```
<a href='https://www.darshan.ac.in/'> <img
src='https://www.darshan.ac.in/Content/media/DU_Logo.svg' width="250"
height="10"/></a>
<pre>
<center><b><h1>Python Programming - 2301CS404</b></center>
    <center><b><h1>Harsh Khant</b></center>
    <center><b><h1>23010101140</b></center>
    <center><b><h1>166</b></center>
<center><b><h1>Lab - 1</b></center>
<pre>
```

01) WAP to print “Hello World”

```
print("Hello World")
```

```
Hello World
```

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

```
#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.

```
a=5
print(type(a))
b="Harsh"
print(type(b))
```

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

04) WAP to calculate simple interest.

```
#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.

```
#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.

```
#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.

```
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.

```
#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.

```
#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.

```
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.

```
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.

```
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.

```
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.

```
#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.

```
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
```

if..else..

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

```
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.

```
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.

```
#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.

```
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]

```
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.

```
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.

```
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

```
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.

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

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

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

```
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

```
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
```

for and while loop

01) WAP to print 1 to 10.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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

```
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.

```
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
```

Python Programming - 2301CS404 Lab - 4

Harsh Khant
23010101140
166

String

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

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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

```
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

```
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

```
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
```

List

Python Programming - 2301CS404 Lab - 5 Harsh Khant 23010101140 166

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

```
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.

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

```
5
```

03) WAP to find the length of a List.

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

```
5
```

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

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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

[]
```

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

```
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.

```
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.

```
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]
```

Tuple

01) WAP to find sum of tuple elements.

```
t1=(1,2,3,4,5)
print(sum(t1))
```

```
15
```

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

```
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.

```
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.

```
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.

```
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)
```

```
(4, 5)
```

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

```
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.

```
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.

```
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.

```
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.

```
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
```

Set & Dictionary

01) WAP to iterate over a set.

```
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.

```
a={1,2,3,4,5}
type(a)

b=list(a)
type(b)

t=tuple(a)
type(t)

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

```
str
```

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

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

```
5
1
```

04) WAP to perform union of two sets.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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)

```
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.

```
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
```

User Defined Function

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

```
def calculate_bmi(mass, height):
    bmi = mass / (height ** 2)
    return bmi
mass = float(input("Enter your mass in kilograms: "))
height = float(input("Enter your height in meters: "))
bmi = calculate_bmi(mass, height)
print(f"Your BMI is: {bmi:.2f}")
```

```
Enter your mass in kilograms:  65
Enter your height in meters:  1.65

Your BMI is: 23.88
```

02) Write a function that add first n numbers.

```
def sum_of_first_n_numbers(n):
    return n * (n + 1) // 2
n = int(input("Enter a positive integer (n): "))
result = sum_of_first_n_numbers(n)
print(f"The sum of the first {n} natural numbers is: {result}")
```

```
Enter a positive integer (n):  25

The sum of the first 25 natural numbers is: 325
```

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

```
def is_prime(num):
    if num <= 1:
        return 0
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return 0
    return 1
number = int(input("Enter a number: "))
result = is_prime(number)
print(f"The number {number} is prime: {result}")
```

```
Enter a number:  13
```

```
The number 13 is prime: 1
```

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

```
def is_prime(num):

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

def primes_between(start, end):
    prime_list = []
    for num in range(start, end + 1):
        if is_prime(num):
            prime_list.append(num)
    return prime_list
start = int(input("Enter the starting number: "))
end = int(input("Enter the ending number: "))

prime_numbers = primes_between(start, end)
print(f"Prime numbers between {start} and {end}: {prime_numbers}")
```

```
Enter the starting number:  25
Enter the ending number:  56

Prime numbers between 25 and 56: [29, 31, 37, 41, 43, 47, 53]
```

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

```
def is_palindrome(s):
    s = s.replace(" ", "").lower()
    # Compare the string with its reverse
    return s == s[::-1]
input_string = input("Enter a string: ")
result = is_palindrome(input_string)
print(f"Is the string a palindrome? {result}")
```

```
Enter a string:  bab

Is the string a palindrome? True
```

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

```
def sum_of_list_elements(lst):
    return sum(lst)
```

```
numbers = list(map(float, input("Enter elements of the list separated
by spaces: ").split()))
result = sum_of_list_elements(numbers)
print(f"The sum of all elements in the list is: {result}")
```

```
Enter elements of the list separated by spaces:  2 5 5 2 2

The sum of all elements in the list is: 16.0
```

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

```
def sum_of_first_elements(tuples_list):
    return sum(t[0] for t in tuples_list)
tuples_list = [(1, 2), (3, 4), (5, 6)]
result = sum_of_first_elements(tuples_list)
print(f"The sum of the first elements is: {result}")
```

```
The sum of the first elements is: 9
```

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

```
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
n = int(input("Enter the term position (n): "))
result = fibonacci(n)
print(f"The {n}th term of the Fibonacci Series is: {result}")
```

```
Enter the term position (n):  11

The 11th term of the Fibonacci Series is: 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

```
def get_student_name(rollno, student_dict):
    return student_dict.get(rollno, "Roll number not found")
dict1 = {101: 'Ajay', 102: 'Rahul', 103: 'Jay', 104: 'Pooja'}
rollno = int(input("Enter the roll number: "))
name = get_student_name(rollno, dict1)
print(f"The name of the student with roll number {rollno} is: {name}")
```

```
Enter the roll number:  102

The name of the student with roll number 102 is: 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

```python
def sum_of_scores_ending_with_zero(scores):
    return sum(score for score in scores if score % 10 == 0)
scores = [200, 456, 300, 100, 234, 678]
result = sum_of_scores_ending_with_zero(scores)
print(f"The sum of scores ending with zero is: {result}")
```

```
The sum of scores ending with zero is: 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'}

```python
def invert_dictionary(original_dict):
    return {value: key for key, value in original_dict.items()}
original_dict = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
inverted_dict = invert_dictionary(original_dict)
print("Original Dictionary:", original_dict)
print("Inverted Dictionary:", inverted_dict)
```

```
Original Dictionary: {'a': 10, 'b': 20, 'c': 30, 'd': 40}
Inverted Dictionary: {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.

```python
def is_pangram(s):
    s = s.lower()
    alphabet_set = set("abcdefghijklmnopqrstuvwxyz")
    return alphabet_set.issubset(set(s))
input_string = input("Enter a string: ")
if is_pangram(input_string):
    print("The string is a pangram.")
```

```
else:
    print("The string is not a pangram.")
```

```
Enter a string:  abcdefghijklmnopqrstuvwxyz

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

```
def count_upper_lower(s):
    no_upper = 0
    no_lower = 0
    for char in s:
        if char.isupper():
            no_upper += 1
        elif char.islower():
            no_lower += 1
    return no_upper, no_lower
s1 = input('Enter a String:')
no_upper, no_lower = count_upper_lower(s1)
print(f"Number of uppercase letters: {no_upper}")
print(f"Number of lowercase letters: {no_lower}")
```

