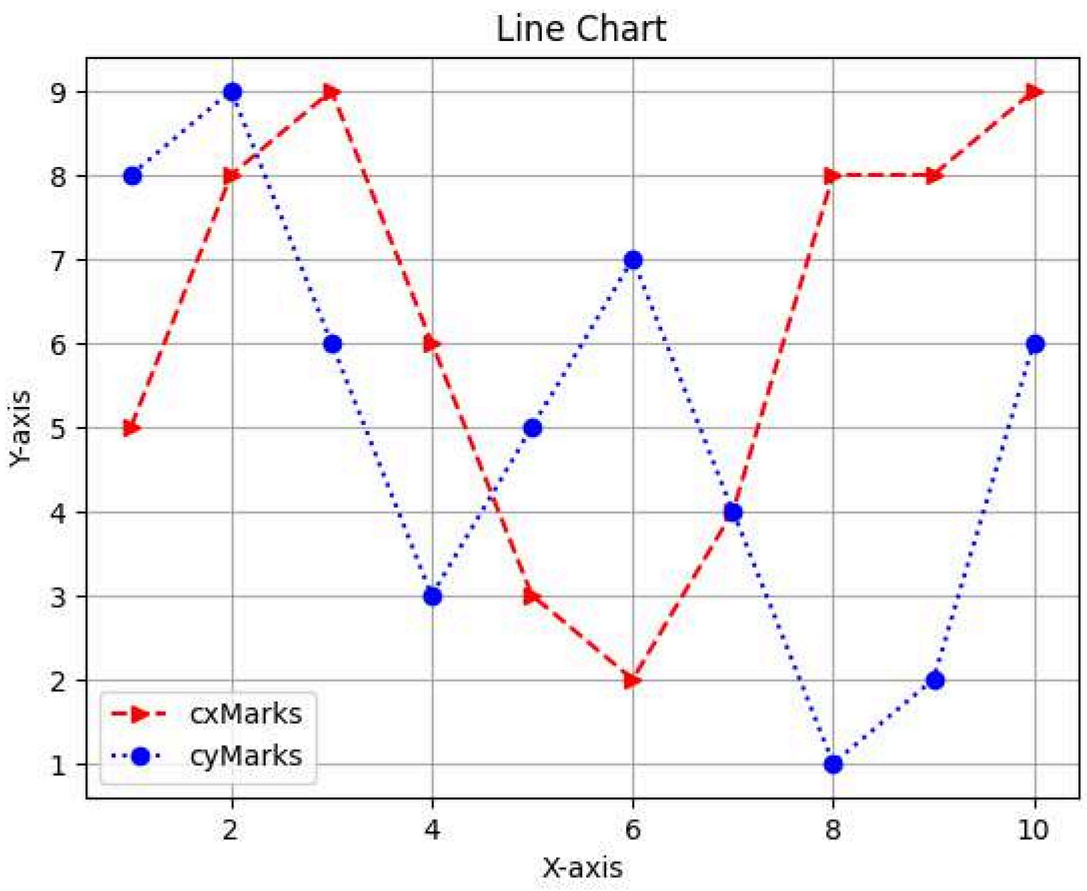
223 | Vishal Baraiya | 23010101014

In [3]:
```
#import matplotlib below
import matplotlib.pyplot as plt
import numpy as np
```

