```
#-----------------------------------------------------------------------------------
---------------------

# there are 2 ways to find the root of a given no. : Bisection method & Newton
Raphason Method.
#-----------------------------------------------------------------------------------
-----------------
#12th September,2021
#here we just use the theory to find the root by graph plotting calculation.

#program-1

'''
def f(x): return x**2-4.0  # defining a function.
x=0.0
t=0.0001

while f(x)<0.0001: # it will run untill it crosses the 0.0001
    x=x+t
print (x-t)


#OUTPUT
1.9999999999997964

'''
'''
#program-2

def f(x): return x**2-96.0  # defining a function.
x=0.0
t=0.0001

while f(x)<0.0001: # it will run untill it crosses the 0.0001
    x=x+t
print ((x),f(x))

#OUTPUT
9.797999999990504 0.0008039998139111049

'''
#-----------------------------------------------------------------------------------
---------------
#19th September
# Bisection Method
#program-1
'''
import sys
```

```
def f(x): return x**2-4.0



a=float(input("enter first guess value"))
b=float(input("enter second guess value"))
if f(a)*f(b)>0:
    print ("No root is Available within the range")
    sys.exit()
while abs (a-b)>=0.001:
    xm=(a+b)*0.5
    if f(xm)==0:
        print ("the root is",xm)
        sys.exit()
    if f(a)*f(xm)<0:
        b=xm
    else:
        a=xm
print("The root is=",(a+b)*0.5)
#OUTPUT-1

enter first guess value5
enter second guess value6
No root is Available within the range


#OUTPUT-2
enter first guess value1
enter second guess value5
the root is 2.0

'''

#-----------------------------------------------------------------------------------
------------------
#Newton Rhapson Method
#Program-2

'''

import sys
def f(x):return x**2-4.0
def h(x):return 2*x
x=float(input("enter the value of approximate root"))
if f(x)==0:
    print ("root is==",x)
    sys.exit
while f(x)>0.0001:
    x=x-f(x)/h(x)
print ("The root is==",x)
```

```
#OUTPUT-1
enter the value of approximate root5
The root is== 2.0000051812194735
#OUTPUT-2
enter the value of approximate root2
root is== 2.0
The root is== 2.0

'''
#------------------------------------------------------------------------------------




#-------------------------------------------------------------------------------
#Topic :Newton FORWARD Interpolation
#26th September
#program-1
#continued at 1st 0ctober,2021
'''
x=[5.0,10.0,15.0,20.0,25.0,30.0]
y=[45.0,105.0,174.0,259.0,364.0,496.0]
d=[]
t=(18.0-x[0])/5.0 #HERE PUT THE REQUIRED VALUE OF OF X INSTEAD OF 18.0(EXAMPLE)
sum=y[0]
coef=t
k=1.0
for i in range (len(y),1,-1):
    for j in range(i-1):
        dif=y[j+1]-y[j]
        d.append(dif)
    sum=sum+coef*d[0]
    coef=((coef*(t-k))/(k+1))
    k=k+1
    y=d
    d=[]


print (sum)

#OUTPUT-INTERPOLATED VALUE
222.826688

#----------------------------------------------------------------------------
#5th October,2021
#NEWTON'S BACKWARD INTERPOLATION
x=[5.0,10.0,15.0,20.0,25.0,30.0]
y=[45.0,105.0,174.0,259.0,364.0,496.0]
```

```
d=[]
n=len(x)-1
t=(18.0-x[n])/5.0
sum=y[n]
coef=t
k=1.0
for i in range (len(y),1,-1):
    for j in range(i-1):
        dif=y[j+1]-y[j]
        d.append(dif)
    sum=sum+coef*d[j]
    coef=coef*(t+k)/(k+1)
    k=k+1
    y=d
    d=[]
print(sum)


#OUTPUT-INTERPOLATED VALUE
222.826688

'''


#-------------------------------------------------------------------------------




#-------------------------------------------------------------------------------



#INTEGRATION
#7th October,2021
'''
# INTEGRATION BY RECTANGULAR METHOD-
def f(x): return 3.0
b=float(input('enter upper limit'))
a=float(input('enter lower limit'))
sum=f(a)*(b-a)
print (sum)
#OUTPUT-
enter upper limit6
enter lower limit2
12.0
'''
#-----------------------------------------------------
```

```
'''
# INTEGRATION BY TRAPIZOIDAL METHOD-
def f(x): return x
b=float(input('enter upper limit: '))
a=float(input('enter lower limit: '))
h=b-a
sum= 0.5*(f(a)+f(b))*h
print (sum)


#OUTPUT-
enter upper limit: 6
enter lower limit: 2
16.0
'''
#----------------------------------------------------
#8th october,2021
'''
# INTEGRATION BY COMPOSITE TRAPIZOIDAL RULE-
def f(x): return x**2
b=float(input('enter upper limit: '))
a=float(input('enter lower limit: '))
n=int(input('enter no of division: '))
h=float((b-a))/n
sum=(f(b)+f(a))*0.5
for i in range (1,n):
    x=a+i*h
    sum=sum+f(x)

print (sum)

#OUTPUT-
enter upper limit: 6
enter lower limit: 2
enter no of division: 10
173.6
'''

#---------------------------------------
#13th November,2021
'''
#INTEGRATION BY SIMPSON METHOD-
def f(x): return x**2
b=float(input('enter upper limit: '))
a=float(input('enter lower limit: '))
h=float((b-a))/2
y0=f(a)
y1=f(0.5*(a+b))
y2=f(b)
sum= (h/3.0)*(y0+4*y1+y2)
```

```
print (sum)


#OUTPUT-
enter upper limit: 6
enter lower limit: 2
69.33333333333333
'''
#-----------------------------------------
#24th November,2021
#SIMPSON COMPOSITE RULE
'''
def f(x): return x**2
b=float(input('enter upper limit: '))
a=float(input('enter lower limit: '))
n=int(input('enter number of division: '))
h=float((b-a))/n
sum1=f(a)+f(b)
sum2=0.0
for i in range (1,n,2):
    x=a+i*h
    sum2=sum2+f(x)
sum3=0.0
for j in range (2,n,2):
    x=a+j*h
    sum3=sum3+f(x)
I=(h/3.0)*(sum1+(4*sum2)+(2*sum3))
print(I)

#OUTPUT-
enter upper limit: 6
enter lower limit: 2
enter number of division: 10
69.33333333333333
'''
#-----------------------------------------
#8TH December,2021
#INTEGRATION BY USING SCIPY MODULE-1
#Caution: if you want to integrate upto nth value of x put (n+1) besides x= command.
#BETTER DID BY GOOGLE COLAB, SCIPY MODULE PACAKE NOT INSTALLED IN IDLE.
'''
import numpy as np
from scipy import integrate
x=np.arange(0,3)
y=x**2
I=integrate.simps(y,x)
print (I)

#OUTPUT-
2.6666666666666665
```

```
'''
#-------------------------------------------------------
#DECEMBER-15 &16 ,2021
#simpson by scipy
'''
from scipy.integrate import simps
import math
f=lambda x: math.sqrt(4-x**2)
I=simps(f,0,2,100)
print I
'''
#by summing method
'''
x=[1.0,2.0,3.0,4.0,5.0]
y=[2.0,4.0,6.0,8.0,10.0]
s=0.0
for i in range(len(x)-1):
    a=((x[i+1]-x[i])y[i])+0.5*(x[i+1]-x[i])*(y[i+1]-y[i]))
    s=s+a
    print s
'''

'''
def f(x): return 3*x**2+2*x
print f(3.5)
'''
#verify by simpson
'''
from scipy.integrate import quad
import numpy as np
f=lambda x: x**2
I=quad(f,1,5)
print (I)
 #OUTPUT-
 41.33333333333333, 4.588921835117313e-13

'''
#------------------------------------------------------

#22nd december,2021
'''
import matplotlib.pyplot as plt
x=[3,4,5]
y=[7,8,9]
plt.plot(x,y,'.')
plt.show()
'''
#23rd december,2021
#pogram-1
'''
```

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,100)
plt.xlim(-5,15)
plt.ylim(-1.5,1.5)
plt.xlabel ('X-axis')
plt.ylabel ('Y-axis')
plt.title ('TITLE')
plt.plot(x,np.sin(x))
plt.show()
'''
#----------------------------------------------------------------




#22nd December,2021
#Euler's Method
'''
def f(x,y):return 3*x*y
x=0.0
y=1
h=0.003
for i in range(1001):
    y=y+h*f(x,y)
    x=x+h
    print(y)

#OUTPUT LAST LINE-
655684.5304321039

'''
#------------------------------------------------
#Runge-Kutta-2nd order(RK2)
'''
def f(x,y):return 3*x*y
x=0.0
y=1.0
h=0.0002
for i in range(10001):
    k1=h*f(x,y)
    k2=h*f(x+h,y+k1)
    y=y+0.5*(k1+k2)
    x=x+h
    print(y)

#OUTPUT-
403.9129321328625
'''
```

```
#---------------------------------------------

#pogram-2

'''
import matplotlib.pyplot as plt
import numpy as np


#Euler method
def f(x,y):return (x+y+1)
x=0.0
x1=0.0

y=0.0
y1=0.0
h=0.01
d1=np.linspace(0,2.01,201)
exact=2*np.exp(d1)-d1-2.0
d2=[]
d3=[]
for i in range(201):
    y=y+h*f(x,y)
    x=x+h
    d2.append(y)
    print(y)

#by RK-2 Method

def f(x1,y1):return (x1+y1+1)
for i in range(201):
    k1=h*f(x1,y1)
    k2=h*f(x1+h,y1+k1)
    y1=y1+0.5*(k1+k2)
    x1=x1+h
    d3.append(y1)
    print(y1)



print(len(d1))
print(len(d2))


plt.plot(d1,d2,'o',label='Euler')
plt.plot(d1,d3,'^',label='RK2')
plt.plot(d1,exact,label='exact')
plt.legend(fontsize=14)
plt.show()
'''
```

#---