```
Enter a String: abccxsGSGABA

Number of uppercase letters: 6
Number of lowercase letters: 6
```

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

```
smallest = lambda x, y: x if x < y else y
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
result = smallest(num1, num2)
print(f"The smallest number is: {result}")
```

```
Enter the first number:  25
Enter the second number:  58

The smallest number is: 25.0
```

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

```
students = ["Alice", "Bob", "Alexander", "Catherine", "David",
"Elizabeth"]
def has_more_than_7_chars(name):
    return len(name) > 7
filtered_names = list(filter(has_more_than_7_chars, students))
print("Names with more than 7 characters:", filtered_names)
```

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

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

```
students = ["alice", "bob", "alexander", "catherine", "david",
"elizabeth"]
def capitalize_first_letter(name):
    return name.capitalize()
capitalized_names = list(map(capitalize_first_letter, students))
print("Names with first letter capitalized:", capitalized_names)
```

```
Names with first letter capitalized: ['Alice', 'Bob', 'Alexander',
'Catherine', 'David', 'Elizabeth']
```

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

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

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

def default_args(a, b=5):
    return a * b

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

def variable_length_keyword(**kwargs):
    return kwargs
```

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

def positional_only_args(a, b, /):
    return a * b

print("Positional Arguments Result:", positional_args(10, 20))
print("Keyword Arguments Result:", keyword_args(b=20, a=10))
print("Default Arguments Result:", default_args(10))
print("Variable-Length Positional Arguments Result:",
variable_length_positional(1, 2, 3, 4, 5))
print("Variable-Length Keyword Arguments Result:",
variable_length_keyword(name="Alice", age=25, city="New York"))
print("Keyword-Only Arguments Result:", keyword_only_args(a=10, b=20))
print("Positional-Only Arguments Result:", positional_only_args(10,
20))
```

```
Positional Arguments Result: 30
Keyword Arguments Result: -10
Default Arguments Result: 50
Variable-Length Positional Arguments Result: 15
Variable-Length Keyword Arguments Result: {'name': 'Alice', 'age': 25,
'city': 'New York'}
Keyword-Only Arguments Result: 30
Positional-Only Arguments Result: 200
```

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('test1.txt','w+')
fp.write('Hello\n llkjhgfghjk')
fp.seek(0)
rd=fp.read()
print(rd)
fp.close()
```

```
Hello
 llkjhgfghjk
```

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

```
fp=open('new.txt','w')
fp.write('hello')
fp.close()
```

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

```
fp=open('readLines.txt','r')
for i in range(5):
    print(fp.readline())
fp.close()
```

```
afhaygfajgf

adnbhfbkjas

jfjlkafn

anfknal

ansfdljkakf
```

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

```
fp=open('readLines.txt','r')
a=fp.read().split()
b=max([len(i) for i in a])
lw=[i for i in a if len(i) == b]
for i in lw:
    print(i)
fp.close()
```

```
afhaygfajgf
adnbhfbkjas
ansfdljkakf
```

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

```
fp=open('test1.txt')
st=fp.read()
li=st.split('\n')
lines=len(li)
li1=st.split()
words=len(li1)
ch=0
for i in li1:
    ch+=len(i)
    ch-=i.count('\n')
print('lines:',lines)
print('words:',words)
print('character:',ch)
```

```
lines: 2
words: 2
character: 16
```

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

```
fp=open('test1.txt')
st=fp.read()
li=st.split('\n')
fp.close()
fp1=open('test2.txt','w')
for i in li:
    fp1.write(str(i)+'\n')
fp1.close()
```

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

```
fp=open('test1.txt')
print(fp.seek(0,2))
fp.close()
```

```
19
```

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

```
def frequency():
    sword = input("Enter the word to count: ")
    fp=open('test1.txt','r')
    d=fp.read()
    w=d.lower().split()
    sword=sword.lower()
    c=w.count(sword)
    print('Count is:',c)
    fp.close()

frequency()
```

```
Enter the word to count:  Hello

Count is: 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.

```


fp=open('test3.txt','w')
ent=[input(f"Enter score for subject {i+1}: ") for i in range(5)]
fp.write('\n'.join(ent))
print(ent)
fp.close()

fp=open('test3.txt','r')
sc=[int(i) for i in fp]
fp.close()

print(sc)
print(max(sc))
```

```
Enter score for subject 1:  10
Enter score for subject 2:  20
Enter score for subject 3:  30
Enter score for subject 4:  040
Enter score for subject 5:  5
```

```
['10', '20', '30', '040', '5']
[10, 20, 30, 40, 5]
40
```

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

(Note: each number should be in new line)

```python
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

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

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

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

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

```python
try:
    with open("file1.txt", "r") as f1, open("file2.txt", "r") as f2,
open("mergedfile.txt", "w") as out:
        out.write(f1.read())
        out.write("\n")
        out.write(f2.read())

    print("Files merged successfully into 'mergedfile.txt'.")
except FileNotFoundError as e:
    print(f"Error: {e}")
```

|

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

```
f = open("demofile.txt", "r")
print(f.tell())

f = open("abG.txt", "r")

f.seek(20)

print(f.tell())

print(f.readline())
f.close()
```

Python Programming - 2301CS404 Lab - 10

Harsh Khant
166
23010101140

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.

```
# 1-----ZeroDivisonError
try:
    a=1
    b=0
    c=a/b
    print(c)
except ZeroDivisionError:
    print('value of b is not zero')


# 2-----ValueError
try:
    a='abc'
    b=int(a)
    print(b)

except ValueError:
    print('You cannot convert string into integer')

# 3-----TypeError
try:
    a='abc'
    b=5
    c=a+b
    print(c)

except TypeError:
    print('You cannot concat string+integer')
```

```
You cannot concat string+integer
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
# 1-----IndexError
li=[1,2,3,4]
try:
    li[5]
except IndexError:
    print('In List index is out of bound ')


# 2-----KeyError
try:
    k={1:'a',2:'b',3:'c'}
    k[4]
except KeyError :
    print('In dictionary key not found')



In dictionary key not found
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
# 1-----FileNotFoundError
try:
    fp=open('abc.txt','r')
except FileNotFoundError:
    print('File not found')

# 2-----ModuleNotFoundError
try:
    import ha
except ModuleNotFoundError:
    print('You can import available module')



You can import available module
```

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

```
try:
    'a'+10
except BaseException as err:
    print(err)
```

```
can only concatenate str (not "int") to str
```

05) WAP to demonstrate else and finally block.

```
try:
    a=10/0
except BaseException as err:
    print(err)
else:
    print(a)
finally:
    print('You can successfully run the code')
```

```
division by zero
You can successfully run the code
```

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.

```
g= input("Enter a list of grades separated by commas: ")
try:
    grades = [int(grade) for grade in g.split(',')]
    print("grades:", grades)
except ValueError:
    print("String can't be converted into integer")
```

```
Enter a list of grades separated by commas:  A

String can't be converted into integer
```

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

```
def divide(a,b):
    try:
        c=a/b
    except :
        print('You cannot divide by zero')

divide(1,0)
```

```
You 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.

```
try:
    age=14
    a=int(age)
    if a<18:
        raise Exception('Enter Valid Age')
    else:
        print('age:',age)
except ValueError as err:
    print(err)
```

```
---------------------------------------------------------------------------
-----
Exception                                 Traceback (most recent call
last)
Cell In[49], line 5
      3 a=int(age)
      4 if a<18:
----> 5     raise Exception('Enter Valid Age')
      6 else:
      7     print('age:',age)

Exception: 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.

```
try:
    name='har'
    if len(name)<5 or len(name)>15:
        raise Exception('InvalidUsernameError')
    else:
        print(name)
except Exception :
    print('Username must be between 5 and 15 characters long')
```

```
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.

```
try:
    import math
    num=-14
    if num<0:
        raise Exception('Cannot calculate the square root of negative
number')
    else:
        b=math.sqrt(num)
        print(b)
except Exception as err:
    print(err)
```

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

Harsh Khant 23010101140 166

Modules

```
import random
import math
import CalculatorModule
import datetime
from datetime import date
```

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.

```

print('Enter a number to select operation:')
print('number 1 for add')
print('number 2 for sub')
print('number 3 for mul')
print('number 4 for div')

a=int(input('Enter number 1'))
b=int(input('Enter number 2'))

ch=int(input('Enter your choice number:'))

match ch:
    case 1:print(CalculatorModule.add(a,b))
    case 2:print(CalculatorModule.sub(a,b))
    case 3:print(CalculatorModule.mul(a,b))
    case 4:print(CalculatorModule.div(a,b))
```

```
Enter a number to select operation:
number 1 for add
number 2 for sub
number 3 for mul
number 4 for div

Enter number 1 5
Enter number 2 10
Enter your choice number: 1

15
```

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

```
a=input('Enter a string')
ans=random.choice(a)
print(ans)
```

```
Enter a string HARSH

H
```

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

```
li=[1,2,3,4,5,6]
ans=random.choice(li)
print(ans)
```

```
1
```

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

```
random.seed(2)
print(random.randint(2,4))
```

```
2
```

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

```
for i in range(3):
    print(random.randrange(100,1000,5))
```

```
575
795
665
```

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

```
tic=[random.randrange(1,1000) for i in range(1,101)]
d=random.sample(tic,2)
print('winner:',d[0],'\nRunnerUp',d[1])
```

```
winner: 723
RunnerUp 637
```

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

```
ans=datetime.datetime.now()
print(ans)
```

```
2025-02-14 13:17:54.438854
```

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

```
ans=datetime.datetime.now()
b=ans-datetime.timedelta(days=7)
print(b)
```

```
2025-02-07 13:20:44.968793
```

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

```
d1=date(2005,2,21)
d2=date(2006,1,21)
print(d2-d1)
```

```
334 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.)

```
a=datetime.date(2005,2,21)
day=a.strftime("%A")
print(day)
```

```
Monday
```

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

```
ans=datetime.datetime.now()
b=ans-datetime.timedelta(days=7)
print(b)
```

```
2025-02-07 13:26:37.908410
```

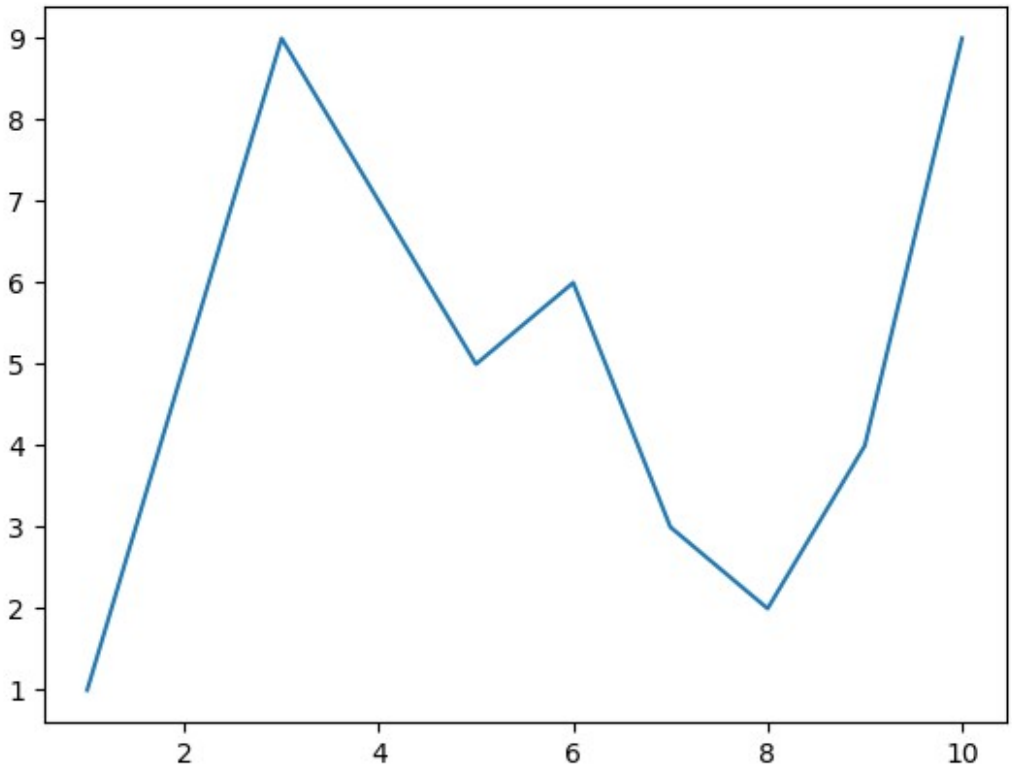
12) WAP to demonstrate the use of the math module.

```
a=5
b=10
print(math.gcd(a,b))
```

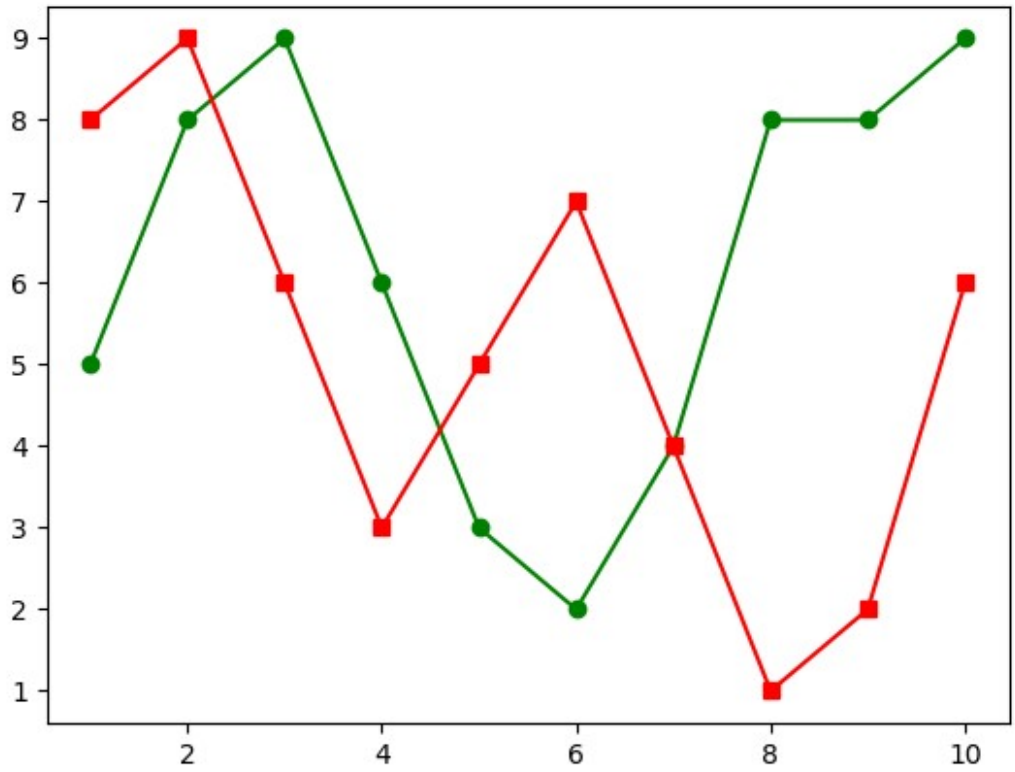
```
5
```

Python Programming - 2301CS404 Lab - 12 Harsh Khant
23010101140
166

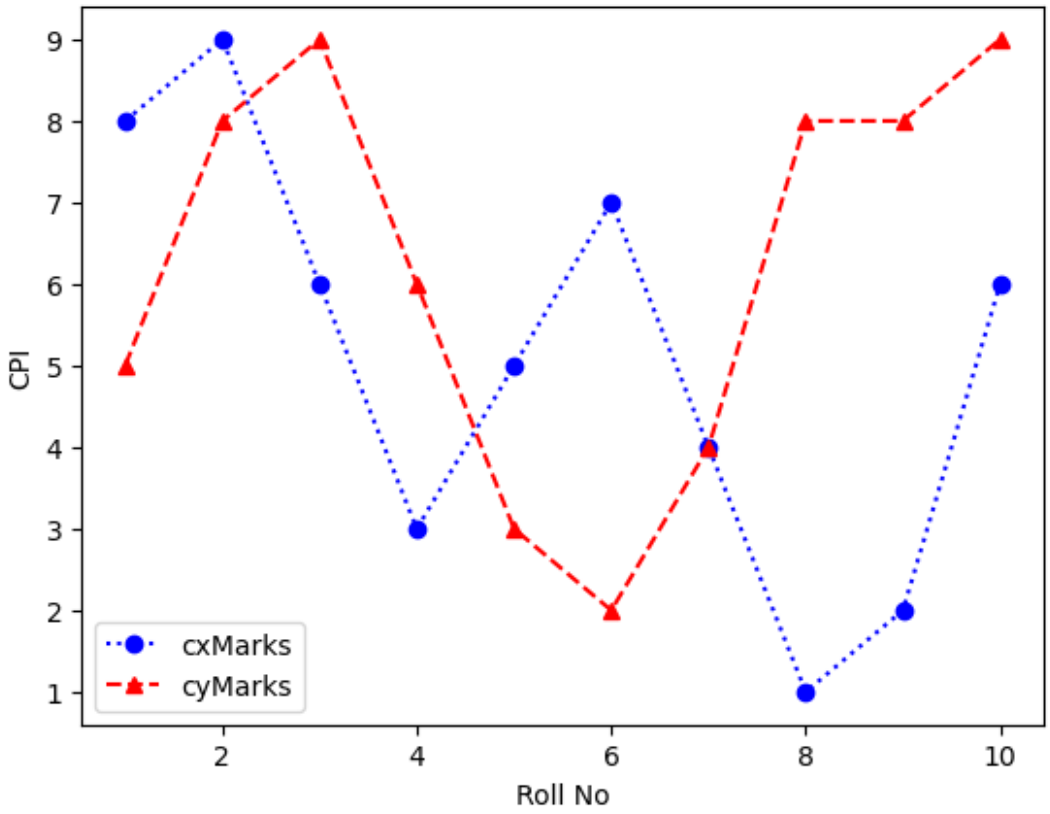
```
#import matplotlib below

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
```




```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]
plt.plot(x,cxMarks,label="cxMarks",color="g",marker="o")
plt.plot(x,cyMarks,label="cyMarks",color="r",marker="s")
plt.show()

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

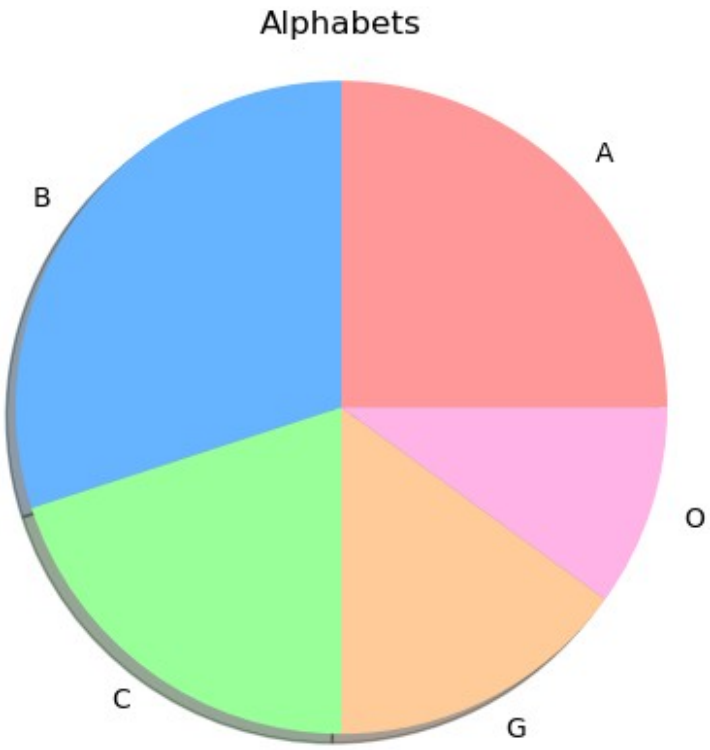



```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks,label="cxMarks",color="b",marker="o",linestyle="dot
ted")
plt.plot(x,cyMarks,label="cyMarks",color="r",marker="^",linestyle="das
hed")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.show()


# write a code to generate below graph
```



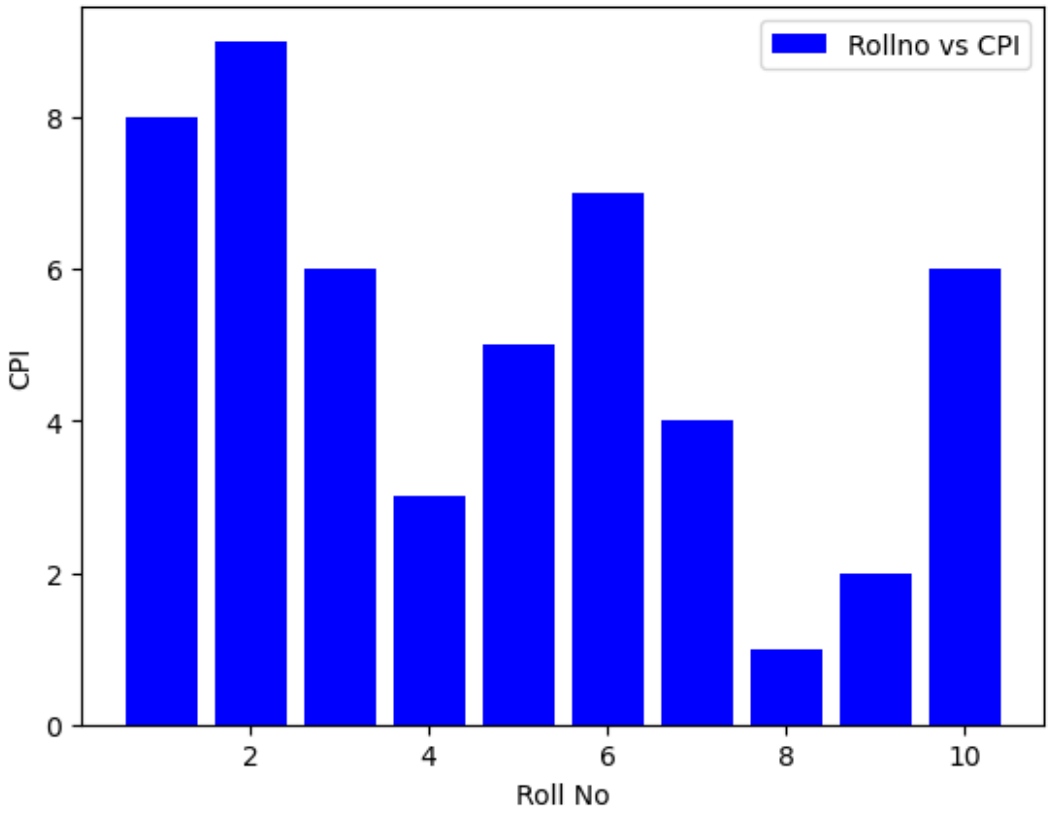
04) WAP to demonstrate the use of Pie chart.

```
labels = ['A', 'B', 'C', 'G', 'O']
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('Alphabets')
plt.show()
```



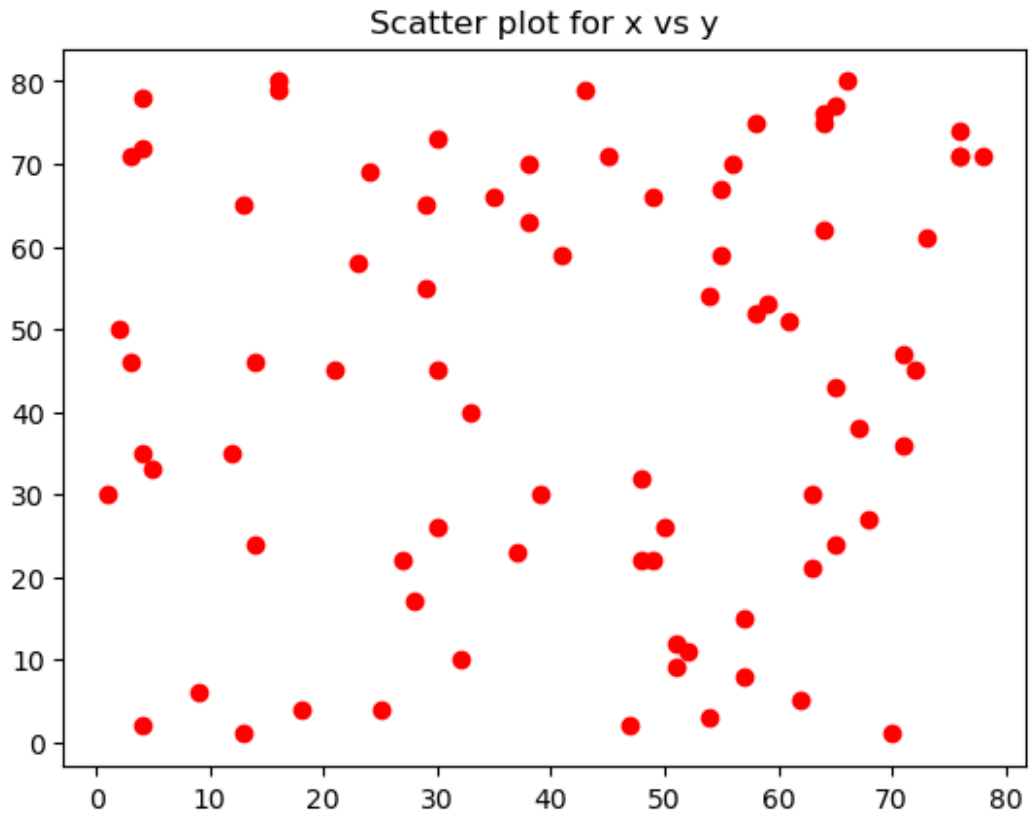
05) WAP to demonstrate the use of Bar chart.

```
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()
```



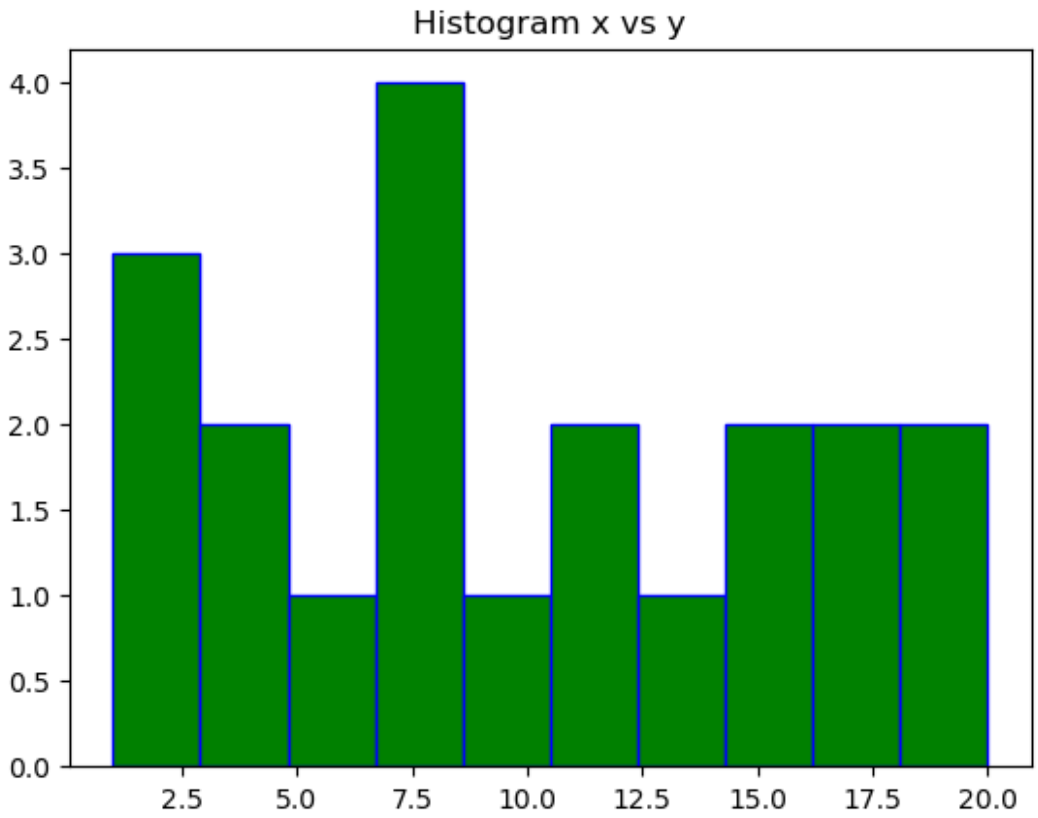
06) WAP to demonstrate the use of Scatter Plot.

```
import random as r
r.seed(1)
x = [r.randint(1,80) for i in range(80)]
y = [r.randint(1,80) for i in range(80)]
plt.scatter(x,y,color="r")
plt.title("Scatter plot for x vs y")
plt.show()
```



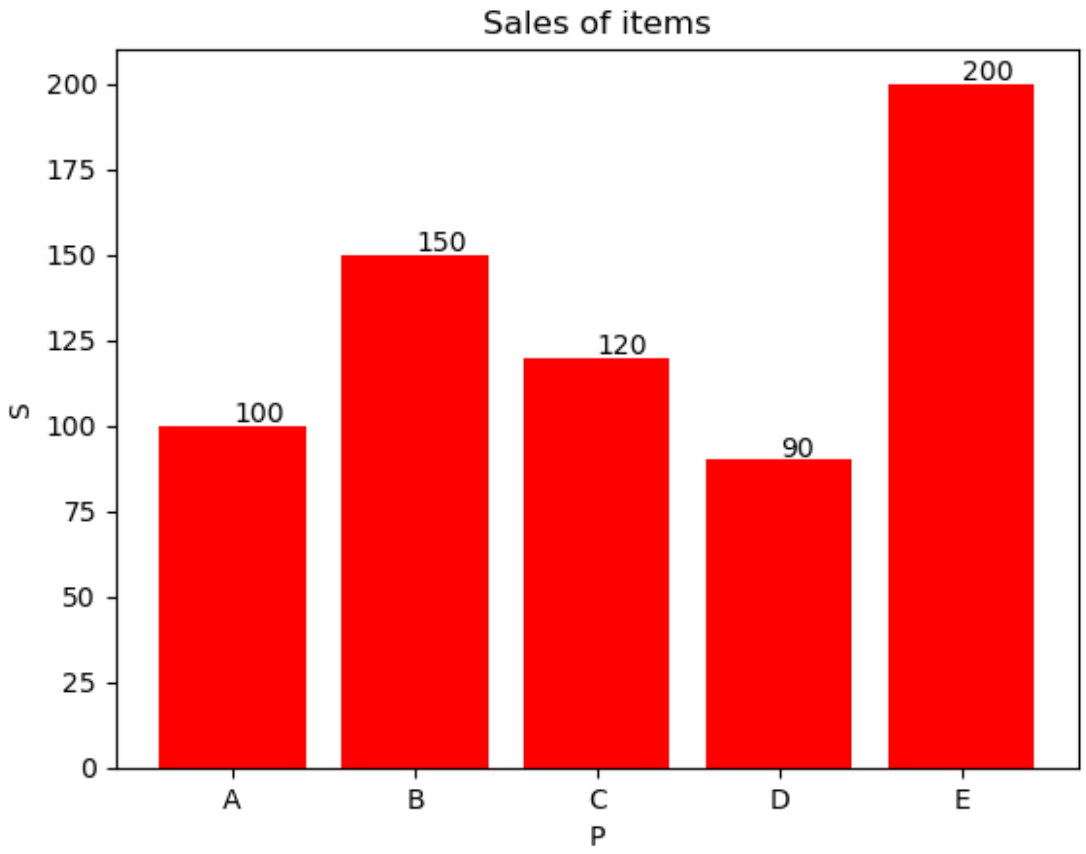
07) WAP to demonstrate the use of Histogram.

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



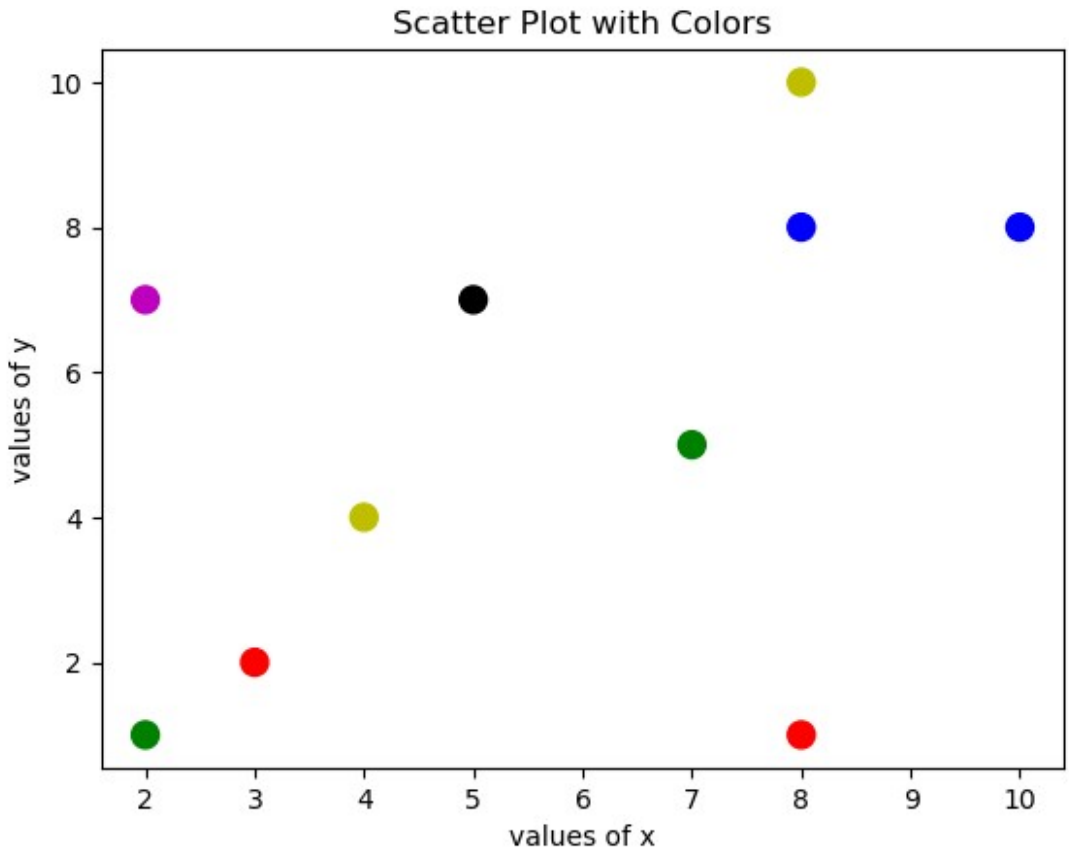
08) WAP to display the value of each bar in a bar chart using Matplotlib.

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



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

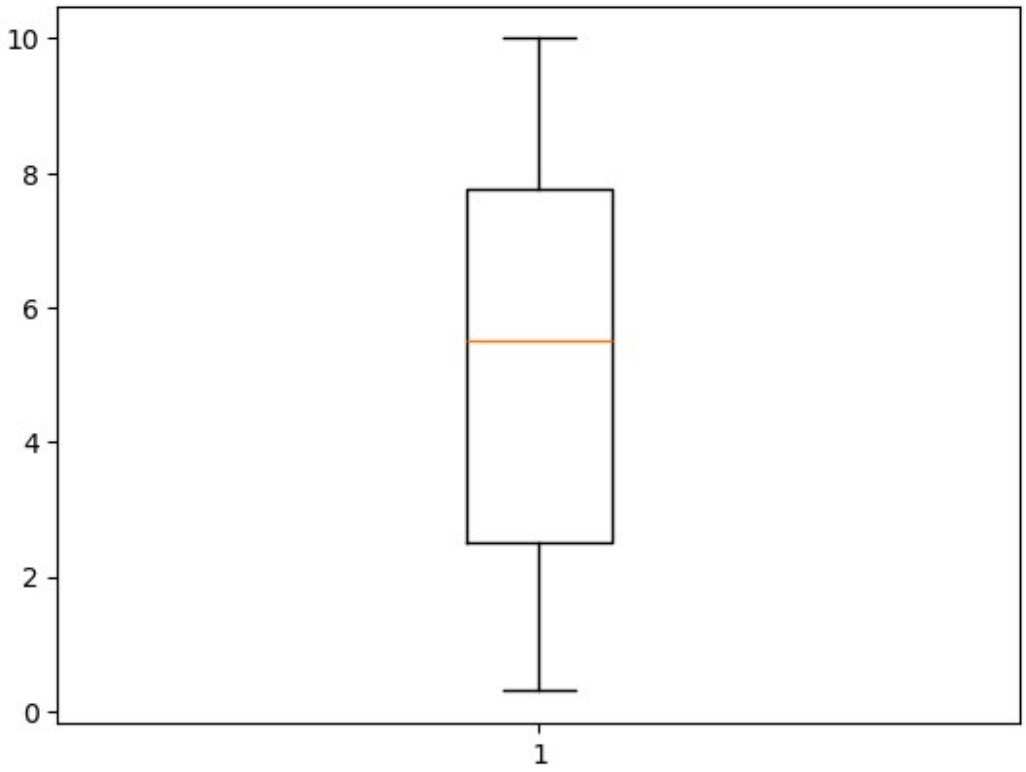
```
r.seed(1)
x = [r.randint(1,10) for i in range(10)]
y = [r.randint(1,10) for i in range(10)]
colors = ['r','b','g','k','m','y','r','b','g','y']

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



10) WAP to create a Box Plot.

```
import matplotlib.pyplot as plt
a=[1,2,.3,4,5,6,7,8,9,10]
plt.boxplot(a)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1bc0ec129c0>,
  <matplotlib.lines.Line2D at 0x1bc0ec12cc0>],
 'caps': [<matplotlib.lines.Line2D at 0x1bc0ec12f90>,
  <matplotlib.lines.Line2D at 0x1bc0ec13290>],
 'boxes': [<matplotlib.lines.Line2D at 0x1bc0ec12720>],
 'medians': [<matplotlib.lines.Line2D at 0x1bc0ec13560>],
 'fliers': [<matplotlib.lines.Line2D at 0x1bc0ec13860>],
 'means': []}
```


10
8
6
4
2
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:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade
    def displayDetails(self):
        print('name:',self.name)
        print('age:',self.age)
        print('garde:',self.grade)

s1=Students('Harsh',20,'A+')
s1.displayDetails()
```

```
name: Harsh
age: 20
garde: 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 :
    def dispalyAccountDetails(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)
    def getAccountDeatils(self):
        self.Account_No=int(input('Enter your account number:'))
        self.User_Name=input('Enter your account User_Name:')
        self.Email=input('Enter your account Email:')
        self.Account_Type=input('Enter your account Account_Type:')
        self.Account_Balance=int(input('Enter your account
Account_Balance:'))

h1=Bank_Account()
```

```
h1.getAccountDeatils()
h1.dispalyAccountDetails()


Enter your account number: 07
Enter your account User_Name: thala
Enter your account Email: thala@gmail.com
Enter your account Account_Type: thalaPrivate
Enter your account Account_Balance: 07

Account_No 7
User_Name thala
Email thala@gmail.com
Account_Type thalaPrivate
Account_Balance 7
```

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

```
import math
class circle:
    def __init__(self,r):
        self.r=r
    def areaOfCircle(self):
        print(math.pi**2)
    def permeterOfCircle(self):
        print(2*math.pi*self.r)
obj=circle(5)
obj.areaOfCircle()
obj.permeterOfCircle()


9.869604401089358
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 display_info(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Salary: ${self.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
        print("Employee information updated successfully.")


emp1 = Employee("Harsh", 30, 50000)
emp1.display_info()

emp1.update_info(age=31, salary=55000)
emp1.display_info()
```

```
Name: Harsh
Age: 30
Salary: $50000
Employee information updated successfully.
Name: Harsh
Age: 31
Salary: $55000
```

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

```
class bank_account:
    Balance=5000
    def depo(self,d):
        self.Balance =self.Balance+d
    def withd(self,w):
        self.Balance= self.Balance-w
    def displayAvailableBalance(self):
        print('Balance is:',self.Balance)
obj1=bank_account()
obj1.depo(5000)
obj1.withd(300)
obj1.displayAvailableBalance()
```

```
Balance is: 9700
```

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 Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        self.items[name] = {"price": price, "quantity": quantity}
        print(f"{name} added successfully.")

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]
            print(f"{name} removed successfully.")
        else:
            print(f"{name} not found in inventory.")

    def update_item(self, name, price=None, quantity=None):
        if name in self.items:
            if price is not None:
                self.items[name]["price"] = price
            if quantity is not None:
                self.items[name]["quantity"] = quantity
            print(f"{name} updated successfully.")
        else:
            print(f"{name} not found in inventory.")

inventory = Inventory()
inventory.add_item("Laptop", 75000, 10)
inventory.update_item("Laptop", quantity=8)
inventory.remove_item("Laptop")
```

```
Laptop added successfully.
Laptop updated successfully.
Laptop removed successfully.
```

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

```
class Vehicle:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def display_info(self):
        print(f"{self.year} {self.brand} {self.model}")
```

```
car = Vehicle("Toyota", "Corolla", 2020)
car.display_info()
```

```
2020 Toyota Corolla
```

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 StudentKit:
    principal_name = "Mr ABC"

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

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

    def generate_certificate(self, days_present):
        print(f"Certificate of Attendance\nStudent:
{self.student_name}\nPrincipal: {StudentKit.principal_name}\nDays
Present: {days_present}")

student = StudentKit("Harsh")
student.attendance(90)
student.generate_certificate(90)
```

```
Harsh attended 90 days.
Certificate of Attendance
Student: Harsh
Principal: Mr ABC
Days Present: 90
```

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

```
        total_hours = self.hour + other.hour + (total_minutes // 60)
        total_minutes = total_minutes % 60
        return Time(total_hours, total_minutes)

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

time1 = Time(2, 45)
time2 = Time(1, 30)
result = time1.add_time(time2)
result.display_time()
```

```
4 hour(s) and 15 minute(s)
```

Continued..

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

```
class areaOfRectangle:
    def __init__(self,l,b):
        self.l=l
        self.b=b
    def calculateArea(self):
        return self.l*self.b
obj=areaOfRectangle(5,6)
obj.calculateArea()
```

```
30
```

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 areaOfSquare:
    def __init__(self,l):
        self.l=l
    def calculateArea(self):
        ans=self.l**2
        self.output(ans)
    def output(self,ans):
        print('Answer:',ans)
obj=areaOfSquare(5)
obj.calculateArea()
```

```
Answer: 25
```

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 areaOfRectangle:
    def __init__(self,l,b):
        self.l=l
        self.b=b
    def calculateArea(self):
        if(self.l!=self.b):
            ans=self.l*self.b
            self.output(ans)
        else:
            print('This is Square')
    def output(self,ans):
        print('Answer:',ans)
obj=areaOfRectangle(5,5)
obj.calculateArea()
```

```
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:
    __side='abc'
    def get_side(self):
        print(self.__side)
    def set_side(self,st):
        self.__side=st
obj=Square()
obj.set_side('Harsh')
obj.get_side()
```

```
Harsh
```

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 __init__(self,amount):
        self.p=amount
    def getProfit(self):
        return self.p


class Loss:
    def __init__(self,amount):
        self.l=amount
    def getLoss(self):
         return self.l


class BalanceSheet(Profit,Loss):
    def __init__(self,profit,loss):
        Profit.__init__(self,profit)
        Loss.__init__(self,loss)

    def balance(self):
        ans=self.p-self.l
        return ans


b1=BalanceSheet(100,50)
print(b1.balance())



50
```

15) WAP to demonstrate all types of inheritance.

```
#Single

class Parent:
    def show(self):
        print("This is the Parent class")
```

```
class Child(Parent):
    def display(self):
        print("This is the Child class")

obj = Child()
obj.show()
obj.display()

#Multiple
class Father:
    def show_father(self):
        print("Father's property")

class Mother:
    def show_mother(self):
        print("Mother's property")

class Child(Father, Mother):
    def display(self):
        print("This is the Child class")

obj = Child()
obj.show_father()
obj.show_mother()
obj.display()


#multilevel
class Grandparent:
    def show_grandparent(self):
        print("This is the Grandparent class")

class Parent(Grandparent):
    def show_parent(self):
        print("This is the Parent class")

class Child(Parent):
    def show_child(self):
        print("This is the Child class")

obj = Child()
obj.show_grandparent()
obj.show_parent()
obj.show_child()


class Father:
    def show_father(self):
        print("Father's property")
```

```
class Mother:
    def show_mother(self):
        print("Mother's property")

class Child(Father, Mother):
    def display(self):
        print("This is the Child class")

obj = Child()
obj.show_father()
obj.show_mother()
obj.display()

class Grandparent:
    def show_grandparent(self):
        print("This is the Grandparent class")

class Parent(Grandparent):
    def show_parent(self):
        print("This is the Parent class")

class Child(Parent):
    def show_child(self):
        print("This is the Child class")

obj = Child()
obj.show_grandparent()
obj.show_parent()
obj.show_child()

#Hierarchical
class Parent:
    def show(self):
        print("This is the Parent class")

class Child1(Parent):
    def display1(self):
        print("This is Child1 class")

class Child2(Parent):
    def display2(self):
        print("This is Child2 class")

obj1 = Child1()
obj2 = Child2()

obj1.show()
obj1.display1()

obj2.show()
```

```
obj2.display2()


#Hybrid
class Base:
    def show_base(self):
        print("This is the Base class")

class Derived1(Base):
    def show_derived1(self):
        print("This is Derived1 class")

class Derived2(Base):
    def show_derived2(self):
        print("This is Derived2 class")

class Derived3(Derived1, Derived2):
    def show_derived3(self):
        print("This is Derived3 class")

obj = Derived3()
obj.show_base()
obj.show_derived1()
obj.show_derived2()
obj.show_derived3()
```

```
This is the Parent class
This is the Child class
Father's property
Mother's property
This is the Child class
This is the Grandparent class
This is the Parent class
This is the Child class
Father's property
Mother's property
This is the Child class
This is the Grandparent class
This is the Parent class
This is the Child class
This is the Parent class
This is Child1 class
This is the Parent class
This is Child2 class
This is the Base class
This is Derived1 class
This is Derived2 class
This is Derived3 class
```

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

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

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

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

person = Person('Harsh', 30)
person.display_info()

employee = Employee('Deep', 28, 50000)
employee.display_info()
```

```
Name: Harsh, Age: 30
Name: Deep, Age: 28
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.

```
from abc import ABC, abstractmethod
```

```python
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 = [Triangle(),Rectangle(),Circle()]

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

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