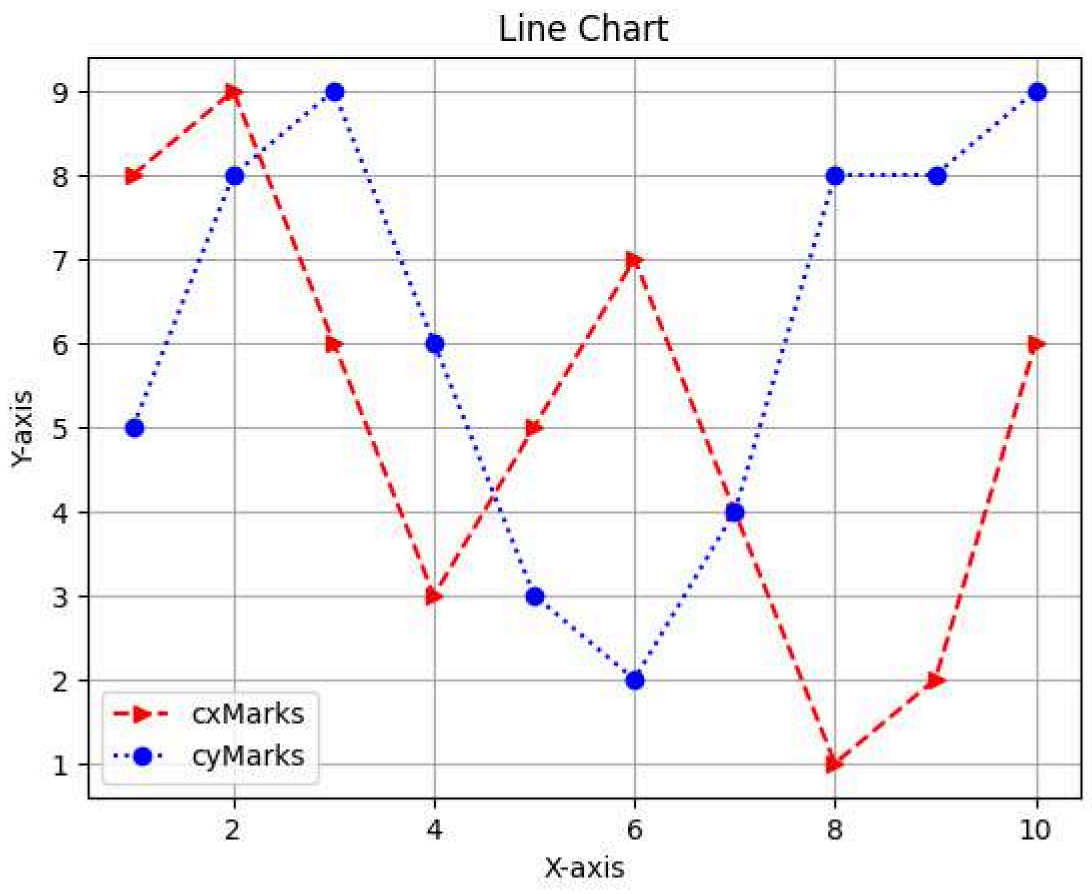
In [58]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
ax = plt.axes()
ax.grid()
# write a code to display the line chart of above x & y
plt.plot(x, y, marker='D', linestyle='-', color='black', label='Data Line')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Line Plot Chart")
plt.legend()
plt.show()
```



In [24]:
```
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]
ax = plt.axes()
ax.grid()
# write a code to display two lines in a line chart (data given above)
plt.plot(x, cxMarks, marker='>', linestyle='--', color='r',label='cxMarks')
plt.plot(x, cyMarks, marker='o', linestyle=':', color='b',label='cyMarks')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Line Chart")
plt.legend()
plt.show()
```

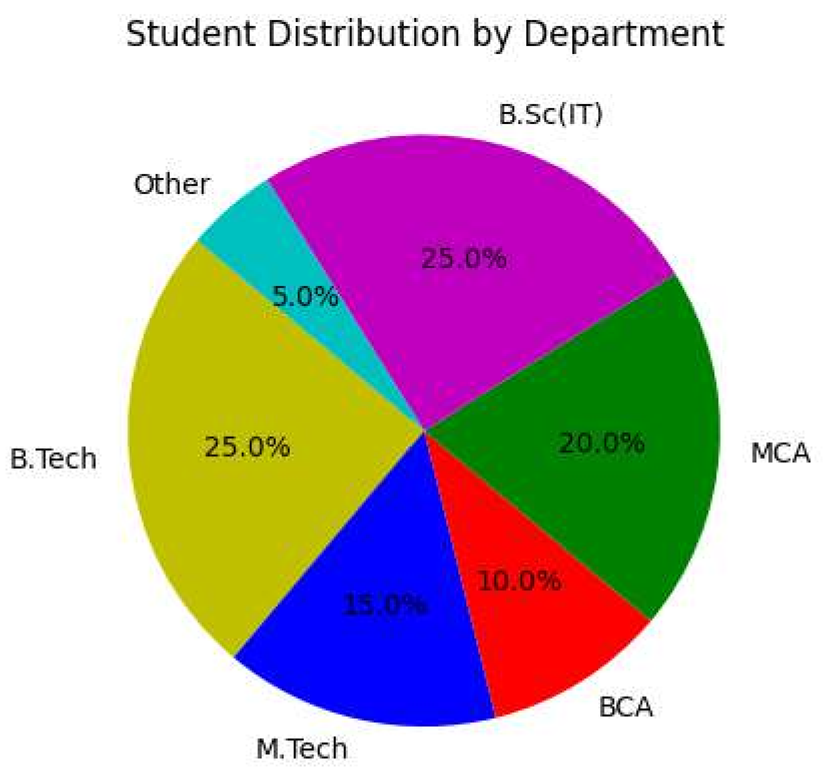


In [23]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

ax = plt.axes()
ax.grid()
# write a code to generate below graph
# write a code to display two lines in a line chart (data given above)
plt.plot(x, cxMarks, marker='>', linestyle='--', color='r',label='cxMarks')
plt.plot(x, cyMarks, marker='o', linestyle=':', color='b',label='cyMarks')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Line Chart")
plt.legend()
plt.show()
```



