```
### 01) WAP to print “Hello World”

print("Hello World")

Hello World
```

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

```
a,b=5,2
print("Addition of two numbers without input::",a+b)
num1=int(input("Enter 1st number"))
num2=int(input("Enter 2nd Number"))
print("Addition of two numbers from user::",num1+num2)


Addition of two numbers without input:: 7

Enter 1st number 2
Enter 2nd Number 3

Addition of two numbers from user:: 5
```

03) WAP to check the type of the variable.

```
a=9
print(type(a))
b="mansi"
print(type(b))

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

04) WAP to calculate simple interest.

```
p=int(input("Enter Principle amount"))
r=int(input("Enter rate"))
n=int(input("Enter time in years"))
ans=(p*r*n)/100
print("Interest id",ans)

Enter Principle amount 2
Enter rate 25
Enter time in years 1

Interest id 0.5
```

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

```
import math
r=int(input("Enter radius"))
area=math.pi*r*r
per=math.pi*2*r
print("Area is::",area)
print("Perimtere is::",per)

Enter radius 1

Area is:: 3.141592653589793
Perimtere is:: 6.283185307179586
```

06) WAP to calculate area of a triangle.

```
num1=int(input("Enter base"))
num2=int(input("Enter height"))
print("Area is::", 0.5*(num1*num2))

Enter base 10
Enter height 5

Area is:: 25.0
```

07) WAP to compute quotient and remainder.

```
num1=int(input("Enter 1st number::"))
num2=int(input("Enter 2nd number"))
print("Quotient is::",num1/num2,"Remainder is::",num1%num2)

Enter 1st number:: 4
Enter 2nd number 2

Quotient is:: 2.0 Remainder is:: 0
```

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

```
num1=int(input("Enter Fahrenheit"))
cel=(num1-32)*(5/9)
print("Degree is::",cel)
num2=int(input("Enter Degree"))
far=(num2+32)*(9/5)
print("Fahrenheit is::",far)

Enter Fahrenheit 32

Degree is:: 0.0

Enter Degree 456

Fahrenheit is:: 878.4
```

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

```
x1=int(input("Enter x coordinate of 1st point "))
y1=int(input("Enter y coordinate of 1st point "))
x2=int(input("Enter x coordinate of 2st point "))
y2=int(input("Enter y coordinate of 2st point "))
ans=((x2-x1)**2-(y2-y1)**2)**0.5
print(ans)
```

```
Enter x coordinate of 1st point  2
Enter y coordinate of 1st point  -6
Enter x coordinate of 2st point  7
Enter y coordinate of 2st point  3

(4.582208742218757e-16+7.483314773547883j)
```

10) WAP to print sum of n natural numbers.

```
num1=int(input("Enter number::"))
ans=0
while(num1!=0):
    ans=ans+num1
    num1=num1-1
print(ans)
```

```
Enter number:: 10

55
```

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

```
num1=int(input("Enter number::"))
ans=0
while(num1!=0):
    ans=ans+num1**2
    num1=num1-1
print(ans)
```

```
Enter number:: 10

385
```

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

```
fname=input("Enter your First name")
lname=input("Enter your last name")
print(fname,lname,sep=" ")
```

```
Enter your First name Nimavat
Enter your last name Mansi
```

```
Nimavat Mansi
```

13) WAP to swap two numbers.

```
num1=int(input("Enter 1st number"))
num2=int(input("Enter 2nd Number"))
print("Before swapping num1=",num1,"num2=",num2)
x=num1
num1=num2
num2=x;
print("After swapping num1=",num1,"num2=",num2)
```

```
Enter 1st number 2
Enter 2nd Number 3

Before swapping num1= 2 num2= 3
After swapping num1= 3 num2= 2
```

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

```
km=int(input("Enter the distance in kilometer::"))
m=km*1000
ft=km*3281
inc=km*39370
cm=km*100000
print("In meter",m)
print("In feet",ft)
print("In Inches",inc)
print("In Centimeter",cm)
```

```
Enter the distance in kilometer:: 123

In meter 123000
In feet 403563
In Inches 4842510
In Centimeter 12300000
```

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

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

```
Enter day 23
Enter month 11
Enter year 2024
```

```
23-11-2024
```

```
# if..else..
```

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

```
a=int(input("Enter your number"))
if(a>0):
            print(a," is positive")
elif(a==0):
    print(a," is zero")
else:
      print(a," is negative")
```

```
Enter your number 2

2  is positive
```

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

```
a=int(input("Enter your number"))
if(a%2==0):
            print(a," is even")
else:
      print(a," is odd")
```

```
Enter your number 13

13  is odd
```

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

```
a=int(input("Enter your number"))
b=int(input("Enter your number"))
if(a>b):
    print(a," is greater than ",b)
else:
    print(b," is greater than ",a)
```

```
Enter your number 2
Enter your number 5

5  is greater than  2
```

```
a=int(input("Enter your number"))
b=int(input("Enter your number"))
print("Maximum from ",a," and ",b," is ",(a if(a>b) else b))
```

```
Enter your number 2
Enter your number 4

Maximum from  2  and  4  is  4
```

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

```
a=int(input("Enter your number"))
b=int(input("Enter your number"))
c=int(input("Enter your number"))
print("Maximum is ",((a if(a>c) else c)  if(a>b) else(b if(b>c) else
c)))

Enter your number 3
Enter your number 1
Enter your number 5

Maximum is  5
```

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]

```
a=int(input("Enter Year"))
if(a%400==0):
    print("Leap Year")
else:
    if(a%4==0 and a%100!=0):
        print("leap year")
    else:
        print(" Not leap year")

Enter Year 1988

leap year
```

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

```
a=int(input("Enter any number"))
if(a%7==1):
    print("Monday")
elif(a%7==2):
    print("Tuesday")
elif(a%7==3):
    print("Wednesday")
elif(a%7==4):
    print("Thursday")
elif(a%7==5):
```

```
    print("Friday")
elif(a%7==6):
    print("Saturday")
else:
    print("Sunday")


Enter any number 7

Sunday
```

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

```
a=int(input("Enter your number"))
b=int(input("Enter your number"))
c=input("Enter Operator")
if(c=="+"):
    print("Ans is ",a+b)
elif(c=="-"):
    print("Ans is ",a-b)
elif(c=="*"):
    print("Ans is ",a*b)
elif(c=="/"):
    print("Ans is ",a/b)
else:
    print("Invalid Operator")

Enter your number 4
Enter your number 5
Enter Operator /

Ans is  0.8
```

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

```
s1=int(input("Enter your marks in subject1:"))
s2=int(input("Enter your marks in subject2:"))
s3=int(input("Enter your marks in subject3:"))
s4=int(input("Enter your marks in subject4:"))
s5=int(input("Enter your marks in subject5:"))
ans=(s1+s2+s3+s4+s5)/5
if(ans<35):
    print("Fail")
```

```
elif(ans<45 and ans>=35):
    print("Pass")
elif(ans<60 and ans>=45):
    print(" Second Class")
elif(ans<70 and ans>=60):
    print("First Class")
else:
    print("Distinction")
```

```
Enter your marks in subject1: 95
Enter your marks in subject2: 78
Enter your marks in subject3: 89
Enter your marks in subject4: 99
Enter your marks in subject5: 70

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.

```
a=int(input("Enter side1:"))
b=int(input("Enter side2:"))
c=int(input("Enter side3:"))
if(a==b and b==c):
    print("Equilateral")

elif(a!=b and b!=c and c!=a):
    max=((a if(a>c) else c)  if(a>b) else(b if(b>c) else c))
    if(max==a):
        if(a**2==(b**2+c**2)):
            print("Right Angle")
        else:
            print("Scalene")
    if(max==b):
        if(b**2==(a**2+c**2)):
            print("Right Angle")
        else:
            print("Scalene")
    if(max==c):
        if(c**2==(b**2+a**2)):
            print("Right Angle")
        else:
            print("Scalene")
else:
    print("Isosceles")
```

```
Enter side1: 2
Enter side2: 6
Enter side3: 8

Scalene
```

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

```
a=int(input("Enter your number"))
b=int(input("Enter your number"))
c=int(input("Enter your number"))
if(a>b):
    if(a>c):
        if(b>c):
            print(b," is second largest")
        else:
            print(c," is second largest")
    else:
            print(a," is second largest")
else:
    if(b>c):
        if(a>c):
            print(a," is second largest")
        else:
            print(c," is second largest")
    else:
            print(b," is second largest")
```

```
Enter your number 30
Enter your number 40
Enter your number 50

40  is second largest
```

11) WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

a. First 1 to 50 units – Rs. 2.60/unit

b. Next 50 to 100 units – Rs. 3.25/unit

c. Next 100 to 200 units – Rs. 5.26/unit

d. above 200 units – Rs. 8.45/unit

```
a = int(input("Enter units: "))
ans = 0

while a > 0:
```

```
    if a >= 1 and a <= 50:
        ans += a * 2.6
        a -= a
    elif a > 50 and a <= 100:
        ans += (a - 50) * 3.25
        a = 50
    elif a > 100 and a <= 200:
        ans += (a - 100) * 5.26
        a = 100
    else:
        ans += (a - 200) * 8.45
        a = 200
print(f"Total cost: {ans}")
```

```
Enter units:  200
```

```
Total cost: 818.5
```

```
isnam
```

Python Programming - 2301CS404 Lab - 3
Nimavat Mansi Dilipbhai|23010101177|9-12-2024

```
# for and while loop
```

01) WAP to print 1 to 10.

```
for i in range(10):
    print(i+1)
```

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

02) WAP to print 1 to n.

```
n=int(input("enter any number:"))
for i in range(1,n+1):
    print(i)
```

```
enter any number: 5
```

```
1
2
3
4
5
```

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

```
n=int(input("enter any number:"))
for i in range(1,n+1,2):
    print(i)
```

```
enter any number: 10
```

```
1
3
5
7
9
```

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

```
n=int(input("enter any number:"))
m=int(input("enter any number:"))
for i in range(n,m+1):
    if(i%2==0 and i%3!=0):
        print(i)
else:
    print("Loop ended")
```

```
enter any number: 1
enter any number: 10

2
4
8
10
Loop ended
```

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

```
n=int(input("enter any number:"))
ans=0;
for i in range(1,n+1):
    ans+=i
print(ans)
```

```
enter any number: 5

15
```

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

```
n=int(input("enter any number:"))
ans=0;
for i in range(1,n+1):
    ans+=i**2
print(ans)
```

```
enter any number: 4

30
```

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

```
n=int(input("enter any number:"))
ans=0;
for i in range(1,n+1):
    if(i%2==0):
        ans-=i
```

```
    else:
        ans+=i
print(ans)
```

```
enter any number: 5

3
```

08) WAP to print multiplication table of given number.

```
n=int(input("enter any number:"))
for i in range(1,11):
    print(n,"x",i,"=",n*i)
```

```
enter any number: 5

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

09) WAP to find factorial of the given number.

