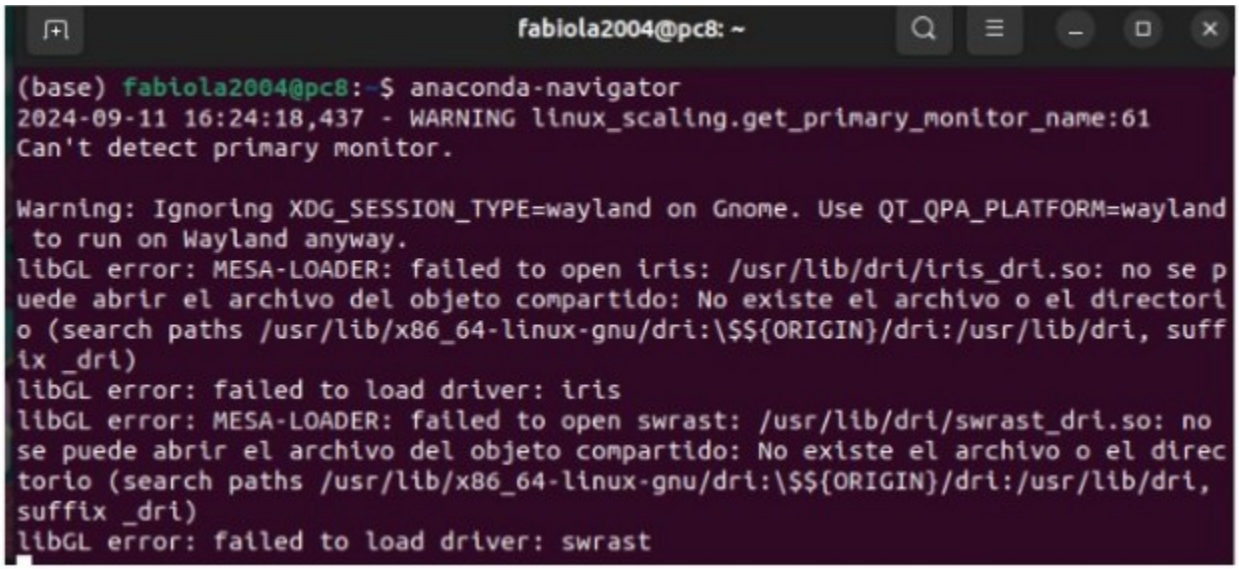
Nombre de la práctica	Matplotlib			No.	
Asignatura:	Simulación	Carrera:	Ingenieria en sistemas computacionales	Duración de la práctica (Hrs)	

NOMBRE DEL ALUMNO: Fabiola Castañeda Mondragón
GRUPO: 3501

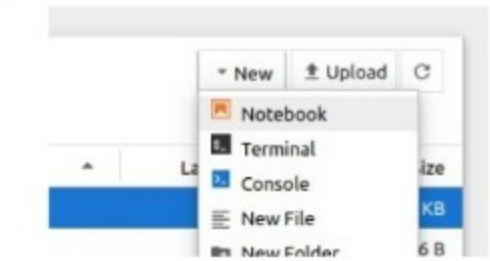
1.- Abrir Anaconda, desde la terminal con el comando anaconda-navigator:



2.- Correr nuestro ambiente virtual:

3.- Abrimos Jupyter:

4.- Creamos un nuevo proyecto:



5.- Escribimos una breve Introducción a Matplotlib:

Introduccion a Matplotlib

Matplotlib es una biblioteca que permite la creacion de figuras y graficos de calidad mediante el uso de Python.

- Permite la creacion de graficos de manera sencilla y eficiente.
- Permite la integracion de graficos y figuras en un Jupyter Notebook.

