Darshan UNIVERSITY

योगः कर्मसु कौशलम्

# Python Programming - 2301CS404

## Lab - 1

**Name: Solanki Harshil**

**Enrollment :- 23010101256**

**Roll no :125**

## 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 [5]:
```
 # without using input()
a, b= 4, 5

c = a + b
print("The adittion of a and b is : ", c)

# with input()
number1 = int(input("Enter the number : "))
number2 = int(input("Enter the number : "))

number3 = number1 + number2
print("The adittion of two number is : ", number3)
```

```
The adittion of a and b is :  24
The adittion of two number is :  3
```

## 03) WAP to check the type of the variable.

In [17]:
```
num = input("Enter something : ")
```

```
print(type(num))
```

```
<class 'str'>
```

## 04) WAP to calculate simple interest.

In [19]:
```
P = int(input("Enter the Principal :"))
R = int(input("Enter the Rate of intrest :"))
T = int(input("Enter the Time period : "))

simpleIntrest = (P * R * T) / 100
print("the simple intrest is :" , simpleIntrest)
```

```
the simple intrest is : 100000000.0
```

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

In [21]:
```
import math

rad = float(input("Enter the radius of the circle "))

print(f' the perimeter is : {math.pi * 2 * rad}')

print(f'the area is : {math.pi * pow(rad, 2)}')
```

```
 the perimeter is : 12.566370614359172
the area is : 12.566370614359172
```

## 06) WAP to calculate area of a triangle.

In [23]:
```
base = int(input("Enter the base of triangle :"))
height = int(input("Enter the height of the triangle :"))

print(f' The are of the  Triangle is : { (1/2) * (base * height) }')
```

```
The are of the  Triangle is : 6.0
```

## 07) WAP to compute quotient and remainder.

In [27]:
```
num1 = int(input(" Enter the first number :"))
num2 = int(input(" Enter the second number :"))

print(f' the quotient is : { num1 / num2}')
print(f' the remainder is : { num1 % num2}')
```

```
the quotient is : 0.5
the remainder is : 5
```

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

In [33]:
```
info = input("write what you want to convert F for fahrenheit to celsius and vice v

if(info == "F"):
```

```
    fahrenheit = float(input(" Enter the temperature in fahrenheit"))
    print(f' the temperature in celsius is : { (fahrenheit - 32) * (5 / 9) }')

else:
    celsius = float(input(" Enter the temperature in celsius"))
    print(f' the temperature in fahrenheit is : {  (9 / 5) * fahrenheit + 32  }')
```

```
the temperature in fahrenheit is : 50.0
```

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

In [35]:
```
x1 = int(input("Enter the x1 cordinate :"))
x2 = int(input("Enter the x2 cordinate :"))
y1 = int(input("Enter the y1 cordinate :"))
y2 = int(input("Enter the y2 cordinate :"))

print(f' the distance between two points in 2 - D space is : { (x2 - x1) ** 2 + (y2
```

```
the distance between two points in 2 - D space is : 0.0
```

## 10) WAP to print sum of n natural numbers.

In [39]:
```
sumOfN = int(input(" Enter the number :"))

print(f' the sum of numbers is { (sumOfN * (sumOfN/2) ) + (sumOfN/2)}')
```

```
the sum of numbers is 66.0
```

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

In [5]:
```
n = int(input("Enter the value of n: "))

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

print(f'the sum of square of first {n} number is : {res}')
```

```
the sum of square of first 2 number is : 5
```

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

In [9]:
```
first = input("Enter your first name :")
last = input("Enter your last name :")
name = first + " "+last

print(f"Your name is {name}")
```

```
Your name is viraj  odedra
```

## 13) WAP to swap two numbers.

In [11]:
```
a1 = int(input("Enter the first number :"))
a2 = int(input("Enter the first number :"))
print(f' before swap the value of a1 and a2 are {a1}, {a2}')
```

```
a1,a2 = a2,a1
print(f' after swapping the value of a1 and a2 are {a1}, {a2}')
```

```
before swap the value of a1 and a2 are 4, 40
after swapping the value of a1 and a2 are 40, 4
```

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

In [15]:
```
dis = int(input('Enter the distance in Kilometer'))

meter = dis * 1000
feet = dis * 3280.84
inches = dis * 39370.08
centimeter = dis * 100000

print(f"Distance in meters: {meter}" )
print(f"Distance in feet: {feet}" )
print(f"Distance in inches: {inches}" )
print(f"Distance in centimeters: {centimeter}" )
```

```
Distance in meters: 1000
Distance in feet: 3280.84
Distance in inches: 39370.08
Distance in centimeters: 100000
```

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

In [31]:
```
day = input("Enter the day: ")
month = input("Enter the month: ")
year = input("Enter the year: ")

# print("Today's date is:" + day, month, year, sep="-")
print(f"Today's Date is {day}-{month}-{year}. ")
```

```
Today's Date is 1-12-2024.
```

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

# Python Programming - 301CC4404

# Lab - 2

## if..else..

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

In [5]:
```
a = int(input("Enter the number : "))

if( a > 0) :
    print("The number you entered is Positive !")
else :
    print("The number you entered is Negative !")
```

```
The number you entered is Negative !
```

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

In [7]:
```
a = int(input("Enter the number : "))

if( a % 2 == 0) :
    print("The number you entered is EVEN !")
else :
    print("The number you entered is ODD !")
```

```
The number you entered is EVEN !
```

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

In [20]:
```
a = int(input("Enter the number : "))
b = int(input("Enter the number : "))

if( a > b) :
    print(f' {a} is largest number !')
else :
    print(f' {b} is largest number !')

c = a if(a > b) else b
print(f'The largest number is {c}')
```

```
 5 is largest number !
The largest number is 5
```

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

In [24]:
```
a = int(input("Enter the number : "))
b = int(input("Enter the number : "))
c = int(input("Enter the number : "))

if( a > b and a > c) :
    print(f'The largest number is : {a} !')

elif( b > c and b > a) :
    print(f'The largest number is : {b} !')

else :
    print(f'The largest number is : {c} !')
```

```
The largest number is : 3
```

## 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 [30]:
```
year = int(input("Enter the Year to check wheather it is Leap or not !"))

if( year % 4 == 0 and (not year % 100 == 0)) or (year % 400 == 0):
    print(f"The entered year {year} is Leap year")

else :
    print(f"The entered year {year} is Leap year")
```

```
The entered year 2024 is Leap year
```

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

In [36]:
```
day = int(input("Enter the number less then 7 :"))

match day :
```

```
    case 1:
        print("Monday")
    case 2:
        print("Tuesday")
    case 3:
        print("Wednesday")
    case 4:
        print("Thrusday")
    case 5:
        print("Friday")
    case 6:
        print("Saturday")
    case 7:
        print("Sunday")
    case _:
        print("Enter the valid input ")
```

```
Enter the valid input
```

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

In [44]:
```
op = input("Enter the operation !!")
num1 = float(input("Enter the first number :"))
num2 = float(input("Enter the first number :"))

match op :
    case '+':
        print(f'{num1} + {num2} = {num1 + num2}')
    case '-':
        print(f'{num1} - {num2} = {num1 - num2}')
    case '*':
        print(f'{num1} * {num2} = {num1 * num2}')
    case '/':
        print(f'{num1} / {num2} = {num1 / num2}')
```

```
4.2 / 2.3 = 1.8260869565217392
```

## 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 [53]:
```
sub1 = int(input("Enter the marks of subject 1 :"))
sub2 = int(input("Enter the marks of subject 2 :"))
sub3 = int(input("Enter the marks of subject 3 :"))
sub4 = int(input("Enter the marks of subject 4 :"))
sub5 = int(input("Enter the marks of subject 5 :"))

total = ((sub1 + sub2 + sub3 + sub4 + sub5)/500) * 100

if(total < 35) :
```

```
    print(f'You get {total} , You Failed!')
elif(total > 35 and total < 45) :
    print(f'You get {total} , You Pass!')
elif(total > 45 and total < 60) :
    print(f'You get {total} , You passed Second class!')
elif(total > 60 and total < 70) :
    print(f'You get {total} , You passed First class!')
elif(total > 70) :
    print(f'You get {total} , You get Distinction!')
```

```
You get 50.2 , You passed Second class!
```

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

In [69]:
```
side1 = int(input("Enter the First side of Triangle !"))
side2 = int(input("Enter the Second side of Triangle !"))
side3 = int(input("Enter the Third side of Triangle !"))


if(side1 == side2 == side3):
    print("Triangle is Equilator")
elif(side1 == side2 or side2 == side3 or side3 == side1):
    print("Triangle is isosceles")
elif(side1 != side2 and side1 != side3 and side2 != side3):
    if(pow(side1, 2) + pow(side2, 2) == pow(side3,2) or pow(side3, 2) + pow(side2,
       print("Triangle is Right - angled Triangle")
else:
    print("The triangle is scalene")
```

```
Triangle is Right - angled Triangle
```

In [ ]:
```

```

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

In [73]:
```
num1 = float(input("Enter the first number :"))
num2 = float(input("Enter the first number :"))
num3 = float(input("Enter the first number :"))

if((num1 > num2 and num1 < num3) or (num1 < num2 and num1 > num3)):
    print(f'The second largest number is : {num1}')
elif((num2 > num1 and num2 < num3) or (num2 < num1 and num2 > num3)):
    print(f'The second largest number is : {num2}')
else:
    print(f'The second largest number is : {num3}')
```

```
The second largest number is : 444.0
```

## 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 [75]:
```
units = float(input("Enter your monthly units :"))
bill = 0

if(units > 0 and units < 50):
    bill = 2.60 * units
elif(units > 50 and units < 100):
    bill = 3.25 * units
elif(units > 100 and units < 200):
    bill = 5.26 * units
elif(units > 200):
    bill = 8.45 * units

print(f"Your monthly BILL is {bill}")
```

```
Your monthly BILL is 31.200000000000003
```

In [ ]:
```

```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

[redacted] - 3001S4444

# Lab - 3

## for and while loop

### 01) WAP to print 1 to 10.

In [78]:
```
for i in range(11) :
    print(i, end = ",")
```

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

### 02) WAP to print 1 to n.

In [80]:
```
num = int(input("Enter how many numbers you want to print : "))

for temp in range(num + 1):
    print(temp)
```

```
0
```

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

In [14]:
```
num = int(input("Enter the number : "))
print(f"The odd number between o to {num} are :")

for temp in range(num + 1):
    if(temp % 2 != 0):
        print(temp, end = " ")
```

```
The odd number between o to 23 are :
1 3 5 7 9 11 13 15 17 19 21 23
```

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

In [16]:
```
num1 = int(input("Enter the first number :"))
num2 = int(input("Enter the second number :"))

print(f"The number between {num1} and {num2} which are divisible 2 and not divisibl

for num in range(num1, num2 + 1):
    if(num % 2 == 0 and num % 3 != 0):
        print(num, end = " ")
```

```
The number between 3 and 12 which are divisible 2 and not divisible 3 is :
4 8 10
```

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

In [20]:
```
num = int(input("Enter the number :"))
print(f" The sum of 1 to {num} is : ", end = "")

sum = 0
for i in range(num + 1):
    sum += i

print(sum)
```

```
The sum of 1 to 12 is : 78
```

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

In [38]:
```
num = int(input("Enter the number :"))
sum = 0
for i in range(1, num + 1):
    sum += i**2

print(f"The sum of given series is : {sum}")
```

```
The sum of given series is : 5
```

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

In [42]:
```
num = int(input("Enter the number :"))
sum = 0

for i in range(num + 1):
    if i % 2 == 0:
        sum -= i
    else:
        sum += i
print(f'The sum of the series : {sum}')
```

```
The sum of the series : 3
```

## 08) WAP to print multiplication table of given number.

In [46]:
```
num = int(input('Enter the number which multiplication table you want !!'))

for i in range(1, 11):
    print(f'{num} X {i} = {num * i}')
```

```
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
```

## 09) WAP to find factorial of the given number.

In [48]:
```
num = int(input('Enter the number which factorial you want to find :'))
sum = 1
for i in range(2, num + 1):
    sum *= i
print(f"The factorial of the {num} is : {sum}")
```

```
The factorial of the 5 is : 120
```

## 10) WAP to find factors of the given number.

In [54]:
```
num = int(input('Enter the number which factors you want to find :'))
sum = 1
factor_list = []
for i in range(1, num//2 + 1):
    if(num % i == 0):
        factor_list.append(i)
print(f"The factor of the {num} are : {factor_list}")
```

```
The factor of the 50 are : [1, 2, 5, 10, 25]
```



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

In [62]:
```
import math
num = int(input("Enter the number to check whether it is prime or not !!"))

for i in range(2, int(math.sqrt(num))):
    if(num % i == 0):
        print(f"The {num} is NOT PRIME !")
        break;
else:
    print(f" The {num} is PRIME !")
```

```
The 10 is NOT PRIME !
```

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

In [70]:
```
num = int(input("Enter the number to get the sum of it's degits !"))
num1 = num
sum = 0;
while num > 0:
    sum += int(num % 10)
    num /= 10
print(f'The sum of digits of {num1} is : {sum}')
```

```
The sum of digits of 123 is : 6
```

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

In [74]:
```
num = int(input("Enter the number to check whether it is PALINDROME or not !"))
temp = num
res = 0
while num > 0:
    res = num % 10 + (res*10)
    num //= 10
if(res == temp):
    print(f"{temp} is Palindrom number !")
else :
    print(f"{temp} is not Palindrom number !")
```

```
12321 is Palindrom number !
```