```
n=int(input("enter any number:"))
ans=1
for i in range(1,n+1):
    ans*=i
print(ans)
```

```
enter any number: 4

24
```

10) WAP to find factors of the given number.

```
n=int(input("enter any number:"))
for i in range(1,n+1):
    if(n%i==0):
        print(i)
else:
    print("Loop ended")
```

```
enter any number: 4
```

```
1
2
4
Loop ended
```

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

```
n=int(input("enter any number:"))
count=0;
for i in range(2,(n//2)+1):
    if(n%i==0):
        count+=1;
if(count!=0):
    print("Not Prime")
else:
    print("Prime")
```

```
enter any number: 4

Not Prime
```

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

```
n=int(input("enter any number:"))
ans=0;
while(n>0):
    ans+=n%10;
    n=n//10;
print(ans)
```

```
enter any number: 123

6
```

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

```
n=int(input("enter any number:"))
temp=n;
ans=0
while(n>0):
    r=n%10;
    ans=ans*10+r
    n=(int)(n/10)
if(ans==temp):
    print("Palindrome")
else:
    print("Not palindrome")
```

```
enter any number: 222

Palindrome
```

14) WAP to print GCD of given two numbers.

```
n=int(input("enter any number:"))
m=int(input("enter any number:"))
gcd=1
i=1
while(i<=n and i<=m):
    if(n%i==0 and m%i==0):
        gcd=i;
    i+=1
print(gcd)
```

```
enter any number: 6
enter any number: 4

2
```

String

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

```
str="Mansi"
str2=str[::-1]
if(str==str2):
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
Not Palindrome
```

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

```
str=input("Enter your String")
str2=str.split()[::-1];
ans=""
for i in str2:
    ans+=i+" "
print(ans)
```

```
Enter your String hi I am Mansi

Mansi am I hi
```

03) WAP to remove ith character from given string.

```
str1=input("Enter your String:")
i=int(input("Enter you index you want to remove:"))
str=str1[0:i]+str1[i+1:]
print(str)
```

```
Enter your String: hi I am Mansi
Enter you index you want to remove: 3

hi  am Mansi
```

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

```
str=input("Enter your String")
count=0
for i in str:
```

```
    count+=1
print("Length=",count)
```

```
Enter your String Hi Iam Mansi

Length= 12
```

05) WAP to print even length word in string.

```
str=input("Enter your String")
s=str.split()
for i in s:
    if(len(i)%2==0):
        print(i)
```

```
Enter your String Hi I am Mansi

Hi
am
```

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

```
str=input("Enter your String")
count=0
for i in str:
    if 'a' in i or 'e' in i or 'i' in i or 'o' in i or 'u' in i:
        count+=1
print(count)
```

```
Enter your String Mansi

2
```

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

```
str=input("Enter your String:")
s=str.split()
ans=""
for i in s:
    if(len(i)==1):
        ans+=i.upper()+" "
    else:
        ans+=i[0:1].upper()+i[1:len(i)-1]+i[len(i)-1].upper()+" "
print(ans)
```

```
Enter your String: hi I am Mansi

HI I AM MansI
```

08) WAP to convert given array to string.

```
x=int(input("Enter your Size"))
a=[]
for i in range(x):
    b=int(input("Enter your value in array"))
    a.append(b)

ans=""
for i in a:
    ans+="{}".format(i)
print(ans)
```

```
Enter your Size 5
Enter your value in array 1
Enter your value in array 1
Enter your value in array 1
Enter your value in array 1
Enter your value in array 1

11111
```

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

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

```
str1=input("Enter your password:")
str2=input("Enter your confirm Password:")
if(str1==str2):
    print("Same")
elif(str1.lower()==str2.lower()):
    print("Case mistake")
else:
    print("Not same")
```

```
Enter your password: Manis
Enter your confirm Password: manis

Case mistake
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

```
str1=input("Enter your String:")
s=str1.split()
x=s[len(s)-1]
y=s[0:len(s)-1]
```

```
ans=""
for i in y:
    for j in i:
        ans+="*"
    ans+=" "
ans+=x
print(ans)
```

```
Enter your String: 1234 5678 9012 3456

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

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

s1 = decimal and s2 = medical are Anagram

```
str1=input("Enter your string1:")
str2=input("Enter your string2 :")
for i in str1:
    if i not in str2:
        print("Not Anagram")
        break
else:
    print("Anagram")
```

```
Enter your string1: mansi
Enter your string2 : darshi

Not Anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

```
str1=input("Enter your String:")
lower=""
upper=""
for i in str1:
    if i.isupper():
        upper+=i
    else:
        lower+=i;
print(upper+lower)
```

```
Enter your String: EHlsarwiwhtwMV

EHMVlsarwiwhtw
```

List

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

```
l1=[1,2,3,4,5]
ans=0
for i in l1:
    ans+=i
print(ans)
```

```
15
```

02) WAP to find largest element in a List.

```
max=0
for i in l1:
    if(max<i):
        max=i
print(max)
```

```
5
```

03) WAP to find the length of a List.

```
print(len(l1))
count=0
for i in l1:
    count+=1
print(count)
```

```
5
5
```

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

```
lst=[1,2,3,4,5]
lst[0], lst[-1] = lst[-1], lst[0]
print(lst)
```

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

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

```
l2=[1,2,3,4,5]
l21=l2[:len(l2)//2]
l22=l2[len(l2)//2::]
l2.append(l21)
print(l2)
```

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

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

```
n=int(input("Enter index1"))
m=int(input("Enter indez2"))
lst=[1,2,3,4,5]
lst[n], lst[m] = lst[m], lst[n]
print(lst)
```

```
Enter index1 2
Enter indez2 4

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

07) WAP to reverse the list entered by user.

```
n=input("Enter elements of list separated by space")
list=n.split()
print(list[::-1])
```

```
Enter elements of list separated by space 1 2 3 4 5

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

08) WAP to print even numbers in a list.

```
list=[1,2,3,4,5]
for i in list:
    if(i%2==0):
        print(i)
```

```
2
4
```

09) WAP to count unique items in a list.

```
n=input("Enter elements of list separated by space")
list=n.split()
uni=[]
```

```
for i in list:
    if i not in uni:
        uni.append(i)
print(len(uni))
```

```
Enter elements of list separated by space 1 2 4 3 5 2 3 4

5
```

10) WAP to copy a list.

```
l=[1,2,3,4,5]
x=l.copy()
print(x)
```

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

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

```
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))
print("Odd numbers in the range:")
for num in range(start, end + 1):
    if num % 2 != 0:
        print(num, end=" ")
```

```
Enter the start of the range:  2
Enter the end of the range:  7

Odd numbers in the range:
3 5 7
```

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

```
n=input("Enter elements of list separated by space")
list=n.split()
x=input("Enter element whose occurence you want to find")
count=0
for i in list:
    if i==x:
        count+=1
print(count)
```

```
Enter elements of list separated by space 1 2 3 4 5
Enter element whose occurence you want to find 2

1
```

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

```
list=[1,4,2,7,9,10,34]
max=0
max1=0
for i in list:
    max=i if max<i else max
list.remove(max)
for i in list:
    max1=i if max1<i else max1
print(max1)
```

```
10
```

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

```

```

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

```
list=[]

for i in range(1,10):
    list.append(i**2)
print(list)
li=[i**2 for i in range(1,10)]
print(li)
```

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

```

```

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

```
l1=[1,2,3,"Name",2.4,5.6,5]
l2=[1,2,4,"Name",3.4,5.6]
l3=[]
for i in l1:
    if i in l2:
      l3.append(i)
print(l3)
```

```
[1, 2, 'Name', 5.6]
```

Tuple

01) WAP to find sum of tuple elements.

```
t1=(1,2,5,4,7,9)
sum=0
for i in t1:
    sum+=i
print(sum)
```

```
28
```

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

```
t1=(1,221,30,6,5,50)
sorted_t1=sorted(t1)
k=int(input("Enter k:"))
if k>len(sorted_t1):
 print("Invalid k")
else:
 print("Minimum:",sorted_t1[:k])
 print("Maximum:",sorted_t1[-k:])
```

```
Enter k: 3

Minimum: [1, 5, 6]
Maximum: [30, 50, 221]
```

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

```
t1=(1,2,4,5,6)
t2=(10,20,30,40,50)
l=[t1,t2]

k = int(input("Enter k"))
count=0
for i in l:
     for j in i:
         if j%k==0:
             count+=1
     if count==len(i):
         print(i)
```

```
Enter k 4
```

