Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 1

01) WAP to print “Hello World”

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

```
Hello World
```

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

In [ ]:
```
#Using Input
a=int(input("Enter Number 1:"))
b=int(input("Enter Number 2:"))
print(a+b)

#Without Input
a=5
b=20
print(a+b)
```

```
Enter Number 1: 5
Enter Number 2: 10
15
25
```

03) WAP to check the type of the variable.

In [ ]:
```
a=5
print(type(a))
b="Harsh"
print(type(b))
```

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

04) WAP to calculate simple interest.

In [ ]:
```
#S.I. = (P × R × T)/100
p=int(input("Enter Principal:"))
r=int(input("Enter Rate of Interest in % per annum:"))
t=int(input("Enter Time:"))

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

```
Enter Principal: 5
Enter Rate of Interest in % per annum: 10
Enter Time: 15
7.5
```

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

In [ ]:
```
#Area:pi*r*r
#perimeter:2*pi*r

import math

r=int(input("Enter reduis of circle"))
area=math.pi*r**2
perimeter=2*math.pi*r

print("Area of Circle is:",area)
print("perimeter of Circle is:",perimeter)
```

```
Enter reduis of circle 5
Area of Circle is: 78.53981633974483
perimeter of Circle is: 31.41592653589793
```

06) WAP to calculate area of a triangle.

In [ ]:
```
#Area: ½ (base × height)

base=int(input("Enter base of triangle:"))
height=int(input("Enter height of triangle:"))

area=(1/2)*(base*height)
print(area)
```

```
Enter base of triangle: 5
Enter height of triangle: 10
25.0
```

07) WAP to compute quotient and remainder.

In [ ]:
```
a=int(input("Enter number 1:"))
b=int(input("Enter number 2:"))

quotient=b//a
remainder=a%b
```

```
print(quotient)
print(remainder)
```

```
Enter number 1: 5
Enter number 2: 10
2
5
```

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

In [ ]:
```
#Degree TO Fahrengeit: C × (9/5) + 32
c=int(input("Enter Temrature in Degree:"))
F=c*(9/5)+32
print(F)

#Fahrengeit To Degree: (F - 32) × 5/9
F=int(input("Enter Temrature in Fahrengeit:"))
print(c)
```

```
Enter Temrature in Degree: 5
41.0
Enter Temrature in Fahrengeit: 125
5
```

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

In [ ]:
```
#d=√((x2 − x1)² + (y2 − y1)²)

import math
x2=int(input("Enter Value of x2:"))
x1=int(input("Enter Value of x1:"))
y2=int(input("Enter Value of y2:"))
y1=int(input("Enter Value of y1:"))

d=math.sqrt((x2-x1)**2 + (y2-y1)**2)
print(d)
```

```
Enter Value of x2: 1
Enter Value of x1: 1
Enter Value of y2: 5
Enter Value of y1: 1
4.0
```

10) WAP to print sum of n natural numbers.

In [ ]:
```
n=int(input("Enter Number to be add:"))
a=(n*(n+1))/2
print(a)
```

```
Enter Number to be add: 10
55.0
```

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

In [ ]:
```
n=int(input("Enter Number to be add:"))
a=((n*(n+1))/2)**2
```

```
print(a)
```

```
Enter Number to be add: 5
225.0
```

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

In [ ]:
```
fName=str(input("Enter your FirstName:"))
lName=str(input("Enter your LastName:"))

FullName=fName+  "  " +lName
print(FullName)
```

```
Enter your FirstName: Harsh
Enter your LastName: Khant
Harsh  Khant
```

13) WAP to swap two numbers.

In [ ]:
```
a,b=5,10
a,b=b,a
print("Value Of A:",a)
print("Value Of B:",b)
```

```
Value Of A: 10
Value Of B: 5
```

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

In [ ]:
```
#1 km=1000 m
#1 km=3280.84 feet
#1 km=39370.1 inch
#1 km=100000 centimeter

a=int(input("Enter distance in Kilometer:"))
meter=a*1000
feet=a*3280.84
inches=a*39370.1
centimeter=a*100000

print("meter is:",meter)
print("feet is:",feet)
print("inches is:",inches)
print("centimeter is:",centimeter)
```

```
Enter distance in Kilometer: 5
5000
16404.2
196850.5
500000
```

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

In [ ]:
```
day=str(input("Enter a day:"))
month=str(input("Enter a month:"))
```

```
year=str(input("Enter a year:"))

date=  day + '-' + month + '-' + year
print(date)
```

```
Enter a day: 23
Enter a month: 11
Enter a year: 2024
23-11-2024
```

In [ ]:

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 2

In [ ]:

if..else..

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

In [1]:
```
a=int(input("Enter a number to check positive or negative:"))
if a>0:
    print("positive")
else:
    print("Negative")
```

```
Enter a number to check positive or negative: 5
positive
```

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

In [2]:
```
n=int(input("Enter a number to check number is odd or even:"))
if n%2==0:
    print("Number is even")
else:
    print("Number is odd")
```

```
Enter a number to check number is odd or even: 5
Number is odd
```

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

In [7]:
```
#Using IF

a=int(input("Enter number 1:"))
b=int(input("Enter number 2:"))

if a>b:
    print("A is largest")
else:
    print("B is largest")

#Using Ternary operator
print("A is Largest" if a>b else "B is Largest")
```

```
Enter number 1: 5
Enter number 2: 10
B is largest
B is Largest
```

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

In [10]:
```
a=int(input("Enter Number 1:"))
b=int(input("Enter Number 2:"))
c=int(input("Enter Number 3:"))

if (a>b) and (a>c):
    largest=a
elif (b>a) and (b>c):
    largest=b
else:
    largest=c
    print(largest)
```

```
Enter Number 1: 5
Enter Number 2: 10
Enter Number 3: 15
15
```

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 [16]:
```
a=int(input("Enter a year:"))
if (a%4==0) and (a%100!=0) or (a%400==0) :
    print("Year is LeapYear")
else:
    print("Year is Not Leap Year")
```

```
Enter a year: 2000
```

```
Year is LeapYear
```

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

In [15]:
```
day=int(input("Enter a number of day:"))

if day==0:
    print("Sunday")

elif day==1:
     print("Monday")

elif day==2:
     print("Tuesday")

elif day==3:
     print("Wednesday")

elif day==4:
     print("Thursday")

elif day==5:
     print("Friday")

elif day==6:
     print("Saturday")

else:
    print("Invalid Number:")
```

```
Enter a number of day: 6
Saturday
```

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

In [42]:
```
a=int(input("Enter number 1:"))
b=int(input("Enter number 2:"))
choice=int(input("Enter 1:Add 2:subtraction 3:Multiplication 4:Division"))
if choice==1:
    add=a+b
    print("Answer of addition:",add)
elif choice==2:
    sub=a-b
    print("Answer of subtraction:",sub)
elif choice==3:
    mul=a*b
    print("Answer of multiplictaion:",mul)
elif choice==4:

    if b!=0:
        #we can use // for get answer in integer instend of floating value
        div=a//b
        print("Answer of division:",div)
    else:
        print("Division by zero is not allowed")
```

```
else:
    print("Invalid choice")
```

```
Enter number 1: 5
Enter number 2: 0
Enter 1:Add 2:subtraction 3:Multiplication 4:Division 4
0 can't be divided
```

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 [3]:
```
a=int(input("Enter a marks of subject 1:"))
b=int(input("Enter a marks of subject 2:"))
c=int(input("Enter a marks of subject 3:"))
d=int(input("Enter a marks of subject 4:"))
e=int(input("Enter a marks of subject 5:"))

avg=((a+b+c+d+e)/500)*100

print("Avg is:",avg)

if avg<35:
    print("Fail")

elif avg>35 and avg<45:
    print("Pass Class")

elif avg>45 and avg<60:
    print("Second Class")

elif avg>60 and avg<70:
    print("First Class")

else:
    print("Distinction")
```

```
Enter a marks of subject 1: 80
Enter a marks of subject 2: 75
Enter a marks of subject 3: 60
Enter a marks of subject 4: 80
Enter a marks of subject 5: 100
Avg is: 79.0
Distinction
```

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 [ ]:
```
import math

p = float(input("Enter first side of triangle : "))
q = float(input("Enter second side of triangle : "))
r = float(input("Enter third side of triangle : "))
```

```
if p == q or p == r :
    print("This is an Isosceles Trianle")

elif p == q and p == r :
    print("This is an Equilateral Trinagle")

elif pow(p, 2) == pow(q, 2) + pow(r, 2) or pow(q, 2) == pow(p, 2) + pow(r, 2) or
    print("This is a right angle trinagle")

else:
    print("This is  Scalene Trinagle")
```

In [ ]:

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

In [12]:
```
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}')
```

```
Enter number 1: 10
Enter number 2: 5
Enter number 2: 4
4
```

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 [7]:
```
unit=int(input("Enter a number of unit:"))

if   unit>1 and unit<=50:
    print(unit*2.60)

elif unit>50 and unit<=100:
    print(unit*3.25)

elif unit>100 and unit<=200:
    print(unit*5.26)

elif unit>200:
    print(unit*8.45)
```

```
Enter a number of unit: 50
130.0
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 3

for and while loop

01) WAP to print 1 to 10.

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

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

02) WAP to print 1 to n.