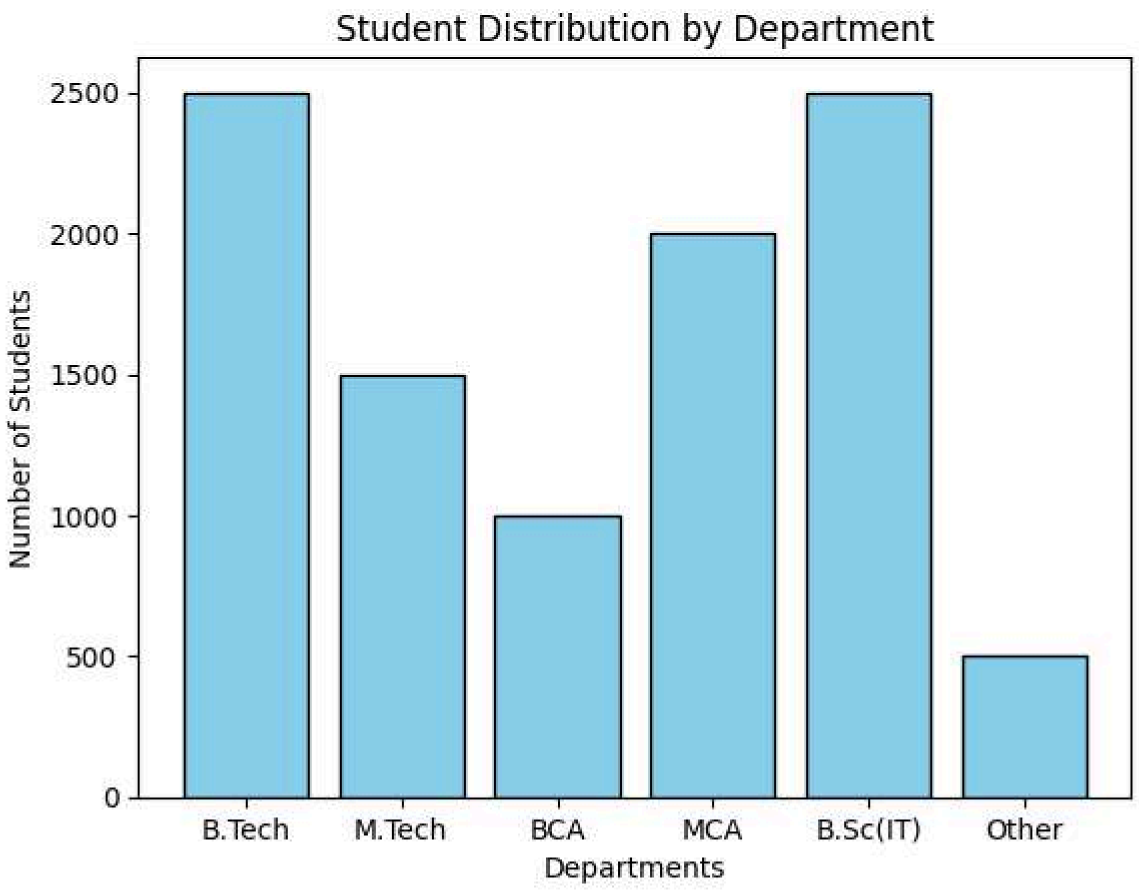
04) WAP to demonstrate the use of Pie chart.

In [7]:
```
dept = ["B.Tech", "M.Tech", "BCA", "MCA", "B.Sc(IT)", "Other"]
x = [2500, 1500, 1000, 2000, 2500, 500]
c = ['y', 'b', 'r', 'g', 'm', 'c']

plt.pie(x, labels=dept, colors=c, autopct='%1.1f%%', startangle=140)
plt.title("Student Distribution by Department")
plt.show()
```



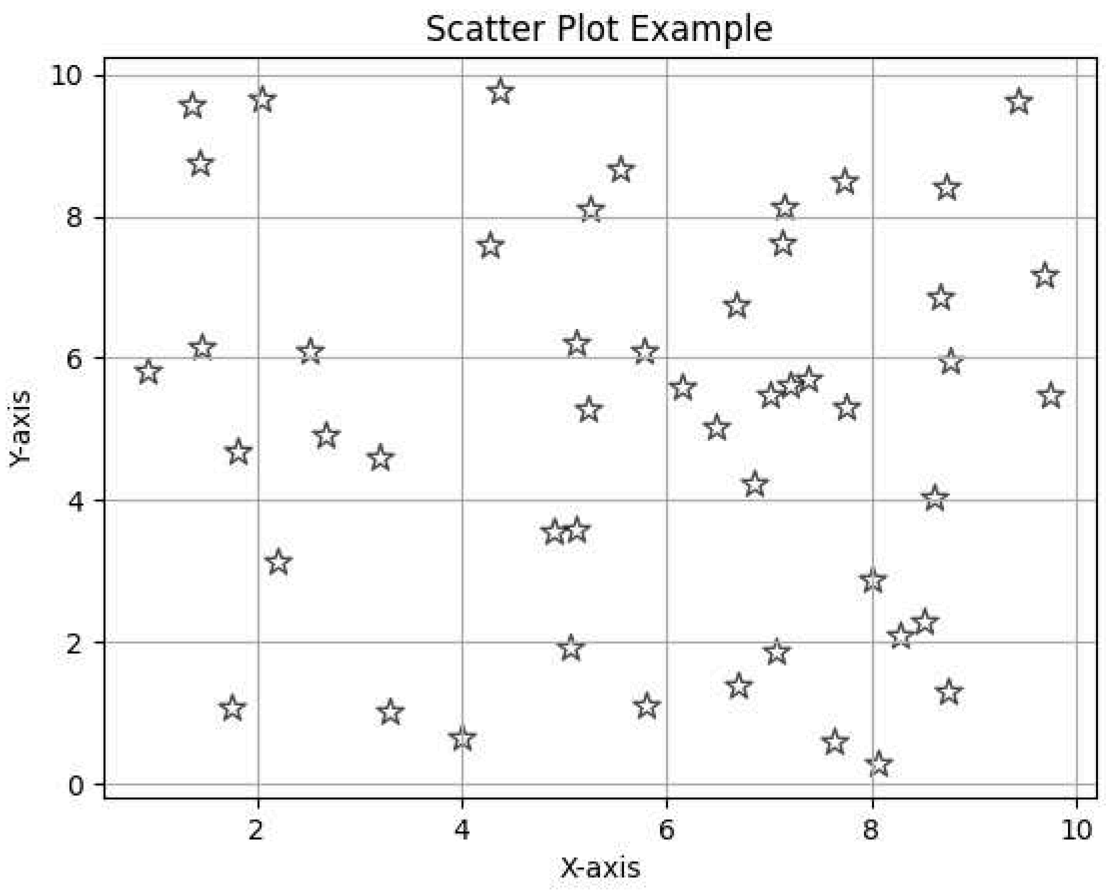
05) WAP to demonstrate the use of Bar chart.