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

```
l1=[1,2,3,4,5]
l=[(i,i**3) for i in l1]
print(l)
```

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

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

```
t1=(1,-2,4,-5,6)
t2=(10,20,30,40,50)
l=[t1,t2]
count=0
for i in l:
    count=0
    for j in i:
        if j>0:
            count+=1
    if count==len(i):
         print(i)
```

```
(10, 20, 30, 40, 50)
```

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

```
l1=[1,2,3,4,50]
t1=(16,18,20,220)
l1.append(t1)
print(l1)
t2=(10,20,30,40,50)
l1=[1,2,2,4,6]
t2=t2+(l1,)
print(t2)
```

```
[1, 2, 3, 4, 50, (16, 18, 20, 220)]
(10, 20, 30, 40, 50, [1, 2, 2, 4, 6])
```

07) WAP to remove tuples of length K.

```
l1=[(1,2,3,4),(10,5,8,20),(3,5,6),(45,60)]
l2=[]
k=int(input("Enter desired length"))
count=0
for i in l1:
     count=0
     for j in i:
         count+=1
```

```
     if count!=k:
         l2.append(i)
print(l2)
```

```
Enter desired length 3
```

```
[(1, 2, 3, 4), (10, 5, 8, 20), (45, 60)]
```

08) WAP to remove duplicates from tuple.

```
t1=(1,2,3,2,20)
t2=()
for i in t1:
     if i not in t2:
         t2+=(i,)
print(t2)
```

```
(1, 2, 3, 20)
```

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

```
t1=(1,2,3,4,5,6)
ans=tuple(i*j for i,j in zip(t1,t1[1:]))
ans
```

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

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

```
t1=(1,2,3,4,5,6)
count=0
for i in t1:
     if(t1.count(i)!=1):
         print('Not distinct')
         break
else:
     print('Distinct')
```

```
Distinct
```

Set & Dictionary

01) WAP to iterate over a set.

```
s={1,2,3,4,5,9,10,7}
for i in s:
    print(i)
```

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

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

```
my_set = {1, 2, 3, 4, 5}
my_list = list(my_set)
print("Set converted to list:", my_list)
my_string = ''.join(map(str, my_set))
print("Set converted to string:", my_string)
my_tuple = tuple(my_set)
print("Set converted to tuple:", my_tuple)
```

```
Set converted to list: [1, 2, 3, 4, 5]
Set converted to string: 12345
Set converted to tuple: (1, 2, 3, 4, 5)
```

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

```
max=1
for i in s:
    max=max if max>i else i
print(max)
```

```
10
```

04) WAP to perform union of two sets.

```
s1={2,"fgh",6,8}
s2={6,9,2,"dfg"}
```

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

```
{2, 'fgh', 6, 8, 9, 'dfg'}
{2, 'fgh', 6, 8, 9, 'dfg'}
```

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

```
print(s1.isdisjoint(s2))
```

```
False
```

06) WAP to remove duplicates from list.

```
l1=[1,2,3,1,2,3]
l1=set(l1)
l1=list(l1)
print(l1)
```

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

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

```
str="Hi I am Mansi Nimavat I Like to play kho-kho"
words=str.split()
uword=set(words)
for i in uword:
    print(i)
```

```
Hi
am
Nimavat
Like
to
kho-kho
I
Mansi
play
```

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

```
s1={2,"fgh",6,8}
s2={6,9,2,"dfg"}
ans=s1-s2
print(ans)
```

```
{8, 'fgh'}
```

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

```
string1 = "listen you"
string2 = "silent you"
string1 = string1.replace(" ", "").lower()
string2 = string2.replace(" ", "").lower()
if set(string1) == set(string2) and len(string1) == len(string2):
    print(f'"{string1}" and "{string2}" are anagrams.')
else:
    print(f'"{string1}" and "{string2}" are not anagrams.')
```

```
"listenyou" and "silentyou" are anagrams.
```

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

```
l1=[1,2,3,4,5]
l2=[3,4,5,6,7]
l3=[5,6,7,8,9]
s1=set(l1)&set(l2)
s2=s1&set(l3)
print(s2)
```

```
{5}
```

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

```
n=input("Enter your String")
vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
vowel_count = sum(1 for char in n if char in vowels)
print(vowel_count)
```

```
Enter your String Mansi
```

```
2
```

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

```
n=input("Enter your String")
binary={'0','1'}
for i in n:
    if i not in binary:
        print("Not binary string")
        break
else:
    print("Binary String")
```

```
Enter your String Mansi
```

```
Not binary string
```

13) WAP to sort dictionary by key or value.

```
d = {2: 56, 100: 2, 3: 323}
myKeys = list(d.keys())
myKeys.sort()
sd = {i: d[i] for i in myKeys}
print(sd)
# myVal=list(d.values())
# myVal.sort()
# print(myVal)


{2: 56, 3: 323, 100: 2}
```

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

```
d1 = {'name1':1,'name3':2,'name2':4,'name4':3}
l1 = list(d1.values())
ans = 0
for i in l1:
    ans += i
print("Sum Of All The Elements is : ",ans)

Sum Of All The Elements 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.

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

Enter Key For Search :  a

5
Key Founded.
```

User Defined Function

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

```
def calculate_bmi(mass, height):
    return mass / height**2
mass = int(input("Enter Mass"))
height =float(input("Enter height"))
bmi = calculate_bmi(mass, height)
print(f"The calculated BMI is: {bmi:.2f}")
```

```
Enter Mass 2
Enter height 3.5

The calculated BMI is: 0.16
```

02) Write a function that add first n numbers.

```
def Addition(n):
    sum=0
    for i in range(1,n+1):
        sum+=i
    return sum
print(Addition(5))
```

```
15
```

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

```
def Prime(n):
    for i in range(2,n//2+1):
        if(n%i==0):
            return 0;
            break;
    return 1;
print(Prime(4))
```

```
0
```

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

```
def prime_range(min,max):
    k=min
    while(k<=max):
        for l in range(2,k//2+1):
            if(k%l==0):
                break
        else:
            print(k)
        k+=1
prime_range(4,15)
```

```
5
7
11
13
```

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

```
def Check_Palindrome(String):
    stringrev=String[::-1]
    if (String==stringrev):
        return True
    return False
print(Check_Palindrome("Mansi"))
print(Check_Palindrome("abba"))
```

```
False
True
```

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

```
def Sum_list(list):
    sum=0
    for i in list:
        sum+=i
    return sum
print(Sum_list([1,2,3,4,5]))
```

```
15
```

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

```
def tup_sum(tup_list):
    return sum(i[0] for i in tup_list)
```

```
tuples_list = [(1, 2), (3, 4), (5, 6)]
print(tup_sum(tuples_list))
```

```
9
```

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

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter the value of n: "))
result = fibonacci(n)
print(f"The {n}th term of the Fibonacci series is: {result}")
```

```
Enter the value of n:  6
```

```
The 6th term of the Fibonacci series is: 8
```

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

```
def dict_name(dict1,roll):
    return dict1.get(roll)
dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
print(dict_name(dict1,102))
```

```
Rahul
```

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

```
def endswith_00(lists):
    sum=0
    for i in lists:
        if(str(i).endswith('00')):
            sum=sum+i
    return sum
scores = [200, 456, 300, 100, 234, 678]
print(endswith_00(scores))
```

```
600
```

11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

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

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

```
def invertdict(dicts):
    inverdict={v:k for k,v in dicts.items()}
    return inverdict
ans= {'a': 10, 'b':20, 'c':30, 'd':40}
print(invertdict(ans))
```

```
{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.

```
def is_pangram(input_string):
    alphabet = set("abcdefghijklmnopqrstuvwxyz")
    input_set = set(input_string.lower())
    return alphabet.issubset(input_set)

is_pangram("the quick brown fox jumps over the lazy dog")
```

```
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

```
def cnt_up_low(s1):
    upc=0
    loc=0
    for i in s1:
        if(i.isupper()):
            upc+=1
        if(i.islower()):
            loc+=1
    return [upc,loc]
```

```
s1 = 'AbcDEfgh'
print(cnt_up_low(s1))
```

```
[3, 5]
```

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

```
 minimum=lambda x,y:x if x<y else y
x=int(input("enter value of x:"))
y=int(input("enter value of y:"))
print("Minimum is ",minimum(x,y))
```

```
enter value of x: 3
enter value of y: 8

Minimum is  3
```

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

```
students = ["Alexander", "Ben", "Catherine", "Diana", "Elizabeth",
"Frank", "Gabrielle", "Harry"]
long_names = list(filter(lambda name: len(name) > 7, students))
print("Names with more than 7 characters:", long_names)
```

```
Names with more than 7 characters: ['Alexander', 'Catherine',
'Elizabeth', 'Gabrielle']
```

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

```
students = ["alexander", "ben", "catherine", "diana", "elizabeth",
"frank", "gabrielle", "harry"]
data=list(map(lambda name:name.title(),students))
print(data)
```

```
['Alexander', 'Ben', 'Catherine', 'Diana', 'Elizabeth', 'Frank',
'Gabrielle', 'Harry']
```

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