## 14) WAP to print GCD of given two numbers.

In [76]:
```
num1 = int(input("Enter the number :"))
num2 = int(input("Enter the number :"))
gcd = 1
minimum = min(num1, num2)
for i in range(minimum, 0, -1):
    if(num1 % i == 0 and num2 % i == 0):
        gcd = i
        break;
print(f'The GCD of {num1} and {num2} is : {gcd}')
```

```
The GCD of 24 and 28 is : 4
```

In [ ]:

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

[redacted] - 2011S4404

# Lab - 4

## String

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

In [7]:
```
str = input("Enter the String :")

if str == str[::-1] :
    print(f"given String {str} is palindrom")
else :
    print(f"Given String {str} is not palindrom")
```

```
given String NayaN is palindrom
```

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

In [21]:
```
str = input('Enter the String to reverse :').split()
str2 = ''

for i in str:
    str2 += i[::-1] + ' '

print(f'The reverse word in given string  is : {str2}')
```

```
The reverse word in given string  is : jariv
```

### 03) WAP to remove ith character from given string.

In [31]:
```
str = input("Enter the string ")
idx = int(input("Enter the position where to delete the character"))

if 0 <= idx < len(str) :
    result = str[: idx] + str[idx + 1 :]
    print(f"Resultant String is : {result}")
else:
    print("Enter valid String or pos of delete character ")
```

```
Resultant String is : vraj
```

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

In [33]:
```
str = input("Enter the String to find it's length")
length = 0

for i in str:
    length += 1
print(f'The length of the string is : {length}')
```

```
The length of the string is 5
```

## 05) WAP to print even length word in string.

In [37]:
```
str = input("Enter the String to find it's length").split()

for word in str:
    if(len(word) % 2 == 0):
        print(word, end = " ")
```

```
hi my name is veeraj
```

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

In [39]:
```
str = input("Enter the String to find it's length").lower()
vowels = 'aeiou'
counter = 0

for ch in str:
    if ch in vowels :
        counter += 1

print(f'The number of vowels in the {str} is : {counter}')
```

```
The number of vowels in the viraj is my name is : 5
```

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

In [49]:
```
str = input("Enter the string :").split()
result = ''

for word in str:
```

```
    result = word 0 .upper() + word 1:len(word) - 1  + word en(word) - 1 .upper()
    print(result, end = " ")
```

```
MY NamE IS NooB
```

## 08) WAP to convert given array to string.

In [57]:
```
str = input("Enter the ARRAY space seperatedly :").split()
str2 = ''

for word in str:
    str2 += word + ' '
print(f"The converted String is : {str2}")
```

```
The converted String is : sYUUUUMMIIIITTTT
```

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

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

In [65]:
```
password = input("Enter the password : ")
confirmPass = input("Enter confirm the password")

if password == confirmPass :
    print('The password saved succesfully!')
elif (password.lower() == confirmPass.lower()):
    print('Only case mistakes are reported!')
else:
    print('Write appropriate password !')
```

```
Only case mistakes are reported!
```

## 10) : Display credit card number.

## card no. : 1234 5678 9012 3456

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

In [99]:
```
cardNo = input("Enter your card number (16 digits) : ").split()
size = len(cardNo)

if(size > 4 or size < 4) :
    print('invalid card number :')
else :
    display = '**** **** ****' + ' ' + cardNo[size - 1]
    print(display)
```

```
4
**** **** **** 2222
```

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

## s1 = decimal and s2 = medical are Anagram

In [89]:
```
str1 = input('Enter the string :')
str2 = input('Enter the string :')

if(len(str1) != len(str2)) :
    print('Not Anagram !')
elif(sorted(str1) == sorted(str2)) :
    print(f'Both {str1} and {str2} are Anagram')
else :
    print(f'Both {str1} and {str2} are NOT Anagram')
```

```
Both viraj and jariv are Anagram
```

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

## input : EHlsarwiwhtwMV

## output : lsarwiwhtwEHMV

In [ ]:
```
str = input("Enter the String :")
is_upper = ''
is_lower = ''

for ch in str :
    if ch.islower() :
        is_upper += ch
    else :
        is_lower += ch

print(is_upper + is_lower)
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

# Python Programming - 2011S4404

# Lab - 5

## List

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

In [1]:
```
l1 = [1, 2, 3, 4, 5]
sum = 0
for i in l1:
    sum += i
print('Sum of all elements in the list is', sum)
```

```
Sum of all elements in the list is 15
```

### 02) WAP to find largest element in a List.

In [2]:
```
l1 = [10, 34, 100, 4, 46, 55, 90]
max = l1[0]
for i in l1:
    if i > max:
        max = i
print('Largest element is', max)
```

```
Largest element is 100
```

### 03) WAP to find the length of a List.

In [5]:
```
l1 = [1, 2, 3, 4, 5, 6]
# len(l1)
length = 0
for i in l1:
    length = length + 1
print('Length of the list is', length)
```

```
Length of the list is 6
```

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

In [11]:
```
a = int(input('Enter how many elements you wan to  add :'))
l1 = []

for i in range(a) :
    l1.append(int(input(f'Enter the {a + 1} element :')))

# temp = l1[0]
# l1[0] = l1[a - 1]
# l1[a - 1] = temp

l1[0], l1[-1] = l1[-1], l1[0]

print(l1)
```

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

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

In [27]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

idx = int(len(li2)/2)
li3 = li2[idx:] + li2[:idx]

print(li3)
```

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

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

In [29]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

idx1 = int(input("Enter the first index to interchange :"))
idx2 = int(input("Enter the second index to interchange :"))

li2[idx1], li2[idx2] = li2[idx2], li2[idx1]

print(li2)
```

```
[1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

## 07) WAP to reverse the list entered by user.

In [31]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

li3 = li2[::-1]

print(li3)
```

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

## 08) WAP to print even numbers in a list.

In [33]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

li3 = [i for i in li2 if (i % 2 == 0)]
print(li3)
```

```
[2, 4, 6, 8]
```

## 09) WAP to count unique items in a list.

In [45]:
```
def uniqueCount(li) :
    unique = set(li)
    return len(unique)

li = input("Enter List :").split()
li2 = [int(i) for i in li]

uniqueCounter = uniqueCount(li2)
print(f'The Counter of the UNIQUE Elements are : {uniqueCounter}')
```

```
The Counter of the UNIQUE Elements are : 4
```

## 10) WAP to copy a list.

In [49]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

li3 = li2.copy()
print(li3)
```

```
[1, 1, 1, 1, 1, 1]
```

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

In [51]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

counterRange = int(input('Enter the Range :'))
```

```
li3 = [i for i in range(0, counterRange) if (i % 2 != 0)]
print(li3)
```

```
[1, 3]
```

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

In [65]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]
d1 = {}

for i in li2 :
    if i not in d1 :
        d1[i] = 1
    else :
        d1[i] += 1

print(d1)
```

```
{1: 2, 2: 2, 3: 3}
```

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

In [75]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

s1 = set(li2)
li3 = list(s1)
li3.sort()

print(li3[len(li3) - 2])
```

```
456
```

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

In [91]:
```
li = input("Enter List :").split()
li2 = [int(i) for i in li]
k = int(input('Enter the frequency :'))
d1 = {}

for i in li2 :
    if i not in d1 :
        d1[i] = 1
    else :
        d1[i] += 1

li3 = []

for i in d1:
    if d1[i] < k :
            li3.append(i)
```

```
print(li3)
```

```
[2]
```

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

In [105…

```
li2 = []

for i in range(10):
    li2.append(i**2)
print(li2)

# OR

li3 = [i**2 for i in range(10)]
print(li3)
```

```
[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 [108…

```
fruits = input('Enter the list of fruits :').split()

fruitStartsB = [fruit.strip() for fruit in fruits if fruit.lower().startswith('b')]

print(fruitStartsB)
```

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

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

In [ ]:

```
li = input("Enter List :").split()
li2 = [int(i) for i in li]

li1 = input("Enter List :").split()
li3 = [int(i) for i in li]

s1 = set(li2)
s2 = set(li3)


li4 = (s1.intersection(s2))
print(li4)
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

# Python Programming - 301CC4404

# Lab - 6

## Tuple

### 01) WAP to find sum of tuple elements.

In [ ]:
```
tuple1 = (10, 20, 30, 40, 50)

tupleSum = sum(tuple1)
print("The sum of tuple elements is:", tupleSum)
```

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

In [5]:
```
tup = tuple(map(int, input("Enter elements separated by space: ").split()))

n = int(input("Enter the value of K: "))

if n > len(tup):
    print("K should not be greater than the number of elements in the tuple.")
else:

    tup2 = sorted(tup)

    minElements = tup2[:n]
    maxElements = tup2[-n:]

    print(f"The {n} minimum elementn are: {minElements}")
    print(f"The {n} maximum elementn are: {maxElements}")
```

```
The 2 minimum elementn are: [1, 2]
The 2 maximum elementn are: [8, 9]
```

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

In [15]:
```python
l1 = [(2, 4, 6), (3, 6, 12), (4, 8, 16), (5, 12, 19)]
k = int(input("Enter the value of K: "))
result = []
for tup in  l1:
    isDiv = True
    for ele in tup :
        if ele % k != 0:
            isDiv = False
            break
    if isDiv:
        result.append(tup)

print("Tuples where all elements are divisible by K:", result)
```

```
Tuples where all elements are divisible by K: [(2, 4, 6), (4, 8, 16)]
```

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

In [ ]:
```python
# num = (input('Enter the elements of the list space seperated :').split()
num = list(map(int, input('Enter the elements of the list space separated: ').split

result = [(n, n**3) for i in num]

print('List of tuple (Number, Cube)', result)
```

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

In [19]:
```python
l1 = [(2, 4, 6), (3, 6, 12), (4, 8, 16), (5, 12, 19), (5, -12, 19), (-1, 2)]

result = []

for tup in l1:
    if all(i > 0 for i in tup):
        result.append(tup)

print("Tuples with all elements greater than 0:", result)
```

```
Tuples with all elements greater than 0: [(2, 4, 6), (3, 6, 12), (4, 8, 16), (5, 12,
19)]
```

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

In [25]:
```
l1 = [1, 2, 3, 4, 5]
t1 = (8, 9, 10, 11)

l1.append(t1)
print("List after adding a tuple:", l1)

l2 = [11, 22, 33, 44, 55]
t2 = tuple(l2) + t1
print("tuple after adding a list:", t2)
```

```
List after adding a tuple: [1, 2, 3, 4, 5, (8, 9, 10, 11)]
tuple after adding a list: (11, 22, 33, 44, 55, 8, 9, 10, 11)
```

## 07) WAP to remove tuples of length K.

In [29]:
```
l1 = [(2, 4, 6), (3, 6, 12), (4, 8, 16), (5, 12, 19), (5, -12, 19), (-1, 2)]

k = int(input('Enter the length to remove that perticular length of the tuple'))

for tup in l1:
    if len(tup) == k :
        l1.remove(tup)

print(l1)
```

```
[(2, 4, 6), (3, 6, 12), (4, 8, 16), (5, 12, 19), (5, -12, 19)]
```

## 08) WAP to remove duplicates from tuple.

In [33]:
```
tup1 = (1, 2, 3, 4, 4, 5, 9, 4, 7 ,11, 12, 11, 13, 6)

unique = tuple(set(tup1))
print("Tuple after removing duplicates:", unique)
```

```
Tuple after removing duplicates: (1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 13)
```

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

In [ ]:
```

```

In [49]:
```
tup1 = (1, 2, 3, 4, 4, 5, 9, 4, 7, 11, 12, 11, 13, 6)

result_list = []

for i in range(len(tup1) - 1):
    temp = tup1[i] * tup1[i + 1]
    result_list.append(temp)

tup2 = tuple(result_list)

print("Resultant tuple after multiplying adjacent elements is given by:", tup2)
```

```
Resultant tuple after multiplying adjacent elements is given by: (2, 6, 12, 16, 20,
45, 36, 28, 77, 132, 132, 143, 78)
```

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

In [51]:
```
tup = (1, 2, 3, 4, 5, 6, 7 ,6000, 7000, 700)

if(len(tup) ==  len(set(tup))):
    print("The tuple has distinct elements.")
else:
    print("The tuple has duplicate elements.")
```

```
The tuple has distinct elements.
```

In [ ]:

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Python Programming - 3011C4404

# Lab - 7

## Set & Dictionary

### 01) WAP to iterate over a set.

In [ ]:
```
x={23,567,76867,4,23,4234,35,65,56}

a= [print(i) for i in x]
```

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

In [ ]:
```
l1=list( [i for i in x])
print(l1)

t1=tuple([ [i for i in x]])
print(t1)

s1=""
for i in x:
    s1+=str(i)

print(s1)
```

```
t1 = tuple(s2)
print(f'Converted tuple from the set is {t1}')
```

```
Converted list frim the set is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Converted string frim the set is {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Converted tuple frim the set is (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

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

In [22]:
```
s1 = {1, 2 ,3 ,4 ,5 ,6, 78, 99, 4444}
maximum = max(s1)
minimum = min(s1)

print(f'the max elements from the set is : {maximum}')
print(f'The min elements from the set is : {minimum}')
```