In [10]:
```python
# Data for the bar chart
dept = ["B.Tech", "M.Tech", "BCA", "MCA", "B.Sc(IT)", "Other"]
students = [2500, 1500, 1000, 2000, 2500, 500]

# Creating the bar chart
plt.bar(dept, students, color='skyblue', edgecolor='black')

# Adding labels and title
plt.xlabel("Departments")
plt.ylabel("Number of Students")
plt.title("Student Distribution by Department")

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



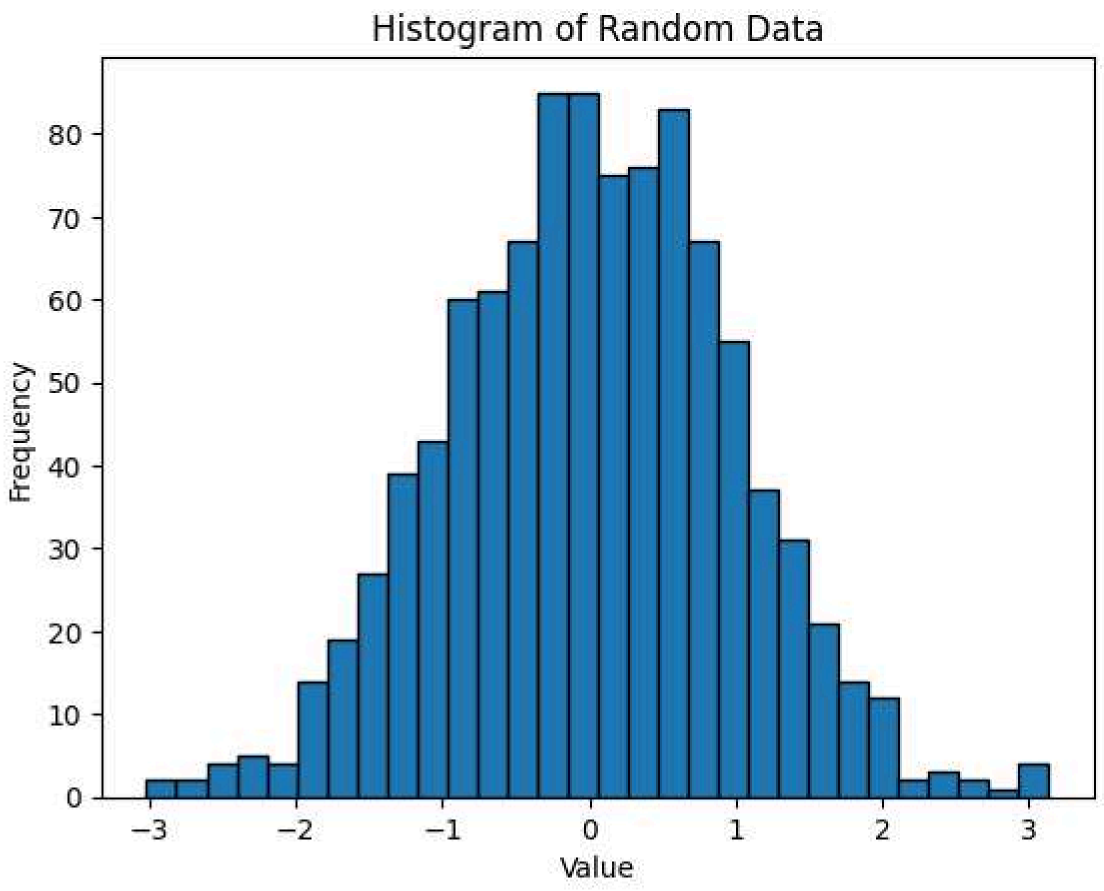
06) WAP to demonstrate the use of Scatter Plot.

In [11]:
```
# Generate random data
no_stars = 50
x = np.random.rand(no_stars) * 10  # Random values between 0 and 10
y = np.random.rand(no_stars) * 10  # Random values between 0 and 10
ax = plt.axes()
ax.grid()
# Create a scatter plot
plt.scatter(x, y, color='white', marker='*', edgecolors='black', alpha=0.7,s=10

# Labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot Example')

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



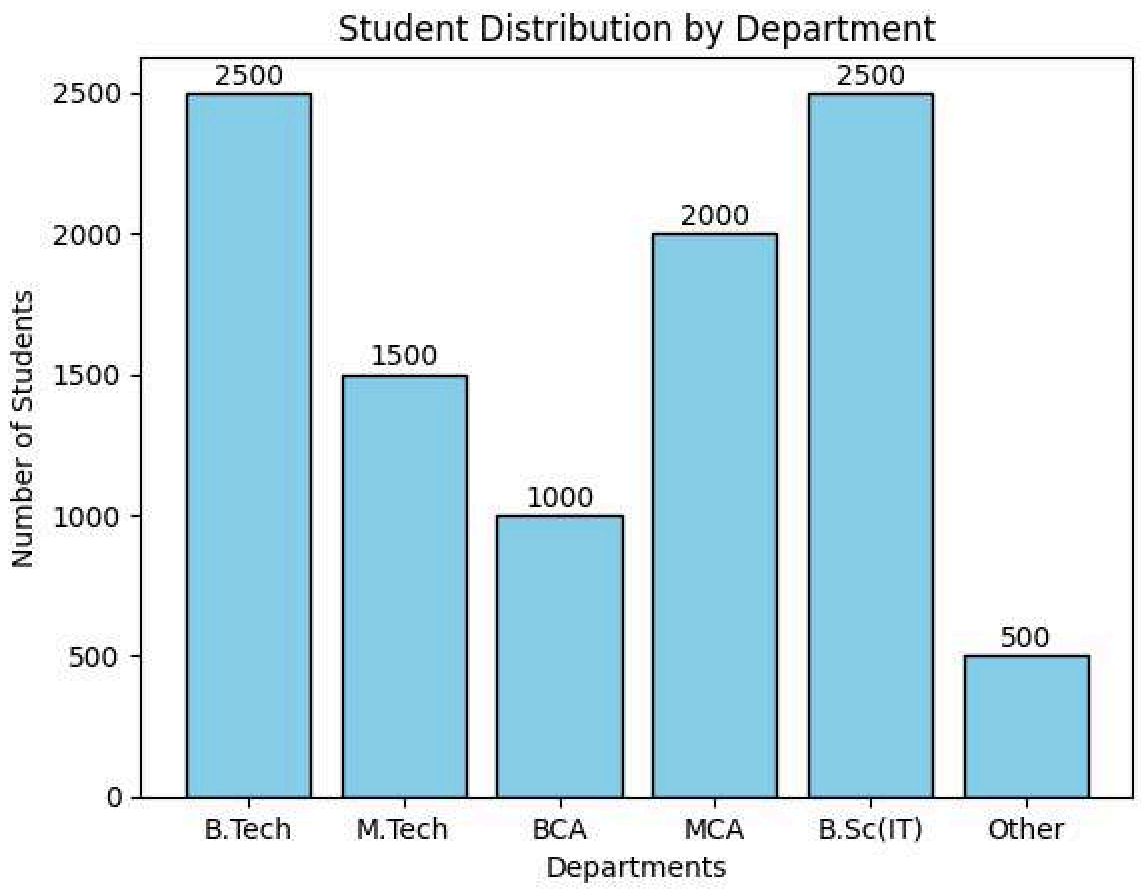
07) WAP to demonstrate the use of Histogram.

In [61]:
```
import matplotlib.pyplot as plt

# Generating random data
data = np.random.randn(1000)  # 1000 random numbers from a normal distribution

# Creating the histogram
plt.hist(data, bins=30, edgecolor='black')

# Adding titles and labels
plt.title('Histogram of Random Data')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Displaying the plot
plt.show()
```



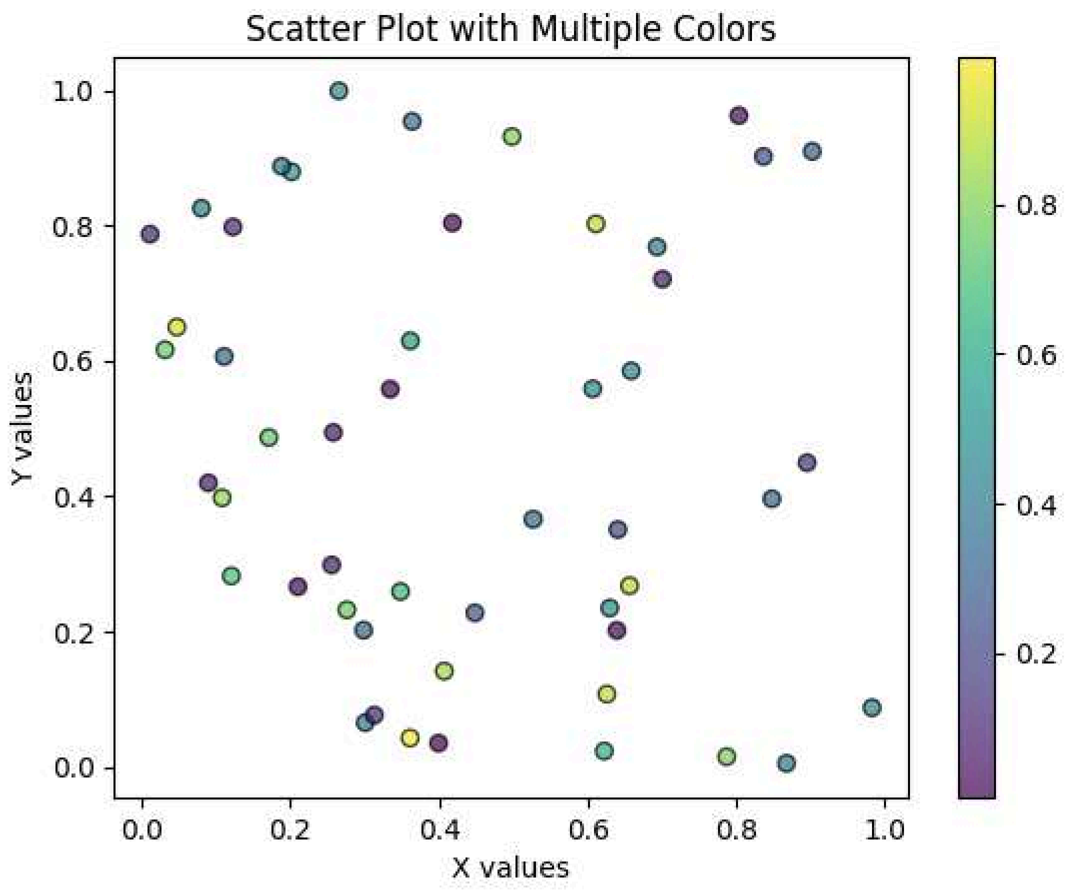
08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [12]:
```
# Data for the bar chart
dept = ["B.Tech", "M.Tech", "BCA", "MCA", "B.Sc(IT)", "Other"]
students = [2500, 1500, 1000, 2000, 2500, 500]

# Creating the bar chart
plt.bar(dept, students, color='skyblue', edgecolor='black')

# Adding labels and title
plt.xlabel("Departments")
plt.ylabel("Number of Students")
plt.title("Student Distribution by Department")

# Adding the value of each bar on top
for i, student_count in enumerate(students):
    plt.text(i, student_count + 10, str(student_count), ha='center', va='bottom'

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



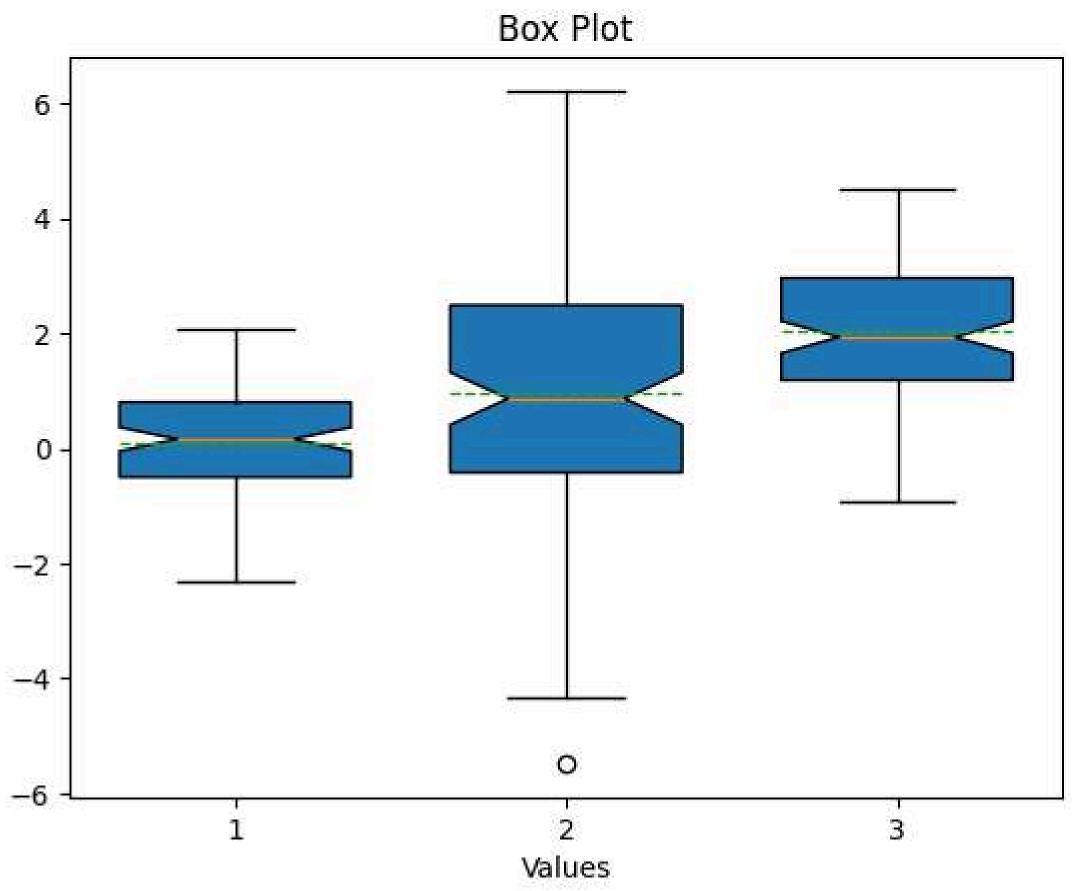
09) WAP create a Scatter Plot with several colors in Matplotlib?

In [16]:
```python
# Data for scatter plot
x = np.random.rand(50)  # Random x values
y = np.random.rand(50)  # Random y values
colors = np.random.rand(50)  # Random colors for each point

