```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  4 08:29:17 2019

@author: Publico
"""

# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 08:24:22 2019

@author: Publico
"""
#Importamos las librerías que nos va a dar las funciones que nos importa
usar. Recuerden,
#pueden reinventar la pólvora o pueden usar las cosas (funciones) que ya
fueron realizadas
#Para esto sirven las librerías
import numpy as np #numpy es una librerìa muy utilizada de herramientas
numéricas
import matplotlib.pyplot as plt #matplotlib.pyplot es otra librerìa de
herramientas gráficas

#Vamos incorporando los datos en una variable que (de forma muy
original!) vamos a llamar "datos"
datos = []
ejemplo = [1,1,1,2,3,4,5]
#Como dato de color, las variables de este tipo, nùmeros separados por
comas entre corchetes, se llaman "listas"


seleccion = ejemplo[2:4] #Asì selecciono los elementos desde la posición
2 hasta la posición 4, incluye al elemento en la posición 2 pero no al
último.
                       #esta variable sería igual a [1,2].

#Ahora queremos armar un histograma, bàsicamente un histograma es una
forma de representar gràficamente
#cantidad de veces que medimos un evento
#en funciòn de alguna caracterìstica del evento. Por ejemplo, el eje Y
podrìa ser cantidad de personas
#con una cierta edad y el eje X las edades posibles. Entonces vamos a
tener tantas columnas como edades
#posibles que tiene la poblaciòn que estamos estudiando. El alto de cada
columna indica cuànta gente tiene
#dicha edad.

#Ahora sí, histograma
plt.hist(ejemplo,10) #la función "hist" pertenece a la librería de
funciones de arriba (matplotlib.pyplot)
#Lo que hace es simplemente graficar un histograma, el primer argumento
(los argumentos son lo que le damos a la funciòn
# para que haga lo que tiene que hacer). En este caso lo que obtendremos
será: en el eje x los valores
# posibles de los datos del ejemplo, y en el eje Y la cantidad de datos
cuyo valor es alguno de los del eje x.
```

```
plt.xlabel('Tiempo (s)') #acá le ponemos la magnitud y sus unidades para
el eje x al graficar
plt.ylabel('Magnitud (unidades correspondientes)') #Magnitud y unidades
para el eje y al graficar
#Numpy tiene funciones que le permiten calcular valores estadìsticos de
cualquier conjunto de datos
np.std(ejemplo)  #desviaciòn estándar
np.median(ejemplo) #la mediana
np.mean(ejemplo) #el valor medio

#Supongamos que queremos hacer varios gràficos para variables y1,y2,y3 en
una misma figura, por lo que
#esperamos que las 3 variables dependan de una misma variable
independiente
plt.figure(0) #Creamos una figura vacia para ir metiendole gràficos
plt.plot(t,y1)
plt.plot(t,y2)
plt.plot(t,y3)
#t es la variable independiente

#Si quisiera graficar por separado, armo una figura vacìa para cada
ploteo (no es la ùnica forma de hacerlo)
plt.figure(1)
plt.plot(t,y1)
plt.figure(2)
plt.plot(t,y2)
plt.figure(3)
plt.plot(t,y3)

#----------------------------------Ajuste----------------------------------
----
#El ajuste de los datos segùn un modelo, en este caso ajustaremos por una
gaussiana
from scipy.optimize import curve_fit #otra librería, de la cual sólo
vamos a importar la función "curve_fit"
# que se encarga de devolver los parámetros del modelo (la gaussiana en
este caso), que mejor ajustan
# a los datos, en el sentido de buscar los cuadrados mìnimos

x = []
y = []

#Definen el mean y el sigma (la desviaciòn estandar) según como vimos en
clase
mean = []
sigma = []

def gauss(x,a,x0,sigma):
    return a*exp(-(x-x0)**2/(2*sigma**2)) #preparamos la función por la
que queremos ajustar

popt,pcov = curve_fit(gauss,x,y,p0=[1,mean,sigma])
plt.plot(x,y,'b+:',label='data')
plt.plot(x,gauss(x,*popt),'ro:',label='fit')
plt.savefig('')
plt.show()
```