6.- Importamos matplotlib, que anteriormente ya lo habia instalado:

Import

```
import matplotlib
import matplotlib.pyplot as plt
```

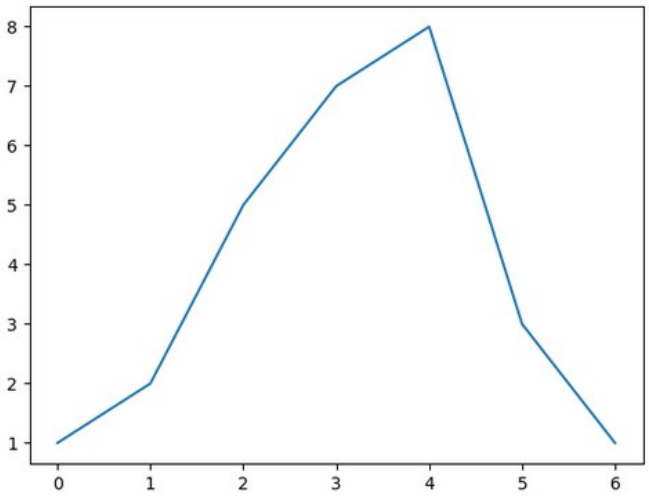
```
# Muestra los graficos integrados dentro de JUpyter Notebook
%matplotlib inline
```

7.- Después de importa matplotlib, Hacemos una representación grafica de los Datos, de esta forma podemos apreciar los datos en una grafica.

Representacion grafica de Datos.

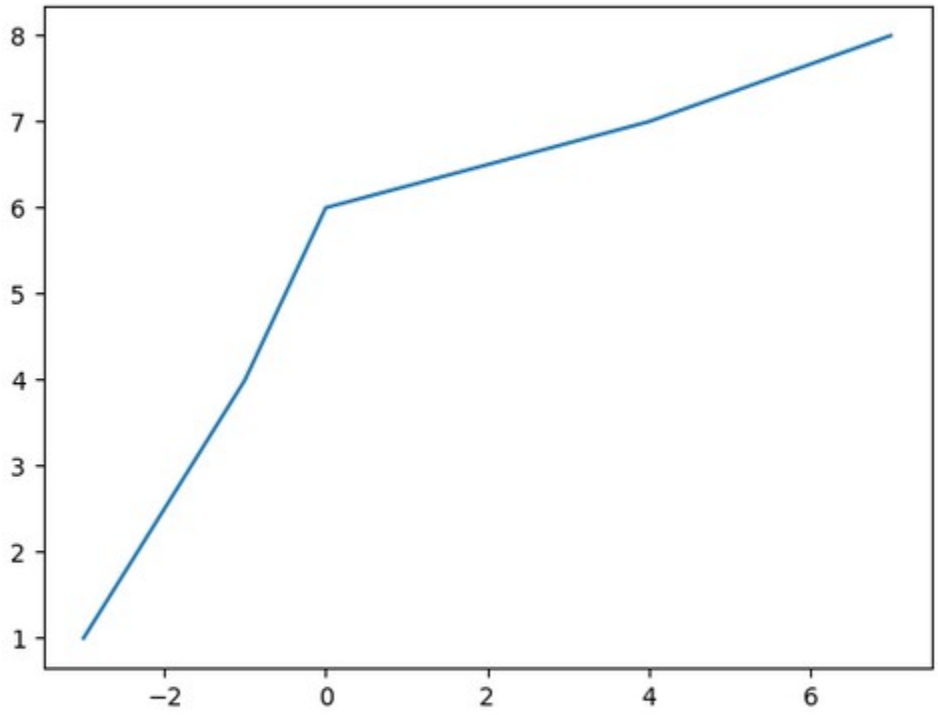
Si a la funcion de trazado se le da una funcion de datos, la usara como coordenadas en el eje vertical, y utilizara el indice de cada punto de datos en el array como la coordenada horizontal.

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



8.- Tambien podemos hacer la modificación de los ejes para que no se vea tan ajustada la grafica.