```python
def positional_args(a, b):
    return a + b

print("Positional Argument",positional_args(10, 20))

print("-----------------------------------------------------")

def keyword_args(a,b):
    return a + b

print("Keyword Arguments",keyword_args(b=10,a=20))

print("-----------------------------------------------------")

def default_args(a,b=100):
    return a + b

print("Default Arguments",default_args(10))
print("Default Arguments",default_args(10,20))

print("-----------------------------------------------------")

def variable_positional_args(*args):
    return sum(args)

print("Variable Legngth Positional(*args)
",variable_positional_args(1, 2, 3, 4))

def variable_keyword_args(**kwargs):
    return kwargs

print("variable length Keyword Arguments
(**kwargs)",variable_keyword_args(name="nandani", age=19))

print("-----------------------------------------------------")

def keyword_only_args(a, *, b):
    return a + b

print("Keyword-Only Keyword-Only ",keyword_only_args(10, b=20))
# print("Keyword-Only Keyword-Only ",keyword_only_args(10, 5))  #
Error: b must be passed as a keyword argument

print("-----------------------------------------------------")

def positional_only_args(a, b, /, c):
    return a + b + c

print("Positional Only Arguments",positional_only_args(10, 10, c=3))
```

```
Positional Argument 30
------------------------------------------------------
Keyword Arguments 30
------------------------------------------------------
Default Arguments 110
Default Arguments 30
------------------------------------------------------
Variable Legngth Positional(*args)  10
variable length Keyword Arguments (**kwargs) {'name': 'nandani',
'age': 19}
------------------------------------------------------
Keyword-Only Keyword-Only  30
------------------------------------------------------
Positional Only Arguments 23
```

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

```
fp=open('Abc.txt')
print(fp.read())
```

```
Hi..I am Mansi
Its 4th Sem
I love python
```

```
fp=open('Abc.txt')
for i in fp.readlines():
    print(i)
```

```
Hi..I am Mansi

Its 4th Sem

I love python
```

```
fp=open('Abc.txt')
print(fp.readlines())
```

```
['Hi..I am Mansi\n', 'Its 4th Sem\n', 'I love python\n']
```

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

```
f=open('new.txt','w')
f.write("Mansi")
f.close()
```

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

```
f=open('Abc.txt','r')
lines=f.readlines()
for i in range(5):
```

```
    print(lines[i])
```

```
Hi..I am Mansi

Its 4th Sem

I love python

Hi..I am Mansi

Its 4th Sem
```

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

```
f=open('new.txt','r')
lines=f.read()
word=lines.split()
maxlength=0
for i in word:
    maxlength=len(i) if len(i)>maxlength else maxlength
for i in word:
    if len(i)==maxlength:
        print(i)
```

```
NimavatMansi
```

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

```
file=open('new.txt','r')
lines = file.readlines()  # Read all lines
num_lines = len(lines)
num_words = sum(len(line.split()) for line in lines)  # Count words
num_chars = sum(len(line) for line in lines)  # Count characters
print("Number of lines:", num_lines)
print("Number of words:", num_words)
print("Number of characters:", num_chars)
```

```
Number of lines: 2
Number of words: 5
Number of characters: 30
```

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

```
file=open('Abc.txt','r')
lines=file.read()
f=open('write.txt','w')
```

```
f.write(lines)
f.close()
```

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

```
import os
filename='Abc.txt'
size = os.path.getsize(filename)  # Get file size in bytes
print(f"Size of '{filename}': {size} bytes")
```

```
Size of 'Abc.txt': 171 bytes
```

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

```
def frequency(filename, word):
    file= open(filename, 'r')
    text = file.read()
    words = text.split()
    count = words.count(word)
    return count
filename = "Abc.txt"
word_to_find = "python"
result = frequency(filename, word_to_find)
print(f"The word '{word_to_find}' appears {result} times in
'{filename}'.")
```

```
The word 'python' appears 4 times in 'Abc.txt'.
```

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.

```
fp=open("marks.txt",'a')
for i in range(5):
    s=input("enter marks for subject"+str(i+1))
    fp.write(s+" ")
fp.close()
max=0
f=open("marks.txt",'r')
for i in f.read().split():
    if int(i)>max:
        max=int(i)
print(max)
f.close()
```

```
enter marks for subject1 3
enter marks for subject2 5
enter marks for subject3 7
```

```
enter marks for subject4 8
enter marks for subject5 3

8
```

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

(Note: each number should be in new line)

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

def write_primes():
    primes = []
    num = 2
    while len(primes) < 100:
        if is_prime(num):
            primes.append(num)
        num += 1

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

write_primes()
```

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

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

def write_primes():
    primes = []
    num = 2
    while len(primes) < 100:
        if is_prime(num):
            primes.append(num)
        num += 1
```

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

def merge_files(file1, file2, output_file):
    with open(output_file, "w") as outfile:
        for fname in [file1, file2]:
            with open(fname, "r") as infile:
                outfile.write(infile.read())

write_primes()
merge_files("primenumbers.txt", "anotherfile.txt", "mergedfile.txt")
```

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

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

def write_primes():
    primes = []
    num = 2
    while len(primes) < 100:
        if is_prime(num):
            primes.append(num)
        num += 1

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

def merge_files(file1, file2, output_file):
    with open(output_file, "w") as outfile:
        for fname in [file1, file2]:
            with open(fname, "r") as infile:
                outfile.write(infile.read())

def replace_word(input_file, output_file, word1, word2):
    with open(input_file, "r") as file:
        data = file.read()

    data = data.replace(word1, word2)
```

```
    with open(output_file, "w") as file:
        file.write(data)

write_primes()
merge_files("primenumbers.txt", "anotherfile.txt", "mergedfile.txt")
replace_word("mergedfile.txt", "updatedfile.txt", "word1", "word2")
```

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

```
with open("example.txt", "w") as file:
    file.write("Hello, this is a file handling example.")

with open("example.txt", "r") as file:
    print("Initial Position:", file.tell())  # Should be 0

    file.seek(10, 0)
    print("After seeking 10 bytes from start:", file.tell())

    file.seek(5, 1)
    print("After seeking 5 bytes forward from current position:",
file.tell())

    file.seek(-5, 2)
    print("After seeking 5 bytes backward from end:", file.tell())

    print("Remaining content:", file.read())
```

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.

```
try:
    a=int(input("Enter 1st value:"))
    b=int(input("Enter 2nd value:"))
    ans=a//b
    print(ans)
except ZeroDivisionError as err:
    print(err)
except ValueError as err:
    print("ValueError")
```

```
Enter 1st value: 4
Enter 2nd value: Mansi

ValueError
```

```
try:
    a=input("Enter 1st value:")
    b=int(input("Enter 2nd value:"))
    ans=a+b
    print(ans)
except TypeError as err:
    print("TypeError")
```

```
Enter 1st value: dfg
Enter 2nd value: 12

TypeError
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
try:
    l1=[2,3,4]
```

```
    print(l1[7])
except IndexError:
    print("IndexError")
```

```
IndexError
```

```
try:
    dic={'name':'Manali'}
    print(dic['namee'])
except KeyError as error:
    print(error)
```

```
'namee'
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
try:
    fp=open("abc.txt")
except FileNotFoundError as err:
    print(err)
```

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

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

```
try:
    print(2//0)
    fp=open("abc.txt")
except Exception as err:
    print(err)
```

```
integer division or modulo by zero
```

05) WAP to demonstrate else and finally block.

```
try:
    a=int(input("Enter 1st value:"))
    b=int(input("Enter 2nd value:"))
    ans=a//b
except ZeroDivisionError:
     print("Error")
else:
    print(ans)
finally:
    print("Done")
```

```
Enter 1st value: 2
Enter 2nd value: 2

1
Done
```

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.

```
try:
    g=input("Enter comma sep values")
    l=[int(i) for i in g.split(',')]
    print(l)
except:
    print("value error")
```

```
Enter comma sep values 1,3,4

[1, 3, 4]
```

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

```
def Divide(a,b):
    try:
        print(a//b)
    except ZeroDivisionError:
        print("ZeroDivisionError")
Divide(1,2)
```

```
0
```

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.

```
a=int(input("Enter your age"))
try:
    if(a<18):
        raise ValueError
```

```
    else:
        print(a)
except ValueError:
    print("Enter Valid Age")
```

```
Enter your age 19
```

```
19
```

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.

```
class InvalidUsernameError(Exception):
    pass
try:
    u=input("Enter your name")
    if len(u)<=15 and len(u)>=5:
        print(u)
    else:
        raise InvalidUsernameError
except InvalidUsernameError:
    print("Username must be between 5 and 15 characters long")
```

```
Enter your name e
```

```
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.

```
class NegativeNumberError(Exception):
    def __init__(self,msg):
        self.msg=msg
try:
    a=int(input("Enter your number"))
    if a>=0:
        print(a)
    else:
```

```
        raise NegativeNumberError("Cannot calculate the square root of
a negative number")
except NegativeNumberError as ee:
    print(ee)


Enter your number -1

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

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.

