```
#Finite Difference Solution of the Schrodinger Equation
#For Simplicity we use system of units such that : m=1 & h_bar=1
#Here we use Potential Energy function of Particle in 1D Box
#import necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#define potential energy function (PARTICLE in 1-D BOX)
def Vpot(x):
      return 1e-90

#enter computational parameters a,b & N
a = float(input("Enter lower limit of the domain (ex: a=-5) : a = "))
b = float(input("Enter upper limit of the domain (ex: b=5) : b = "))
N = int(input("Enter the number of grid points (ex: N=100) : N = "))

#define x grid and h step size
x = np.linspace(a,b,N)
h = x[1]-x[0]

#create kinetic energy matrix
T = np.zeros((N-2)**2).reshape(N-2,N-2)
for i in range(N-2):
      for j in range(N-2):
            if i==j:
                  T[i,j]=-2
            elif np.abs(i-j)==1:
                  T[i,j]=1
            else:
                  T[i,j]=0

#create potential energy matrix
V = np.zeros((N-2)**2).reshape(N-2,N-2)
for i in range(N-2):
    for j in range(N-2):
        if i==j:
            V[i,j]= Vpot(x[i+1])
        else:
            V[i,j]=0

#create hamiltonian matrix
H = -T/(2*h**2) + V

#find eigenvalues and eigenvectors of first 4 lowest states, then sort
them in ascending order
val,vec = np.linalg.eig(H)
z = np.argsort(val)
z = z[0:4]
energies=(val[z])
print("First 4 state Energies : ",energies)

#plot wavefunctions for first 4 lowest states
plt.figure(figsize=(12,10))
for i in range(len(z)):
    y = []
    y = np.append(y,vec[:,z[i]])
```

```
    y = np.append(y,0)  #Final Boundary Condition | y(N)=0
    y = np.insert(y,0,0) #Initial Boundary Condition | y(0)=0
    plt.plot(x,y,lw=3, label="{} ".format(i))
    plt.xlabel('x', size=14)
    plt.ylabel('$\psi$(x)',size=14)
plt.legend()
plt.grid()
plt.title('Wavefunctions for first 4 lowest states',size=14)
plt.show()
```