In [8]:
```
n=int(input("Enter number to print:"))

for x in range(n):
    print(x+1)
```

```
Enter number to print: 5
1
2
3
4
5
```

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

In [5]:
```
n=int(input("Enter number to print:"))

for x in range(0,n,2):
    print(x+1)
```

```
Enter number to print: 5
1
3
5
```

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

In [13]:
```
n1=int(input("Enter number to print1:"))
n2=int(input("Enter number to print2:"))

for i in range(n1,n2+1):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
Enter number to print1: 12
Enter number to print2: 15
14
```

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

In [20]:
```
n=int(input("Enter number to print:"))
sum=0
for x in range(1,n+1):
    sum=sum+x

print(sum)
```

```
Enter number to print: 3
6
```

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

In [21]:
```
n=int(input("Enter number to print:"))
sum=0
for x in range(1,n+1):
    sum=sum+(x*x)

print(sum)
```

```
Enter number to print: 5
55
```

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

In [23]:
```
n=int(input("Enter number to print:"))
sum=0
min=0

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

```
    sum=sum+i

for i in range(2,n+1,2):
    min=min+i

print(sum-min)
```

```
Enter number to print: 10
-5
```

08) WAP to print multiplication table of given number.

In [25]:
```
n=int(input("Enter number to print:"))

for i in range(1,11):
     print(f"{n} * {i} = {n*i}")
```

```
Enter number to print: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

09) WAP to find factorial of the given number.

In [26]:
```
n=int(input("Enter number to print:"))
fact=1
for i in range(1,n+1):
    fact*=i
print(fact)
```

```
Enter number to print: 3
6
```

10) WAP to find factors of the given number.

In [29]:
```
n=int(input("Enter number to print:"))
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

```
Enter number to print: 5
1
5
```

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

In [32]:
```
n=int(input("Enter number to print:"))
count=0
for i in range(1,n+1):
    if(n%i==0):
```

```
        count=count+1
if count==2:
    print("Number is prime")
else:
    print("Number is not prime")
```

```
Enter number to print: 3
Number is prime
```

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

In [34]:
```
n=int(input("Enter number to print:"))
count=0
while n!=0:
    count +=n%10
    n//=10
print(count)
```

```
Enter number to print: 166
13
```

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

In [10]:
```
n = int(input("Enter number to check: "))
con = str(n)
reverse_string = ''

for x in con:
    reverse_string = x + reverse_string

if con == reverse_string:
    print(f"{n} is a palindrome.")
else:
    print(f"{n} is not a palindrome.")
```

```
Enter number to check:  121
121 is a palindrome.
```

14) WAP to print GCD of given two numbers.

In [12]:
```
import math

a=int(input("Enter number to print1:"))
b=int(input("Enter number to print2:"))

while b!=0:
    a,b=b,a%b
print(f"Gcd of the number is:{a} ")

#second way use inbuilt method of math
ans=math.gcd(a,b)
print(ans)
```

```
Enter number to print1: 48
Enter number to print2: 18
Gcd of the number is:6
6
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 4

String

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

In [2]:
```
a=input("Enter your string:")
if a==a[::-1]:
    print("String is palindrome")
else:
    print("String is not palindrome")
```

```
Enter your string: aabaa
String is palindrome
```

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

In [11]:
```
a=input("Enter a String:")
li=a.split(" ")
li.reverse()
s2=' '.join(li)
print(s2)
```

```
Enter a String: i am
am i
```

03) WAP to remove ith character from given string.

In [17]:
```
a=input("Enter a String:")
i=int(input("Enter index of character to remove:"))

a=a[0:i]+a[i+1::1]
```

```
print(a)
```

```
Enter a String: harsh
Enter index of character to remove: 1
arsh
```

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

In [18]:
```
s=input("Enter your String:")
count=0
for i in s:
    count=count+1;
print("Count is:",count)
```

```
Enter your String: harsh
Count is: 5
```

05) WAP to print even length word in string.

In [23]:
```
s=input("Enter your String:")
st=s.split()
for w in st:
    if len(w)%2==0:
        print(w)
```

```
Enter your String: i am harsh
am
```

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

In [25]:
```
s=input("Enter your String:")
count=0
stg=['a','e','i','o','u']
for i in s:
    if i in stg:
        count=count+1
        print(count)
```

```
Enter your String: harsh
1
```

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

In [38]:
```
s = input("Enter your String: ")
result = []

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

print(' '.join(result))
```

```
Enter your String:  i am hasrh
```

```
I AM HasrH
```

08) WAP to convert given array to string.

In [8]:
```
li=['a','b','c']
j=''.join(li)

print(j)
print(type(j))
```

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

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

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

In [13]:
```
password=input("Enter your password:")
confirm_password=input("Enter your confirm password")

if password==confirm_password:
    print("Password is same")

elif password.upper()==confirm_password.upper() :
    print("In password Case Problem")

else:
    print("Error")
```

```
Enter your password: Harsh1234
Enter your confirm password Harsh123
Error
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [28]:
```
card_no=1234567890123456
stg=str(card_no)

output="**** **** **** ****"+' '+ stg[-4:]
print(output)
```

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

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

s1 = decimal and s2 = medical are Anagram

In [7]:
```
s1 = input('Enter String 1: ')
s2 = input('Enter String 2: ')
```

```
isAnagram = True

if len(s1) == len(s2):
    for i in s1:
        if i not in s2:
            isAnagram = False
            break
    if isAnagram:
        print('String are anagram')
    else:
        print('String are not anagram')
else:
    print('String are not anagram')
```

```
Enter String 1:  decimal
Enter String 2:  medical
String are anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [14]:
```
stg=input("Enter your input String")
upper=''
lower=''
for i in stg:
    if i.islower():
        lower=lower+i
    elif i.isupper():
        upper=upper+i
output=lower+upper
print(output)
```

```
Enter your input String  EHlsarwiwhtwMV
lsarwiwhtwEHMV
```

In [ ]:
```

```

List

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 5

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

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

```
15
```

02) WAP to find largest element in a List.

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

Out[4]: 5

03) WAP to find the length of a List.

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

Out[5]: 5

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

In [6]:
```
li=[1,2,3,4,5]
li[0],li[-1]=li[-1],li[0]
print(li)
```

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

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

In [8]:
```
li=[1,2,3,4,5,6,7,8,9]
l=li[len(li)//2:]+li[:len(li)//2]
print(l)
```

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

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

In [19]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)

print(li)
a=int(input("Enter a index  number:"))
b=int(input("Enter a index number:"))
li[a-1],li[b-1]=li[b-1],li[a-1]
print(li)
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 15
Enter a number: 016
Enter a number: -1
[5, 10, 15, 16]
Enter a index  number: 1
Enter a index number: 2
[10, 5, 15, 16]
```

07) WAP to reverse the list entered by user.

In [18]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
print(li[::-1])
```

```
Enter a number: 5
Enter a number: 0
Enter a number: 5
Enter a number: 6
Enter a number: -1
[6, 5, 0, 5]
```

08) WAP to print even numbers in a list.

In [20]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
for i in li:
    if i%2==0:
        print(i)
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 15
Enter a number: -1
10
```

09) WAP to count unique items in a list.

In [24]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
li2=[]
for i in li:
    if li2.count(i)==0:
        li2.append(i)
print(len(li2))
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 5
Enter a number: -1
2
```

10) WAP to copy a list.

In [26]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
l2=[]
l2=li.copy()
print(l2)
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 5
Enter a number: -1
[5, 10, 5]
```

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

In [27]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
a=int(input("Enter a strting index  number:"))
b=int(input("Enter a ending index number:"))
for i in range(a,b+1):
    if li[i]%2==1:
        print(li[i])
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 15
Enter a number: 20
Enter a number: -1
Enter a strting index  number: 2
Enter a ending index number: 3
15
```

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

In [33]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
a=set(li)
for i in a:
    print('count of',i,':',li.count(i))
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 6
Enter a number: 7
Enter a number: 6
Enter a number: 7
Enter a number: 5
Enter a number: -1
count of 10 : 1
count of 5 : 2
count of 6 : 2
count of 7 : 2
```

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

In [38]:
```
li=[]
while True:
    i=int(input("Enter a number:"))
    if i<0:
        break
    li.append(i)
```

```
li2=li.copy()
li2.remove(max(li2))
print(max(li2))
```

```
Enter a number: 5
Enter a number: 10
Enter a number: 15
Enter a number: -1
10
```

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

In [7]:
```
k=int(input("Enter Frequency: "))
li=[]
l=[10,20,30,40,20,20,30,40,20,30]
for i in l:
 if l.count(i)>k and li.count(i)==0:
     li.append(i)
li
```

```
Enter Frequency:  6
```

Out[7]:
```
[]
```

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

In [21]:
```
for i in range(0,10):
    sq=i**2
    print(sq)

#using list comprehension
sq=[i**2 for i in range(0,10)]
print(sq)
```

```
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 [17]:
```
fruits = input("Enter fruits separated by spaces: ").split()

ans = [f for f in fruits if f[0].lower() == 'b']

print("Fruits starting with 'b':", ans)
```

```
Enter fruits separated by spaces:  banana apple pineapple berry blackberry
```

```
Fruits starting with 'b': ['banana', 'berry', 'blackberry']
```

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

In [22]:
```
li1 = [1, 2, 3, 4, 5]
li2 = [4, 5, 6, 7, 8]

common_elements = [i for i in li1 if i in li2]

print("Common elements:", common_elements)
```

```
Common elements: [4, 5]
```

In [ ]:
```

```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 6

Tuple

01) WAP to find sum of tuple elements.