```
import Calculator as c
print(c.add(4,4))
print(c.sub(4,4))
print(c.multi(4,4))
print(c.div(4,4))
```

```
8
0
16
1.0
```

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

```
import random
st="my name is Mansi"
print(random.choice(st))
```

```
a
```

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

```
import random
li=[1,2,3,4,5,6,7,8,9,'dgfh']
print(random.choice(li))
```

```
6
```

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

```

```

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

```
import random
print("random number is divisible by 5:",random.randrange(100,999,5))
print("random number is divisible by 5:",random.randrange(100,999,5))
print("random number is divisible by 5:",random.randrange(100,999,5))
```

```
random number is divisible by 5: 275
random number is divisible by 5: 635
random number is divisible by 5: 380
```

06) WAP to generate 100 random lottery tickets and pick two lucky tickets from it and announce them as Winner and Runner up respectively.

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

```
import datetime
print(datetime.date.today())
t=datetime.time(10,20,40)
print(t)
```

```
2025-03-06
10:20:40
```

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

```
from datetime import datetime, timedelta
date_str = "2025-03-06"
date_obj = datetime.strptime(date_str, "%Y-%m-%d")
new_date = date_obj - timedelta(days=7)
print("New Date:", new_date.strftime("%Y-%m-%d"))
```

```
New Date: 2025-02-27
```

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

```
from datetime import datetime
date1_str = input("Enter the first date (YYYY-MM-DD): ")
date2_str = input("Enter the second date (YYYY-MM-DD): ")
date1 = datetime.strptime(date1_str, "%Y-%m-%d")
date2 = datetime.strptime(date2_str, "%Y-%m-%d")
days_difference = abs((date2 - date1).days)
print("Number of days between the two dates:", days_difference)
```

```
Enter the first date (YYYY-MM-DD):  2025-02-27
Enter the second date (YYYY-MM-DD):  2025-02-27
```

```
Number of days between the two dates: 0
```

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

```
from datetime import datetime
date_input = input("Enter a date (DD-MM-YYYY): ")
date_obj = datetime.strptime(date_input, "%d-%m-%Y")
print( date_obj.strftime("%A"))



Enter a date (DD-MM-YYYY):  19-01-2006

Thursday
```

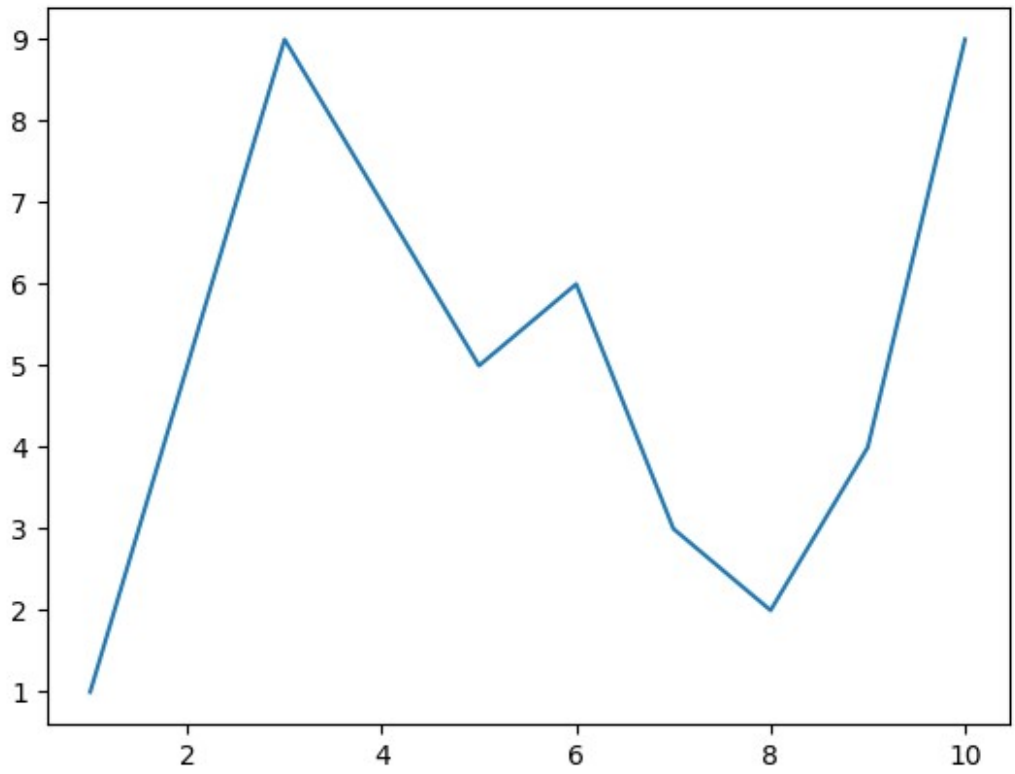
11) WAP to demonstrate the use of date time module.

```
import datetime
print(datetime.date.today())
t=datetime.time(10,20,40)
print(t)

2025-03-06
10:20:40
```

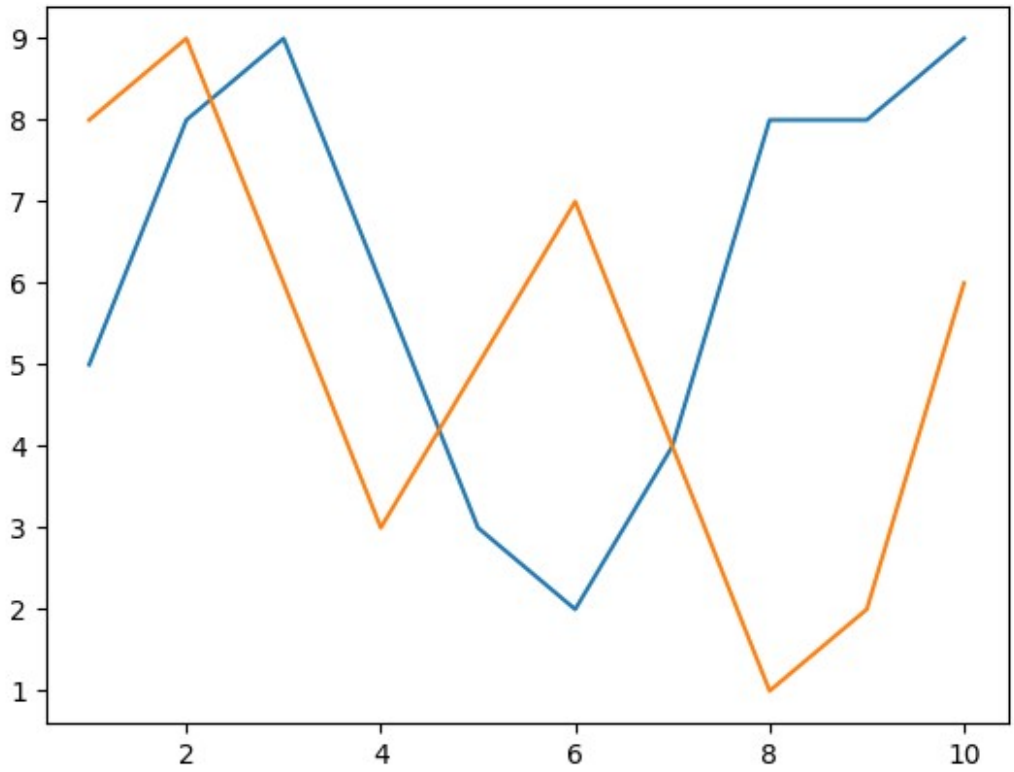
12) WAP to demonstrate the use of the math module.

```
import math
print(math.pi)

3.141592653589793
```

```
#import matplotlib below
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt
%matplotlib inline
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
```

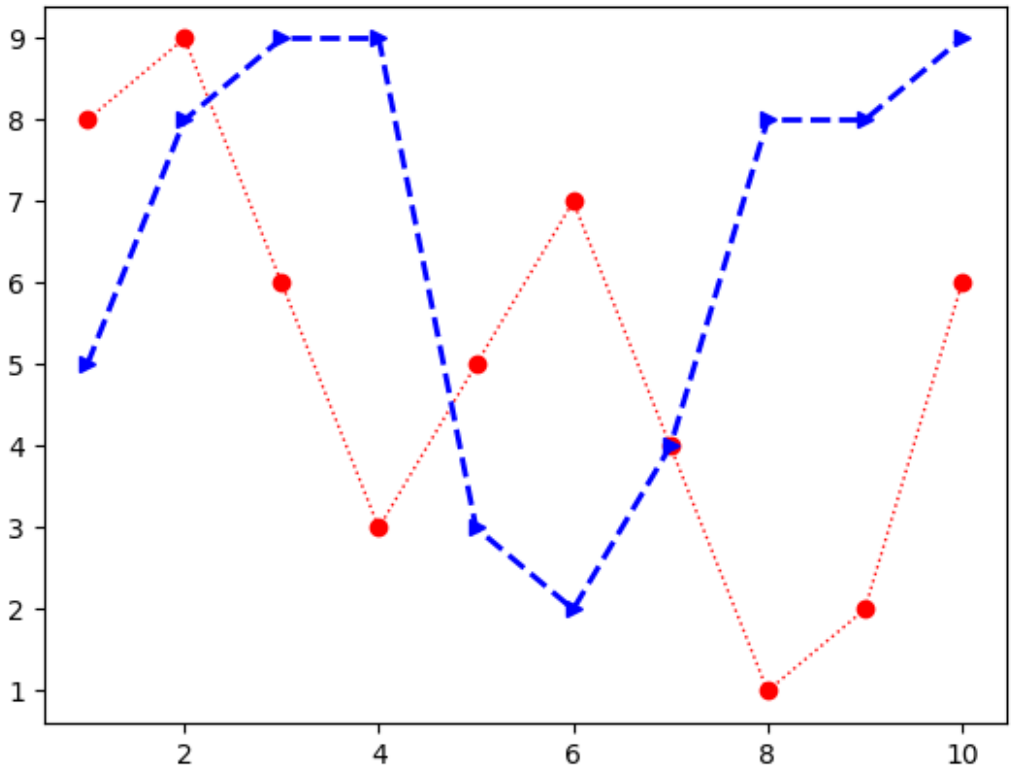