```
the max elements from the set is : 4444
The min elements from the set is : 1
```

## 04) WAP to perform union of two sets.

In [25]:
```
s1 = {1, 2, 3, 4, 5, 6, 7, 8}
s2 = {2 ,3 ,4 ,5, 99, 888 ,44}

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

```
{1, 2, 3, 4, 5, 6, 7, 8, 99, 44, 888}
```

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

In [29]:
```
l1 = [1, 2, 3, 4, 5, 6, 7, 8]
l2 = [2 ,3 ,4 ,5, 99, 888 ,44]

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

if(s1 & s2) :
    print('The given Two lists have at-least one common element ')
else :
    print('The given Two lists have not at-least one common element ')
```

```
The given Two lists have at-least one common element
```

## 06) WAP to remove duplicates from list.

In [31]:
```
l1 = [1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5]

s1 = set(l1)
print('The unique elements are ', s1)
```

```
The unique elements are  {1, 2, 3, 4, 5, 6, 7, 8}
```

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

In [43]:
```
str1 = input('Enter the words space seperatedly :').split()

#s1 = set(str1)
#print('The Unique words are :', s1)
d1 = dict();
for word in str1:
    if word in d1 :
        d1[word] += 1
    else:
        d1[word] = 1

s1 = set()
for word, counter in d1.items() :
    if d1[word] == 1:
        s1.add(word)


print(s1)
```

```
{'hello', 'viraj'}
```

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

In [45]:
```
s1 = {1, 2, 3, 4, 5, 6, 7, 8}
s2 = {2 ,3 ,4 ,5, 99, 888 ,44}
s1 = s1 - s2
print(s1)
```

```
{8, 1, 6, 7}
```

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

In [63]:
```
str1 = 'eat'
str2 = 'tea'

if(len(str1) != len(str2)):
    print("Enter valid Strings to check ..")

else :
    for i in set(str1) :
        if(str1.count(i) != str2.count(i)):
            print('Not Anagram !!')
            break;
    else :
        print('Anagram !')
```

```
Anagram !
```

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

In [67]:
```
l1 = [1, 2, 3, 4, 3, 5, 6, 7, 8]
l2 = [11, 2, 33, 4, 77,5, 6, 7, 8]
l3 = [1, 21, 39, 4, 33, 555, 6, 7, 8]

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

print(s1 & s2& s3)
```

```
{8, 4, 6, 7}
```

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

In [73]:
```
str1 = input('Enter the string !!')
vowels = 'aeiou'
counter = 0
for i in set(str1.lower()):
    if i in vowels:
        counter += str1.count(i)

print("The count of the vowels in the string is :", counter)
```

```
The count of the vowels in the string is : 10
```

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

In [79]:
```
str1 = '10010100001'
s1 = set(str1)
str2 = '01'
flag = True
for i in s1:
    if i not in str2 :
        flag = False
        break

if(flag) :
    print('The entered string is binary string !')
else:
    print('The entered string is not binary string !')
```

```
The entered string is binary string !
```

## 13) WAP to sort dictionary by key or value.

In [85]:
```
d1 = {'a' : 1, 'b' : 3, 'c' : 4, 'd' : 10, 'e' : 0}
print('Before Sort' , d1)

keys = list(d1.keys())
keys.sort()
```

```
d2 = {i:d1[i] for i in keys}
print('The sorted dictionary ', d2)
```

```
Before Sort {'a': 1, 'b': 3, 'c': 4, 'd': 10, 'e': 0}
The sorted dictionary  {'a': 1, 'b': 3, 'c': 4, 'd': 10, 'e': 0}
```

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

In [91]:
```
n = int(input('Enter the size of the dictionary'))
d1 = dict()

for i in range(1, n + 1):
    key = input('Entre the key')
    val = int(input('Enter the val of the key'))

    d1[key] = val

print('the sum of the value of the dictionary is : ', sum(d1.values()))
```

```
the sum of the value of the dictionary is :  8
```

## 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 [95]:
```
dict1 = {'a' : 5, 'c' : 8, 'e' : 2}
key = input('Enter the key :')

if key in d1.keys():
    print(dict1[key])
else:
    print('Key Not Found')
```

```
Key Not Found
```

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

# Python Programming - 2011CS404

# Lab - 8

## User Defined Function

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

In [5]:
```
def bmiCalc(mass, height) :
    return (mass/height**2)

print(bmiCalc(60, 2))
```

```
15.0
```

### 02) Write a function that add first n numbers.

In [9]:
```
def addFirstN(n):
    sum = 0
    for i in range(n + 1):
        sum += i
    return sum

n = int(input('Enter the n to get the sum of the first n :'))
print(f'The Sum of the first {n} number is :', addFirstN(n))
```

```
The Sum of the first $4 number is : 10
```

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

In [51]:
```
import math
def primeNumber(n) :
    m = n
    for i in range(2, int(math.sqrt(n)) + 1):
        if (m % i == 0) :
            return 0;
    return 1;


n = int(input('Enter the n to Check if it is Prime number or not :'))
str = primeNumber(n)

if(str == 1) :
    print(f'The given number {n} is prime')
else:
    print(f'The given number {n} is prime')
```

```
The given number 5 is prime
```

In [ ]:
```

```

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

In [62]:
```
# import math
# def primeNumberList(l2) :
#     l3 = [];
#     for i in l2 :
#         for j in range(2, int(math.sqrt(i)) + 1):
#             if (i % j == 0) :
#                 break;
#         l3.append(i)
#     return l3

def primeNumberList(n1, n2):
    prime = []
    for i in range(n1, n2 + 1):
        if primeNumber(i) == 1 :
            prime.append(i)
    return prime

n1 = int(input('Enter the first number :'))
n2 = int(input('Enter the first number :'))
primeNumberList(n1, n2)


# l1 = input('Enter the elements space seperated ').split()
# l2 = [int(i) for i in l1]
# primeNumberList(l2)
```

Out[62]:
```
[2, 3, 5, 7]
```

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

In [70]:
```
def isPalindrome(str1):
    if str1 == str1[::-1] :
        return True
    else:
        return False

isPalindrome('Viraj')
```

Out[70]: False

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

In [72]:
```
def sumOfList(l2) :
    return sum(l2)

l1 = input('Enter the elements space seperated ').split()
l2 = [int(i) for i in l1]
sumOfList(l2)
```

Out[72]: 27

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

In [78]:
```
def sumOfFirst(l1):
    return sum(t1[0] for t1 in l1)

l1 = [(1, 2, 3), (4, 5, 6)]
sumOfFirst(l1)
```

Out[78]: 5

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

In [89]:
```
def fibbonacci(n) :
    if n == 1 :
        return 1
    if n == 0 :
        return 0
    else:
        return fibbonacci(n - 1) + fibbonacci(n - 2)

n = int(input('Enter the nth turm of the number :'))
fibbonacci(n)
```

Out[89]: 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 [91]:
```
def findValue(roll) :
    return dict1[roll]

dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
print(findValue(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 [98]:
```
def sumList(l2):
    sum = 0
    for i in l2:
        if i % 10 == 0:
            sum += i

    return sum

l1 = input('Enter the elements space seperated ').split()
l2 = [int(i) for i in l1]
sumList(l2)
```

Out[98]: 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 [108…
```
def invertDict(dict1):
    dict2 = dict()
    for i,j in dict1.items():
        dict2[j] = i
    return dict2
```

```
dict1 = {'a': 10, 'b':20, 'c':30, 'd':40}
invertDict(dict1)
```

Out[108… `{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 [112…

```
def panagram(s1):
    s1 = s1.lower()
    s2 = 'abcdefghijklmnopqrstuvwxyz'

    return set(s1).issuperset(s2)


s1 = "the quick brown fox jumps over the lazy dog"
panagram(s1)
```

Out[112… `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 [120…

```
def countUpperAndLower(s1):
    l1 = [0, 0]
    for i in s1:
        if( i.isupper()):
            l1[0] += 1
        elif( i.islower()):
            l1[1] += 1
        else :
            continue
    return l1

s1 = 'Hello i am odedra Viraj ,this is python'
countUpperAndLower(s1)
```

Out[120… `[2, 29]`

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

In [134…

```
smallest = lambda x, y : y if x > y else x

smallest(2, 4)
```

Out[134… 2

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

In [147…
```
l1 = ['Dhairya', 'Pruthviraj', 'Viraj', 'Kai']
# names = list(filter(name in l1 name if len(name) > 7))
# print(names)

names = list(filter (lambda name : len(name) > 7, l1))
print(names)
```

```
['Pruthviraj']
```

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

In [151…
```
l1 = ['dhairya', 'pruthviraj', 'viraj', 'kai']

names = list(map( lambda name : name.capitalize(), l1))
print(names)
```

```
['Dhairya', 'Pruthviraj', 'Viraj', 'Kai']
```

## 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 [ ]:
```
#positional Argument
def greetPositional(name, age):
    print(f'Ki hal chal {name} Paji, aap {age} ho gye ')
#keyword Argument
def greetKeyword(name, age):
    print(f'Ki hal chal {name} Paji, aap {age} ho gye ')

#Default Argument
def greetKeyword(name, age = 22):
    print(f'Ki hal chal {name} Paji, aap {age} ho gye ')

#Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)


# 1. Positional Arguments
def greet_positional(name, age):
    print(f"Hello {name}, you are {age} years old.")
```

```python
# 2. Keyword Arguments
def greet_keyword(name, age):
    print(f"Hello {name}, you are {age} years old.")

# 3. Default Arguments
def greet_default(name, age=18):
    print(f"Hello {name}, you are {age} years old.")

# 4. Variable Length Positional Arguments (*args)
def greet_args(*names):
    for name in names:
        print(f"Hello {name}!")

# 5. Variable Length Keyword Arguments (**kwargs)
def greet_kwargs(**people_info):
    for name, age in people_info.items():
        print(f"Hello {name}, you are {age} years old.")

# 6. Keyword-Only Arguments
def greet_keyword_only(name, *, age):
    print(f"Hello {name}, you are {age} years old.")

# 7. Positional-Only Arguments
def greet_positional_only(name, /, age):
    print(f"Hello {name}, you are {age} years old.")


print("Positional Arguments:")
greet_positional("Alice", 25)

print("\nKeyword Arguments:")
greet_keyword(name="Alice", age=25)

print("\nDefault Arguments:")
greet_default("Alice")
greet_default("Bob", 30)

print("\nVariable Length Positional Arguments (*args):")
greet_args("Alice", "Bob", "Charlie")

print("\nVariable Length Keyword Arguments (**kwargs):")
greet_kwargs(Alice=25, Bob=30, Charlie=35)

print("\nKeyword-Only Arguments:")
greet_keyword_only("Alice", age=25)

print("\nPositional-Only Arguments:")
greet_positional_only("Alice", 25)
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Python Programming - 201CCS404

# Lab - 9

## 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 [15]:
```
#in the form of the string
fp = open("file1.txt", 'r')
print(fp.read())

fp.seek(0, 0)
line1 = fp.readline()
print(line1, end="")
line2 = fp.readline()
print(line2, end="")
line3 = fp.readline()
print(line3, end="")

fp.seek(0, 0)
list1 = fp.readlines()
print(list1)

fp.close()
```

```
this is the text file.
creted by the admin.
file
demo file

12345678910
this is the text file.
creted by the admin.
file
['this is the text file.\n', 'creted by the admin.\n', 'file\n', 'demo file\n',
'\n', '12345678910']
```

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

In [17]:
```
fp = open('file2.txt', 'w');
```

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

In [19]:
```
fp = open('file1.txt', 'r')
line1 = fp.readline()
print(line1, end="")
line2 = fp.readline()
print(line2, end="")
line3 = fp.readline()
print(line3, end="")
line4 = fp.readline()
print(line4, end="")
line5 = fp.readline()
print(line5, end="")
fp.close()

counter = 0
for i in fp:
    if counter <= 5:
        print(i)
    else:
        break

fp.close()
```

```
this is the text file.
creted by the admin.
file
demo file
```

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

In [27]:
```
fp = open('file1.txt', 'r')
longestWord = ''

for i in fp :
    for j in i.split():
```

2

```
        if len(j) > len(longestWord):
            longestWord = j

print(longestWord)
fp.close()
```

```
asdfasdfasdfasfdadfsadsfafd
```

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

In [35]:
```
fp = open('file1.txt')
lines, words, characters = 0, 0, 0

for i in fp:
    lines += 1
    characters  += len(i)
    words += len(i.split())
print('The lines, words, character  in the file are :', lines, characters, words)
fp.close()

# fp.seek(0, 0)
# for i in fp:
#     for j in i.split():
#         words += 1
# print('The words in the file are :', words)
```

```
The lines, words, character  in the file are : 12 141 16
```

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

In [42]:
```
fp = open('file1.txt')
fv = open('copy.txt', 'w')

for i in fp:
    fv.write(i)

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

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

In [52]:
```
fp = open('copy.txt', 'r')
fp.seek(0, 0)
fp.seek(0, 2)
print('The size of the text file is :', fp.tell() + 1)
```

```
The size of the text file is : 75
```

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

In [60]:
```
def frequencyCount(specificWord):
    fp = open('file1.txt', 'r')
    file = fp.read().lower()
    freq = file.split().count(specificWord.lower())
    print(f'The Occurence of the {specificWord} is {freq}')