In [1]:
```
t1=(1,2,3,4,5)
print(sum(t1))
```

```
15
```

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

In [2]:
```
t=(1,2,3,4,5)
k=int(input('Enter a Value of K:'))
temp=list(t)
temp.sort()
print('Minimum',k,'Elements:',tuple(temp[:k]))
print('Maximum',k,'Elements:',tuple(temp[-k:]))
```

```
Enter a Value of K: 2
Minimum 2 Elements: (1, 2)
Maximum 2 Elements: (4, 5)
```

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

In [4]:
```
li=[(1,2,3),(6,36,12),(5,565)]
temp=[]
k=int(input('Enter a Value of K:'))
for i in li:
    isValid=True
    for j in i:
```

```
        if j%k!=0:
            isValid=False
    if isValid:
        temp.append(i)
print(temp)
```

```
Enter a Value of K: 2
[(6, 36, 12)]
```

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

In [3]:
```
li=[1,3,65,23,7,3,6,2,7,3,5,8]
t=[(i,i**3) for i in li]
print(t)
```

```
[(1, 1), (3, 27), (65, 274625), (23, 12167), (7, 343), (3, 27), (6, 216), (2, 8),
(7, 343), (3, 27), (5, 125), (8, 512)]
```

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

In [7]:
```
t1 = [(-1,-2),(-3,4),(5,-6)]
ans = []
for i in range(len(t1)) :
    for j in range(len(t1[i])) :
        if t1[i][j] >= 0 :
            ans.append(t1[i][j])

tuple(ans)
```

Out[7]:
```
(4, 5)
```

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

In [8]:
```
t1 = []
a = [1,2,3]
t1.append(a)
print(tuple(t1))

t2 = []
a = [1,2,3]
t2.append(tuple(a))
print(t2)
```

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

07) WAP to remove tuples of length K.

In [10]:
```
li = [(1, 2), (3, 4, 5), (6, 7), (8, 9, 10)]
K = 2

result = []

for i in li:
```

```
    length = 0
    for a in i:
        length += 1
    if length != K:
        result.append(i)

print("Original list:", li)
print("list:", result)
```

```
Original list: [(1, 2), (3, 4, 5), (6, 7), (8, 9, 10)]
list: [(3, 4, 5), (8, 9, 10)]
```

08) WAP to remove duplicates from tuple.

In [11]:
```
t=(1,2,3,4,5,5)
c=list(t)
r=set(c)
s=tuple(r)
print(s)
```

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

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

In [20]:
```
t = (1, 2, 3, 4, 5)
l = list(t)
mu = [l[i] * l[i+1] for i in range(len(l)-1)]
tp = tuple(mu)
print(tp)
```

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

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

In [21]:
```
t = (1, 2, 3, 4, 5)
isU=True
for i in t:
    if t.count(i)!=1:
        isU=False
if isU:
    print('distinct')
else:
    print('not distinct')
```

```
distinct
```

In [ ]:

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 7

Set & Dictionary

01) WAP to iterate over a set.

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

```
1
2
3
4
5
```

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

In [4]:
```
a={1,2,3,4,5}
type(a)

b=list(a)
type(b)

t=tuple(a)
type(t)

s=str(a)
type(s)
```

Out[4]: str

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

In [5]:
```
s={1,2,3,4,5}
print(max(s))
print(min(s))
```

```
5
1
```

04) WAP to perform union of two sets.

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

u=s.union(e)
print(u)
```

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

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

In [9]:
```
s=set([1,2,3,4,5])
e=set([3,4,5,6])

i=s.intersection(e)
j=set(i)
print(j)
```

```
{3, 4, 5}
```

06) WAP to remove duplicates from list.

In [12]:
```
s=set([1,2,3,4,5,5])
print(s)
```

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

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

In [14]:
```
s='how are you you !'
b=s.split(' ')
a=set(b)
print(a)
```

```
{'!', 'how', 'you', 'are'}
```

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

In [16]:
```
s=set([1,2,3,4,5])
e=set([3,4,5,6])

e=e.symmetric_difference(s)
print(e)
```

```
{1, 2, 6}
```

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

In [22]:
```
st1=input('Enter your String')
st2=input('Enter your String')
s1 = ''.join(st1.split()).lower()
s2 = ''.join(st2.split()).lower()
if(sorted(s1)==sorted(s2)):
    print('Anagram')
else:
    print('Not Anagram')
```

```
Enter your String pub
Enter your String bup
Anagram
```

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

In [23]:
```
s=set([1,2,3,4,5])
e=set([3,4,5,6])
q=set([3,4,5,6])

b=s&e&q
print(b)
```

```
{3, 4, 5}
```

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

In [26]:
```
st1=input('Enter your String')
v={'a','e','i','o','u'}
count=0
for i in st1:
    if v & set(i.lower()):
         count=count+1
print(count)
```

```
Enter your String Harsh
1
```

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

In [36]:
```
st1 = input('Enter your String: ')
bset = {'0', '1'}
s=set(st1)

#using subset concept

if s <= bset :
    print('String is Binary')
else:
    print('String is not Binary')
```

```
Enter your String:  11
```

```
String is Binary
```

13) WAP to sort dictionary by key or value.

In [43]:
```
tdict = {"1": "ABC","2": "DEF","3":6}
a=list(tdict.keys())
b=list(tdict.values())
a.sort()
d = {i: tdict[i] for i in a}
print(d)
```

```
{'1': 'ABC', '2': 'DEF', '3': 6}
```

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

In [6]:
```
udict = {}

n = int(input('how many items to add in the dictionary:'))

for i in range(n):
    key = input(f"Enter key {i + 1}: ")
    value = int(input(f"Enter numeric value for '{key}': "))
    udict[key] = value

print("The dictionary is:", udict)

tsum = sum(udict.values())
print("The sum of all values is:", tsum)
```

```
how many items to add in the dictionary: 2
Enter key 1:  harsh
Enter numeric value for 'harsh':  7
Enter key 2:  HARSH
Enter numeric value for 'HARSH':  3
The dictionary is: {'harsh': 7, 'HARSH': 3}
The sum of all values is: 10
```

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 [17]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2, 'd' : 6}
e = dict1.keys()
t = 'd'

print(e)

if t in e:
    p = dict1[t]
    print('Key found:',p)
else:
    print('Key not Found')
```

```
dict_keys(['a', 'c', 'e', 'd'])
Key found: 6
```

In [ ]:

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 8

User Defined Function

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

In [31]:
```
def BMI(weight,height):
    bmi=weight/(height**2);
    return bmi

BMI(56,1.80)
```

Out[31]: 17.28395061728395

02) Write a function that add first n numbers.

In [41]:
```
def add(n):
    sum=0
    for i in range(1,n+1):
        sum=sum+i
    return sum

add(5)
```

Out[41]: 15

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

In [61]:
```
def prime(a):
    if(a==0 or a==1):
```

```
        return 0;
    elif(a>1):
        for i in range(2,a):
            if(a%i==0):
                return 0
        else:
            return 1

prime(43)
```

Out[61]: 1

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

In [68]:
```
li=[]
def Primeno(a,b):
    for i in range(a,b):
        if(prime(i)):
            li.append(i)
Primeno(1,10)
print(li)
```

```
[2, 3, 5, 7]
```

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

In [91]:
```
st=input("Enter a string:")
def palindrome(st):
    if(st[::-1]==st):
        return True
    else:
        return False
palindrome(st)
```

```
Enter a string: lol
```

Out[91]: True

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

In [93]:
```
li=[1,2,3,4,5,6,7,8,9]
def ans():
    Sum=sum(li)
    return Sum
ans()
```

Out[93]: 45

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

In [107…
```
li=[(0,1,2),(4,5,6),(7,8,9)]
def listsum():
```

```
    sum=0
    for i,j,k in li:
            sum=sum+i
    print(sum)
listsum()
```

```
11
```

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

In [157…

```
def fibonacci(a,b,n):
    if n==1:
        return a
    return fibonacci(b,a+b,n-1)

print(fibonacci(0,1,12))
```

```
89
```

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

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

In [31]:

```
dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
def nameofstudent(n):
    a=dict1.get(n)
    return a

result=nameofstudent(103)
print(result)
```

```
Jay
```

10) Write a function to get the sum of the scores ending with zero.

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

Ans = 200 + 300 + 100 = 600

In [17]:

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

result=sumofScores(scores)
print(result)
```

```
600
```

11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

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

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

In [39]:
```
ini_dict={'a': 10, 'b':20, 'c':30, 'd':40}
def invertDictonary(ini_dict):
    inv_dict = dict(zip(ini_dict.values(), ini_dict.keys()))
    return inv_dict

invertDictonary(ini_dict)
```

Out[39]:
```
{10: 'a', 20: 'b', 30: 'c', 40: 'd'}
```

12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

In [53]:
```
string="the quick brown fox jumps over the lazy dog"

def pangram(string):
    string=string.replace(" ","")
    string=string.lower()
    x=list(set(string))
    x.sort()
    x="".join(x)
    alphabets="abcdefghijklmnopqrstuvwxyz"
    if(x==alphabets):
        print("The string is a pangram")
    else:
        print("The string is not a pangram")

pangram(string)
```

```
The string is a pangram
```

13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

In [65]:
```
s1 = "AbcDEfgh"
def countUPLW(s1):
    lower=0
    upper=0
    for i in s1:
        if i.islower():
            lower+=1
        else:
            upper+=1
    print(f"Lower case are:{lower}")
    print(f"Upper case are:{upper}")
```

```
countUPLW(s1)
```

```
Lower case are:5
Upper case are:3
```

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

In [67]:
```
min_number = lambda a, b : min(a,b)

print(min_number(5, 8))
```

```
5
```

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

In [3]:
```
students=['Alexander','Benjamin','Jonathan','Malay','Meet','Dhairya']
def myFunc(x):
  if len(x) > 7:
    return True
  else:
    return False

names = filter(myFunc, students)

for x in names:
  print(x)
```

```
Alexander
Benjamin
Jonathan
```

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

In [62]:
```
students = ['alexander', 'benjamin', 'jonathan', 'malay', 'meet', 'dhairya']

def uppercaseusingMap(names):
    return list(map(str.capitalize, names))

result = uppercaseusingMap(students)
print(result)
```

```
['Alexander', 'Benjamin', 'Jonathan', 'Malay', 'Meet', 'Dhairya']
```

17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)
5. Keyword-Only & Positional Only Arguments

In [92]:
```
# Positional Arguments
print("Positional Arguments::")
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)

