Nombre de la práctica	Introducción a Matplotlib			No.	3
Asignatura:	Simulación	Carrera:	Ingeniería en sistemas	Duración de la práctica (Hrs)	

Nombre del alumno: Jesús Navarrete Martínez
Grupo: 3501
II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula
III. Material empleado:

- Equipo de computo

IV. Desarrollo de la práctica:

Introducción a Matplotlib

Matplotlib es una biblioteca que permite la creación de figuras y gráficos de calidad mediante el uso e python.

- Permite la creación de gráficos de manera sencilla y eficiente.
- Permite la integración de gráficos y figuras en un jupyter Notebook

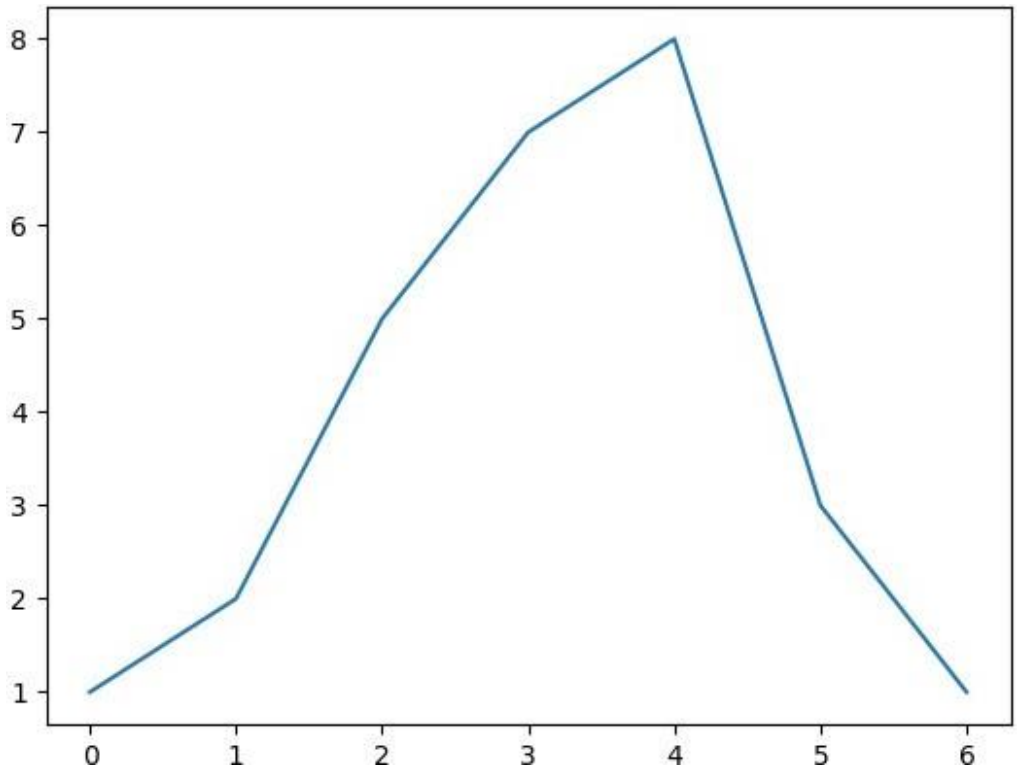
Import

```
import matplotlib
import matplotlib.pyplot as plt

# Muestra los gráficos integrados dentro de jupyter Notebook
%matplotlib inline
```

Representación grafica de Datos.

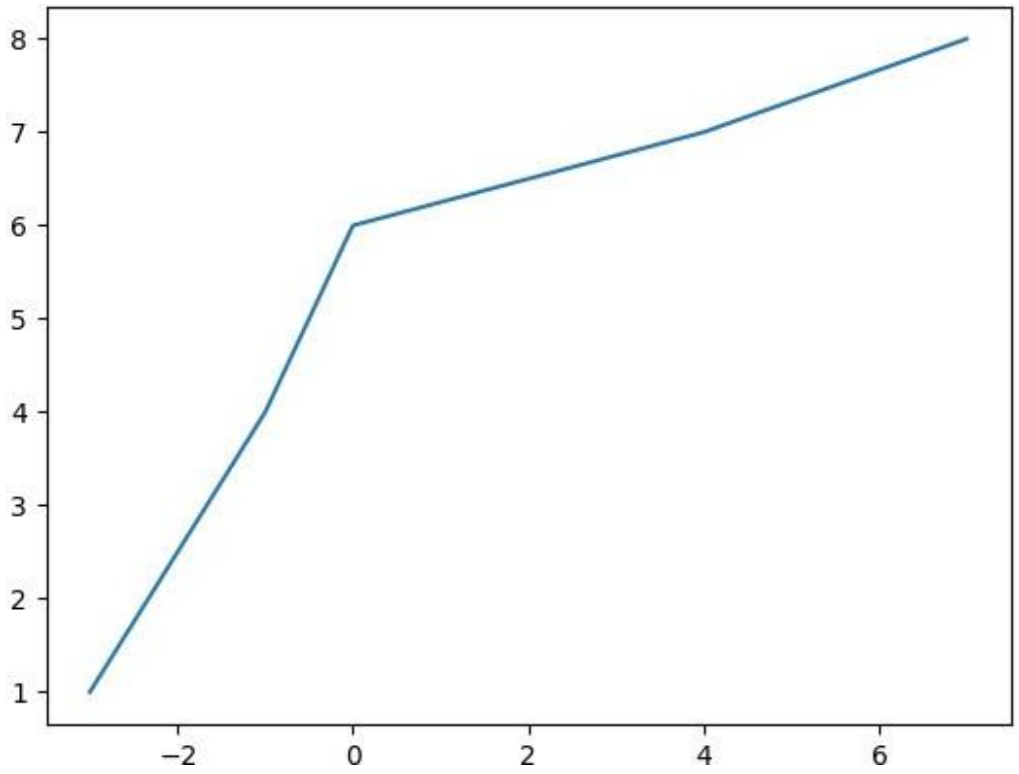
Si a la función de trazado se le da una matriz de datos, la usara como coordenadas en el eje vertical, y utilizara el índice de cada punto de datos en el array como la coordenada horizontal.

```
plt.plot([1,2,5,7,8,3,1])
plt.show()
```