    # for i in fp:
    #     for word in i.split():
    #         if word == specificWord:
    #             counter += 1
    # fp.close()


frequencyCount('pruthviraj')
```

```
The Occurence of the pruthviraj is 16
```

## 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 [90]:
```
fp = open('marks.txt', 'w')

for i in range(0, 5):
    marks = (input('Enter the subject mark :'))
    fp.write(str(marks) + '\n')
fp.close()
fp = open('marks.txt')
highest = 0
highest = [int(i.strip()) for i in fp]
print(max(highest))
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 [105…
```
import math
count = 0
num = 2
def primeNumber(num):
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

fp = open('prime', 'w')
while count < 100 :
```

```
    if(primeNumber(num)):
        count += 1
        fp.write(str(num) + '\n')

    num += 1
fp.close()
```

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

In [111…

```
fp1 = open('copy.txt')
fp = open('merged.txt', 'w')
fp2 = open('marks.txt')

fp.write(fp1.read())
fp.write(fp2.read())

fp1.close()
fp.close()
fp2.close()
```

Out[111… 10

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

In [115…

```
fp = open('file1.txt', 'r')
fp2 = open('file2.txt', 'w')

word = 'viraaaj'

for i in fp :
    fp2.write(i.replace('viraj', word))
```

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

In [ ]:

```
# Random Access of file
#  tell()
#  tell() is used to obtain the current position of the file pointer within the
# file.
 fp = open("file1.txt","r")
 print("Before reading pointer is at:", fp.tell()) #file pointer always
starts from 0.
 fp.read(2)
 print("After reading 2 characters pointer is at: ",fp.tell())
 fp.read(3)
 print("After reading 5 characters pointer is at: ",fp.tell())
 fp.read()
 print("after going to the end of the file, pointer is at:", fp.tell())
 fp.close()
#  Before reading pointer is at: 0
```

```
#  After reading 2 characters pointer is at:  2
#  After reading 5 characters pointer is at:  5
#  after going to the end of the file, pointer is at: 34
#  seek(offset, startpoint_for_offset)
#  offset : how many positions the pointer will move
#  startpoint_for_offset : specifies the point from where pointer will
# move specified as offset
#  0 -> from the beginning of the file (default)
#  1 -> from the current position of file pointer
#  2 -> from the end of the file
 fp = open("file2.txt","r")
 print("Before reading: ",fp.tell())
 fp.read(6)
 print("After reading 6 characters, pointer is at:", fp.tell())
 fp.seek(0,0) # 0 characters from beginning of the file i.e. at 0
 print("After seeking pointer to beginning of the file:", fp.tell())
 fp.close()
 # Before reading:  0
 # After reading 6 characters, pointer is at: 6
 # After seeking pointer to beginning of the file: 0
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Python Programming - 201CCS404

# Lab - 10

## 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 [10]:
```
try :
    num1 = int(input('Enter the number :'))
    num2 = int(input('Enter the divisor :'))

    print(num1 / num2)

except ZeroDivisionError :
    print("can't divivde by the ZERO !")
except ValueError :
    print('Enter the values preperly !')
except TypeError :
    print('Enter the INTEGERS only !')
```

```
Enter the values preperly !
```

### 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [18]:
```
try :
    l1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    idx = int(input('Enter the index to get the element at that idx :'))
    print('The value at the given index is ', l1[idx])
except IndexError:
    print('Index is out of the range !')


try :
    my_dict = {'name': 'Viraj', 'gender': 'Male', 'city': 'New York'}
    print(my_dict['age'])
except KeyError :
    print('Key not found !')
```

```
The value at the given index is  1
Key not found !
```

## 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [26]:
```
try :
    file1 = open("abc.txt")
except FileNotFoundError :
    print('File Not Found !!')

try:
    import non_existent_module
except ModuleNotFoundError as e:
    print(e)
```

```
File Not Found !!
No module named 'non_existent_module'
```

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

In [30]:
```
try :
    num1 = int(input('Enter the number :'))
    num2 = int(input('Enter the divisor :'))

    print(num1 / num2)

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

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

## 05) WAP to demonstrate else and finally block.

In [38]:
```
try :
    num1 = int(input('Enter the number :'))
    num2 = int(input('Enter the divisor :'))

    print(num1 / num2)

except ZeroDivisionError :
    print("can't divivde by the ZERO !")
except ValueError :
    print('Enter the values preperly !')
except TypeError :
    print('Enter the INTEGERS only !')
else :
    print('Program DOne !!')

finally :
    print('Finally, Program ENDED !!!')
```

```
3.0
Program DOne !!
Finally, Program ENDED !!!
```

## 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 [44]:
```
try :
    num = input('Enter the grades comma seperated').split(',')

    list1 = [int(grade) for grade in num]
    print(list1)

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

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

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

In [58]:
```
def division(n1, n2):
    try :
        return n1 / n2
    except ZeroDivisionError as exp:
```

```
        print(exp)

n1 = int(input("Enter the first number :"))
n2 = int(input("Enter the divisor number :"))
division(n1, n2)
```

```
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 [60]:
```
try :
    age = int(input('Enter your AGE :'))

    if age < 18 :
        raise ValueError('Enter the Valid Age ,you are Under Age')
    else :
        print('Your age is :', age)
except ValueError as exp :
    print("ERROR ! ", exp)
```

```
ERROR !  Enter the Valid Age ,you are Under Age
```

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

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

### otherwise print the given username.

In [76]:
```
class InvalidUsernameError(Exception):
    def __init__(self, msg):
        self.msg = msg
try:
    login = (input("Enter a number: "))
    if (len(login) < 3 or len(login) > 15):
        raise InvalidUsernameError("Username must be between 3 and 15 characters !!
    else:
        print('The username is ', login)

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

```
Username must be between 3 and 15 characters !!
```

## 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 [82]:
```
import math
class NegativeNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg
try:
    num1 = int(input("Enter a number: "))
    if (num1 < 0):
        raise NegativeNumberError(" Can't find the square root of the Negative numb
    else:
        print('The square root is ', math.sqrt(num1))

except NegativeNumberError as err:
    print(err)


# or

# try:
#     num1 = int(input("Enter a number: "))
#     if (num1 < 0):
#         raise NegativeNumberError(" Can't find the square root of the Negative nu
#     else:
#         print('The square root is ', math.sqrt(num1))

# except NegativeNumberError as err:
#     print(err)
```

```
Can't find the square root of the Negative numbers !!
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

# Python Programming - 2011CS404

# Lab - 11

## 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 [40]:
```
import  Calculator as c

print(c.add(2,3))
print(c.sub(2,3))
print(c.mul(2,3))
print(c.div(2,3))
```

```
5
-1
6
0.6666666666666666
```

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

In [5]:
```
import random as r
print(r.choice("hello"))
```

```
e
```

```
r
```

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

In [26]:
```
l1 = input('Enter the string space seperately !').split()

print(l1[random.randint(0, len(l1) - 1)])
```

```
viraj
```

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

In [53]:
```
import random
random.seed(1)

print(random.randint(1, 6))
random.seed(1)
print(random.randint(1, 6))
random.seed(1)
print(random.randint(1, 6))
random.seed(1)
print(random.randint(1, 6))
random.seed(1)
print(random.randint(1, 6))
random.seed(1)
print(random.randint(1, 6))
random.seed(1)

#or
# random.seed(2)
# print(random.choice(range(1, 6)))
```

```
2
2
2
2
2
2
1
```

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

In [63]:
```
import random
def randomGenerator() :
    counter = 0
    l1 = list()
    while(counter < 3) :
        num1 = random.choice(range(100, 999))
        if(num1 % 5 == 0) :
            l1.append(num1)
            counter += 1
```

```
    return l1

l2 = randomGenerator()
print(l2)
```

```
[695, 840, 980]
```

## 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 [67]:
```
import random
def generateTicket():
    winner = random.choice(range(1, 101))
    runnerUp = random.choice(range(1, 101))

    print(f'And the Final Winner is {winner}')
    print(f'And the Final Winner is {runnerUp}')


generateTicket()
```

```
And the Final Winner is 70
And the Final Winner is 57
```

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

In [74]:
```
import datetime
date = datetime.datetime.now().date()
time = datetime.datetime.now().time()
print(date, time)
```

```
2025-02-19 10:44:03.738278
```

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

In [90]:
```
# from datetime import datetime, timedelta

# date = str(input('Enter ATP date(Format Mmmmm Dd): '))
# date = datetime.strptime(date, "%B %d")
# prev_date = (date - timedelta(days=7)).strftime('%B %d')

import datetime
from datetime import timedelta

given_date = datetime.datetime.now().date()
newDate = given_date - timedelta(days=7)
print(newDate)
```

```
2025-02-12
```

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

In [92]:
```
from datetime import datetime

# format YYYY-MM-DD
date_str1 = input("Enter the first date (YYYY-MM-DD): ")
date_str2 = input("Enter the second date (YYYY-MM-DD): ")

date1 = datetime.strptime(date_str1, '%Y-%m-%d').date()
date2 = datetime.strptime(date_str2, '%Y-%m-%d').date()

days_difference = abs((date1 - date2).days)

print(f"Number of days between the dates: {days_difference}")
```

```
Number of days between the dates: 31
```

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

In [98]:
```
import datetime

def find_day_of_week(date):
    date_obj = datetime.datetime.strptime(date, "%Y-%m-%d")

    day_number = date_obj.weekday()
    days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Su

    return days[day_number]

date_input = input("Enter the date (YYYY-MM-DD): ")
print(f"The day of the week is: {find_day_of_week(date_input)}")
```

```
The day of the week is: Tuesday
```

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

In [96]:
```
import datetime

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

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

# 3. Get the current time
current_time = now.time()
print("Current Time:", current_time)

# 4. Get the current year, month, day, etc.
year = now.year
month = now.month
day = now.day
hour = now.hour
```

```
minute = now.minute
second = now.second

print(f"Year: {year}, Month: {month}, Day: {day}, Hour: {hour}, Minute: {minute}, S

# 5. Create a specific date and time
specific_date = datetime.date(2025, 2, 19)
specific_time = datetime.time(14, 30)
specific_datetime = datetime.datetime(2025, 2, 19, 14, 30)

print("Specific Date:", specific_date)
print("Specific Time:", specific_time)
print("Specific DateTime:", specific_datetime)

# 6. Date and Time formatting
formatted_date = now.strftime("%A, %B %d, %Y")
formatted_time = now.strftime("%I:%M:%S %p")
print("Formatted Date:", formatted_date)
print("Formatted Time:", formatted_time)

# 7. Add days to a date
future_date = today + datetime.timedelta(days=10)
print("Future Date (10 days later):", future_date)

# 8. Subtract days from a date
past_date = today - datetime.timedelta(days=5)
print("Past Date (5 days ago):", past_date)

# 9. Calculate the difference between two dates
date1 = datetime.date(2025, 2, 19)
date2 = datetime.date(2025, 3, 1)
difference = date2 - date1
print("Difference between two dates:", difference.days, "days")

# 10. Get the day of the week for a specific date
day_of_week = date1.strftime("%A")
print(f"Day of the Week for {date1}: {day_of_week}")
```

```
Current Date and Time: 2025-02-19 11:03:44.760427
Current Date: 2025-02-19
Current Time: 11:03:44.760427
Year: 2025, Month: 2, Day: 19, Hour: 11, Minute: 3, Second: 44
Specific Date: 2025-02-19
Specific Time: 14:30:00
Specific DateTime: 2025-02-19 14:30:00
Formatted Date: Wednesday, February 19, 2025
Formatted Time: 11:03:44 AM
Future Date (10 days later): 2025-03-01
Past Date (5 days ago): 2025-02-14
Difference between two dates: 10 days
Day of the Week for 2025-02-19: Wednesday
```

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

In [94]:
```python
import math

# 1. Mathematical constants
print("Value of pi:", math.pi)
print("Value of Euler's number (e):", math.e)

# 2. Basic mathematical operations
print("Square root of 16:", math.sqrt(16))  # Square root
print("Power of 2 raised to 3:", math.pow(2, 3))  # Power
print("Absolute value of -10:", math.fabs(-10))  # Absolute value

# 3. Trigonometric functions
angle_in_radians = math.radians(30)  # Convert degrees to radians
print("sin(30 degrees):", math.sin(angle_in_radians))
print("cos(30 degrees):", math.cos(angle_in_radians))
print("tan(30 degrees):", math.tan(angle_in_radians))

# 4. Rounding functions
print("Floor of 3.7:", math.floor(3.7))  # Rounds down
print("Ceiling of 3.2:", math.ceil(3.2))  # Rounds up
print("Round of 3.5:", round(3.5))  # Rounds to nearest integer

# 5. Factorial
print("Factorial of 5:", math.factorial(5))  # Factorial

# 6. Logarithmic functions
print("Natural logarithm of 10:", math.log(10))  # Natural log (base e)
print("Logarithm base 10 of 1000:", math.log10(1000))  # Log base 10

# 7. Greatest common divisor (GCD)
print("GCD of 36 and 60:", math.gcd(36, 60))  # GCD of two numbers

# 8. Exponential function
print("Exponential of 3:", math.exp(3))  # e^3

# 9. Modulus
print("Modulus of 10 by 3:", math.fmod(10, 3))  # Modulus (remainder)
```

```
Value of pi: 3.141592653589793
Value of Euler's number (e): 2.718281828459045
Square root of 16: 4.0
Power of 2 raised to 3: 8.0
Absolute value of -10: 10.0
sin(30 degrees): 0.49999999999999994
cos(30 degrees): 0.8660254037844387
tan(30 degrees): 0.5773502691896257
Floor of 3.7: 3
Ceiling of 3.2: 4
Round of 3.5: 4
Factorial of 5: 120
Natural logarithm of 10: 2.302585092994046
Logarithm base 10 of 1000: 3.0
GCD of 36 and 60: 12
Exponential of 3: 20.085536923187668
Modulus of 10 by 3: 1.0
```