# Create the scatter plot
plt.scatter(x, y, c=colors, cmap='viridis', edgecolor='black', alpha=0.7)

# Adding labels and title
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Scatter Plot with Multiple Colors")

# Display the plot
plt.colorbar()  # Show color bar to indicate the color scale
plt.show()
```



10) WAP to create a Box Plot.

In [20]:
```
# Data for the box plot
data = [np.random.normal(0, 1, 100),  # Group 1
        np.random.normal(1, 2, 100),  # Group 2
        np.random.normal(2, 1, 100)]  # Group 3

# Creating the box plot
plt.boxplot(data, patch_artist=True, notch=True, vert=0, widths=0.7, meanline=Tr

# Adding labels and title
plt.xlabel("Values")
plt.title("Box Plot")

# Display the plot
plt.show()
```


Box Plot
6
4
2
0
−2
−4
−6
1
2
3
Values

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

223 | Vishal Baraiya | 23010101014

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 [4]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

s1 = Students("Karan Aahir",55,'F')
print(f" name = {s1.name} \n Age = {s1.age} \n Grade = {s1.grade}")
```

```
name = Karan Aahir
Age = 55
Grade = F
```

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 [8]:
```
class Bank_Account:
    Account_No = 0
    User_Name = ""
    Email = ""
    Account_Type = ""
    Account_Balance = 0.0

    def GetAccountDetails(self):
        self.Account_No = int(input("Enter the Account Number : "))
        self.User_Name = input("Enter the User Name : ")
        self.Email = input("Enter the Email : ")
        self.Account_Type = input("Enter the Account Type : ")
        self.Account_Balance = float(input("Enter the Account Blance : "))

    def DisplayAccountDetails(self):
        print(f'Account Number : {self.Account_No}')
        print(f'User Name : {self.User_Name}')
        print(f'Email : {self.Email}')
        print(f'Account Type : {self.Account_Type}')
        print(f'Account Balance : {self.Account_Balance}')

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