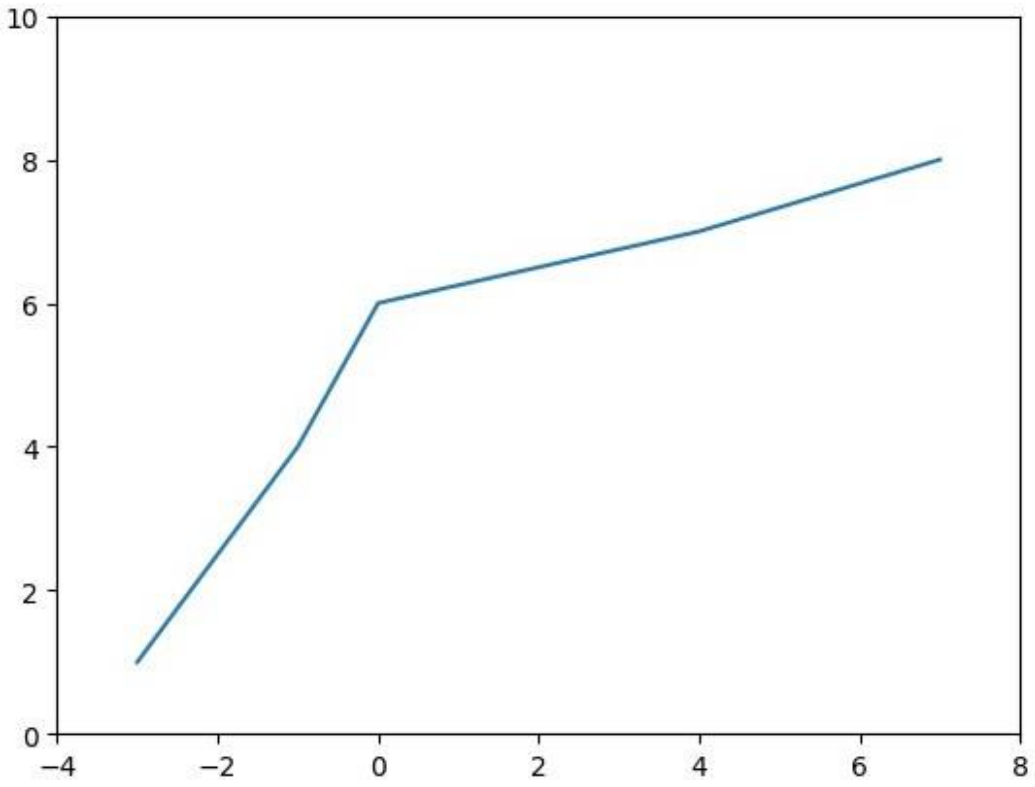
También se puede proporcionar dos matrices: una para el eje horizontal, y otra para el eje vertical.

```
plt.plot([-3,-1,0,4,7], [1,4,6,7,8])
plt.show()
```