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



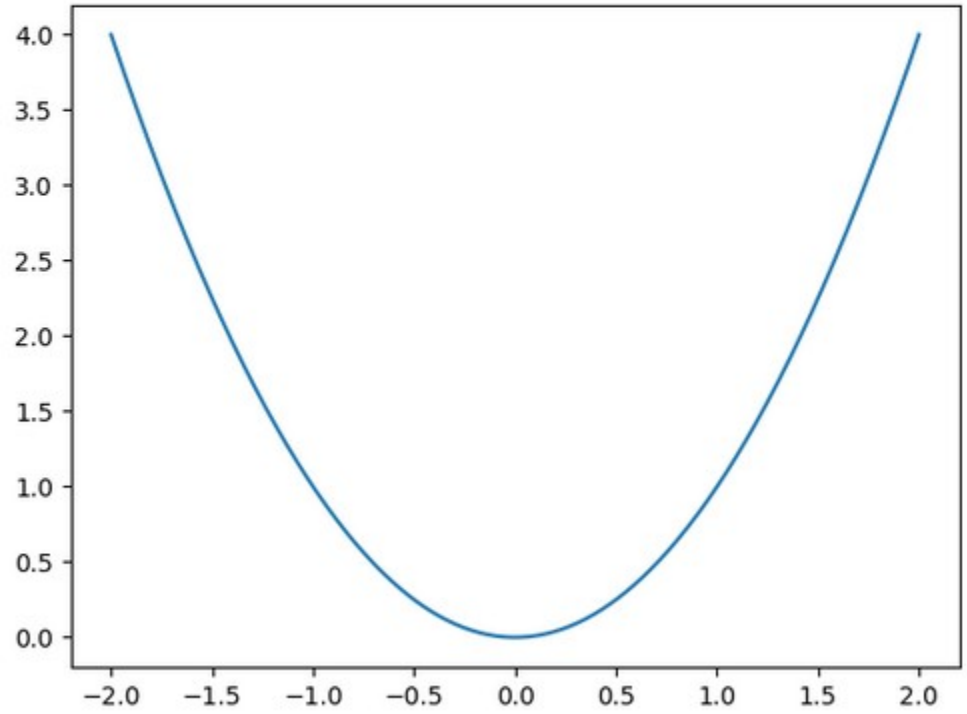
Pueden modificarse las longitudes de los ejes para que la figura no se vea tan ajustada.

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 aplicar una funcion matematica.

9.- Aquí lo que hacemos es modificar el estilo de la grafica para que tenga mas informacion.

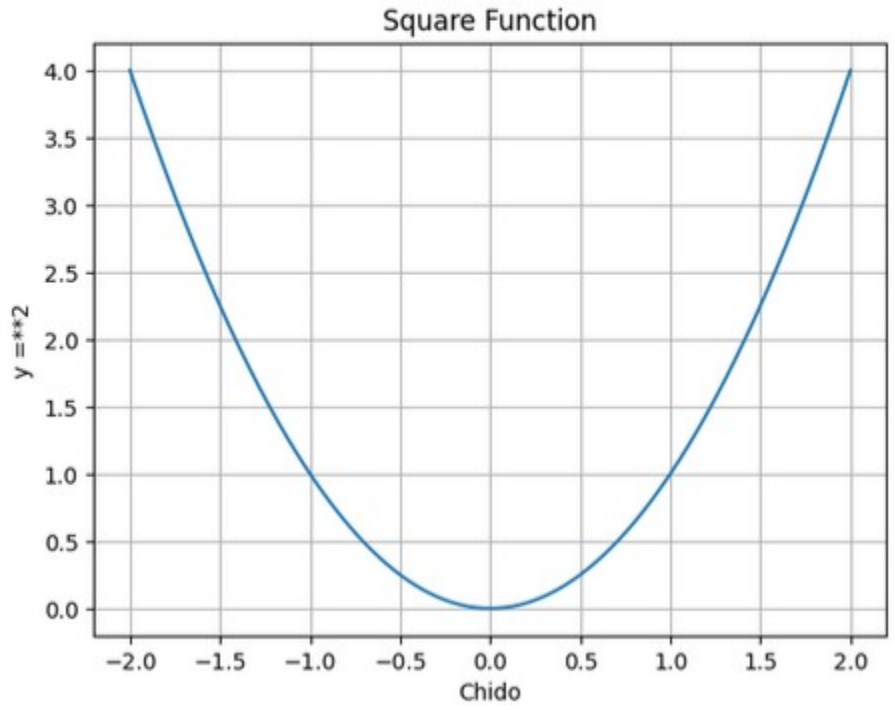
```
import numpy as np
x = np.linspace(-2, 2, 500)
y = x**2

plt.plot(x, y)
plt.show()
```



Tambien puede modificarse el estilo de la grafica para que contenga mas informacion.

10.- Cambiamos el estilo de las funciones, y también podemos superponer graficas.

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

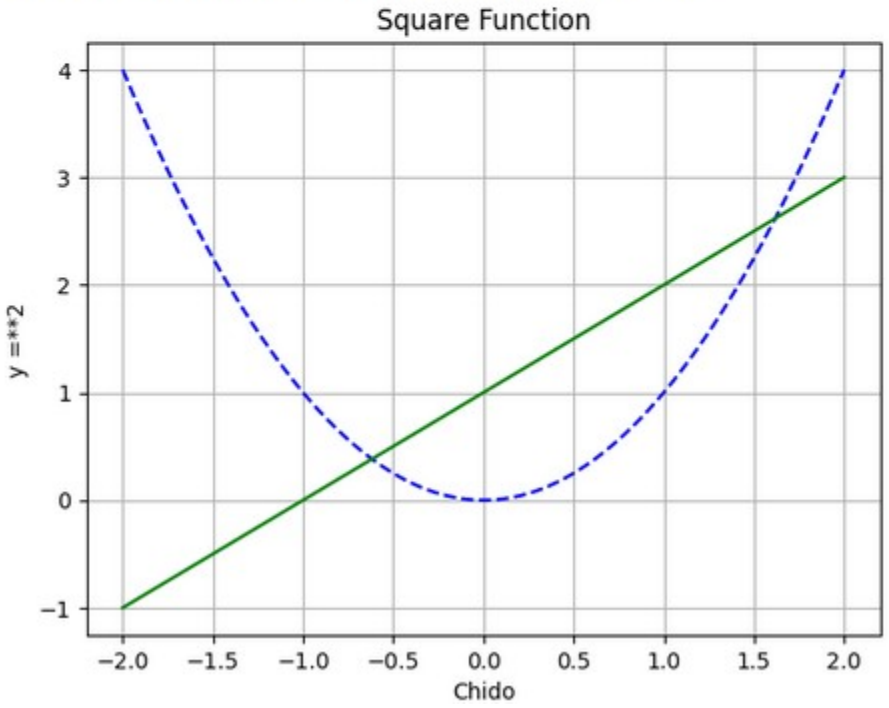


Pueden superponerce gràficas y cambiar el estilo de las funciones.

11.- Implementamos mas graficas.

```
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
```

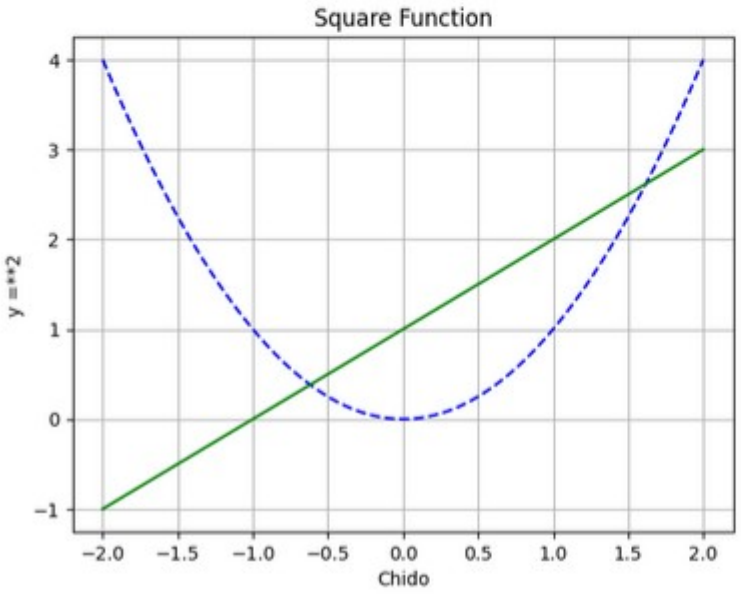
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



12.- Separamos en diferentes lineas las funciones.

```
# 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--')
plt.plot (x, y2, 'g')
plt.show
```

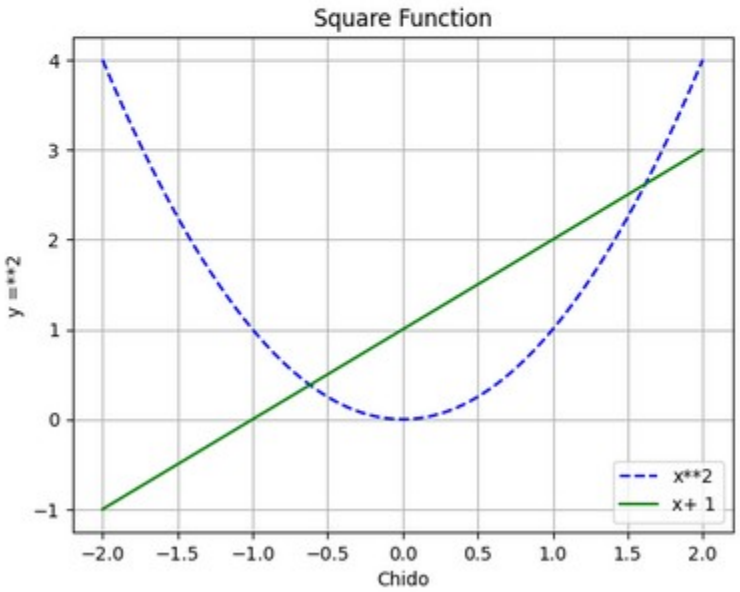
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



13.- Generamos dos graficas que no se superpongan.

```
]: 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
```

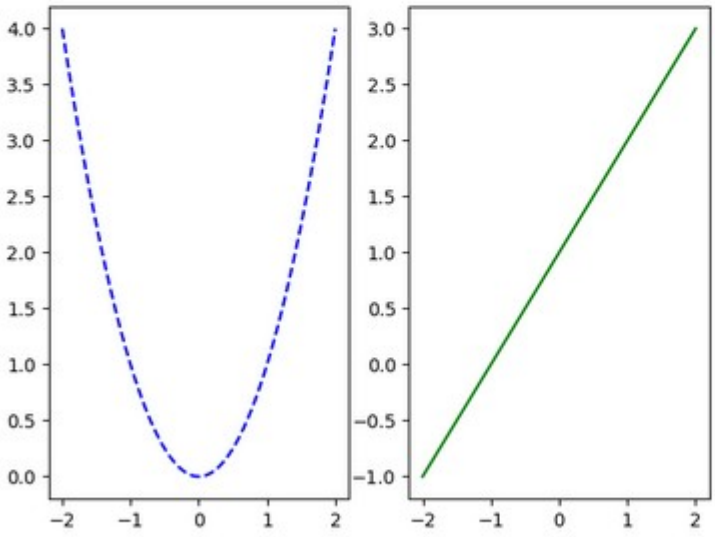
```
]: <function matplotlib.pyplot.show(close=None, block=None)>
```



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

14.- Realizamos las graficas sin que se interpongan, separadas y mas grandes.

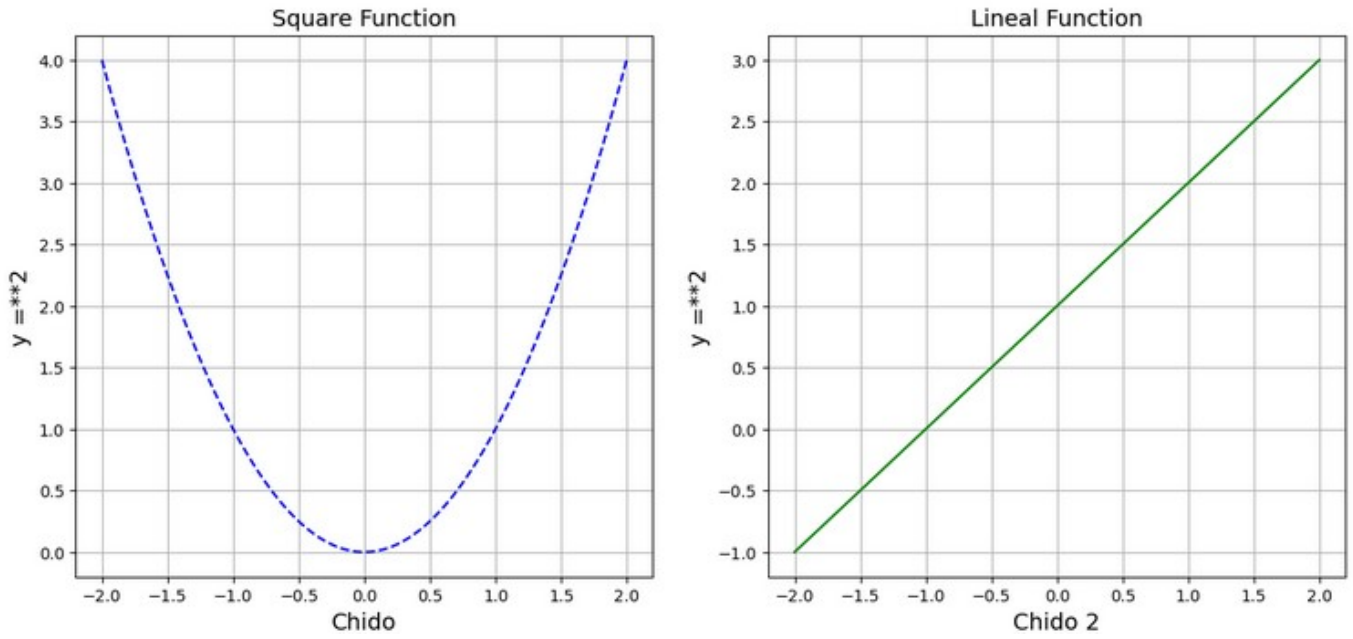
```
4]: import numpy as np
x = np.linspace(-2, 2, 500)
y = x**2
y2 = x + 1

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

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

plt.show()
```



Para que las graficas no queden tan ajustadas, se puede hacer la figura mas grande.

15.- Realizamos las graficas mas separadas, y se les agregan mas información como el titulo y información en los ejes x y y.

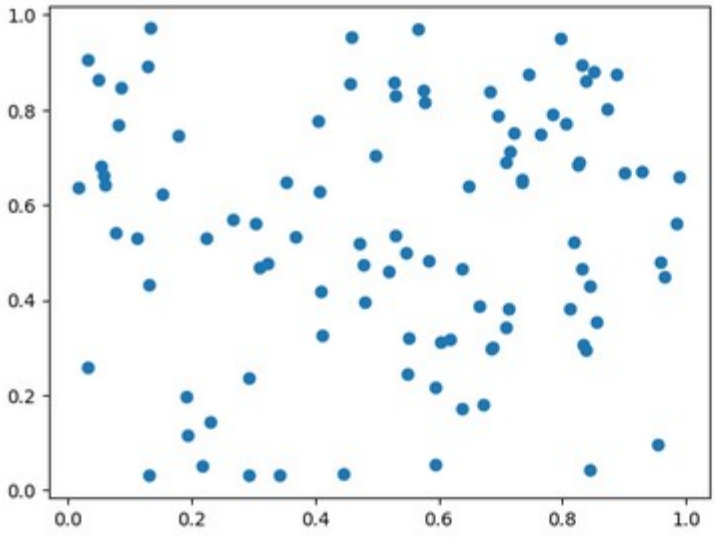
```
plt.figure(figsize = (14,6))
plt.subplot(1, 2, 1) # 1 Rows, 2 Columns, 1st Subplot
plt.plot(x, y, 'b--')
plt.title("Square Function", fontsize = 14)
plt.xlabel("Chido", fontsize = 14)
plt.ylabel("y =**2", fontsize = 14)
plt.grid(True)

plt.subplot(1, 2, 2) #1 Rows, 2 Columns, 2nd Subplot
plt.plot(x, y2, 'g')
plt.title("Lineal Function", fontsize = 14)
plt.xlabel("Chido 2", fontsize = 14)
plt.ylabel("y =**2", fontsize = 14)
plt.grid(True)

plt.show()
```



16.- Lo que realizamos es simular datos con números aleatorios, en este caso son 100 datos.

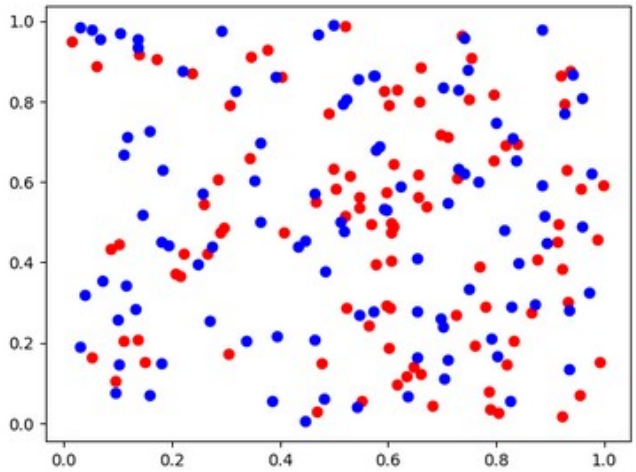
Scatter Plots

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



17.- En este ejemplo igual se simulan datos pero en este caso con dos funciones dd diferente color para diferenciarlos, pero siguen siendo datos aleatorios.

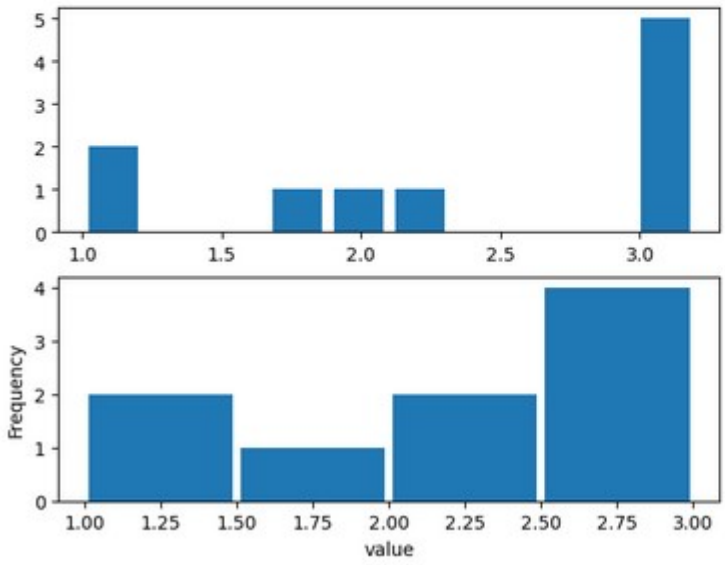
```
28]: from numpy.random import rand
     x, y = rand(2, 100)
     x2, y2 = rand(2, 100)
     plt.scatter(x, y, c ='red')
     plt.scatter(x2, y2, c = 'blue')
     plt.show()
```



18.- Generamos dos gráficos en una misma figura mediante histogramas.

Histogramas

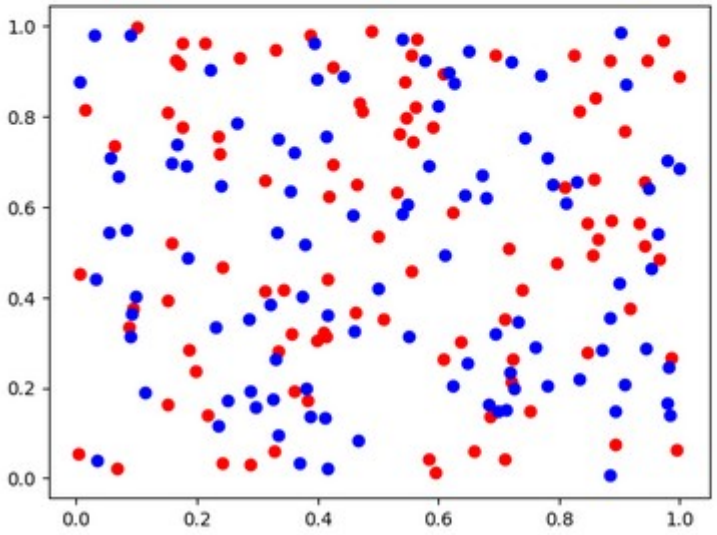
```
data = [1, 1.1, 1.8, 2, 2.1, 3.2, 3, 3, 3, 3]
plt.subplot(211)
plt.hist(data, bins = 10, rwidth = 0.8)

plt.subplot(212)
plt.hist(data, bins = [1, 1.5, 2, 2.5, 3], rwidth = 0.95)
plt.xlabel('value')
plt.ylabel('Frequency')

plt.show()
```



19.- Aquí lo que hacemos es guardar la figura que ya hemos realizado, con ayuda de savefig.

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)
```



20.- Grafica Guardada.

Conclusión.
Matplotlib es una biblioteca de visualización de datos de Python ampliamente utilizada y versátil.
Sus usos comunes son: Análisis de datos científicos y estadísticos, Visualización de datos en investigación, Creación de informes y presentaciones, desarrollo de aplicaciones web y móviles etc.
Matplotlib es utilizada e indispensable para cualquier representación de datos en Python
Su flexibilidad y facilidad de uso la convierten en una opción ideal para visualizar y analizar datos de manera efectiva.