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

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

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

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

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

# Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwar
print("Variable Legngth Positional(*args) & variable length Keyword Arguments (*
# *args
def my_function(*kids):
  print("*args:The youngest child is " + kids[2])

my_function("alexander", "benjamin", "Jonathan")
#**kwargs
def my_function(**kid):
  print("**kargs:His last name is " + kid["lname"])

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

# Keyword-Only & Positional Only Arguments
print("Keyword-Only & Positional Only Arguments::")
```

```
Positional Arguments::
Hi, I am Prince
My age is  20

Keyword Arguments::
The youngest child is Jonathan

Default Arguments::
I am from India
I am from Australia

Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwarg
s)::
*args:The youngest child is Jonathan
**kargs:His last name is benjamin

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

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

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 [11]:
```
f = open("newFile.txt", "r")
print(f.read())
f.close()

f = open("newFile.txt", "r")
print(f.readline())
f.close()

f = open("newFile.txt", "r")
print(f.readlines())
f.close()
```

```
Hello World!!!!!!!!!!
Good Mornings........
Hello World!!!!!!!!!!

['Hello World!!!!!!!!!!\n', 'Good Mornings........']
```

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

In [49]:
```
f = open("new.txt","w")
f.write("Hi Hello from python")
f.close()
```

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

In [31]:
```
f = open("newFile.txt", "r")
for i in range(5):
    print(f.readline())
f.close()
```

```
Hello World!!!!!!!!!!

Good Mornings........

1

2

3
```

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

In [55]:
```
f = open("newFile.txt", "r")
words = f.read().split()
print (max(words, key=len))
f.close()
```

```
Mornings........
```

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

In [51]:
```
#No.of lines
f = open("newFile.txt","r")
lines=len(f.readlines())
print(lines)
f.close()

#No. of words
f = open("newFile.txt","r")
lines=len(f.readline())
print(lines)
f.close()

#No. of characters
f = open("newFile.txt","r")
lines=len(f.read())
print(lines)
f.close()
```

```
9
22
57
```

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

In [59]:
```
f1 = open("newFile.txt","r")
f2 = open("newFile2.text","w")
for i in f1:
    f2.write(i)
f1.close()
f2.close()
```

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

In [67]:
```
import os
sz = os.path.getsize("newFile.txt")
print(sz)
```

```
57
```

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

In [71]:
```
def frequency(file_content,specific_word):
    # Method-1
    count = file_content.count(specific_word)
    # Method-2
    # count = 0
    # for i in file_content:
    #     if i == specific_word:
    #         count += 1
    return count

fp = open("newFile.txt","r")
data = fp.read()
print(frequency(data.split(),'Hello'))
fp.close()
```

```
1
```

09) WAP to get the score of five subjects from the user, store them in a file. Fetch those marks and find the highest score.

In [5]:
```
fp = open("newFile.txt", "w+")
for i in range(1, 6):
    mark = input(f"Enter marks of subject {i}: ")
    fp.write(mark + "\n")
fp.seek(0)
l1 = [int(mark.strip()) for mark in fp.readlines()]
print(f"The highest score is: {max(l1)}")
fp.close()
```

```
Enter marks of subject 1:  5
Enter marks of subject 2:  10
Enter marks of subject 3:  15
Enter marks of subject 4:  20
Enter marks of subject 5:  25
The highest score is: 25
```

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

(Note: each number should be in new line)

In [3]:
```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_numbers(n, filename):
    count = 0
    num = 2
    with open(filename, 'w') as fp:
        while count < n:
            if is_prime(num):
                fp.write(f"{num}\n")
                count += 1
            num += 1

prime_numbers(100, "primenumbers.txt")
```

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

In [13]:
```
f1 = open("newFile.txt", "r")
f2 = open("new.txt", "r")
fp = open("mergedFile.txt", "w")
fp.write(f1.read())
fp.write(f2.read())
f1.close()
f2.close()
fp.close()

print("Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.
```

```
Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.txt'
```

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

In [15]:
```
f1 = open("mergedFile.txt", "r")
content = f1.read()
updated_content = content.replace('Hi', 'bye')
new_file = open("updated_content.txt", "w")
new_file.write(updated_content)
f1.close()
new_file.close()

print("The word replacement has been done and saved to 'updated_content.txt'")
```

```
The word replacement has been done and saved to 'updated_content.txt'
```

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

In [3]:
```
with open('newFile.txt', 'w') as fp:
    fp.write("Hello, this is a sample file for demonstrating tell() and seek()."

with open('newFile.txt', 'r') as fp:
    # Case 1: Tell the current position from the beginning
    print("Case 1: Current position from beginning (before reading):", fp.tell()
    print("Reading first 5 characters:")
    print(fp.read(5))  # Read first 5 characters
    print("Position after reading 5 characters:", fp.tell())

    # Case 2: Seek from the beginning (SEEK_SET)
    fp.seek(0, 0)  # Move the pointer to the beginning
    print("\nCase 2: Seek from the beginning to position 0:", fp.tell())
```

```
Case 1: Current position from beginning (before reading): 0
Reading first 5 characters:
Hello
Position after reading 5 characters: 5

Case 2: Seek from the beginning to position 0: 0
```

In [ ]:

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

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 [14]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # ValueError
    value = int("not_a_number")

    # TypeError
    result = "string" + 10

#handling using seperate except block
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
except ValueError:
    print("Error: Invalid value provided.")
except TypeError:
    print("Error: Type mismatch encountered.")

try:
    # ZeroDivisionError
    result = 10 / 0
```

```
    # ValueError
    value = int("not_a_number")

    # TypeError
    result = "string" + 10

#handling using single except block
except (ZeroDivisionError, ValueError, TypeError) as e:
    print(f"Error: {str(e)}")
```

```
Error: Cannot divide by zero.
Error: division by zero
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [42]:
```
#Index Error
try:
    l1=[1,2,3,4,5,6]
    print(l1[6])
except IndexError as e:
    print(f'Index Error:{str(e)}')

#KeyError
try:
    d1={'a':1,'b':2,'c':3}
    print(d1['d'])
except KeyError as e:
    print(f'KeyError: {str(e)}')
```

```
Index Error:list index out of range
KeyError: 'd'
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [50]:
```
#FileNotFoundError
try:
    fp=open("a.txt", "r")
    fp.read()
except FileNotFoundError as e:
    print(f'FileNotFoundError:{str(e)}')

#ModuleNotFoundError
try:
    import a

except ModuleNotFoundError as e:
    print(f'ModuleNotFoundError:{str(e)}')
```

```
FileNotFoundError:[Errno 2] No such file or directory: 'a.txt'
ModuleNotFoundError:No module named 'a'
```

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

In [63]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # IndexError
    # my_list = [1, 2, 3]
    # print(my_list[5])

    # ValueError
    # value = int("not_a_number")

    # KeyError
    # my_dict = {'a': 1}
    # print(my_dict['b'])

    # FileNotFoundError
    # with open("non_existent_file.txt", "r") as file:
    #     content = file.read()

    # ModuleNotFoundError
    # import non_existent_module

except Exception as e:
    print(f"An error occurred: {str(e)}")
```

```
An error occurred: division by zero
```

05) WAP to demonstrate else and finally block.

In [57]:
```
try:
    result = 10 / 2
    print(result)

except ZeroDivisionError as e:
    print(f"An error occurred: {str(e)}")

else:
    print("he try block executed successfully.")

finally:
    print("This block is always executed")
```

```
5.0
he try block executed successfully.
This block is always 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 [75]:
```
grades_input = input("Enter a list of grades separated by commas: ")

try:
    grades = [int(grade.strip()) for grade in grades_input.split(',')]
    print("Grades:", grades)

except ValueError:
    print("Error: Some of the values you entered cannot be converted to integers
```

```
Enter a list of grades separated by commas:  10,20.3.50,60
Error: Some of the values you entered cannot be converted to integers. Please ent
er valid numbers.
```

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

In [65]:
```
def divide(a, b):
    try:
        result = a / b

    except ZeroDivisionError:
        return "Error: Cannot divide by zero."

    else:
        return result

a = float(input("Enter the a: "))
b = float(input("Enter the b: "))

result = divide(a, b)
print(f"Result: {result}")
```

```
Enter the a:  5
Enter the b:  0
Result: Error: Cannot divide by zero.
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

In [71]:
```
def ageInvalid():
    try:
        age = int(input("Enter your age: "))
        if age < 18:
            raise ValueError("Enter Valid Age")
        else:
            print(f"Your age is {age}")
    except ValueError as e:
        print(f"Error: {str(e)}")

ageInvalid()
```

```
Enter your age:  5
Error: Enter Valid Age
```

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

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

otherwise print the given username.

In [81]:
```
class InvalidUsername(Exception):
    pass
def custom_exception():
    try:
        name=input("Enter your name::")
        if(len(name)<5 or len(name)>15):
            raise InvalidUsername("Username must be between 5 and 15 characters
    except InvalidUsername as e:
        print(f"Error:{str(e)}")
custom_exception()
```

```
Enter your name:: raj
Error:Username must be between 5 and 15 characters long
```

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

if the given number is negative.

otherwise print the square root of the given number.