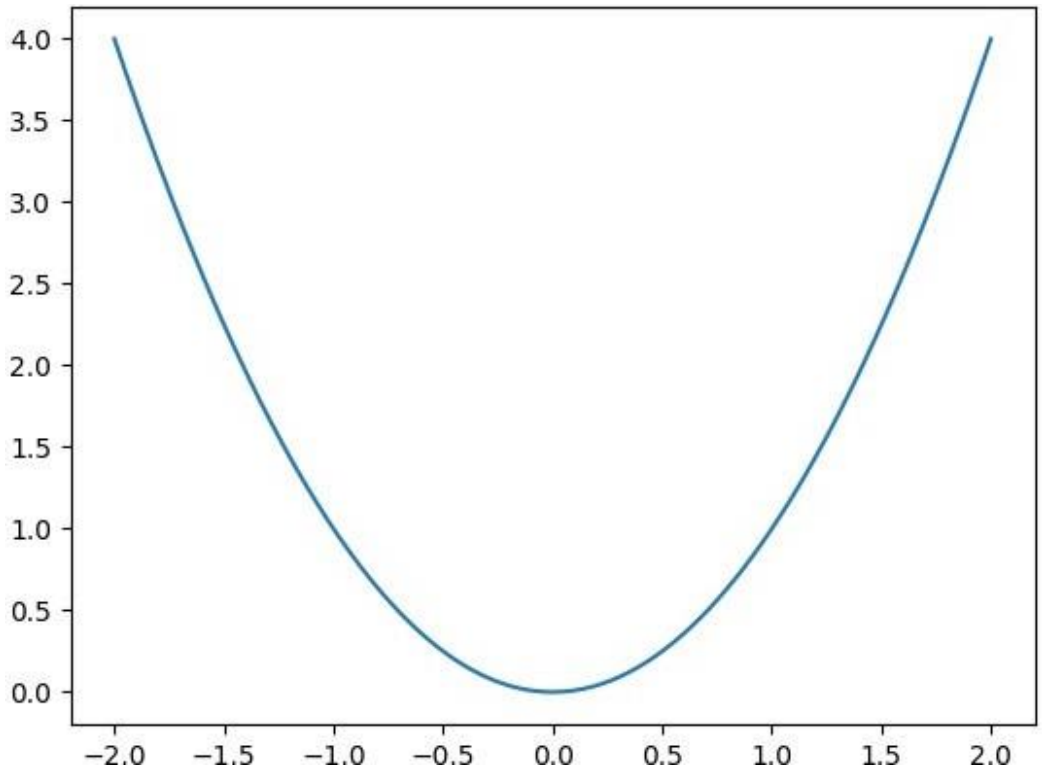
Pueden modificarse las longitudes de los ejes para que la figura no se vea tan ajustada.

```
plt.plot ([-3,-1,0,4,7], [1,4,6,7,8])
plt.axis([-4,8,0,10]) #[xmin,xmax, ymin, ymax]
plt.show()
```



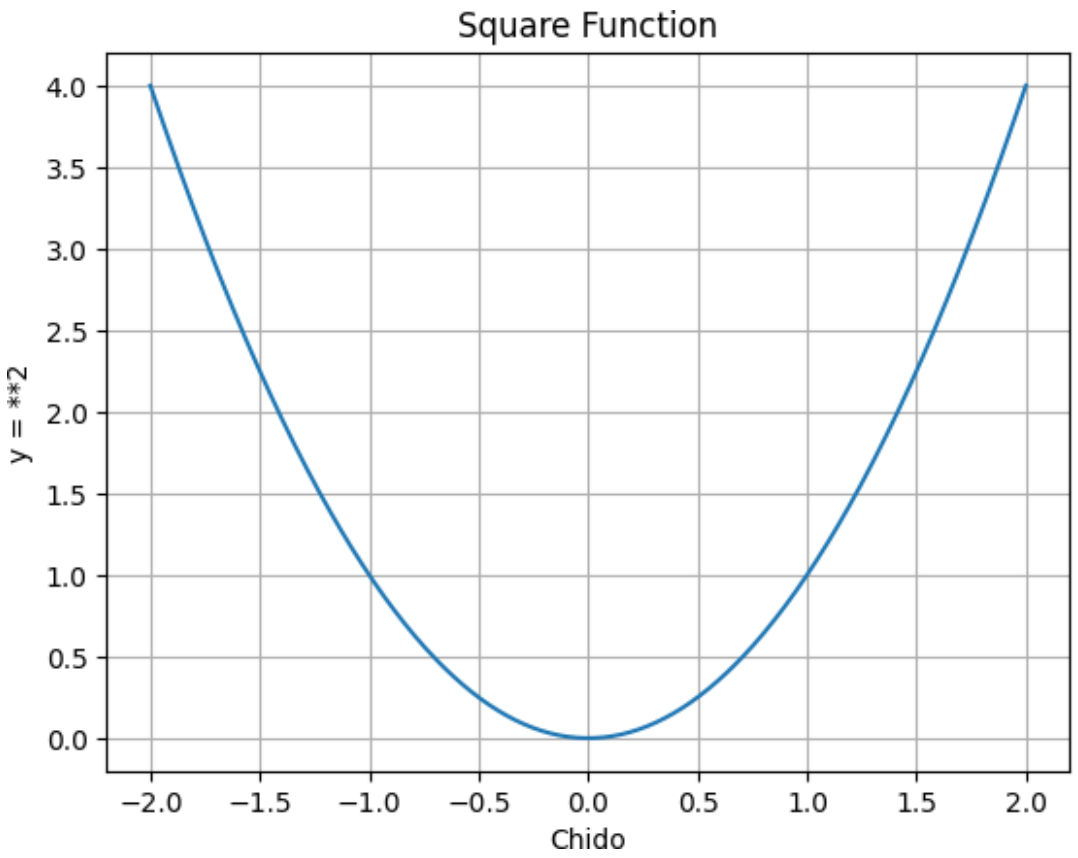
Se sigue el mismo procedimiento para pintar una funcion matematica.

```
import numpy as np
x=np.linspace(-2,2,500)
y=x**2
plt.plot(x,y)
plt.show()
```