```
Account Number : 123456789
User Name : Karan
Email : karan1234@gmail.com
Account Type : demate
Account Balance : 150.0
```

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

In [15]:

```
class Circle:

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

    def findPerimeter(self):
        return 2 * math.pi * self.r

    def findArea(self):
        return math.pi * self.r * self.r

c = Circle(14)
print(f"Area = {c.findArea()}")
print(f"Perimeter = {c.findPerimeter()}")
```

```
Area = 615.7521601035994
Perimeter = 87.96459430051421
```

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

In [16]:

```
class Employees:

    def __init__(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
    def UpdateEmployeeDetails(self):
        self.name = input("Enter the Name : ")
        self.age = int(input("Enter the Age : "))
        self.salary = int(input("Enter the Salary : "))

    def DisplayEmplooyeeDetails(self):
        print(f"Name = {self.name}")
        print(f"age = {self.age}")
        print(f"Salry = {self.salary}")

e = Employees('Karan Aahir',18,120000)
e.UpdateEmployeeDetails()
e.DisplayEmplooyeeDetails()
```

```
Name = Rishil
age = 12
Salry = 7800
```

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

In [21]:

```
class Bank_Account:

    def __init__(self,accountno,name,balance):
        self.Account_No = accountno
        self.Name = name
        self.Balance = balance

    def deposit(self,Amount):
        self.Balance += Amount
        print(f"Hello! {self.Name} your {Amount} is Suuccessfully Deposited.")

    def withdraw(self,Amount):
        if (Amount <= self.Balance) and (Amount >= 0):
            self.Balance -= Amount
            print(f"Hello! {self.Name}, your {Amount} is successfully withdrawn.
        else:
            print(f"Insufficient Balance!")
            print(f"Your Withdraw Ammount = {Amount} and your Current Balance =

    def display(self):
        print(f"Account Number = {self.Account_No}")
        print(f"Name = {self.Name}")
        print(f"Current Balance = {self.Balance}")

b = Bank_Account(102321,'Karan Aahir',200)
b.display()
b.deposit(50)
b.display()
b.withdraw(100)
b.display()
```

```
Account Number = 102321
Name = Karan Aahir
Current Balance = 200
Hello! Karan Aahir your 50 is Suuccessfully Deposited.
Account Number = 102321
Name = Karan Aahir
Current Balance = 250
Insufficient Balance!
Your Withdraw Ammount = 1000 and your Current Balance = 250.
Account Number = 102321
Name = Karan Aahir
Current Balance = 250
```

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 [1]:
```
class Inventory:
    def __init__(self):
        self.inventory = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.inventory:
            print(f"Item '{item_name}' already exists.")
        else:
            self.inventory[item_name] = {'price': price, 'quantity': quantity}
            print(f"Item '{item_name}' added to inventory.")

    def remove_item(self, item_name):
        if item_name in self.inventory:
            del self.inventory[item_name]
            print(f"Item '{item_name}' removed from inventory.")
        else:
            print(f"Item '{item_name}' not found in inventory.")

    def update_item(self, item_name, price=None, quantity=None):
        if item_name in self.inventory:
            self.inventory[item_name]['price'] = price
            self.inventory[item_name]['quantity'] = quantity
            print(f"Item '{item_name}' updated.")
        else:
            print(f"Item '{item_name}' not found in inventory.")

    def view_inventory(self):
        if not self.inventory:
            print("Inventory is empty.")
        else:
            for item_name, details in self.inventory.items():
                print(f"{item_name} - ${details['price']} x {details['quantity']

# Example usage:
i = Inventory()

# Add items
i.add_item("Laptop", 1200, 10)
i.add_item("Smartphone", 800, 20)

# View current inventory
i.view_inventory()
```

```
# Update item
i.update_item("Laptop", quantity=15)

# Remove item
i.remove_item("Smartphone")

# View updated inventory
i.view_inventory()
```

```
Item 'Laptop' added to inventory.
Item 'Smartphone' added to inventory.
Laptop - $1200 x 10 in stock
Smartphone - $800 x 20 in stock
Item 'Laptop' updated.
Item 'Smartphone' removed from inventory.
Laptop - $1200 x 15 in stock
```

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

In [5]:
```
class Student:
    def __init__(self, Name, Age, StudentID, Grade):
        self.Name = Name
        self.Age = Age
        self.StudentID = StudentID
        self.Grade = Grade

    def display(self):
        return f"Student ID: {self.StudentID}, Name: {self.Name}, Age: {self.Age

s = Student("Karan", 20, "101", "A")
print(s.display())
```

```
Student ID: 101, Name: Karan, Age: 20, Grade: A
```

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 [4]:
```
class StudentKit:
    PrincipalName = "Mr. ABC"

    def __init__(self, name):
        self.StudentName = name
        self.AttendanceDays = 0

    def Attendance(self, days):
        self.AttendanceDays = days

    # Method to generate a certificate
    def GetCertificate(self):
        print(f"Certificate of Attendance")
        print(f"This is to certify that {self.StudentName}")
        print(f"has attended {self.AttendanceDays} days of class.")
        print(f"Principal: {StudentKit.PrincipalName}")

name = input("Enter the student's name: ")
student = StudentKit(name)

days = int(input("Enter the number of days present in class: "))
student.Attendance(days)
student.GetCertificate()
```

```
Certificate of Attendance
This is to certify that Vishal
has attended 12 days of class.
Principal: Mr. ABC
```

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

In [7]:
```
class Time :
    hour = 0
    minute = 0

    def __init__(self,h,m):
        self.hour = h
        self.minute = m

    def addTime(self,t1,t2):
        self.hour = t1.hour + t2.hour
        self.minute = t1.minute + t2.minute

        if (self.minute >= 60):
            self.hour += self.minute // 60
            self.minute = self.minute % 60

    def display(self):
        print(f"Hour : {self.hour} Minute: {self.minute}")

t1 = Time(12,34)
t2 = Time(3,45)
t3 = Time(0,0)

t3.addTime(t1,t2)
t1.display()
t2.display()
t3.display()
```

```
Hour : 12 Minute: 34
Hour : 3 Minute: 45
Hour : 16 Minute: 19
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

223 | Vishal Baraiya | 23010101014

Continued..