In [79]:
```
import math
class NegativeNumberError(Exception):
    pass
def calculate_square_root():
    try:
        number = float(input("Enter a number to calculate its square root: "))
        if number < 0:
            raise NegativeNumberError("Cannot calculate the square root of a neg
        sqrt_result = math.sqrt(number)
        print(f"The square root of {number} is: {sqrt_result}")
    except NegativeNumberError as e:
        print(f"Error: {str(e)}")
    except ValueError:
        print("Error: Please enter a valid number.")
calculate_square_root()
```

```
Enter a number to calculate its square root:  25
The square root of 25.0 is: 5.0
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

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 [1]:
```
from another import add,multiply,divide,subtract
print(add(5,6))
print(subtract(10,5))
print(multiply(5,6))
print(divide(4,2))
```

```
11
5
30
2.0
```

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

In [164…
```
import random
a='Hello World'
print(random.choice(a))
```

```
o
```

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

In [32]:
```
li=[1,2,3,4,5,6,7,8,9]
print(random.choice(li))
```

```
5
```

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

In [90]:
```
random.seed(2)

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

print(roll_dice())
```

```
1
```

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

In [100…
```
def get_random_integer():
    count = 0
    l1 = []
    while count != 3:
        random_number = random.randint(100, 999)
        if random_number % 5 == 0:
            l1.append(random_number)
            count += 1
    return l1

print(get_random_integer())
```

```
[470, 265, 770]
```

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 [114…
```
import random

tickets = [''.join(random.choices('0123456789', k = 6)) for _ in range(100)]

winner = random.choice(tickets)
tickets.remove(winner)
runner_up = random.choice(tickets)

print(f"Winner: Ticket {winner}")
print(f"Runner-up: Ticket {runner_up}")
```

```
Winner: Ticket 751582
Runner-up: Ticket 968558
```

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

In [118…
```
import datetime
current_time = datetime.datetime.now()
```

```
print(current_time)
```

```
2025-02-12 12:58:30.892162
```

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

In [129…

```
new_date = datetime.datetime.now() - datetime.timedelta(weeks = 1)
print(f"Date after subtracting a week: {new_date}")
```

```
Date after subtracting a week: 2025-02-05 13:01:37.878858
```

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

In [133…

```
start_date = datetime.date(2025, 2, 4)
end_date = datetime.date(2025, 2, 12)
no_of_days = abs(start_date - end_date)
print(no_of_days)
```

```
8 days, 0:00:00
```

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

In [143…

```
date1=datetime.date(2025,2,12)
dayname=date1.strftime("%A")
print(dayname)
```

```
Wednesday
```

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

In [147…

```
print(datetime.datetime.now())
```

```
2025-02-12 13:06:45.029571
```

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

In [155…

```
import math

print(math.factorial(5))
print(math.gcd(10,5))
print(math.ceil(5.25))
print(math.lcm(23,5))
print(math.pow(23,5))
print(math.sqrt(81))
```

```
120
5
6
115
6436343.0
9.0
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

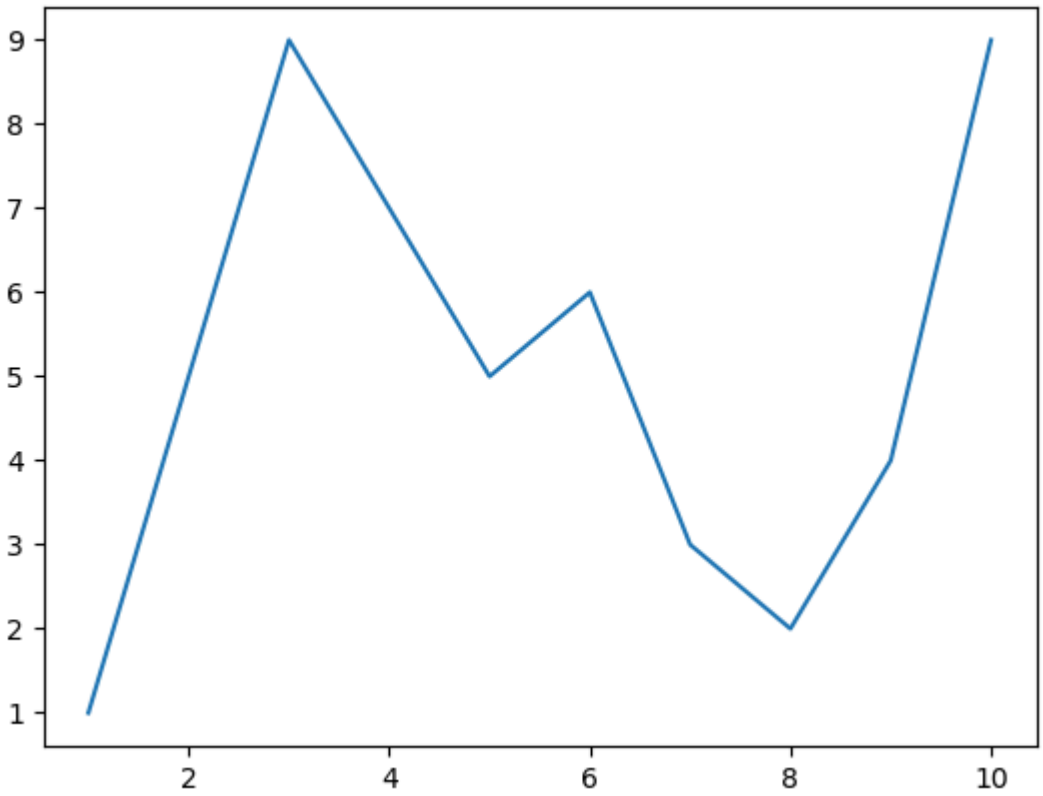
Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 12

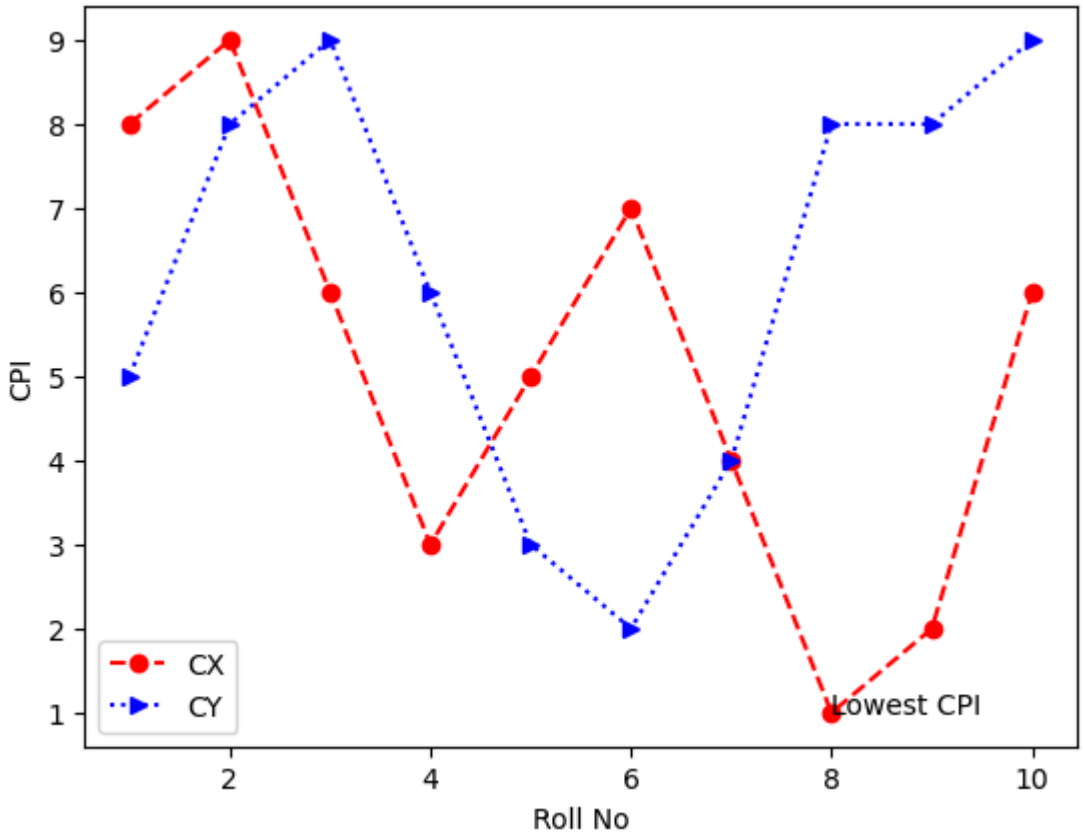
In [ ]:
```
#import matplotlib below
```

In [4]:
```
import matplotlib.pyplot as plt
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
plt.plot(x,y)
plt.show()
# write a code to display the line chart of above x & y
```