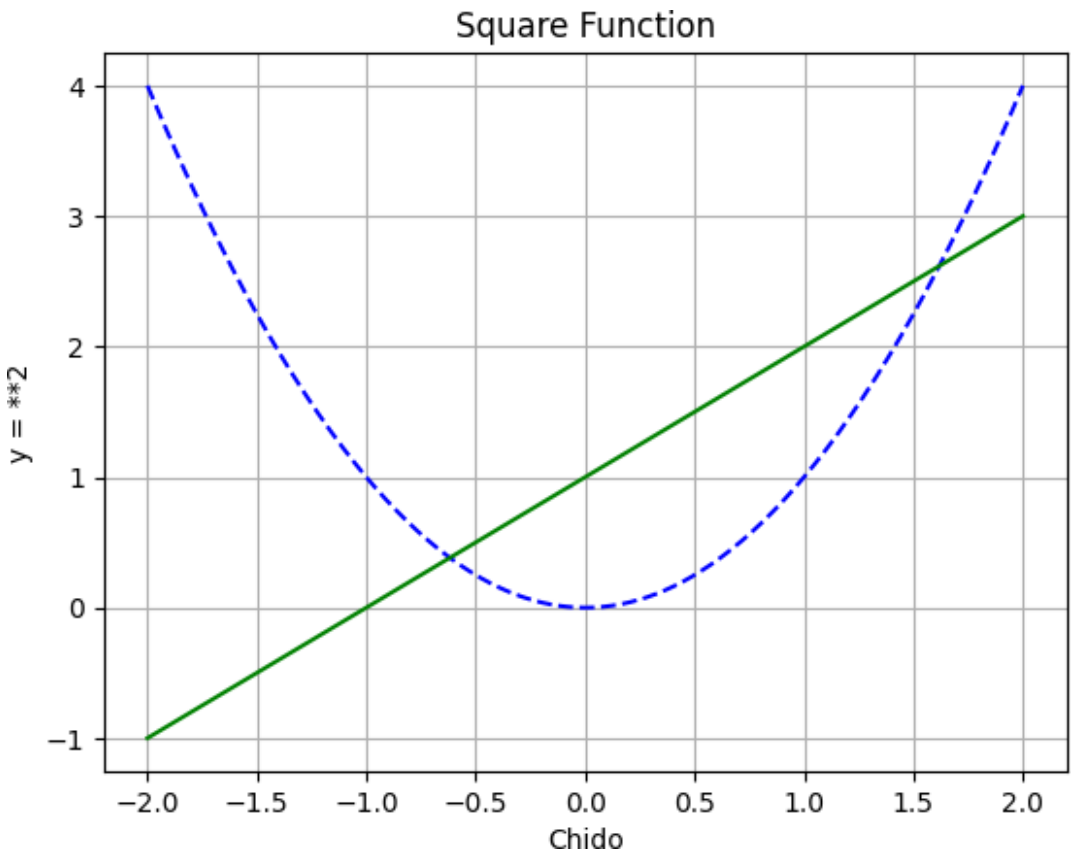
También puede modificarse el estilo de la gráfica para que contenga más información

```
plt.plot(x,y)
plt.title ("Square Function")
plt.xlabel("Chido")
plt.ylabel("y = **2")
plt.grid(True)
plt.show()
```



Pueden superponerse gráficas y cambiar el estilo de las funciones

```
import numpy as np
x= np.linspace(-2,2,500)
y= x**2
y2 = x + 1
plt.title ("Square Function")
plt.xlabel("Chido")
plt.ylabel("y = **2")
plt.grid(True)
plt.plot(x,y,'b--', x, y2, 'g')
plt.show()
```



Separando en diferentes líneas las funciones

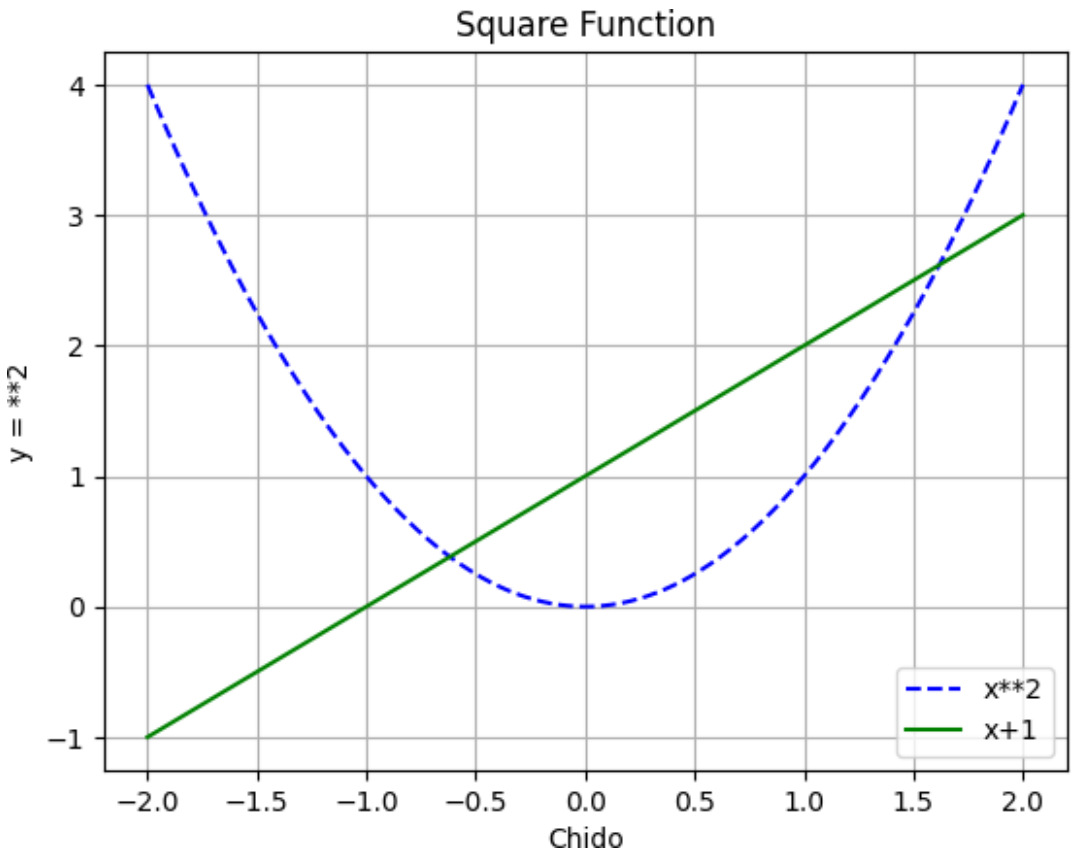
import numpy as np x= np.linspace(-2,2,500) y= x**2 y2 = x + 1 plt.title ("Square Function") plt.xlabel("Chido") plt.ylabel("y = 2") plt.grid(True)