In [ ]:

Darshan UNIVERSITY

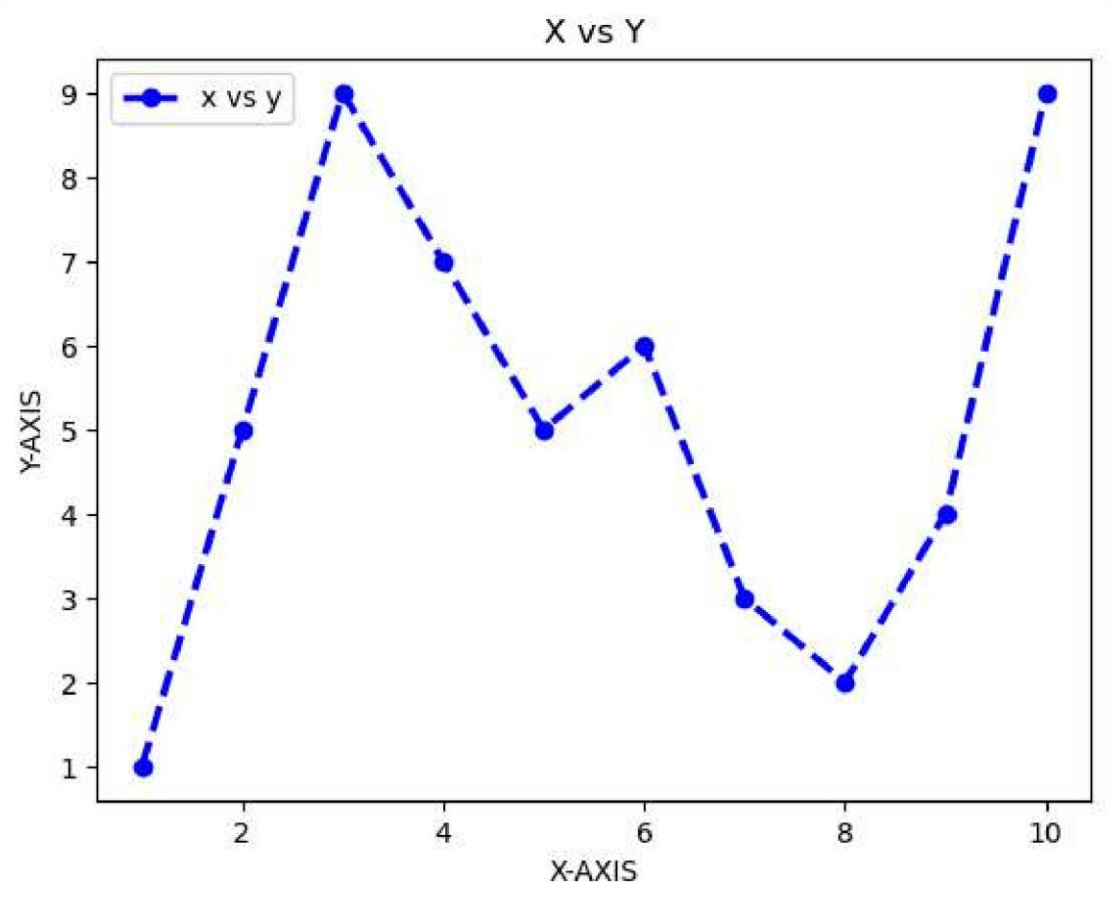
योग: कर्मसु कौशलम्

# Python Programming - 2311CS404

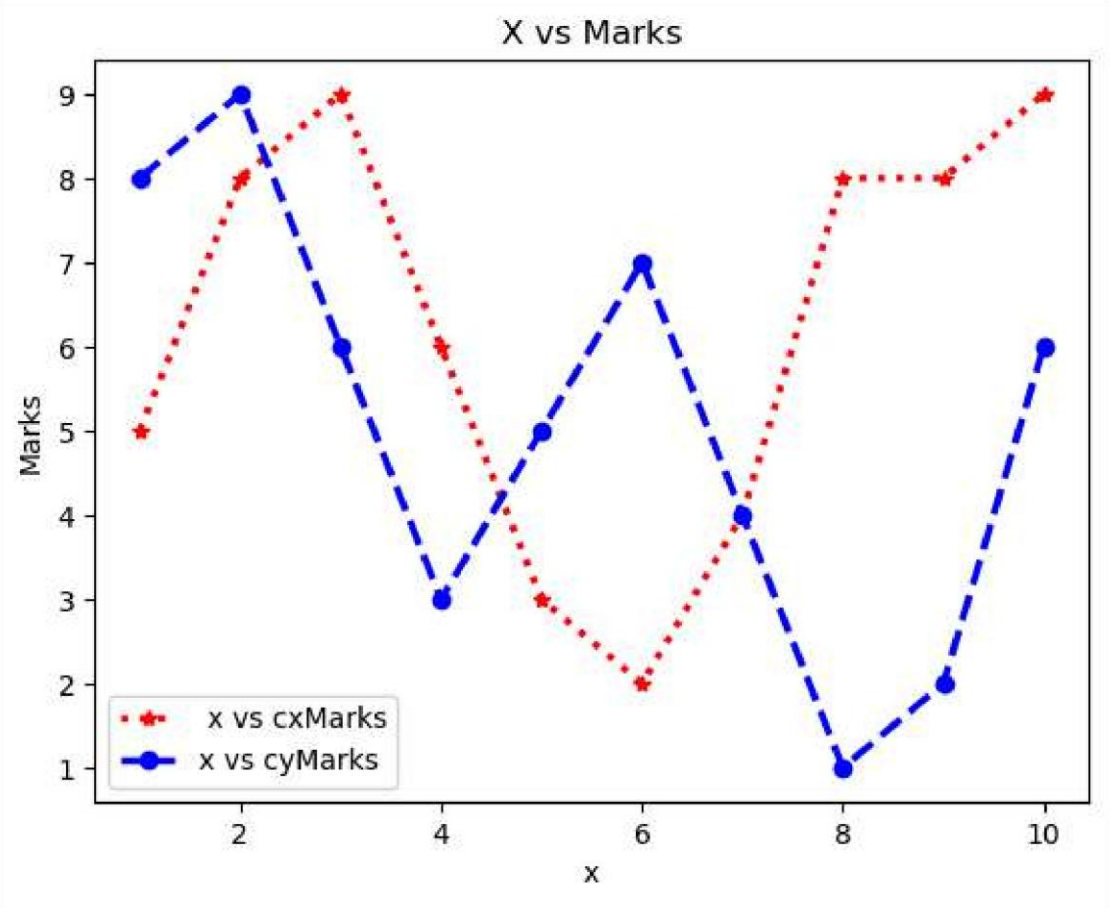
# Lab - 12

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

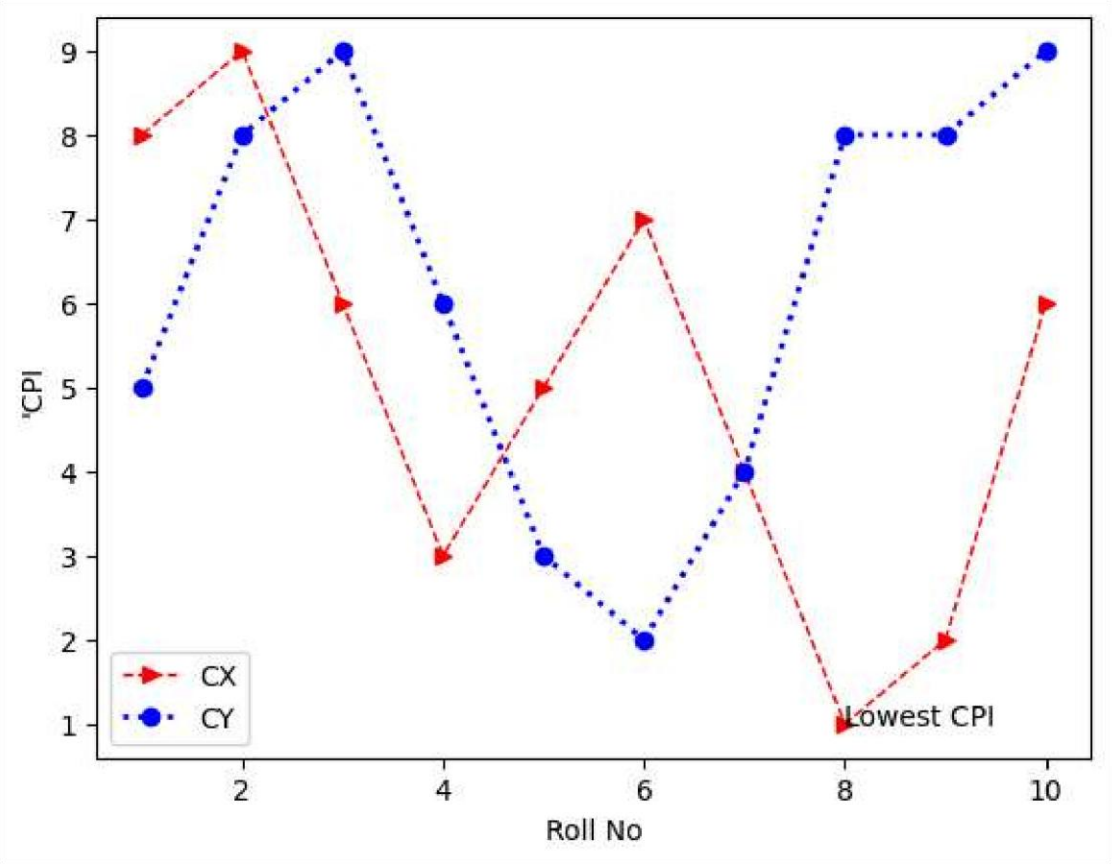
In [16]:
```
# Data
x = range(1, 11)
y = [1, 5, 9, 7, 5, 6, 3, 2, 4, 9]

# Plotting
plt.plot(x, y, label='Line')

# Labels and title
plt.xlabel("x - axis")
plt.ylabel("y - axis")
plt.title("x vs y")

# Grid and Legend
plt.grid(True)
plt.legend()

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



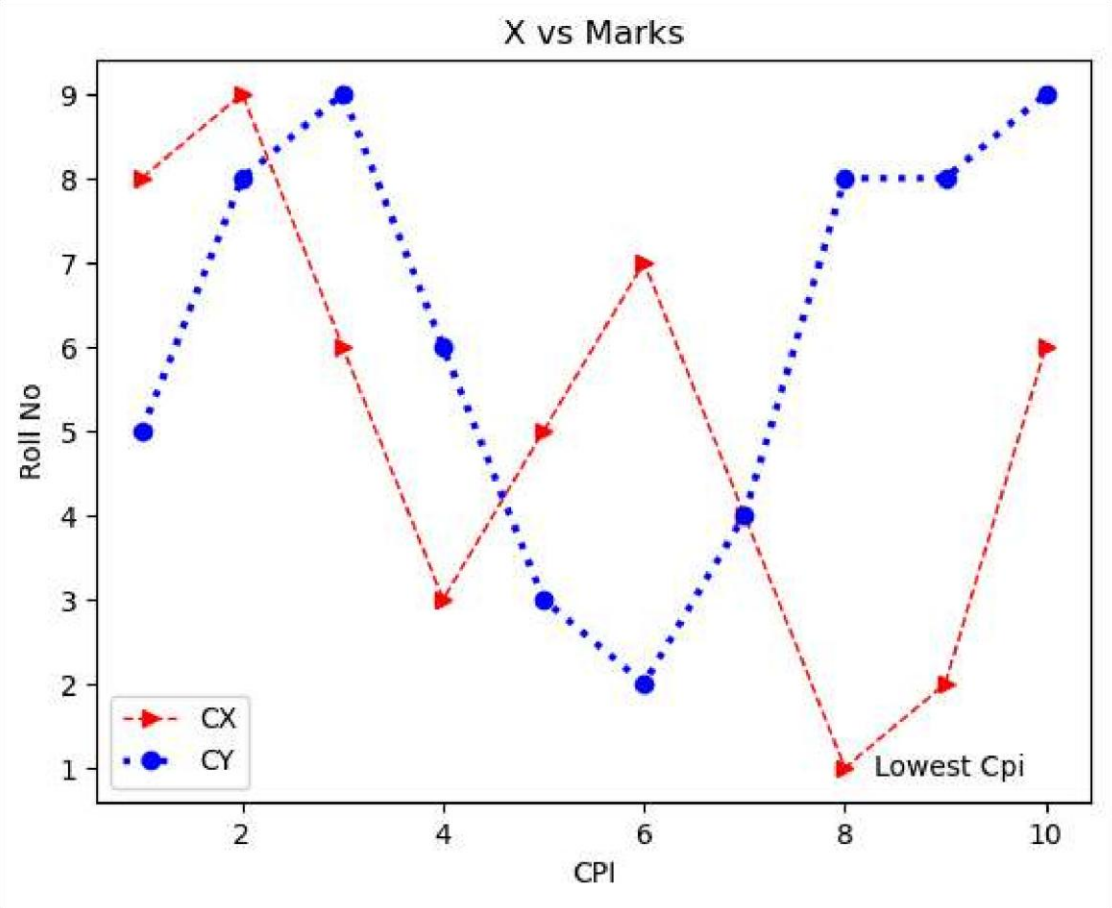

In [7]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

plt.plot(x,cxMarks, ls=":", lw=2.5, c="r", marker="*",label = " x vs cxMarks")
plt.plot(x, cyMarks, ls="--", lw=2.5 , c="b", marker="o", label = "x vs cyMarks")
plt.xlabel("x")
plt.ylabel("Marks")
plt.title("X vs Marks")
plt.legend()
plt.show()

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




In [13]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

# plt.plot(x,cxMarks, ls="-", lw=1, c="r", marker="^",label = " x vs cxMarks")
# plt.plot(x, cyMarks, ls="-", lw=2.5 , c="b", marker="o", label = "x vs cyMarks")
# plt.xlabel("x")
# plt.ylabel("Marks")
# plt.title("X vs Marks")
# plt.annotate("Lowest Point", xy=[2.2,19])
# plt.legend()
# plt.show()


# write a code to generate below graph
```

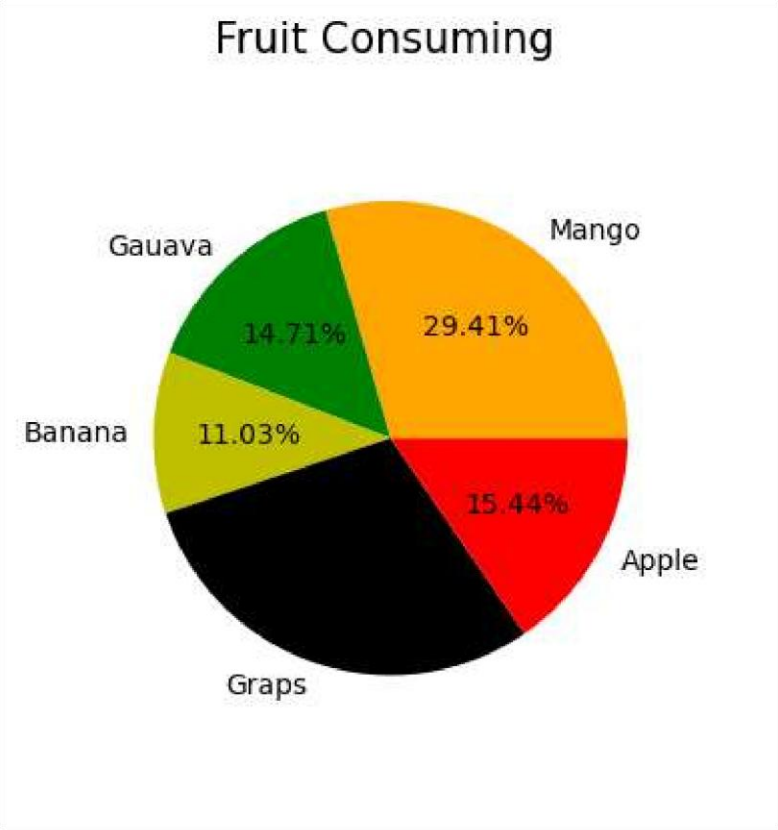



In [27]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

plt.plot(x,cxMarks, ls="--", lw=1, c="r", marker=">",label = "CX")
plt.plot(x, cyMarks, ls="dotted", lw=2.5 , c="b", marker="o", label = "CY")
plt.xlabel("CPI")
plt.ylabel("Roll No")
plt.title("X vs Marks")
plt.annotate("Lowest Cpi", xy=[8.3, 0.9])
plt.legend()
plt.show()
```



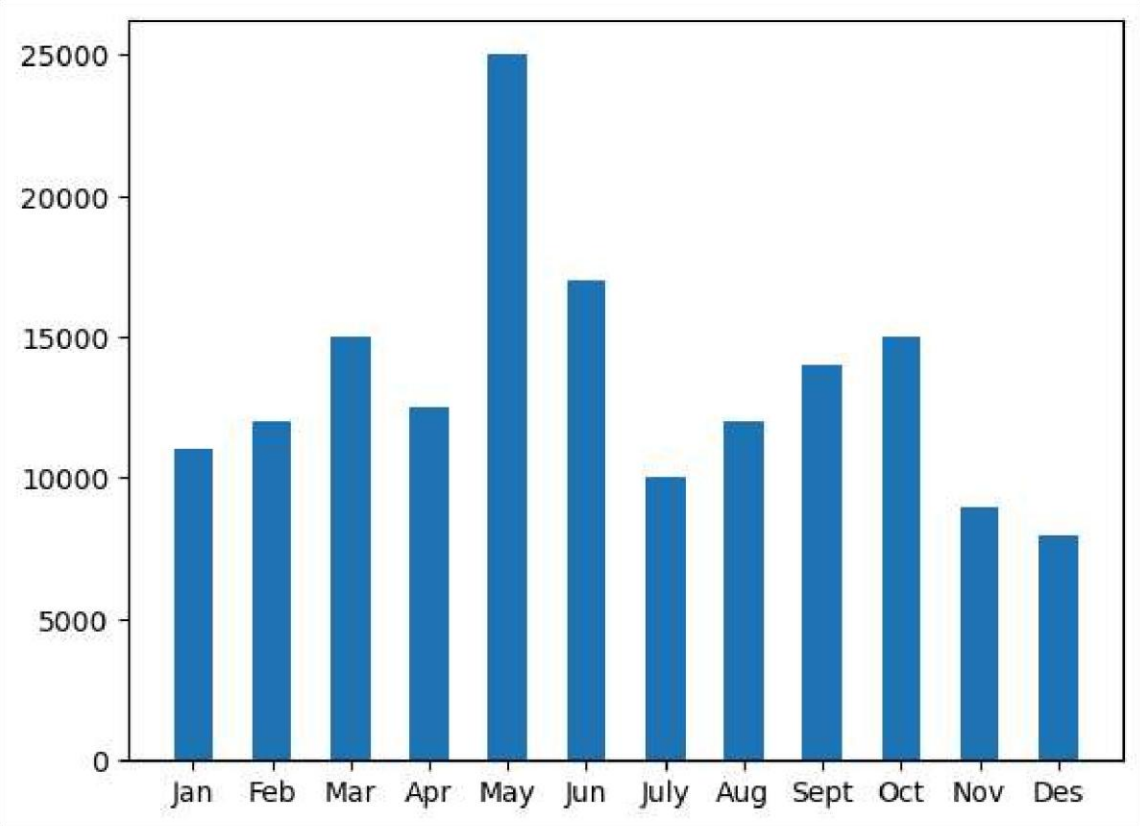

## 04) WAP to demonstrate the use of Pie chart.

In [33]:
```
dept = ["Mango", "Gauava", "Banana", "Graps", "Apple"]
stud = [4000, 2000, 1500, 4000, 2100]
x = ["orange","g","y","k","r"]
plt.pie(stud, labels = dept, colors=x, autopct="%1.2f%%", radius=0.8)
plt.title("Fruit Consuming ",fontsize=15)
plt.show()
```



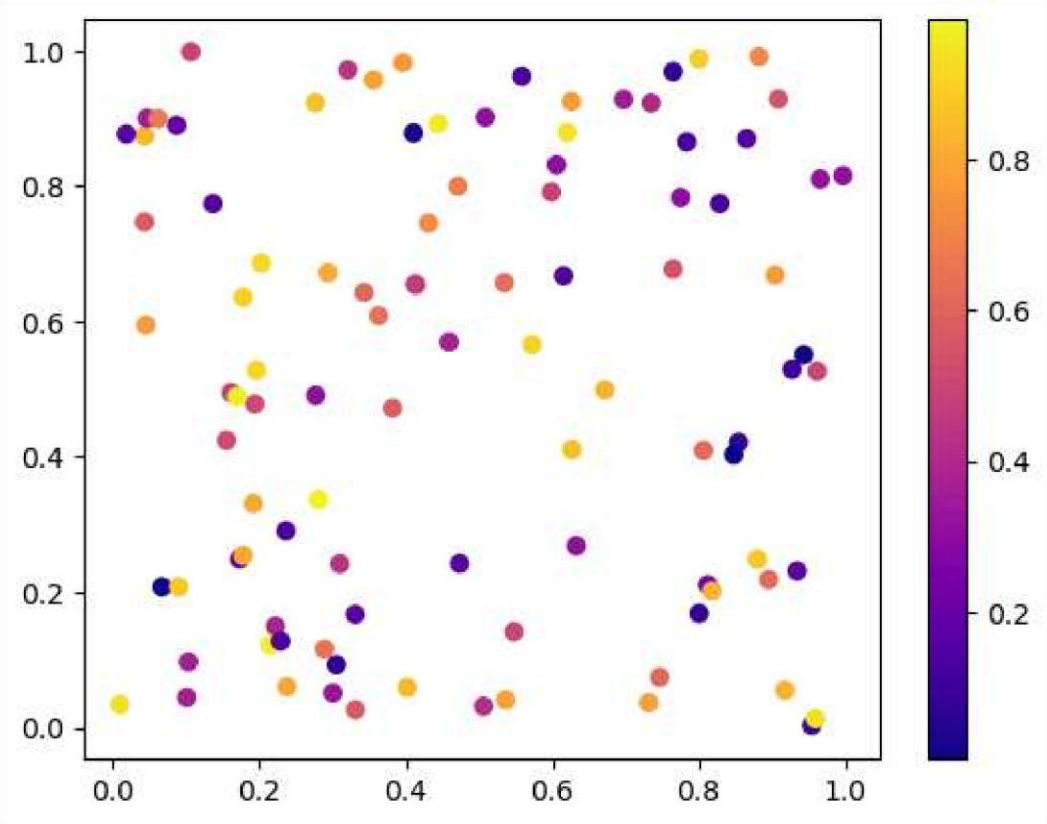

## 05) WAP to demonstrate the use of Bar chart.

In [35]:
```
mon = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov
visiters = [11000, 12000, 15000, 12500, 25000, 17000, 10000, 12000, 14000, 15000, 9
plt.bar(mon, visiters, width=0.5)
plt.show()
```



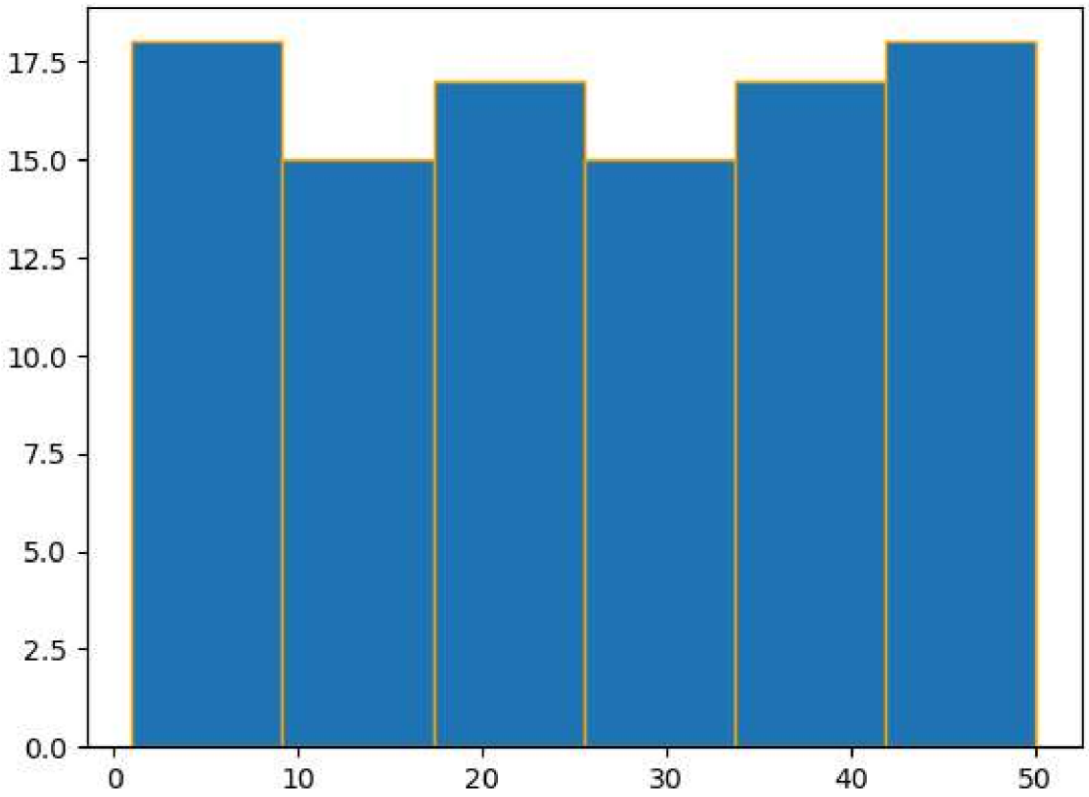

## 06) WAP to demonstrate the use of Scatter Plot.

In [41]:
```
random.seed(4)
x = [random.random() for i in range(100)]
y = [random.random() for i in range(100)]
z = [random.random() for i in range(100)]
plt.scatter(x,y, c=z,cmap="plasma")
plt.colorbar()
plt.show()
```



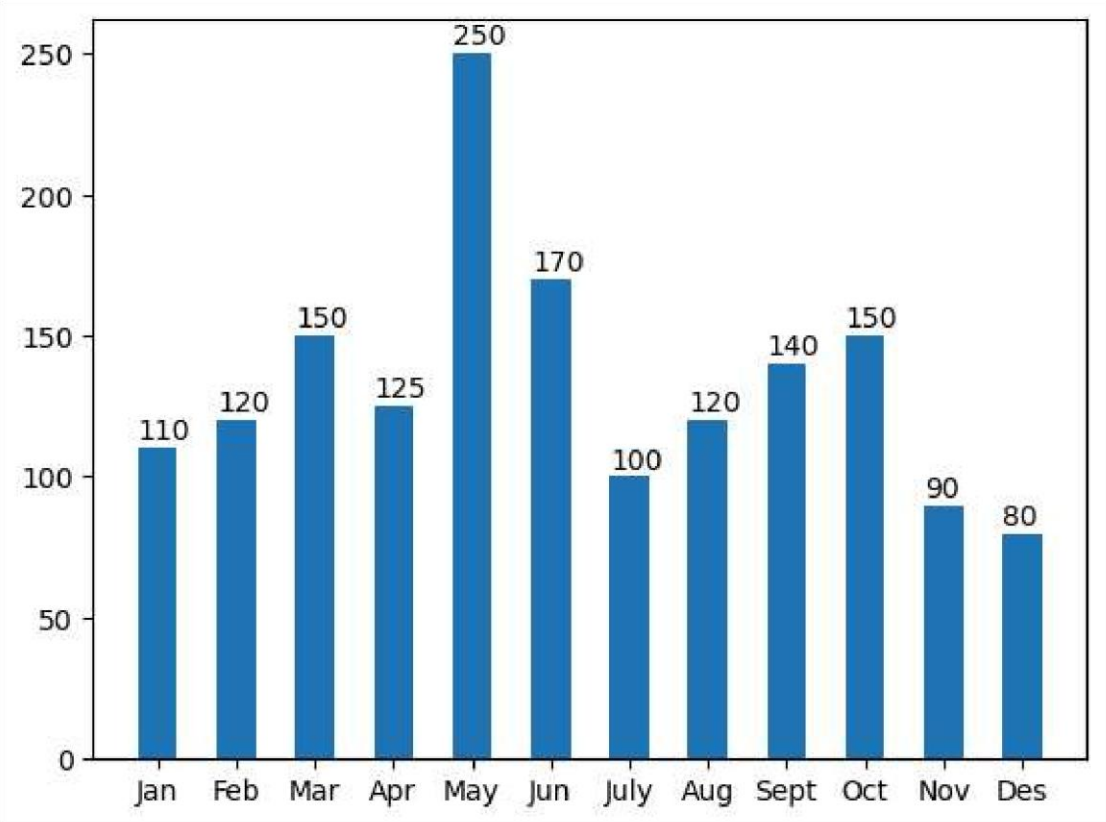

## 07) WAP to demonstrate the use of Histogram.

In [43]:
```
random.seed(14)
age = [random.randint(1,50) for i in range(100)]
plt.hist(age, edgecolor="orange", bins=6,histtype="bar")
plt.show()
```



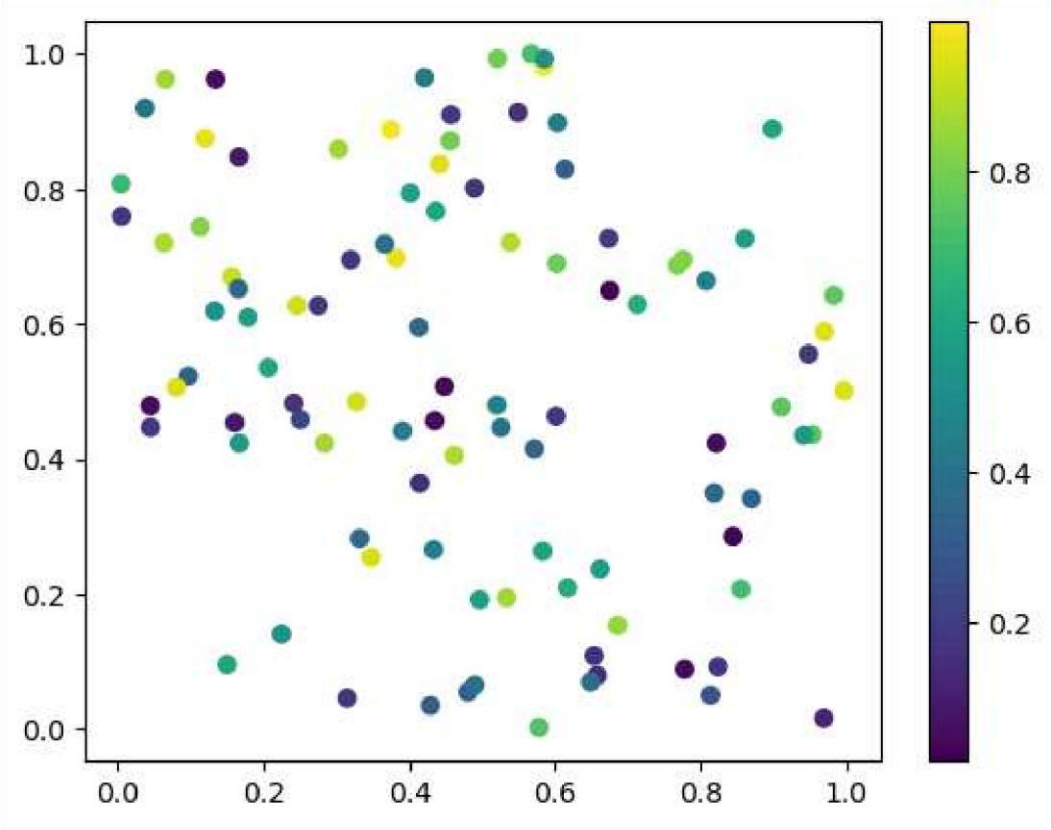

## 08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [57]:
```
mon = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov
visiters = [110, 120, 150, 125, 250, 170, 100, 120, 140, 150, 90, 80]
bars = plt.bar(mon, visiters, width=0.5)
for i in bars:
    yc = i.get_height()
    plt.text(i.get_x(), yc + 3, f"{yc}")
plt.show()
```



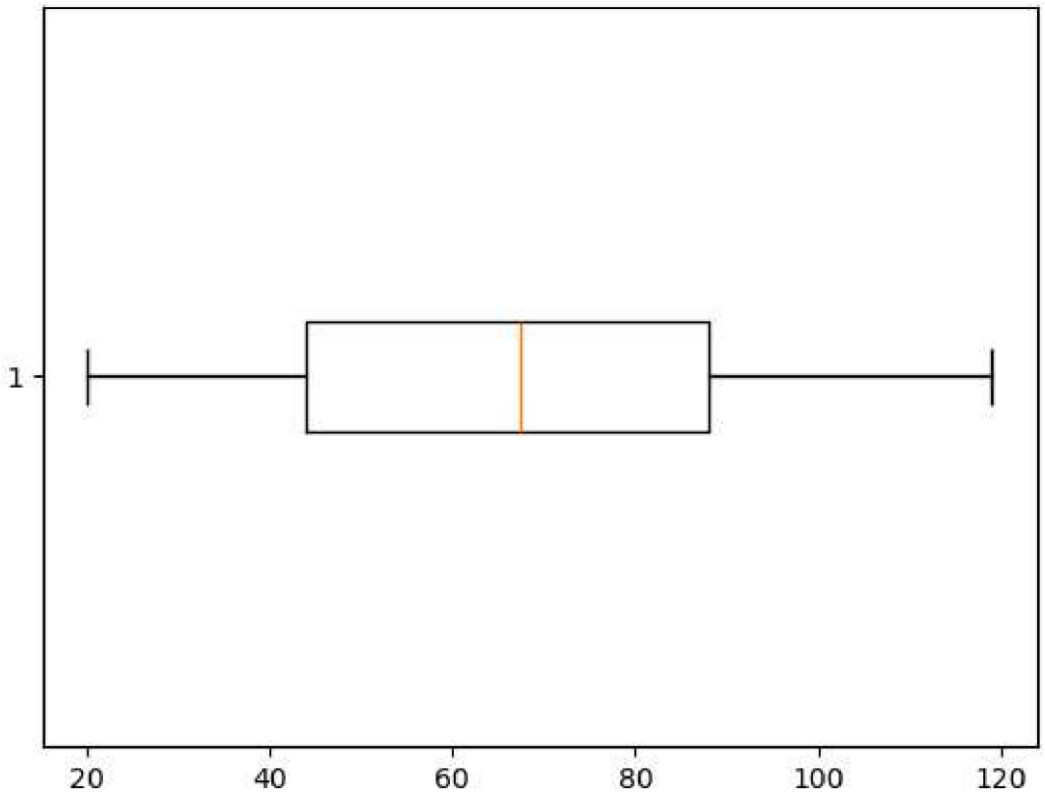

## 09) WAP create a Scatter Plot with several colors in Matplotlib?

In [63]:
```
random.seed(10)
x = [random.random() for i in range(100)]
y = [random.random() for i in range(100)]
z = [random.random() for i in range(100)]
plt.scatter(x,y, c=z,cmap="viridis")
plt.colorbar()
plt.show()
```

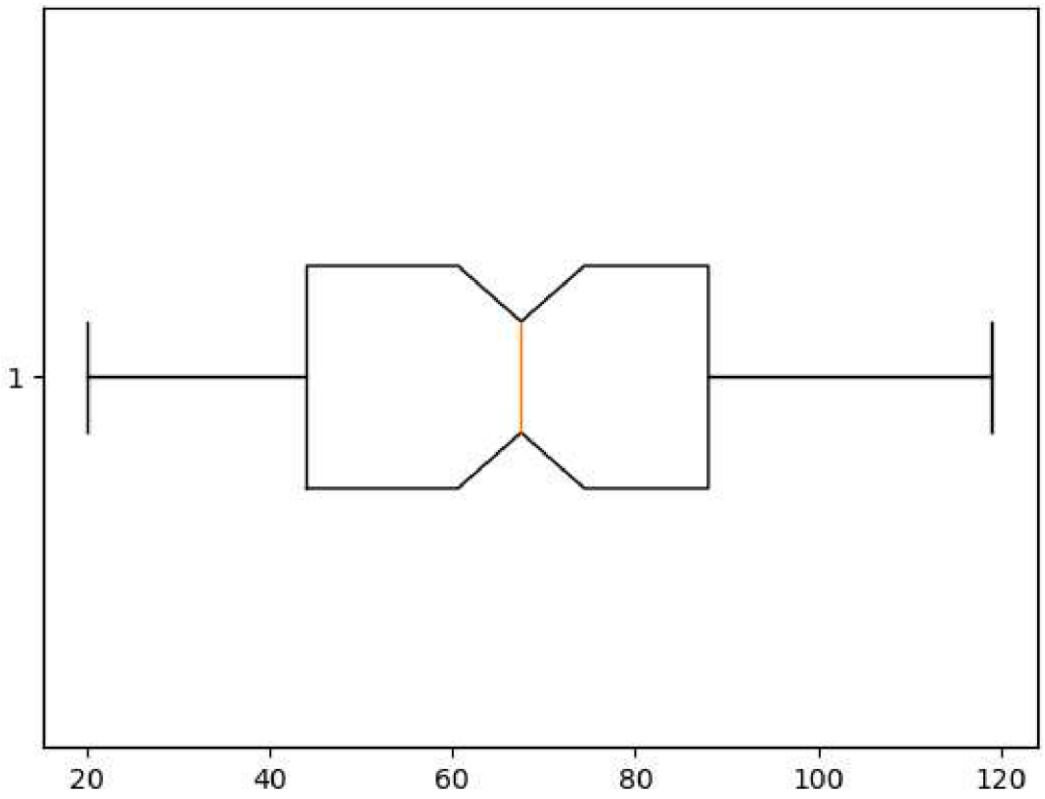



## 10) WAP to create a Box Plot.

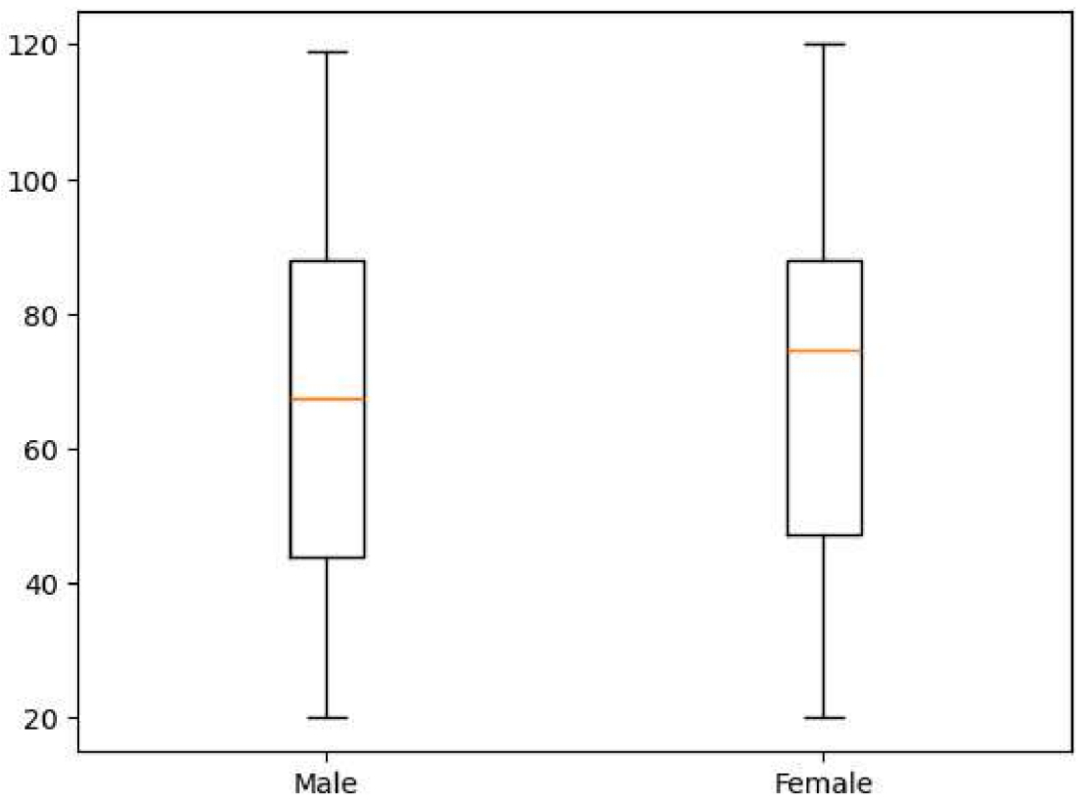
In [65]:
```
random.seed(10)
time = [ random.randint(20,120) for i in range(100)]
plt.boxplot(time, vert=False)
plt.show()
```



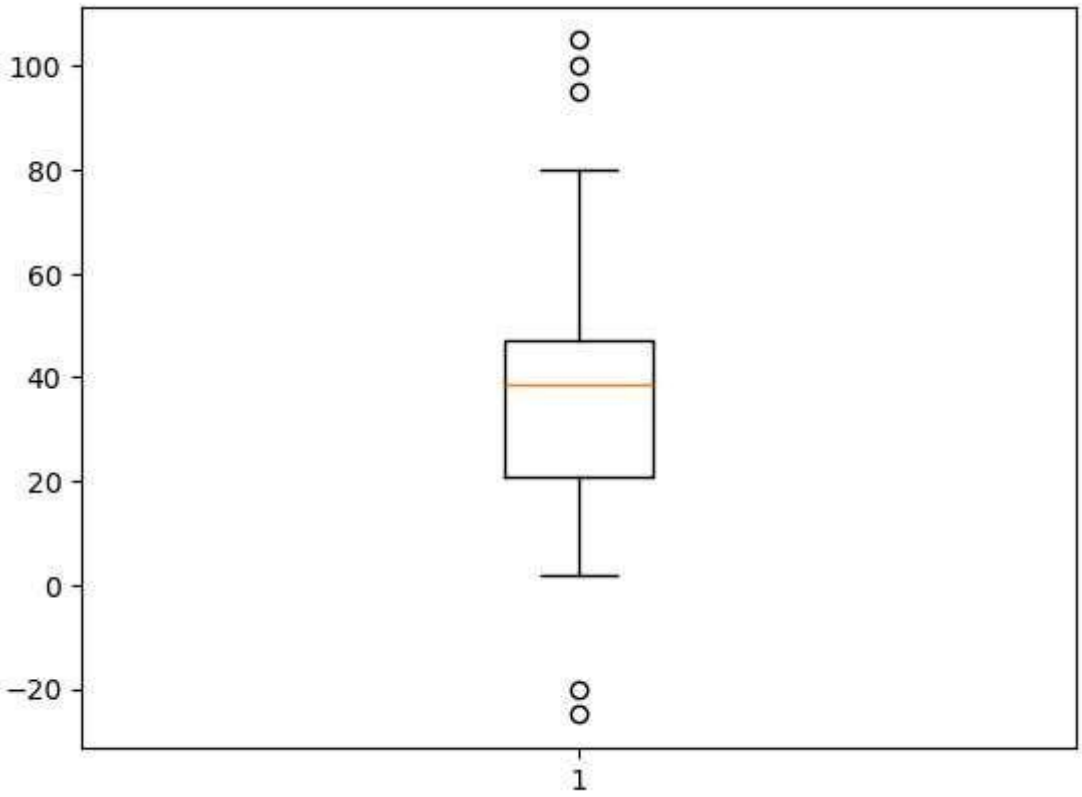

In [67]:
```
random.seed(10)
time = [ random.randint(20,120) for i in range(100)]
plt.boxplot(time, vert=False, notch=True, widths=0.3)
plt.show()
```



In [69]:
```
random.seed(10)
time_male = [ random.randint(20,120) for i in range(100)]
time_female = [ random.randint(20,120) for i in range(100)]
plt.boxplot([time_male,time_female], tick_labels=["Male", "Female"])
plt.show()
```




In [71]:
```
x = [-20,-25,47, 24, 47, 26, 11, 27, 47, 39, 47, 20, 2, 38, 21, 42, 21, 5, 48,70,80
plt.boxplot(x)
plt.show()
```




In [ ]:

In [ ]:

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Python Programming - 201CCS404

# Lab - 13

**Manav Kotecha || 23010101145 || 137**

## 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:
    name = ""
    age = 18
    grade = ""

    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def display(self):
        print(f"{self.name}, {self.age}, {self.grade}")

stu1 = Students("Viraj", 17, "B")

stu1.display()
```

```
Viraj, 17, B
```