In [19]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
```

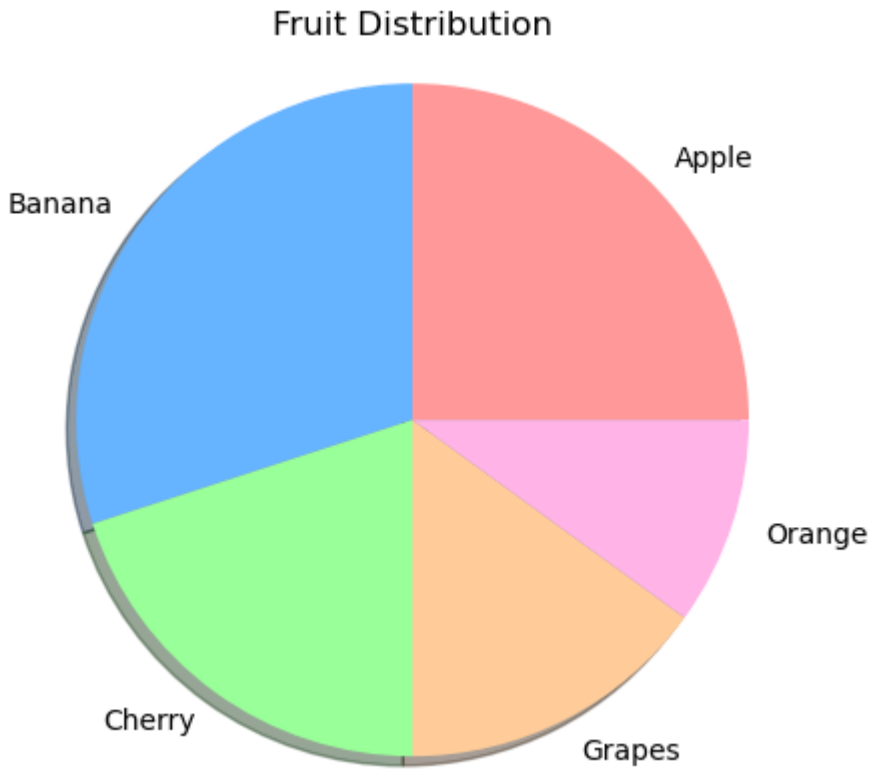
```
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks,label="CX",color="r",marker="o",linestyle="--")
plt.plot(x,cyMarks,label="CY",color="b",marker=">",linestyle="dotted")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.annotate('Lowest CPI',xy=[8,1])
plt.show()
```



04) WAP to demonstrate the use of Pie chart.

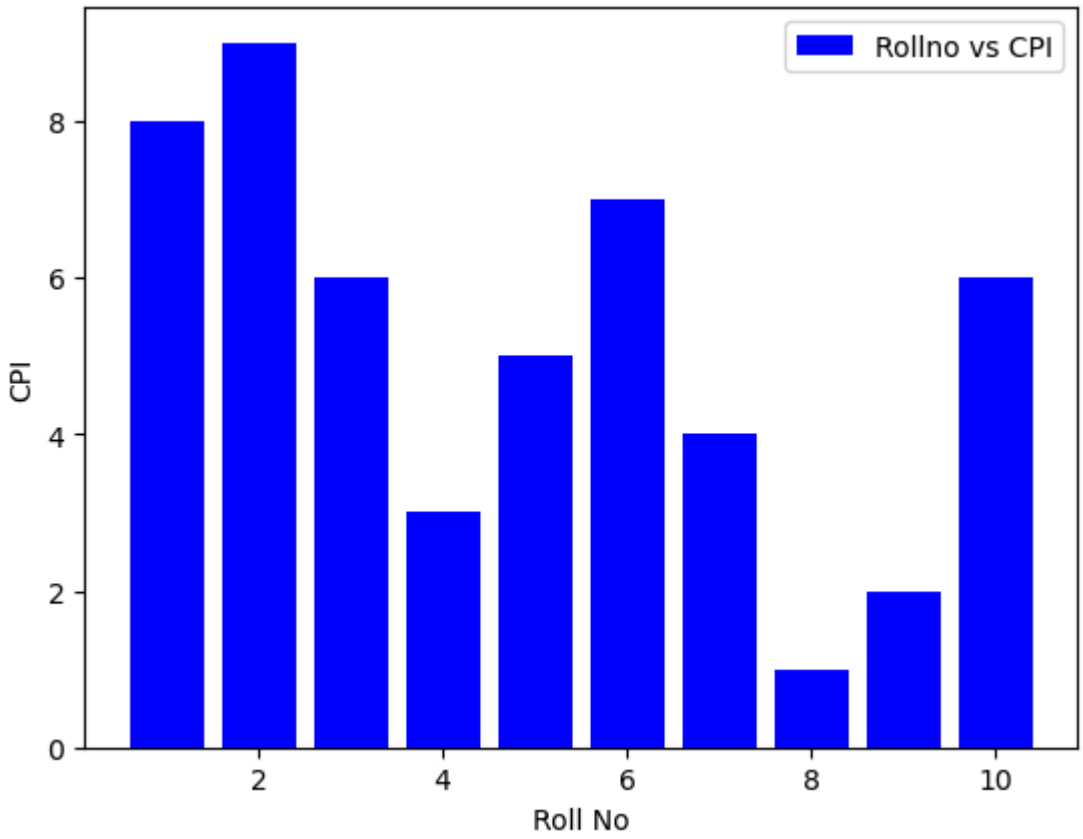
In [114…

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

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



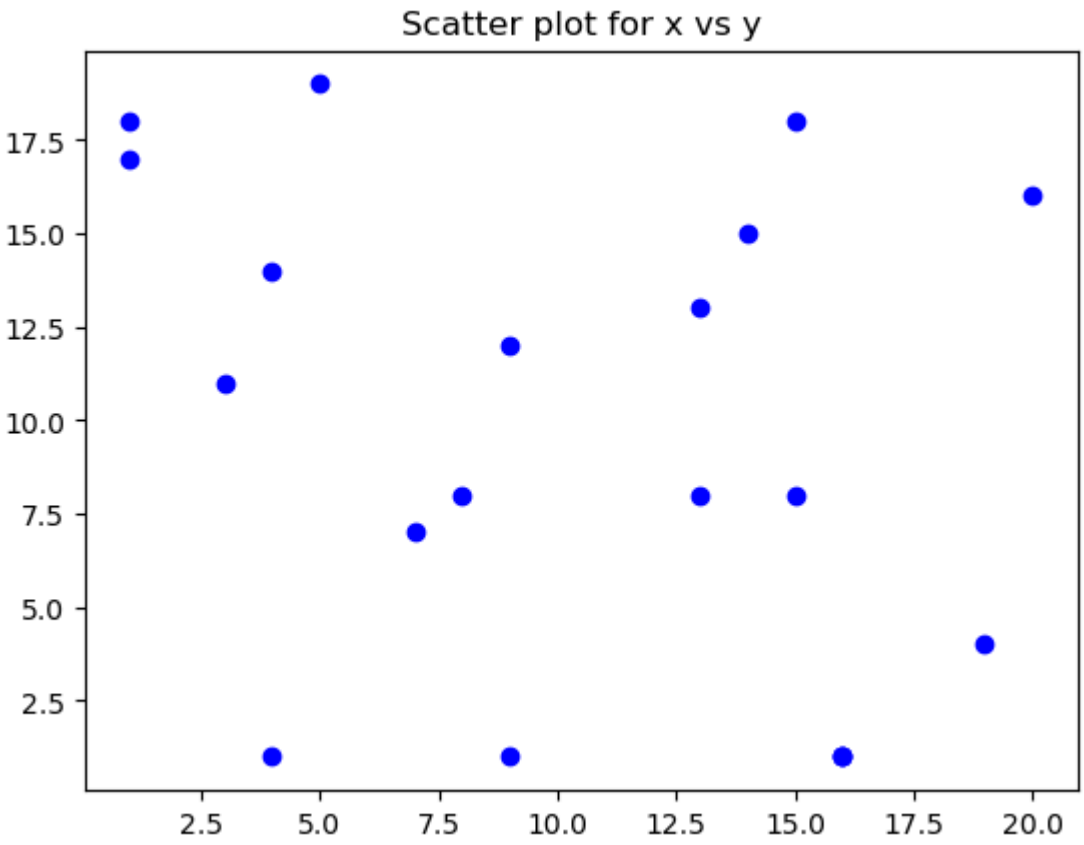
05) WAP to demonstrate the use of Bar chart.

In [58]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
plt.bar(x,cxMarks,label="Rollno vs CPI",color="b")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.show()
```



06) WAP to demonstrate the use of Scatter Plot.

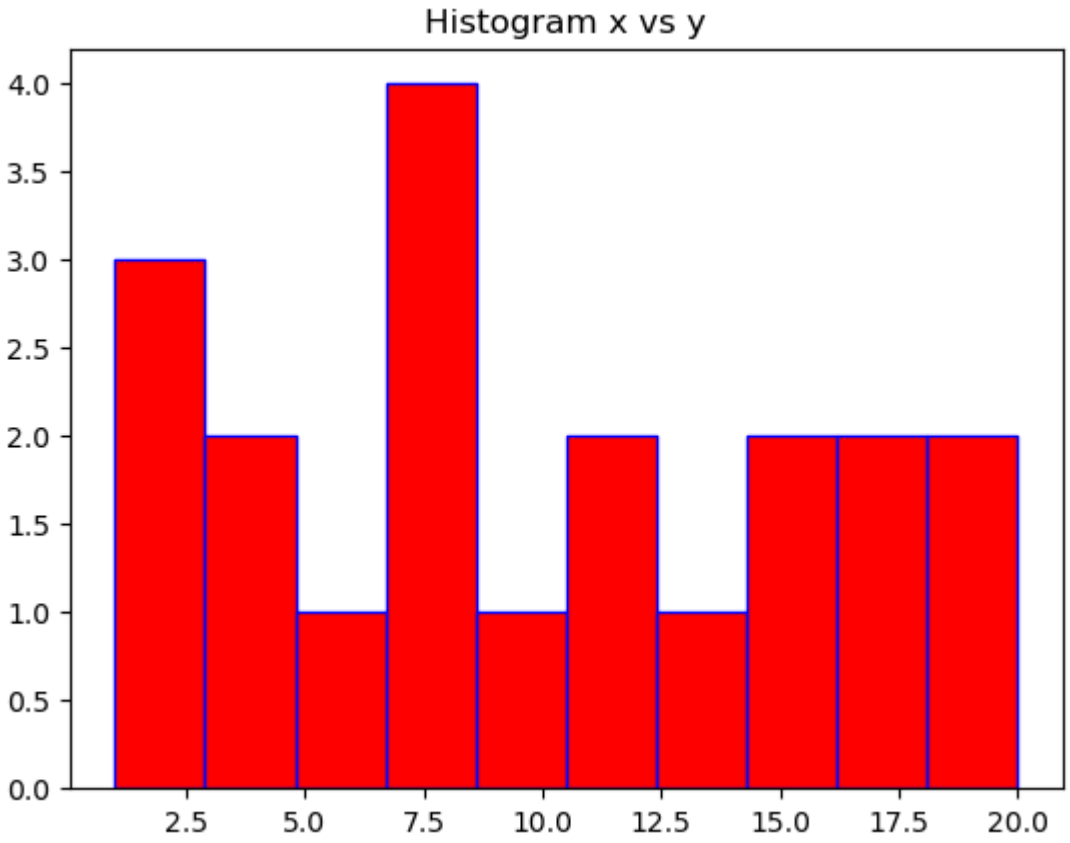
In [82]:
```
import random as r
r.seed(1)
x = [r.randint(1,20) for i in range(20)]
y = [r.randint(1,20) for i in range(20)]
plt.scatter(x,y,color="b")
plt.title("Scatter plot for x vs y")
plt.show()
```



07) WAP to demonstrate the use of Histogram.