plt.plot(x,y,'b--') plt.plot(x, y2, 'g')

plt.show()

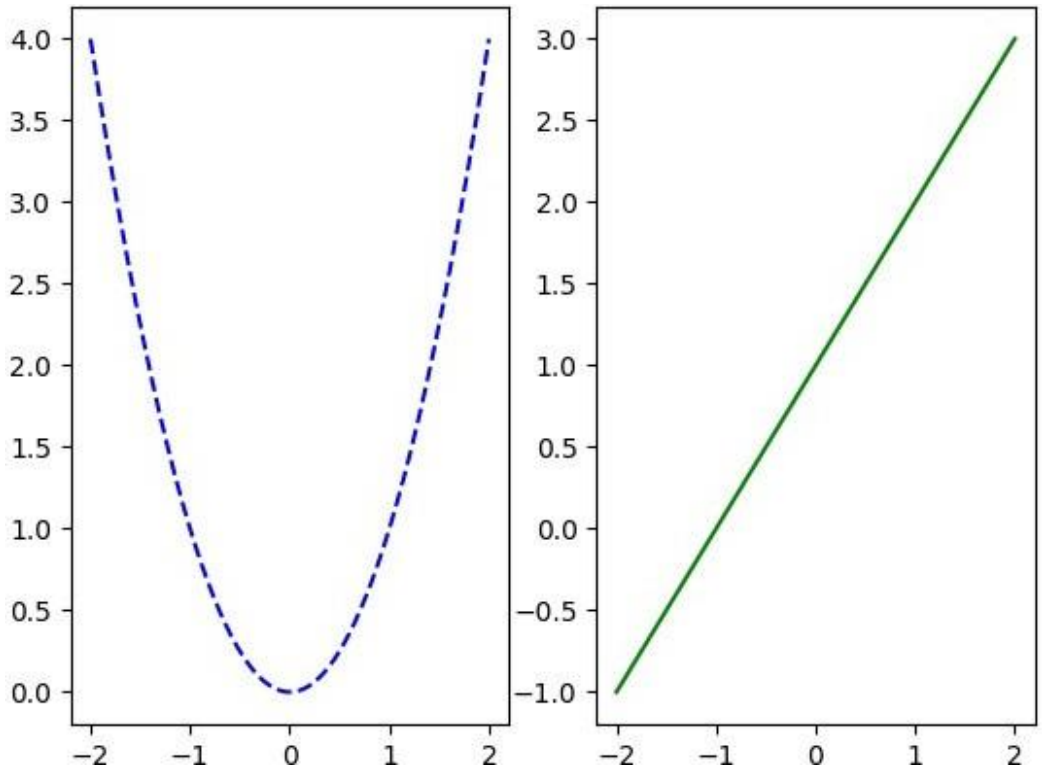
```
# Separando en diferentes lineas las funciones
import numpy as np
x= np.linspace(-2,2,500)
y= x**2
y2 = x + 1
plt.title ("Square Function")
plt.xlabel("Chido")
plt.ylabel("y = **2")
plt.grid(True)

plt.plot(x,y,'b--',label = "x**2")
plt.plot(x, y2, 'g',label= "x+1")
plt.legend(loc="best") # La situa en la mejor localizacion
```

```
plt.show()
```



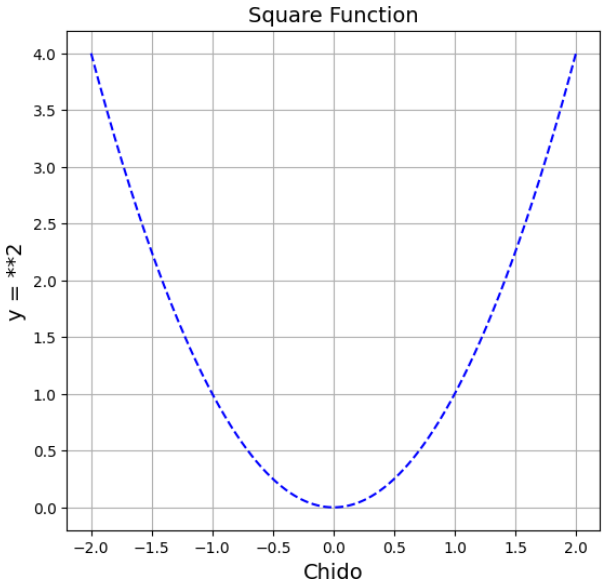
También puede crearse dos graficas que no se superpongan. Estas graficas se organizan en un grid y se denominan subplots.