```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarkss = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]
plt.plot(x,cxMarkss)
plt.plot(x,cyMarks)
plt.show()
# write a code to display two lines in a line chart (data given above)
```



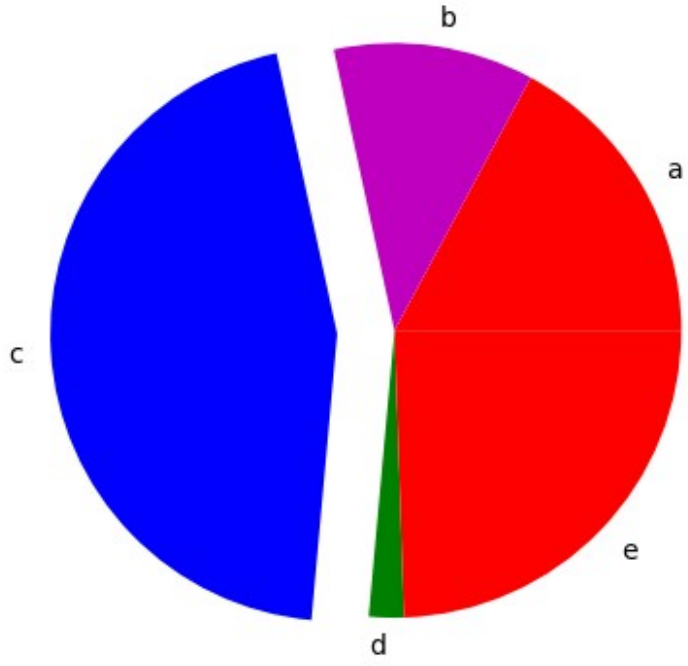

```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,9,3,2,4,8,8,9]
plt.plot(x,cxMarks,c='r',lw=1,ls=':',marker="o")
plt.plot(x,cyMarks,c='b',ls='--',marker='>',lw=2)

# write a code to generate below graph
```

```
[<matplotlib.lines.Line2D at 0x2154fe8e350>]
```



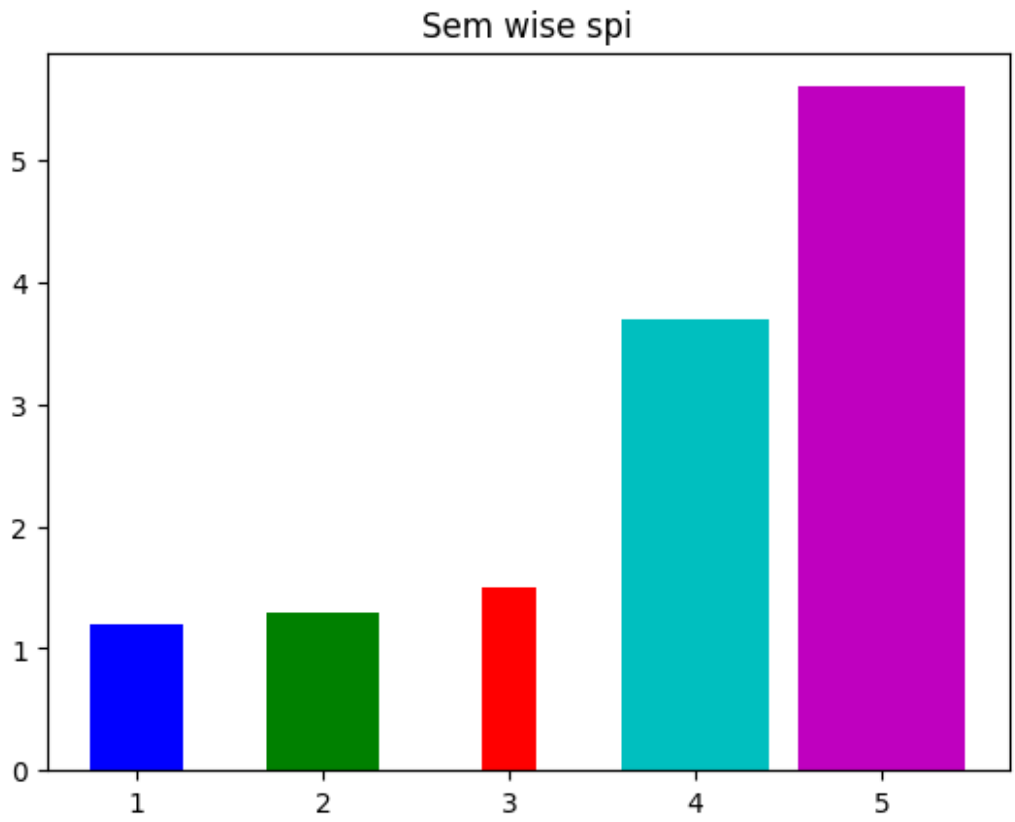
04) WAP to demonstrate the use of Pie chart.

```
values=[305,201,805,35,436]
ns=["a","b","c","d","e"]
c=["r","m","b","g","r"]
e=[0,0,0.2,0,0]
plt.pie(values,labels=ns,explode=e,colors=c)
plt.show()
```



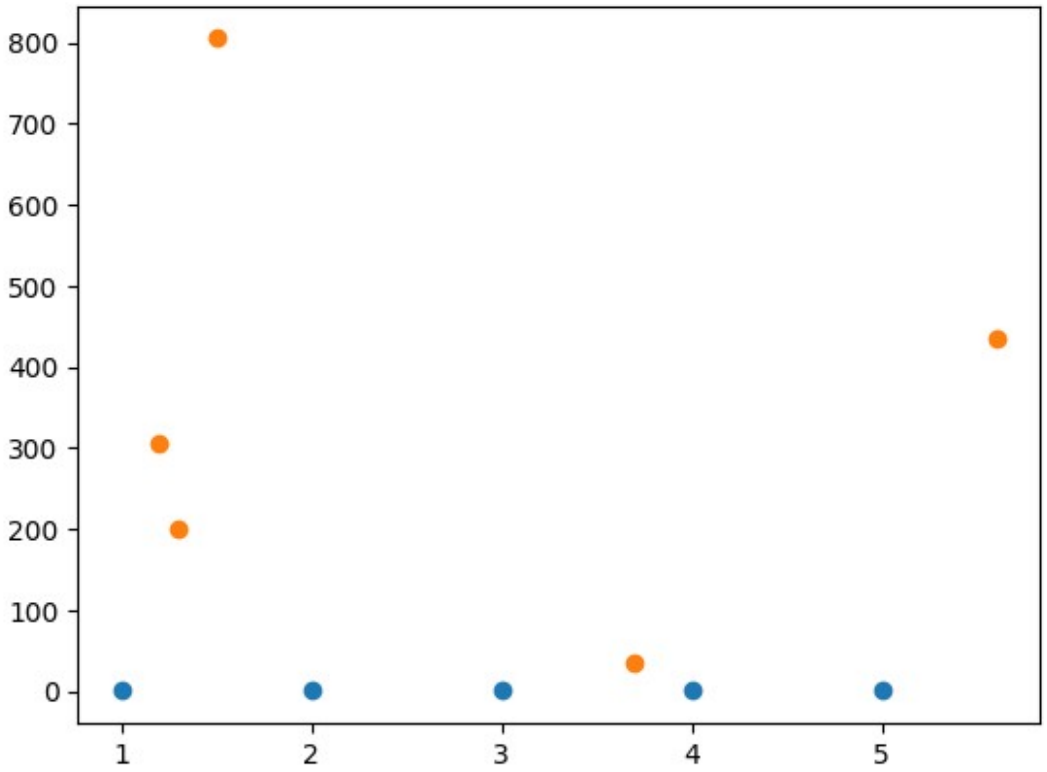
05) WAP to demonstrate the use of Bar chart.

```
import matplotlib.pyplot as plt


x=[1,2,3,4,5]
y=[1.2,1.3,1.5,3.7,5.6]
l=["1","2","3","4","5"]
c=['b','g','r','c','m']
w=[0.5,0.6,0.3,0.8,0.9]
plt.title("Sem wise spi")
plt.bar(x,y,label=l,color=c,width=w)
plt.show()
```



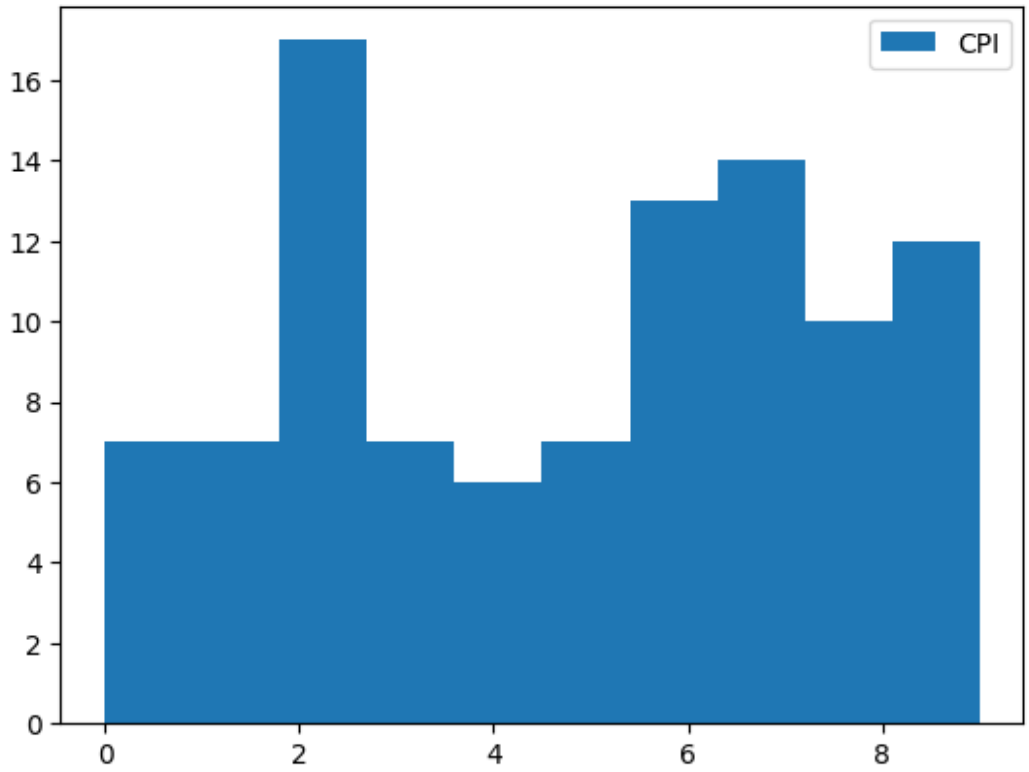
06) WAP to demonstrate the use of Scatter Plot.

```
x=[1,2,3,4,5]
y=[1.2,1.3,1.5,3.7,5.6]
w=[0.5,0.6,0.3,0.8,0.9]
values=[305,201,805,35,436]
plt.scatter(x,w)
plt.scatter(y,values)
plt.show()
```



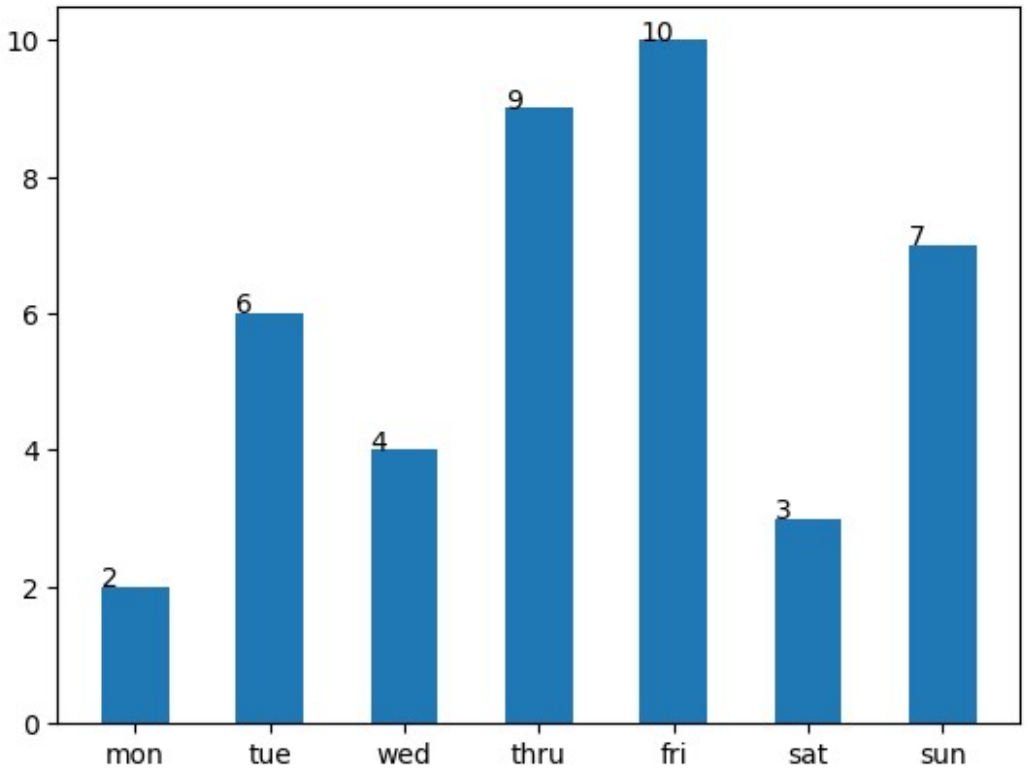
07) WAP to demonstrate the use of Histogram.

```
import numpy as np
cp=np.random.randint(0,10,100)
plt.hist(cp,bins=10,histtype='stepfilled',align='mid',label="CPI")
plt.legend()
plt.show()
```



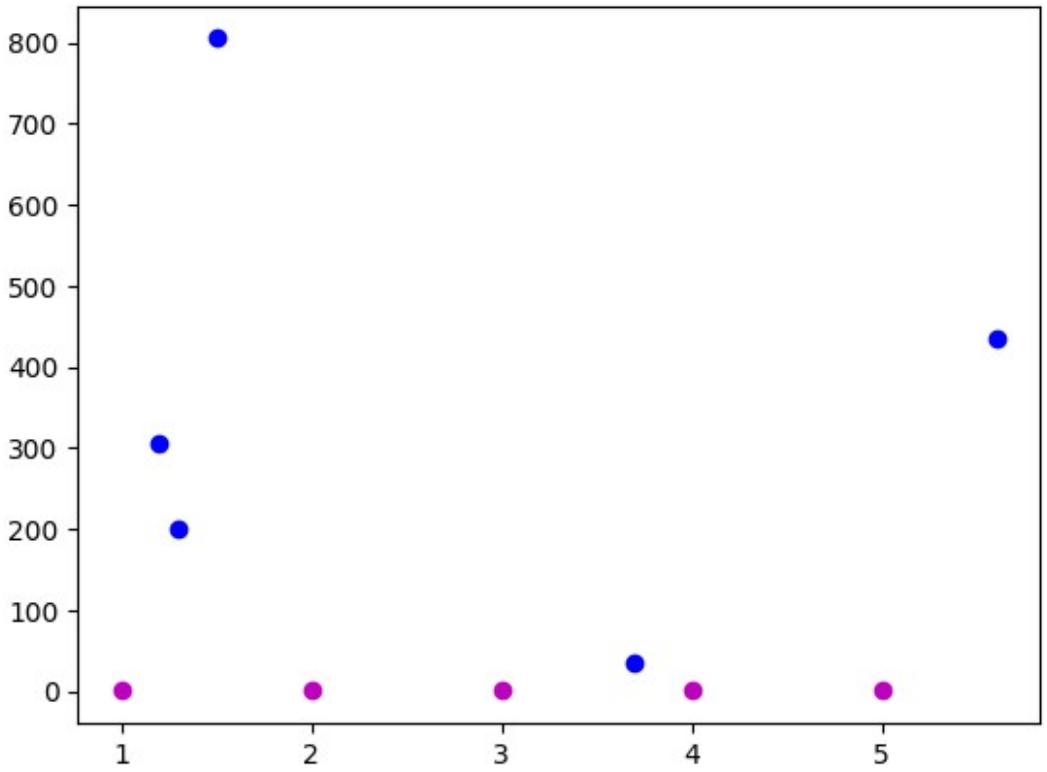
08) WAP to display the value of each bar in a bar chart using Matplotlib.

```
day=["mon","tue","wed","thru","fri","sat","sun"]
val=[2,6,4,9,10,3,7]
bar=plt.bar(day,val,0.5)
for i in bar:
    y=i.get_height()
    plt.text(i.get_x(),y,f"{y}")
plt.show()
```



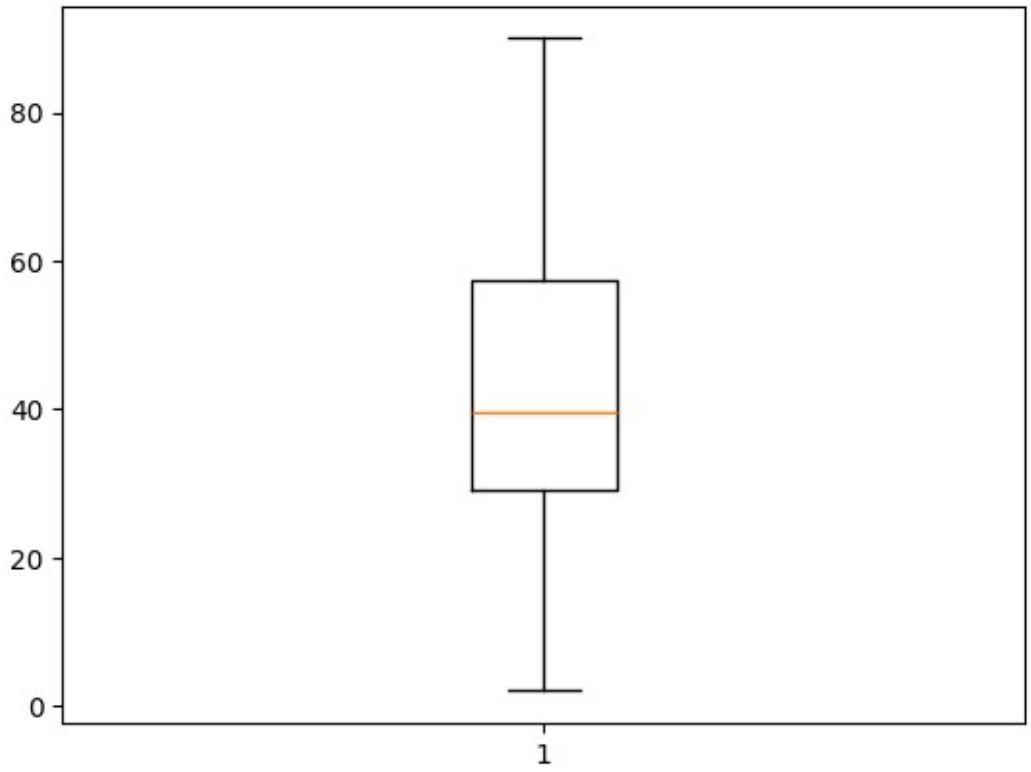
09) WAP create a Scatter Plot with several colors in Matplotlib?

```
x=[1,2,3,4,5]
y=[1.2,1.3,1.5,3.7,5.6]
w=[0.5,0.6,0.3,0.8,0.9]
values=[305,201,805,35,436]
plt.scatter(x,w,color='m')
plt.scatter(y,values,color='b')
plt.show()
```



10) WAP to create a Box Plot.