In [108…

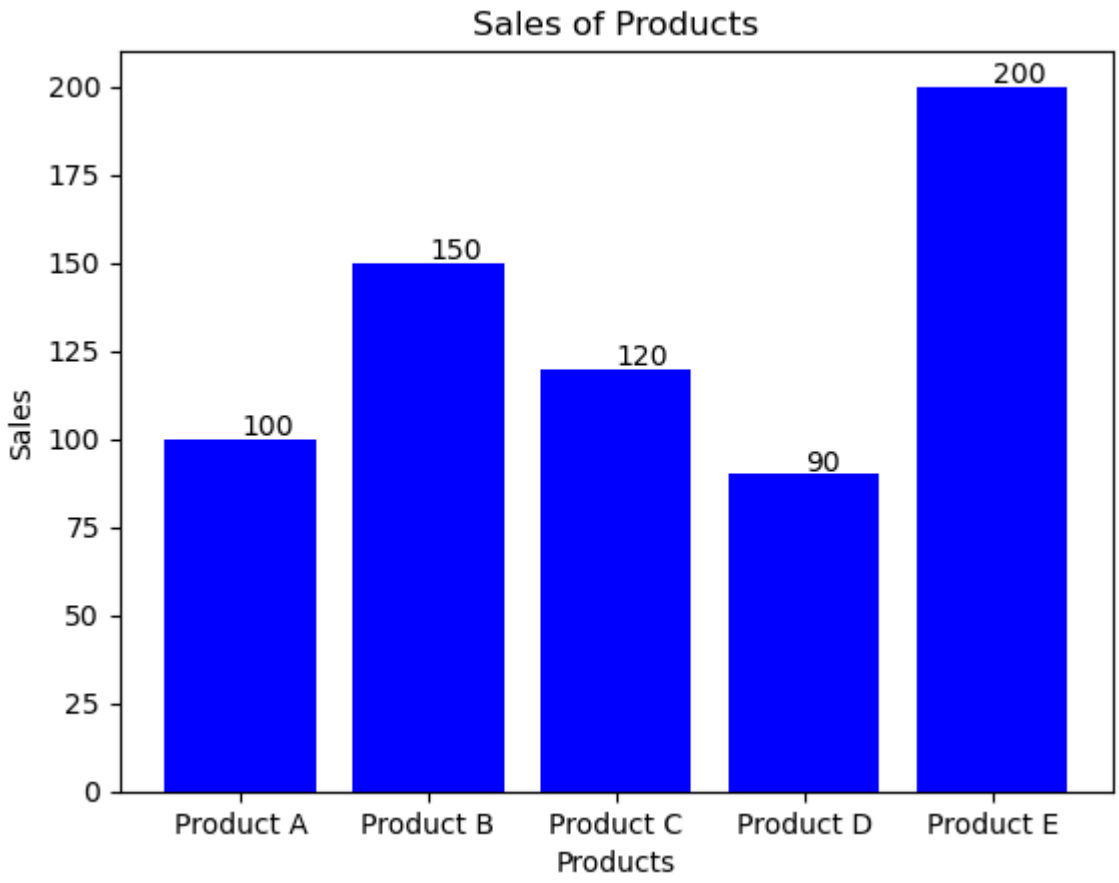
```
r.seed(5)
data = [r.randint(1,20) for i in range(20)]
plt.hist(data,edgecolor="b",color="r")
plt.title("Histogram x vs y")
plt.show()
```



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

In [128…

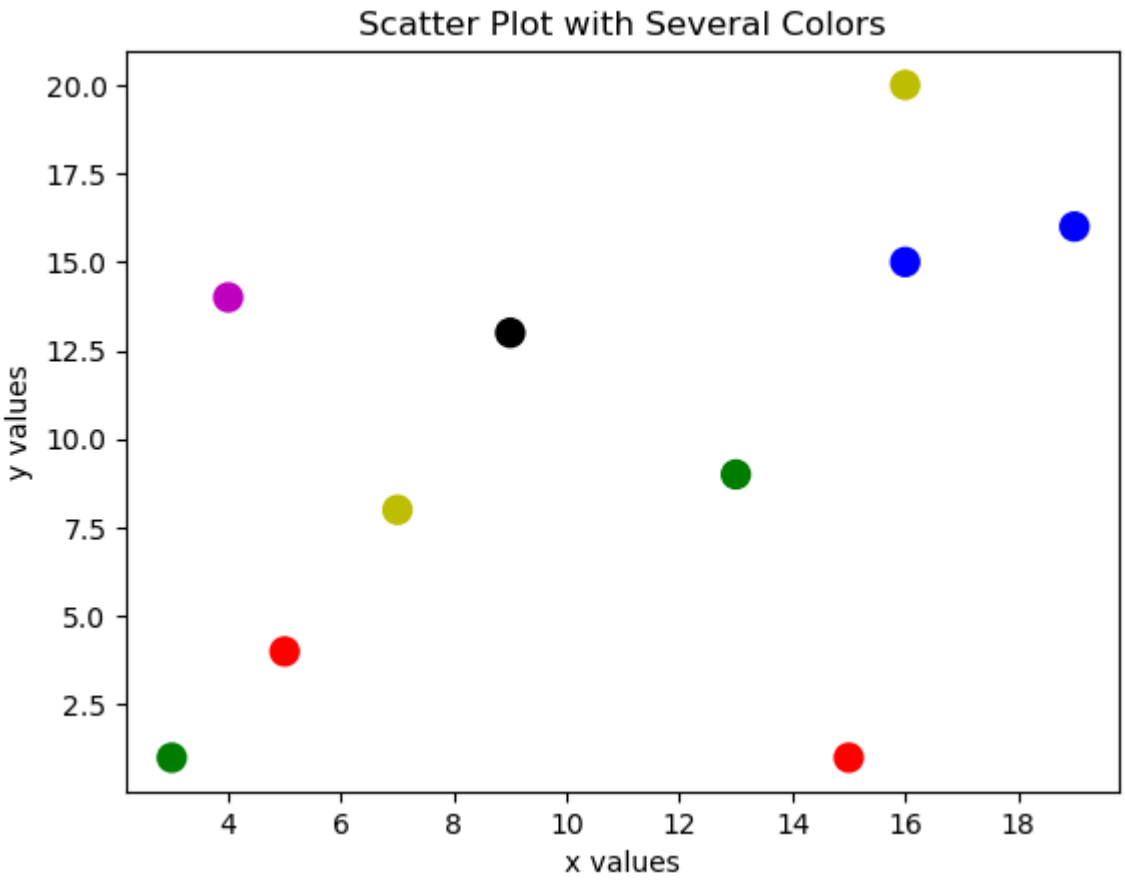
```
products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
sales = [100, 150, 120, 90, 200]
plt.bar(products, sales, color='b')
for i in range(len(products)):
    plt.text(i, sales[i]+1,str(sales[i]))
plt.title('Sales of Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()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



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

In [134…

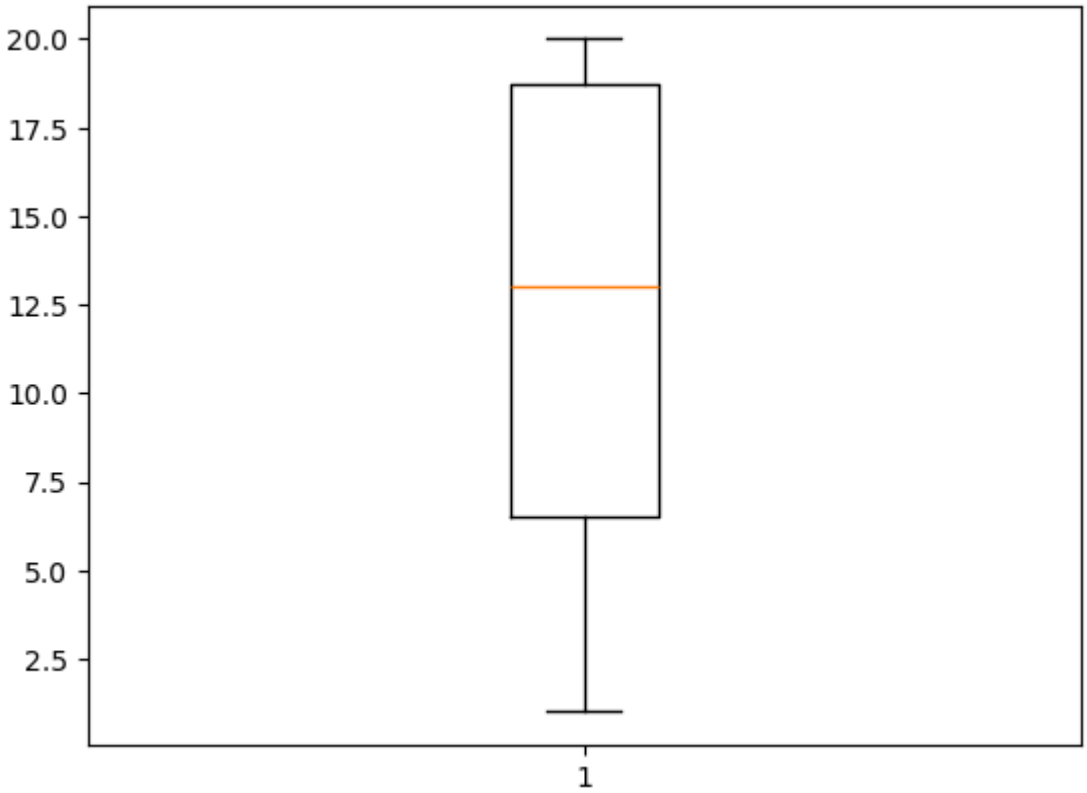
```
r.seed(1)
x = [r.randint(1,20) for i in range(10)]
y = [r.randint(1,20) for i in range(10)]
colors = ['r','b','g','k','m','y','r','b','g','y']

plt.scatter(x, y, color=colors, s=100)
plt.title('Scatter Plot with Several Colors')
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```



10) WAP to create a Box Plot.

In [158…

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



Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 13

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 [7]:
```
class Student:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade

obj=Student("Malay",19,"A")

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

```
Malay
19
A
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

In [23]:
```
class Bank_Account:
    def __init__(self):
        pass
```

```
    def GetAccountDetails(self):
        self.Account_no=int(input("Enter Account_no:"))
        self.User_Name=input("Enter User_Name:")
        self.Email=input("Enter Email:")
        self.Account_Type=input("Enter Account_Type:")
        self.Account_Balance=int(input("Enter Account_Balance:"))

    def DisplayAccountDetails(self):
        print(self.Account_no)
        print(self.User_Name)
        print(self.Email)
        print(self.Account_Type)
        print(self.Account_Balance)

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

```
Enter Account_no: 213210
Enter User_Name: Malay
Enter Email: panaramalay@gmail.com
Enter Account_Type: Savings
Enter Account_Balance: 1000000000000000
213210
Malay
panaramalay@gmail.com
Savings
1000000000000000
```

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

In [36]:
```
import math
class Circle:
    def area(self,r):
        print(f"Area:{math.pi*r*r}")

    def perimeter(self,r):
        print(f"Perimeter:{2*math.pi*r}")

obj=Circle()
obj.area(7)
obj.perimeter(7)
```

```
Area:153.93804002589985
Perimeter:43.982297150257104
```

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

In [15]:
```
class Employee:
    def __init__(self,name,age,salary):
        self.name=name
        self.age=age
        self.salary=salary

    def updatedetails(self,name,age,salary):
```

```
        if name:
            self.name=name
        if age:
            self.age=age
        if salary:
            self.salary=salary

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

obj=Employee("Malay",20,555500000)
obj.displaydetails()
obj.updatedetails("Malay",19,1000000)
obj.displaydetails()
```

```
Employee Information:
Name: Malay
Age: 20
Salary: 555500000
Employee Information:
Name: Malay
Age: 19
Salary: 1000000
```

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

In [28]:
```
class Bank_Account:
    def __init__(self,acc_no,balance):
        self.acc_no=acc_no
        self.balance=balance

    def deposit(self,amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited Rs.{amount}. Current balance: Rs.{self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self,amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew Rs.{amount}. Current balance: Rs.{self.balance}
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

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

obj=Bank_Account(151322,10000000)
obj.deposit(1000)
obj.withdraw(123)
obj.check_balance()
```

```
Deposited Rs.1000. Current balance: Rs.10001000
Withdrew Rs.123. Current balance: Rs.10000877
Balance:Rs.10000877
```

06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

In [38]:
```python
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} {item_name}(s) to inventory.")

    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if self.items[item_name]['quantity'] >= quantity:
                self.items[item_name]['quantity'] -= quantity
                print(f"Removed {quantity} {item_name}(s) from inventory.")
            else:
                print(f"Not enough {item_name} in inventory to remove.")
        else:
            print(f"{item_name} not found in inventory.")

    def update_price(self, item_name, new_price):
        if item_name in self.items:
            self.items[item_name]['price'] = new_price
            print(f"Updated price of {item_name} to ${new_price}.")
        else:
            print(f"{item_name} not found in inventory.")

    def display_inventory(self):
        if not self.items:
            print("Inventory is empty.")
        else:
            print("Inventory Details:")
            for item_name, details in self.items.items():
                print(f"{item_name}: Price - ${details['price']}, Quantity - {de

inventory = Inventory()
inventory.add_item("Laptop", 1200, 10)
inventory.add_item("Phone", 800, 15)
inventory.display_inventory()
inventory.remove_item("Laptop", 5)
inventory.update_price("Phone", 850)
inventory.display_inventory()
inventory.remove_item("Laptop", 6)
inventory.update_price("Tablet", 300)
```

```
Added 10 Laptop(s) to inventory.
Added 15 Phone(s) to inventory.
Inventory Details:
Laptop: Price - $1200, Quantity - 10
Phone: Price - $800, Quantity - 15
Removed 5 Laptop(s) from inventory.
Updated price of Phone to $850.
Inventory Details:
Laptop: Price - $1200, Quantity - 5
Phone: Price - $850, Quantity - 15
Not enough Laptop in inventory to remove.
Tablet not found in inventory.
```

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

In [36]:
```python
class Car:
    def __init__(self, make, model, year, color):
        self.make = make
        self.model = model
        self.year = year
        self.color = color

    def display_car_info(self):
        print(f"Car Information:")
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
        print(f"Color: {self.color}")

car1 = Car("Toyota", "Century", 2021, "Blue")
car1.display_car_info()
car2 = Car("Honda", "Civic", 2020, "Red")
car2.display_car_info()
```

```
Car Information:
Make: Toyota
Model: Century
Year: 2021
Color: Blue
Car Information:
Make: Honda
Model: Civic
Year: 2020
Color: Red
```

08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

In [40]:
```
class StudentKit:
    principal = "Mr ABC"

    def __init__(self, student_name):
        self.student_name = student_name
        self.attendance = 0

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

    def generate_certificate(self):
        if self.attendance >= 75:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Passed (Attendance is sufficient)")
        else:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Failed (Attendance is insufficient)")

student_name = input("Enter the student's name: ")
student = StudentKit(student_name)
days_present = int(input("Enter the number of days present in class: "))
student.record_attendance(days_present)
student.generate_certificate()
```

```
Enter the student's name:  Malay
Enter the number of days present in class:  262
Attendance recorded: 262 days
Certificate of Attendance

Principal: Mr ABC
Student: Malay
Days Present: 262
Status: Passed (Attendance is sufficient)
```

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

In [42]:
```
class Time:
    def __init__(self, hour=0, minute=0):
        self.hour = hour
        self.minute = minute

    def add_time(self, other):
        total_minutes = (self.hour * 60 + self.minute) + (other.hour * 60 + othe
        total_hour = total_minutes // 60
        total_minute = total_minutes % 60
        return Time(total_hour, total_minute)

    def display_time(self):
        print(f"{self.hour} hour(s) and {self.minute} minute(s)")
```

```
time1 = Time(2, 45)
time2 = Time(3, 30)
total_time = time1.add_time(time2)
print("Time 1:")
time1.display_time()
print("Time 2:")
time2.display_time()
print("Total Time after adding:")
total_time.display_time()
```

```
Time 1:
2 hour(s) and 45 minute(s)
Time 2:
3 hour(s) and 30 minute(s)
Total Time after adding:
6 hour(s) and 15 minute(s)
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Meet Jani | 23010101103 | 159

Lab - 13

Continued..

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

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

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

rect = Rectangle(10, 7)

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

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

```
Area of rectangle: 70
```

11) Calculate the area of a square.

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

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

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

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

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

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

12) Calculate the area of a rectangle.

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

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

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

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

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

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

rect1 = Rectangle(10, 5)

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

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

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

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

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

In [25]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

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

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

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

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

14) Create a class Profit that has a method named getProfit that accepts profit from the user.

Create a class Loss that has a method named getLoss that accepts loss from the user.

Create a class BalanceSheet that inherits from both classes Profit and Loss and calculates the balanace. It has two methods getBalance() and printBalance().

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

class Loss:
    def __init__(self):
```

```
        self.loss=0
    def getLoss(self):
        self.loss=float(input("Enter loss:"))

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

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

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

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

```
Enter profit: 100000000
Enter loss: 5
Profit: 100000000.0
Loss: 5.0
Balance: 99999995.0
```

15) WAP to demonstrate all types of inheritance.

In [69]:

```
# Single Inheritance
class Animal:
    def speak(self):
        print("Animal speaks")

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

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

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

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

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

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

```
        Employee.__init__(self, employee_id)

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

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

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

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

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

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

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

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

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

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

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

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

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


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

```
dog.bark()
print()

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

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

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

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

```
Single Inheritance:
Animal speaks
Dog barks

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

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

Hierarchical Inheritance:
Circle Area: 78.5
Rectangle Area: 50

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

16) Create a Person class with a constructor that takes two arguments name and age.

Create a child class Employee that inherits from Person and adds a new attribute salary.

Override the **init** method in Employee to call the parent class's **init** method using the super() and then initialize the salary attribute.

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

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

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

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

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

```
Name: ABC
Age: 30
Salary: 50000
```

17) Create a Shape class with a draw method that is not implemented.

Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors.

Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

In [73]:
```
from abc import ABC, abstractmethod

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

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

class Circle(Shape):
```

```
    def draw(self):
        print("Drawing a circle")

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

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

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

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

In [ ]: