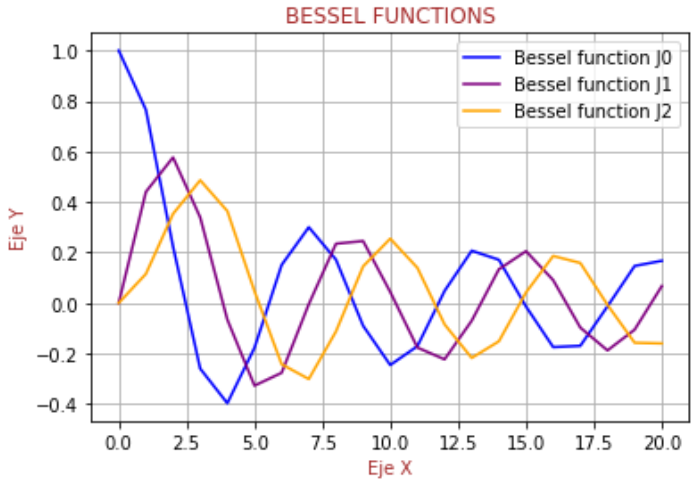
```
import math
from math import pi, cos, sin,sqrt
import numpy
import matplotlib.pyplot as plt
```

```
N=1000
a=0
b=pi
h=((b-a)/N)


def f(m, x, tetha):
    return (cos((m*tetha)-x*(sin(tetha))))


def J(m, x):
  acum=0

  fa=f(m,x,a)
  fb=f(m,x,b)
  for k in range(2,N,2):
      t=a+(h*k)
      acum=(f(m,x,t))+acum

  I=0
  acum2=0
  for k in range(1,N,2):
      t=a+(h*k)
      acum2=(f(m,x,t)+acum2)
  I=(h/(3*pi))*(fa+fb+(acum*2)+(acum2*4))
  return I
```

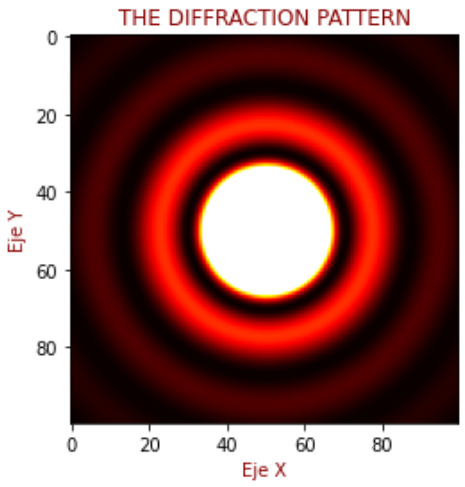
```
J0=[None]*21
J1=[None]*21
J2=[None]*21 #Creamos los arreglos con el espacio que necesitamos
X=[None]*21
for x in range(0,21):
  J0[x]=J(0,x)
  J1[x]=J(1,x)
  J2[x]=J(2,x)
  X[x]=x
print("\x1b[1;31mJ0=",J0)
print("\x1b[1;31mJ1=",J1)
print("\x1b[1;31mJ2",J2)
print("\x1b[1;32mx=",X)

plt.grid() #cuadricula
plt.title("BESSEL FUNCTIONS",color="brown")
plt.ylabel('Eje Y',color='brown')
plt.xlabel('Eje X',color='brown')
plt.plot(X, J0, color='blue',label="Bessel function J0",linewidth= 1.5)
```

```
plt.plot(X, J1, color='purple',label="Bessel function J1",linewidth= 1.5)
plt.plot(X, J2, color='orange',label="Bessel function J2",linewidth= 1.5)
plt.legend()
plt.show()
```

```
J0= [1.0000000000000002, 0.7651976865579666, 0.2238907791412356, -0.2600519549019335, -0.39714980986
J1= [-1.9095836023552694e-17, 0.44005058574493316, 0.5767248077568733, 0.33905895852593637, -0.06604
J2 [-2.4794980883295165e-17, 0.11490348493190047, 0.35283402861563806, 0.4860912605858908, 0.3641281
x= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```




```
def In(lamda,r): #La función definida para la intensidad de la luz
    k=(2*pi/lamda)
    F=((J(1,k*r))/(k*r))**2
    return F

LO=0.5        # Se hace la conversión de nm a um ya que la gráfica debe cubrir valores en unidades de um
n=100  #El tamaño que decidimos para n y así para el número de pasos, con esto podemos manejar el tamaño
l=0.015 #Separación

I=numpy.empty([n,n],numpy.float) #Creamos un arreglo que no contenga valores inicializados.
for i in range(n):
    y=l*(i-n/2)
    for j in range(n):
        x=l*(j-n/2)
        r=math.sqrt((x**2)+(y**2))

        if r <=0.001:
            I[i,j]= 0.5 #Recordando que el límite cuando r tiende a 0, I tiende a 1/2
        else:
            I[i,j] = In(LO,r)


plt.imshow(I,vmax=0.01,cmap='hot')

plt.title("THE DIFFRACTION PATTERN",color="darkred")
plt.ylabel('Eje Y',color='darkred')
plt.xlabel('Eje X',color='darkred')
plt.show()
```


THE DIFFRACTION PATTERN
Eje Y
Eje X
0
20
40
60
80
0
20
40
60
80