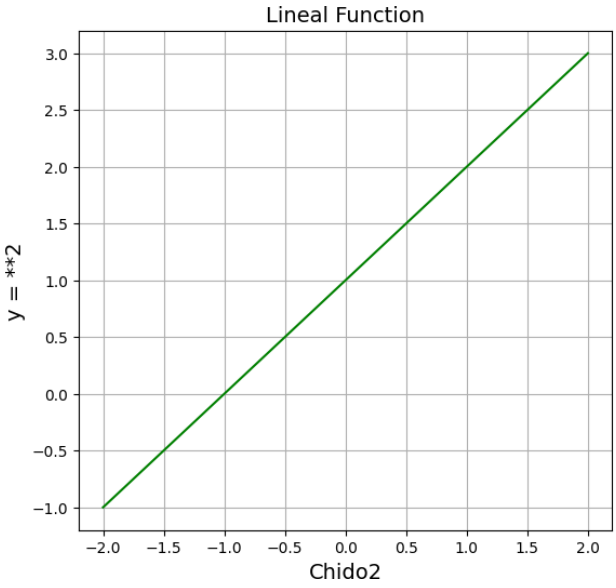
```
import numpy as np
x=np.linspace(-2,2,500)
y=x**2
y2=x+1

plt.subplot(1,2,1) # 1 Rows, 2 Columns, 1st subplot
plt.plot(x,y,'b--')

plt.subplot(1,2,2) #1 ROW, 2 Columns, 2nd subplot
plt.plot(x,y2,'g')

plt.show()
```



Para que las gráficas no queden tan ajustadas, se puede hacer la figura más grande

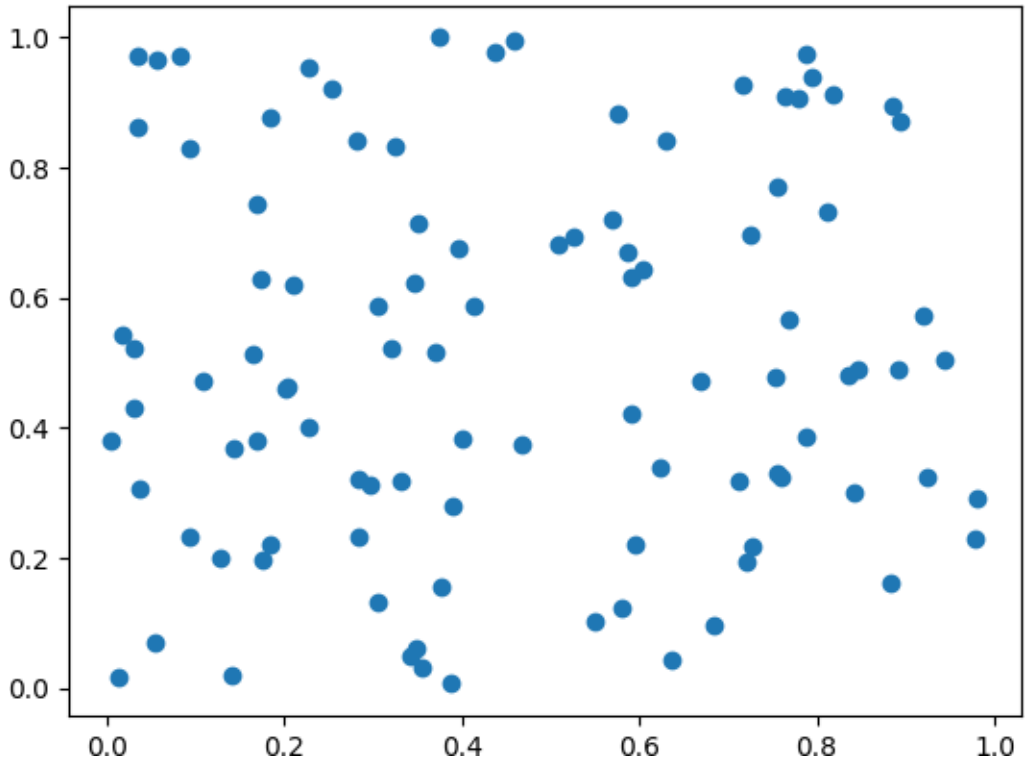
```
plt.figure(figsize = (14,6))

plt.subplot(1,2,1) # 1 Rows, 2 Columns, 1st subplot
plt.plot(x,y,'b--')

plt.title ("Square Function", fontsize=14)
plt.xlabel("Chido", fontsize=14)
plt.ylabel("y = **2", fontsize=14)
plt.grid(True)

plt.subplot(1,2,2) #1 ROW, 2 Columns, 2nd subplot
plt.plot(x,y2,'g')

plt.title ("Lineal Function", fontsize=14)
plt.xlabel("Chido2", fontsize=14)
plt.ylabel("y = **2", fontsize=14)
plt.grid(True)
plt.show()
```



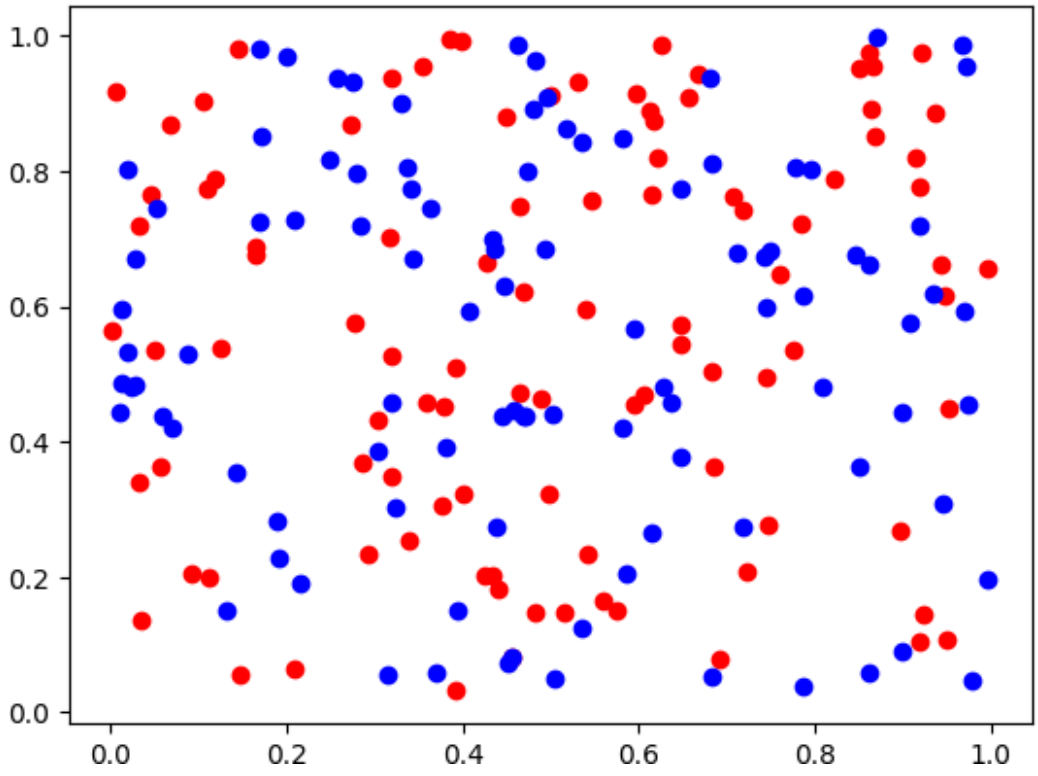


Scatter Plots

```
from numpy.random import rand
x,y = rand(2,100)
plt.scatter(x,y)
plt.show()
```

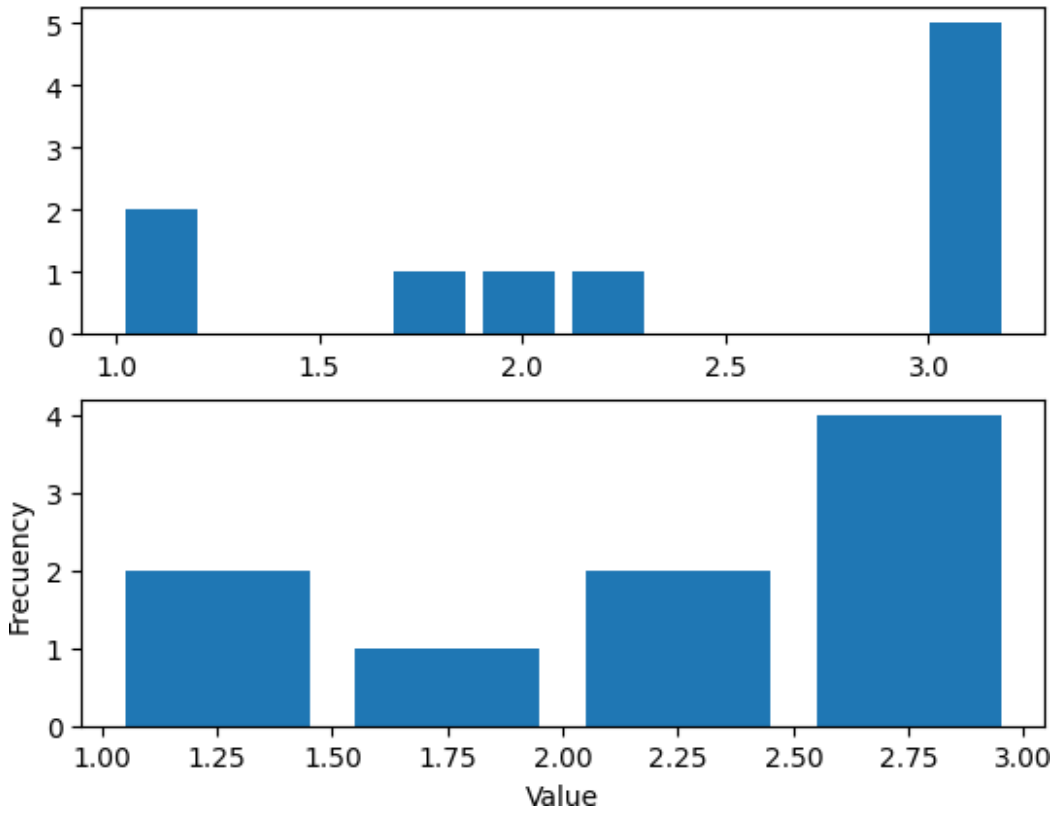


```
from numpy.random import rand
x,y =rand(2,100)
x2,y2 = rand(2,100)
plt.scatter(x,y,c='red')
plt.scatter(x2,y2,c='blue')
plt.show()
```



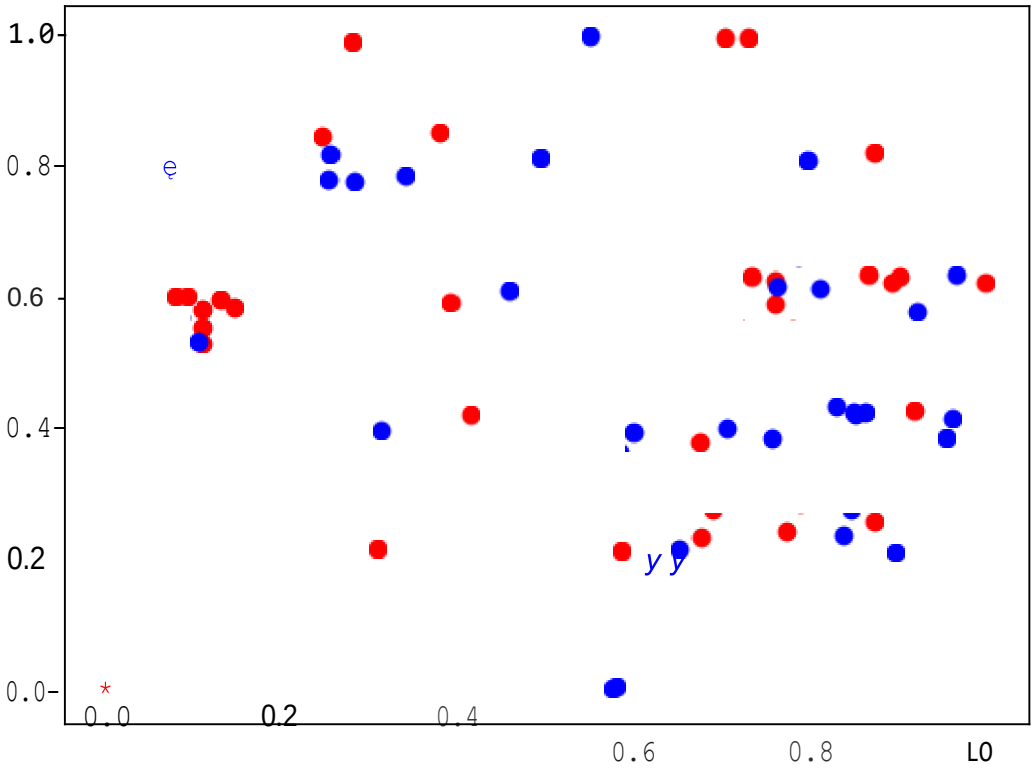
Histogramas

```
data = [1,1.1,1.8,2,2.1,3.2,3,3,3,3]
plt.subplot(211)
plt.hist(data, bins=10,rwidth =0.8)
plt.subplot(212)
plt.hist(data, bins=[1,1.5,2,2.5,3],rwidth =0.8)
plt.xlabel('Value')
plt.ylabel('Frecuency')
plt.show()
```



Guardar las figuras

```
from numpy.random import rand
x,y =rand(2,100)
x2,y2 = rand(2,100)
plt.scatter(x,y,c='red')
plt.scatter(x2,y2,c='blue')

plt.savefig("3501_Mi_Grafica_Chida.png",  transparent=True)
```



V. Conclusiones:

En esta práctica, se exploraron los fundamentos de Matplotlib, una de las bibliotecas más utilizadas para la visualización de datos en Python. A través de diversos ejemplos, se pudo observar la versatilidad de esta herramienta para crear gráficos simples y complejos, como líneas, barras, histogramas y gráficos de dispersión. Además, se demostró la facilidad con la que Matplotlib permite personalizar cada aspecto de los gráficos, desde los ejes y etiquetas hasta los colores y estilos de las líneas. Esta práctica ha sentado las bases para poder abordar la visualización de datos de manera efectiva, una habilidad esencial para el análisis de datos en diversos campos. El dominio de Matplotlib facilitará el entendimiento y la comunicación de información de manera gráfica, permitiendo transformar datos en insights visualmente comprensibles.