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

In [4]:
```python
class Rectangle:
    l = 0
    b = 0

    def __init__(self,l,b):
        self.l = l
        self.b = b


def findArea(r : Rectangle):
    return r.l * r.b

r = Rectangle(10,20)
print(f"Area = {findArea(r)}")
```

```
Area = 200
```

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]:
```python
class square:
    l = 0
    a = 0

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

    def area(self):
        self.a = self.l * self.l
        self.output()

    def output(self):
        print(f"Area : {self.a}")

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

```
Area : 100
```

12) Calculate the area of a rectangle.

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

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

In [14]:
```
class Rectangle:
    length = 0
    breath = 0

    def __init__(self,length,breath):
        self.length = length
        self.breath = breath

    @classmethod
    def compare(cls,l,b):
        if l==b:
            print("THIS IS SQAURE.")
            return None
        else:
            return Rectangle(l,b)

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

    def output(self,ans):
        print(f"Area : {ans}")

r1=Rectangle.compare(9,8)
if r1:
    r1.area()
```

```
Area : 72
```

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 [15]:
```
class squre:
    _side = 0

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

    def get_side(self):
        return self._side

s = squre()
s.set_side(12)
print(s.get_side())
```

```
12
```

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 [4]:
```
class Profit:

    def getProfit(self):
        self.profit = int(input("Enter Your Profit : "))

class Loss:

    def getLoss(self):
        self.loss = int(input("Enter Your Loss : "))

class BalanceSheet(Profit,Loss):

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

    def printBalance(self):
        print(f"Balance = {self.balance}")

a = BalanceSheet()

a.getProfit()
a.getLoss()
a.getBalance()
a.printBalance()
```

```
Balance = 500
```

15) WAP to demonstrate all types of inheritance.

In [3]:

```
# 1. Single Inheritance
class Animal:
    def sound(self):
        print("Animal makes sound")

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

# 2. Multiple Inheritance
class Father:
    def father_name(self):
        print("Father's name is John")

class Mother:
    def mother_name(self):
        print("Mother's name is Mary")

class Child(Father, Mother):
    def child_name(self):
        print("Child's name is Sam")

# 3. Multilevel Inheritance
class Grandparent:
    def grandparent_name(self):
        print("Grandparent's name is George")

class Parent(Grandparent):
    def parent_name(self):
        print("Parent's name is Henry")

class ChildOfParent(Parent):
    def child_name(self):
        print("Child's name is Lisa")

# 4. Hierarchical Inheritance
class Vehicle:
    def type_of_vehicle(self):
        print("This is a vehicle")

class Car(Vehicle):
    def car_type(self):
        print("This is a car")

class Bike(Vehicle):
    def bike_type(self):
        print("This is a bike")

# 5. Hybrid Inheritance (Combination of Multiple and Multilevel)
class A:
    def method_a(self):
        print("Method from class A")

class B(A):
    def method_b(self):
        print("Method from class B")

class C:
    def method_c(self):
        print("Method from class C")

class D(B, C):
    def method_d(self):
        print("Method from class D")


# Demonstrating Single Inheritance
print("Single Inheritance:")
dog = Dog()
dog.sound()  # Inherited method
dog.bark()   # Class method

# Demonstrating Multiple Inheritance
print("\nMultiple Inheritance:")
child = Child()
child.father_name()  # Method from Father class
child.mother_name()  # Method from Mother class
child.child_name()   # Method from Child class

# Demonstrating Multilevel Inheritance
print("\nMultilevel Inheritance:")
child_of_parent = ChildOfParent()
child_of_parent.grandparent_name()  # Inherited method from Grandparent
child_of_parent.parent_name()       # Inherited method from Parent
child_of_parent.child_name()        # Method from ChildOfParent

# Demonstrating Hierarchical Inheritance
print("\nHierarchical Inheritance:")
car = Car()
car.type_of_vehicle()  # Inherited from Vehicle
car.car_type()         # Method from Car class

bike = Bike()
bike.type_of_vehicle()  # Inherited from Vehicle
bike.bike_type()        # Method from Bike class

# Demonstrating Hybrid Inheritance
print("\nHybrid Inheritance:")
d = D()
d.method_a()  # Inherited from A
d.method_b()  # Inherited from B
d.method_c()  # Inherited from C
d.method_d()  # Method from D class
```

```
Single Inheritance:
Animal makes sound
Dog barks

Multiple Inheritance:
Father's name is John
Mother's name is Mary
Child's name is Sam

Multilevel Inheritance:
Grandparent's name is George
Parent's name is Henry
Child's name is Lisa

Hierarchical Inheritance:
This is a vehicle
This is a car
This is a vehicle
This is a bike

Hybrid Inheritance:
Method from class A
Method from class B
Method from class C
Method from class D
```

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 [19]:
```
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("Karan", 30, 60000)
emp.display()
```

```
Name: Karan, Age: 30
Salary: 60000
```

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 [20]:

```python
from abc import ABC, abstractmethod

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

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

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

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

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

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

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

In [ ]: