```
#/usr/bin/python
#!*-* coding:utf-8 *-*
# Este script es sofware libre. Puede redistribuirlo y/o modificarlo bajo
# los terminos de la licencia pública general de GNU, según es publicada
# por la free software fundation bien la versión 3 de la misma licencia
# o de cualquier versión posterior. (según su elección ).
# Si usted hace alguna modificación en esta aplicación, deberá siempre
# mencionar el autor original de la misma.
# Autor:
# Universidad  Distrital Francisco Jose
# Grupo de fisica e informatica
# Dr Julian Andres Salamanca Bernal
# Diego Alberto Parra Garzón
# Colombia, Bogota D.C.
from matplotlib.widgets import  RectangleSelector
from numpy import *
import numpy as np
import matplotlib.pyplot as pl
import os
import subprocess
import math
import time
import shutil
import Gnuplot
from matplotlib.widgets import Cursor
from pylab import *
class Estadistica:
    def Cargar(self):
#   self.f= np.loadtxt('datos/dats1.dat')
    self.x , self.ym = np.loadtxt('datos/dats1.dat', unpack=True,  usecols=[0,1])
#   self.x , self.y = np.loadtxt('dats1.dat', unpack=True,  usecols=[0,1])
    self.y = self.ym*10
    self.n = np.size(self.x)
    self.k = np.ceil(1 + np.log2(self.n)) # comando ceil redondea el numero al mayor entero
    self.c = np.sort(self.y) #comando sort guarda los datos de y
#   self.lon = self.c[110]/8
    print self.n
#   print "El tamaño de la muestra es: ", self.n , "\nEl tamaño del intervalo es ", self.k, self.lon

    def Grafica(self):
    def onselect(eclick, erelease):
        print(' startposition : (%f, %f)' % (eclick.xdata, eclick.ydata))
        print(' endposition   : (%f, %f)' % (erelease.xdata, erelease.ydata))
        print(' used button   : ', eclick.button)
        self.xinicial = round(eclick.xdata,2)
        self.yinicial= round(eclick.ydata,11)
        self.xfinal = round(erelease.xdata,2)
        self.yfinal = round(erelease.ydata,11)
        print self.xinicial, self.yinicial , self.xfinal, self.yfinal

    def toggle_selector(event):
        print(' Key pressed.')
        if event.key in ['Q', 'q'] and toggle_selector.RS.active:
```

```
54                print(' RectangleSelector deactivated.')
55                toggle_selector.RS.set_active(False)
56            if event.key in ['A', 'a'] and not toggle_selector.RS.active:
57                print(' RectangleSelector activated.')
58                toggle_selector.RS.set_active(True)
59
60        pl.xlabel('Distancia [cm]')
61        pl.ylabel('Intensidad [micro W]')
62            pl.title('LONGITUD DE ONDA DIODO INFRARROJO \n')
63        fig = figure
64        ax = subplot(111)
65      #      pl.legend(loc='upper left')
66            ax.plot(self.x, self.y, 'o--')
67        cursor = Cursor(ax, useblit=True, color='red', linewidth=2)
68        toggle_selector.RS = RectangleSelector(ax, onselect, drawtype='line')
69        connect('key_press_event', toggle_selector)
70        pl.subplots_adjust(right=0.97)
71            pl.subplots_adjust(left=0.18)
72            pl.subplots_adjust(bottom=0.13)
73            pl.subplots_adjust(top=0.87)
74            pl.subplots_adjust(wspace=0.32)
75            pl.subplots_adjust(hspace=0.71)
76        pl.show()
77
78        def Estadistica(self):
79        self.distancia = self.xfinal - self.xinicial
80        d = 4.81*10**(-6)
81        x = self.distancia*10**(-2)
82        y = 45*10**(-2)
83        tetharad = math.atan(x/y)
84        tethagra = math.degrees(tetharad)
85        sintetha= math.sin(tetharad)
86        self.lamda = round((2*d*sintetha)*10**(9),0)
87        self.error = round(pow(pow(850-self.lamda,2),0.5), 2)
88        self.error1 = round(pow(pow(100-self.lamda*100/850, 2), 0.5), 3)
89        print tethagra, "\n ", sintetha
90        print "la longitud de onda aproximada para el diodo infrarrojo es",  self.
          lamda  , "nanometros, con un error de ", self.error1 , " %"
91
92        def Grafica1(self):
93        self.x1 = self.x/100
94        self.y1 = self.y/1000000
95        pl.subplot(221)
96        pl.plot(self.x1,self.y1, 'o--')
97        pl.title('Datos Capturados  \n')
98        pl.xlabel('Distancia [m]')
99        pl.ylabel('Intensidad [W]')
100       pl.ylim(0, 0.001)
101           pl.plot([self.xinicial/100, self.xinicial/100], [0, 0.0005], '-')
102           pl.plot([self.xfinal/100, self.xfinal/100], [0, 0.0005], '-')
103 #     pl.ylim(0, 0.02)
104
105       def Grafica2(self):
106       pl.subplot(222)
107           pl.plot(self.x1, self.y1, 'o--')
```

```
108             pl.title('Patrones de Interferencia \n')
109         pl.xlabel('Distancia [m]')
110         pl.ylabel('Intensidad [W]')
111         pl.text(0.001, 0.0008, r' x1 = ' + str(self.xinicial/100))
112         pl.text(0.001, 0.0006, r' x2 = ' + str(self.xfinal/100))
113         pl.text(0.001, 0.0004, r' x2-x1 = ' + str(self.distancia/100))
114         pl.ylim(0, 0.001)
115             pl.plot([self.xinicial/100, self.xinicial/100], [0, 0.0005], '-')
116             pl.plot([self.xfinal/100, self.xfinal/100], [0, 0.0005], '-')
117             pl.plot(self.x1, self.y1, 'o--')
118     #   pl.ylim(0, 0.022)
119
120         def Grafica3(self):
121         pl.subplot(212)
122             pl.plot(self.x1, self.y1, 'o--')
123         pl.ylim(0, 0.001)
124             pl.plot([self.xinicial/100, self.xinicial/100], [0, 0.0005], '-')
125             pl.plot([self.xfinal/100, self.xfinal/100], [0, 0.0005], '-')
126     #         pl.plot([self.xinicial/100, self.xfinal/100], [0, 0.0000000001], '-')
127         pl.text(0.001, 0.0008, r' m:1 Lamda IRE =' +  str(self.lamda) + ' [nm} +/- ' + ↵
          str(self.error) + ' [nm]')
128         pl.text(0.001, 0.0006, r' Distancia entre asintotas '+str(self.distancia/100)  ↵
         + '[m] ')
129         pl.text(0.001, 0.0004, r' Error porcentual  ' + str(self.error1) + ' %')
130     #   pl.ylim(0, 0.022)
131         pl.xlabel('Distancia [m]')
132         pl.ylabel('Intensidad [W]')
133             pl.title('LONGITUD DE ONDA DIODO INFRARROJO \n')
134
135         def Plotear(self):
136         pl.subplots_adjust(right=0.97)
137             pl.subplots_adjust(left=0.18)
138             pl.subplots_adjust(bottom=0.13)
139             pl.subplots_adjust(top=0.87)
140             pl.subplots_adjust(wspace=0.62)
141             pl.subplots_adjust(hspace=0.71)
142             pl.savefig('datos/Graficas.png')
143             pl.show()
144
145         def ordena(self):
146         os.system("python bin/o_Carpetas1.py")
147
148         def __init__(self):
149         self.Cargar()
150         self.Grafica()
151         self.Estadistica()
152         self.Grafica1()
153         self.Grafica2()
154         self.Grafica3()
155         self.Plotear()
156         self.ordena()
157
158     esto = Estadistica()
159
```