Primeros pasos con Matplotlib

Matplotlib es una paquete Python para crear gráficos de calidad tanto estáticos, animados como interactivos.

La fuente oficial de referencia es matplotlib.org.

Este cuaderno supone que el paquete ya está instalado en el sistema. Si no es el caso puedes seguir esta instrucciones de instalación.

Preparación del entorno.

Lo primero que necesitamos es preparar el entorno de trabajo.

In [1]:
```
%matplotlib notebook
import matplotlib as mpl
import matplotlib.pyplot as plt
print('Matplotlib version: {}'.format(mpl.__version__))
import numpy as np
np.set_printoptions(floatmode='fixed', precision=3)
import pandas as pd
```

```
Matplotlib version: 3.5.2
```

En la línea 1 `%matplotlib notebook` sólo es necesaria si vamos a utilizar el paquete dentro de un entorno interactivo *Jupyter* (como por ejemplo nuestro caso).

Si estamos usando el entorno interactivo IPython usamos el comando `%matplotlib inline`. En ambos casos lo que estamos diciendo es que los gráficos generados se muestran dentro del entorno interactivo en vez de abrir una ventana gráfica independiente.

Si por el contrario estamos usando el paquete Matplotlib en un *script* python el comando `%matplotlib` no es necesario.

La línea 2 importa el paquete Matplotlib con el alias `mpl`. El paquete Matplotlib contiene varios sub paquetes de los cuales el paquete Pyplot es sobre el que vamos a trabajar. Para facilitar el uso importamos en la línea 3 el paquete Pyplot con el alias `plt`.

Ademas en las líneas 5 y 7 importamos los paquetes Numpy y Pandas usados para crear los datos necesarios para generar los gráficos.

"Hola Mundo" con Matplotlib.

Para comprobar que el entorno está correctamente instalado y tener una primera visión de la forma de trabajar, vamos generar un gráfico de ejemplo. No importa si aún no entiendes bien cómo se genera, lo importante es que puedas generar el gráfico.

Ejercicio 01: Crear un objeto Series 'sen' que represente la función seno() en el intervalor $[0, 2\pi]$ con cien muestras. El nombre de la series será 'sen(x)'.

La salida debería ser algo parecido a lo siguiente:

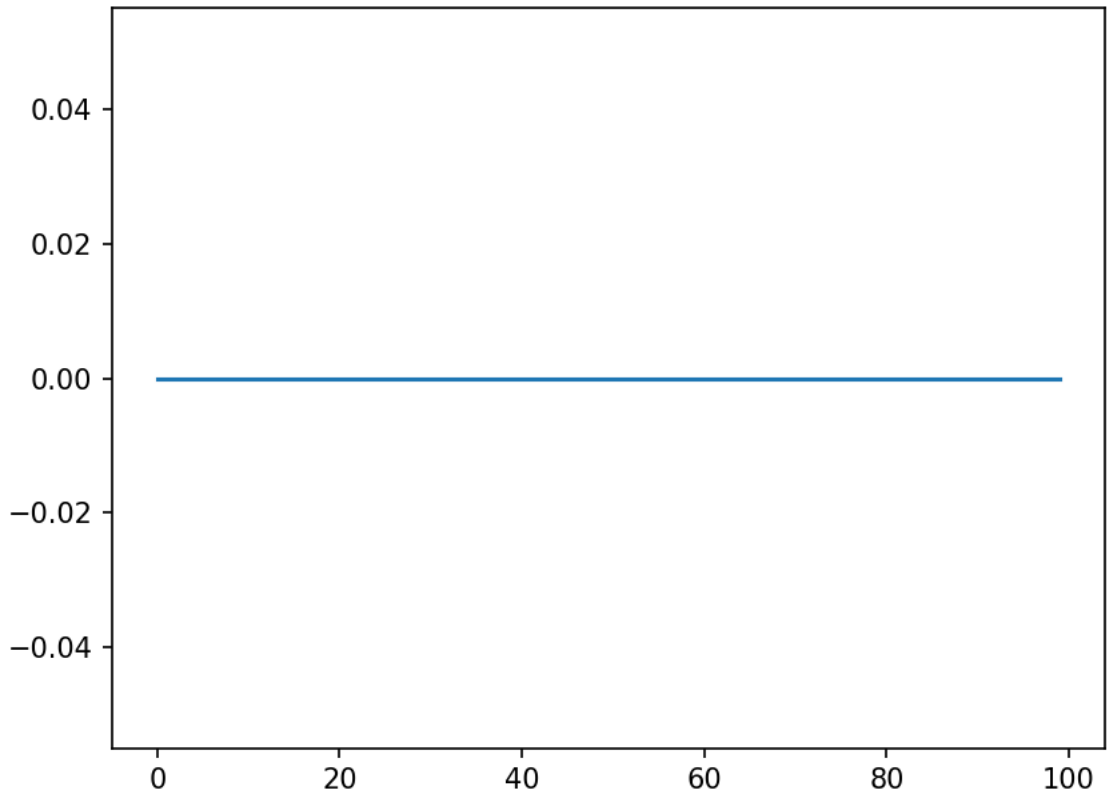
```
Serie:
 0.000000    0.000000e+00
0.063467    6.342392e-02
0.126933    1.265925e-01
0.190400    1.892512e-01
0.253866    2.511480e-01
                ...
6.029319   -2.511480e-01
6.092786   -1.892512e-01
6.156252   -1.265925e-01
6.219719   -6.342392e-02
6.283185   -2.449294e-16
Name: sen(x), Length: 100, dtype: float64
```

In [5]:
```
sen = pd.Series(np.zeros(100))
#Pon tu código aquí.
#Sugerencia: utiliza np.linspace para generar el índice de valores de x.,
#utliza la ufunc np.sin.

#

print('Serie:\n', sen)
```

```
Serie:
 0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
     ...
95    0.0
96    0.0
97    0.0
98    0.0
99    0.0
Length: 100, dtype: float64
```

Ahora vamos a generar nuestro primer gráfico.