```
plt.boxplot([50,34,58,2,56,87,34,9,23,78,45,29,29,90])
plt.show()
```


80
60
40
20
0
1

OOP

01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

```
class Students:
    name="Mansi";
    age=18;
    grade="A";
s=Students();
print(s.name,s.age,s.grade)
```

```
Mansi 18 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.

```
class Bank_Account:
    Account_No = ''
    User_Name = ''
    Email = ''
    Account_Type = ''
    Account_Balance = 0
    def
GetAccountDetails(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 DisplayAccountDetails(self):
        print("Account_No :",self.Account_No)
        print("User_Name :",self.User_Name)
        print("Email :",self.Email)
        print("Account_Type :",self.Account_Type)
        print("Account_Balance :",self.Account_Balance)

#This is for main method
if __name__ == '__main__':
```

```
    Bn = Bank_Account()

Bn.GetAccountDetails('123654987','Shruti','shruti@gmail.com','valid',2
0000000)
    Bn.DisplayAccountDetails()
```

```
Account_No : 123654987
User_Name : Shruti
Email : shruti@gmail.com
Account_Type : valid
Account_Balance : 20000000
```

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

```
import math
class Circle:
    r=0
    def Area(self,r):
        self.r=r
        return math.pi*r*r
    def peri(self,r):
        self.r=r;
        return 2*math.pi*self.r
c=Circle()
print(c.Area(7))
print(c.peri(5))
```

```
153.93804002589985
31.41592653589793
```

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

```
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):
        return f"Name: {self.name}, Age: {self.age}, Salary:
{self.salary}"

# Example usage
emp1 = Employee("Alice", 30, 50000)
print(emp1.display_info())  # Output: Name: Alice, Age: 30, Salary:
$50000

print("\nupdate information......\n")
emp1.update_info(age=31, salary=55000)
print(emp1.display_info())  # Output: Name: Alice, Age: 31, Salary:
$55000
```

```
Name: Alice, Age: 30, Salary: 50000

update information......

Name: Alice, Age: 31, Salary: 55000
```

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

```
class Bank:
    balance=0
    def deposit(self,balance,money):
        self.balance=balance;
        balance+=money
        return balance
    def Withdraw (self,balance,money):
        self.balance=balance
        balance-=money
        return balance
    def check_balance(self):
        return self.balance
b=Bank();
print(b.deposit(2000,3000))
print(b.Withdraw(2000,1000))
print(b.check_balance())
```

```
5000
1000
2000
```

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

```
class Managing_inventory:
    def _init_ (self):
        pass

    def AddItems(self):
        self.name = input("Enter your items name: ")
        self.price = float(input("Enter item price :"))
        self.quantity = int(input("Enter a quantity of your item:"))

    def DisplayItem(self):
        print("Item name is :",self.name)
        print("Item price is :",self.price)
        print("Item quantity is : ",self.quantity)

    def  UpdateItem(self,name=None,price=None,quantity=None):
        if name:
            self.name = name
        if price:
            self.price = price
        if quantity:
            self.quantity = quantity

    def DeleteItem(self):
        self.name = None
        self.price = None
        self.quantity = None

items = Managing_inventory()
items.AddItems()

print("\n after add data...\n")
items.DisplayItem()

print("\n after update Items........\n")
items.UpdateItem(name='laptop' , price =1000000)
items.DisplayItem()

print("\n after delete Items........\n")
items.DeleteItem()
items.DisplayItem()
```

```
Enter your items name:  wrsfgvtr
Enter item price : 24
Enter a quantity of your item: 3
```

```
 after add data...

Item name is : wrsfgvtr
Item price is : 24.0
Item quantity is :  3

 after update Items........

Item name is : laptop
Item price is : 1000000
Item quantity is :  3

 after delete Items........

Item name is : None
Item price is : None
Item quantity is :  None
```

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

```
class profile:
    def __init__(self,name,age):
        self.name=name
        self.age=age
p=profile("Mansi",19)
print(p.name,p.age)
```

```
Mansi 19
```

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.

```
class student_kit:
    def __init__(self,name,days):
        self.name=name
        self.days=days

    def attendance(self,days):
        self.days=days
    def generate_certificate(self):
            return (f"Certificate of Attendance\n"
```

```
                    f"This is to certify that {self.name} has attended
{self.days} days of class.\n")
s=student_kit("mansi",100)
print(s.generate_certificate())
```

```
Certificate of Attendance
This is to certify that mansi has attended 100 days of class.
```

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

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

    def add(self, other):
        total_minutes = self.minute + other.minute
        extra_hours = total_minutes // 60
        minutes = total_minutes % 60
        hours = self.hour + other.hour + extra_hours
        return Time(hours, minutes)

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

time1 = Time(2, 50)
time2 = Time(1, 30)
time3 = time1.add(time2)
time3.display()
```

```
4 hours 20 minutes
```

Continued..

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

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

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

def calculate_area(rect):
    return rect.area()

rect1 = Rectangle(10, 15)

print("Area of the rectangle:", calculate_area(rect1))
```

```
Area of the rectangle: 150
```

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().

```
class square:
    def __init__(self,length):
        self.length=length;
    def area(self):
        ans=self.length*self.length
        self.ouptut(ans)
    def ouptut(self,ans):
        print(ans)
s=square(10)
s.area()
```

```
100
```

12) Calculate the area of a rectangle.

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

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

```
class Rectangle:
    def __init__(self, length, width):
        if length == width:
            print("THIS IS SQUARE.")
        else:
            self.length = length
            self.width = width

    def area(self):
        return self.output(self.length * self.width)

    def output(self, area):
        print("Area of the rectangle: ",area)

    @classmethod
    def validate(cls, length, width):
        if length == width:
            print("THIS IS SQUARE.")
            return None
        return cls(length, width)

side1 = int(input("Enter side1:"))
side2 = int(input("Enter side2:"))
rect1 = Rectangle.validate(side1, side2)
if rect1:
    rect1.area()

side1 = int(input("Enter side1:"))
side2 = int(input("Enter side2:"))
rect2 = Rectangle.validate(side1, side2)
if rect2:
    rect2.area()
```

```
Enter side1: 3
Enter side2: 5

Area of the rectangle:  15

Enter side1: 3
Enter side2: 3
```

```
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.

```
class Square:
    def __init__(self, side):
        self.__side = side  # Private attribute

    def get_side(self):
        return self.__side

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

sq = Square(50)

print("Initial Side:", sq.get_side())

sq.set_side(80)
print("Updated Side:", sq.get_side())

sq.set_side(-3)
```

```
Initial Side: 50
Updated Side: 80
Side length must be positive!
```

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

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

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

```
class profit:
    def getProfit(self):
        self.profit=int(input("Enter profit"))
class Loss:
```

```
     def getLoss(self):
        self.loss=int(input("Enter Loss"))
class BalanceSheet(profit,Loss):
    def __init__(self):
        self.profit = 0
        self.loss = 0
        self.bal = 0
    def getBalance(self):
        self.bal=self.profit-self.loss
    def printBal(self):
        print(self.bal)

bs = BalanceSheet()

bs.getProfit()
bs.getLoss()

bs.getBalance()
bs.printBal()
```

```
Enter profit 300
Enter Loss 200

100
```

15) WAP to demonstrate all types of inheritance.

```
# 1. Single Inheritance
class Parent:
    def show(self):
        print("Single Inheritance: Parent class")

class Child(Parent):
    pass

# 2. Multiple Inheritance
class Father:
    def fatherFeature(self):
        print("Multiple Inheritance: Feature from Father")

class Mother:
    def motherFeature(self):
        print("Multiple Inheritance: Feature from Mother")

class Child2(Father, Mother):
    pass

# 3. Multilevel Inheritance
class Grandparent:
    def grandFeature(self):
```

```
        print("Multilevel Inheritance: Feature from Grandparent")

class Parent2(Grandparent):
    pass

class Child3(Parent2):
    pass

# 4. Hierarchical Inheritance
class Base:
    def baseFeature(self):
        print("Hierarchical Inheritance: Base class feature")

class Derived1(Base):
    pass

class Derived2(Base):
    pass

# 5. Hybrid Inheritance (Combination of multiple types)
class A:
    def featureA(self):
        print("Hybrid Inheritance: Feature A")

class B(A):
    def featureB(self):
        print("Hybrid Inheritance: Feature B")

class C(A):
    def featureC(self):
        print("Hybrid Inheritance: Feature C")

class D(B, C):
    pass

print("\n--- Single Inheritance ---")
c1 = Child()
c1.show()

print("\n--- Multiple Inheritance ---")
c2 = Child2()
c2.fatherFeature()
c2.motherFeature()

print("\n--- Multilevel Inheritance ---")
c3 = Child3()
c3.grandFeature()

print("\n--- Hierarchical Inheritance ---")
d1 = Derived1()
```

```
d2 = Derived2()
d1.baseFeature()
d2.baseFeature()

print("\n--- Hybrid Inheritance ---")
d = D()
d.featureA()
d.featureB()
d.featureC()
```

```
--- Single Inheritance ---
Single Inheritance: Parent class

--- Multiple Inheritance ---
Multiple Inheritance: Feature from Father
Multiple Inheritance: Feature from Mother

--- Multilevel Inheritance ---
Multilevel Inheritance: Feature from Grandparent

--- Hierarchical Inheritance ---
Hierarchical Inheritance: Base class feature
Hierarchical Inheritance: Base class feature

--- Hybrid Inheritance ---
Hybrid Inheritance: Feature A
Hybrid Inheritance: Feature B
Hybrid Inheritance: Feature C
```

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.

```
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 display(self):
        print("Name:",self.name, "\nAge:", self.age, "\nSalary:",
self.salary)

emp = Employee("Shruti", 18, 5500000)
emp.display()
```

```
Name: Shruti
Age: 18
Salary: 5500000
```

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.

```
from abc import ABC, abstractmethod #Abstract Base Class(Module)

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
```