## 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]:
```python
class Bank_Account:
    def __init__(self, account_no, user_name, email, account_type, account_balance)
        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):
        self.Account_No = input("Enter Account Number: ")
        self.User_Name = input("Enter User Name: ")
        self.Email = input("Enter Email: ")
        self.Account_Type = input("Enter Account Type (Saving/Checking): ")
        self.Account_Balance = float(input("Enter Account Balance: "))

    def DisplayAccountDetails(self):
        print("\nAccount Details:")
        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:.2f}")

def main():
    bank_account = Bank_Account("", "", "", "", 0.0)

    bank_account.GetAccountDetails()

    bank_account.DisplayAccountDetails()

if __name__ == "__main__":
    main()
```

```
Enter Account Number: 23010101178
Enter User Name: Viraj
Enter Email: 23010101178
Enter Account Type (Saving/Checking): Saving
Enter Account Balance: 100000

Account Details:
Account Number: 23010101178
User Name: Viraj
Email: 23010101178
Account Type: Saving
Account Balance: ₹100000.00
```

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

In [7]:
```python
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

radius = float(input("Enter the radius of the circle: "))
circle = Circle(radius)
print("Area:", round(circle.area(), 2))
print("Perimeter:", round(circle.perimeter(), 2))
```

```
Enter the radius of the circle: 10
Area: 314.16
Perimeter: 62.83
```

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

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

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

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

emp = Employee("Viraj", 17, 50000)
emp.display_info()
print("\n")
emp.update_info(salary=55000)
emp.display_info()
```

```
Name: Viraj
Age: 17
Salary: 50000


Name: Viraj
Age: 17
Salary: 55000
```

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

In [10]:
```python
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited: ₹{amount}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds!")
        else:
            self.balance -= amount
            print(f"Withdrew: ₹{amount}")

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

account = BankAccount()
account.deposit(500)
account.withdraw(200)
account.check_balance()
```

```
Deposited: ₹500
Withdrew: ₹200
Balance: ₹300
```

## 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 [13]:
```python
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        self.items[name] = {'price': price, 'quantity': quantity}

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]
```

```
    def update_item(self, name, price=None, quantity=None):
        if name in self.items:
            if price: self.items[name]['price'] = price
            if quantity: self.items[name]['quantity'] = quantity

    def display_items(self):
        for name, details in self.items.items():
            print(f"{name}: ₹{details['price']}, Quantity: {details['quantity']}")

inventory = Inventory()
inventory.add_item("Laptop", 1200, 5)
inventory.update_item("Laptop", quantity=10)
inventory.display_items()
inventory.remove_item("Laptop")
```

```
Laptop: ₹1200, Quantity: 10
```

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

In [14]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

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

person = Person("Meet", 28)
person.display_info()
```

```
Name: Meet, Age: 28
```

## 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 [17]:
```
class StudentKit:
    principal_name = "toon mausi"

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

    def attendance(self, days_present):
        self.days_present = days_present

    def certificate(self):
```

```
        print(f"Certificate for {self.student_name}: {self.days_present} days prese

student = StudentKit("Viraj")
student.attendance(90)
student.certificate()
```

```
Certificate for Viraj: 90 days present in class.
Signed by: toon mausi
```

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

In [18]:

```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add(self, other):
        total_minutes = (self.hour + other.hour) * 60 + (self.minute + other.minute
        result_hour = total_minutes // 60
        result_minute = total_minutes % 60
        return Time(result_hour, result_minute)

    def display(self):
        print(f"{self.hour} hours and {self.minute} minutes")

time1 = Time(2, 45)
time2 = Time(1, 30)
result_time = time1.add(time2)
result_time.display()
```

```
4 hours and 15 minutes
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Python Programming - 2011CS404

# Lab - 13

## Continued..

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

In [3]:
```
class ractangle :

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

    @staticmethod
    def calculate_area(r1) :
        return r1.length * r1.width

r1 = ractangle(4, 5)
area = ractangle.calculate_area(r1)
print(f'the area of rectangle is : {area}')
```

```
the area of rectangle is : 20
```

## 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 [4]:
```
class square :

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

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

    def displayArea(self) :
        print(f'The area of the square is  : {self.area()}')

s1 = square(4)
s1.displayArea()
```

```
The area of the square is  : 16
```

## 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 [12]:
```
class Rectangle :

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

    @classmethod
    def compare(cls, length, width) :
        return (length != width)

    def calcArea(self) :
        Rectangle.output(self.length * self.width)

    def output(res) :
        print(f'The  area of the Rectangle is : {res}')
```

```
if Rectangle.compare(4, 50) :
    r2 = Rectangle(4, 50)
    r2.calcArea()
else:
    print('THIS IS SQUARE')
```

```
The  area of the Rectangle is : 200
```

## 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 [13]:
```
class Square:
    __side = 0
#     def __inti__(self, side) :
#         self.__side = side

    def get_side(self) :
        return self.__side

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

s1 = Square()
s1.set_side(4)
s1.get_side()
```

Out[13]:
```
4
```

## 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 get_profit(self, profit) :
        self._profit = profit

class Loss:

    def get_loss(self, loss) :
        self._loss = loss

class BalanceSheet(Loss, Profit) :
```

```
    def getBalance(self, balance) :
        self._balance = balance + self._profit + self._loss

    def printBalance(self) :
        print(f'The Current Balance is : {self._balance}')


b1 = BalanceSheet()
b1.get_profit(40000)
b1.get_loss(10000)
b1.getBalance(20000)
b1.printBalance()
```

```
The Current Balance is : 70000
```

## 15) WAP to demonstrate all types of inheritance.

In [16]:
```
# Single Inheritance
class Fruit:
    def taste(self):
        print("Fruit tastes sweet")

class Apple(Fruit):
    def color(self):
        print("Apple is red or green")

# Multiple Inheritance
class Fruit:
    def taste(self):
        print("Fruit tastes sweet")

class Citrus:
    def acidity(self):
        print("Citrus fruits are acidic")

class Orange(Fruit, Citrus):
    def color(self):
        print("Orange is orange")

# Multilevel Inheritance
class Fruit:
    def taste(self):
        print("Fruit tastes sweet")

class Citrus(Fruit):
    def acidity(self):
        print("Citrus fruits are acidic")

class Orange(Citrus):
    def color(self):
        print("Orange is orange")

# Hierarchical Inheritance
class Fruit:
```

```
    def taste(self):
        print("Fruit tastes sweet")

class Apple(Fruit):
    def color(self):
        print("Apple is red or green")

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

# Creating objects and calling methods to demonstrate each inheritance
# Single Inheritance
print("Single Inheritance:")
apple = Apple()
apple.taste()
apple.color()

# Multiple Inheritance
print("\nMultiple Inheritance:")
orange = Orange()
orange.taste()
orange.acidity()
orange.color()

# Multilevel Inheritance
print("\nMultilevel Inheritance:")
orange2 = Orange()
orange2.taste()
orange2.acidity()
orange2.color()

# Hierarchical Inheritance
print("\nHierarchical Inheritance:")
apple2 = Apple()
apple2.taste()
apple2.color()

banana = Banana()
banana.taste()
banana.color()
```

```
Single Inheritance:
Fruit tastes sweet
Apple is red or green

Multiple Inheritance:
Fruit tastes sweet
Citrus fruits are acidic
Orange is orange

Multilevel Inheritance:
Fruit tastes sweet
Citrus fruits are acidic
Orange is orange

Hierarchical Inheritance:
Fruit tastes sweet
Apple is red or green
Fruit tastes sweet
Banana is yellow
```

## 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

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

    def print_details(self) :
        print(f"{self.name} - {self.age} - {self.salary}")
E1 = Employee('Pruthvi', 22, 22002200)
E2 = Employee('Viraj', 22, 22002200)
E3 = Employee('Dhairy', 22, 230023)

E1.print_details()
E2.print_details()
E3.print_details()
```

```
Pruthvi - 22 - 22002200
Viraj - 22 - 22002200
Dhairy - 22 - 230023
```

## 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 [ ]:
```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()
```