In [6]:
```
plt.figure() #crea una nueva figura que pasa a ser la figura "actual".
plt.plot(sen, '-') #dibuja una gráfica de líneas en la figura "actual".

#plt.show() no es necesario en entornos interactivos. Solo para scripts.
```



Out[6]: `[<matplotlib.lines.Line2D at 0x7f1ca2a38dc0>]`

La línea 1 crea un objeto `Figure` que pasa a ser la figura "actual". Este objeto inicialmente representa un lienzo en blanco.

En la línea 2 generamos un gráfico de línea en la figura "actual".

Como ves se ha mostrado automáticamente el gráfico. Esto es debido a que ejecutamos el comando `%matplotlib notebook`, es decir, estamos en un entorno interactivo.

Si no estamos en un entorno interactivo deberíamos haber descomentado la última línea para ejecutar la sentencia `plt.show()`. Esto provoca que se cree una ventana independiente y en ella se renderiza el gráfico.

Ejercicio 02: en el mismo rango de valores de x vamos a dibujar la función $\cos(x)$.

La salida debería ser algo parecido a lo siguiente:

```
Serie:
 0.000000    1.000000
0.063467    0.997987
0.126933    0.991955
0.190400    0.981929
0.253866    0.967949
              ...
6.029319    0.967949
6.092786    0.981929
6.156252    0.991955
6.219719    0.997987
```

```
6.283185    1.000000
Name: cos(x), Length: 100, dtype: float64
```

In [7]:
```
cos = pd.Series(np.zeros(100))
#Pon tu código aquí.
#Sugerencia: utiliza la ufunc np.cos para generar los valore del coseno.,

#

print('Serie:\n', cos)
```

```
Serie:
 0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
     ...
95    0.0
96    0.0
97    0.0
98    0.0
99    0.0
Length: 100, dtype: float64
```

Ahora vamos a dibujar el gráfico para la función coseno.

In [8]:
```
plt.plot(cos, '--') # dibuja una gráfica de líneas en la figura "actual"
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x7f1ca2f6fee0>]
```

Como ves, la gráfica mostrada anteriormente se ha actualizado ya que hemos modificado la figura "actual" con la nueva gráfica. Podemos forzar la actualización usando el comando `plt.draw()`.

Si lo que queremos es generar un figura nueva para una gráfica del seno(x)^2 , deberemos crear primero otra figura "actual" y después utilizar el comando para dibujar la nueva gráfica:

In [ ]:
```
plt.figure() #Crea una nueva figura y esta figura pasa a ser la figura "actual"
plt.plot(sen*sen, '-.') #dibuja una gráfica de línea en la figura "actual"
```

Guardando la figura en un fichero.

Si necesitamos almacenar la figura generada en un fichero tenemos dos alternativas:

- En la figura visualizada (o la ventana gráfica abierta) se muestra una botonera para interactuar con el gráfico generado y entre ellos hay un botón para almacenar.
- Si la figura es generada en un *script* podemos utilizar la función `plt.savefig()` para almacenarla.

Interfaz tipo Matlab o interfaz orientada a objetos.

Matplotlib permite trabajar con una interfaz muy parecida a la ofrecida por el entorno *MatLab*. Esta ha sido la forma utilizada en los ejemplos anteriores. Su principal característica es que se mantiene un "estado" (por ejemplo la figura "actual") y todas las operaciones de dibujo se realizan sobre este estado.

Una segunda alternativa que Matplotlib ofrece es una interfaz orientada a objetos (OO). Con la interfaz OO se trabaja especificando explícitamente sobre qué objeto (por ejemplo Figura) se ejecuta las operaciones.

La interfaz OO es más potente y la recomendada a aprender si se parte desde cero, pero la interfaz *Matlab* es más simple de usar y lógicamente es más amigable para los usuarios que conocen de dicho entorno.

Afortunadamente Matplotlib permite combinar ambas alternativas al mismo tiempo y esto hace más fácil unas operaciones (usando la interfaz tipo *Matlab*) y a la vez permite realizar operaciones avanzadas usando la interfaz orientada a objetos.

Veamos un ejemplo que combina ambas interfaces para dibujar las funciones seno y coseno ya vistas.

In [ ]:
```
fig = plt.figure() #Crea una nueva figura y pasa a ser la figura "actual".
ax = plt.axes() #Añade unos ejes a la figura "actual" y los devuelve.
ax.plot(sen, '-') #dibuja una gráfica de línea en los ejes ax.
ax.plot(cos, '--') #dibuja una gráfica de línea en los ejes ax.
```

Las líneas 1 y 2 usan la interfaz tipo *Matlab* para crear la figura "actual" y obtener el objeto Axes asociado a la misma (la región donde dibujar). En las líneas 3 y 4 dibujamos las gráficas de líneas sobre el objeto Axes que obtuvimos en la línea 2 (interfaz OO).

In [ ]:
```

```