

HOJA DE AYUDA

pandas

[Link a la documentación de Pandas](#)

El módulo pandas brinda herramientas para el análisis y manipulación de datos.

```
import pandas as pd
```

Comando	Concepto
<code>df = pd.read_csv ("archivo.csv")</code>	Lee un archivo CSV convirtiendo su contenido en un DataFrame que es guardado en la variable df .
<code>df.describe()</code>	Genera estadísticas descriptivas de df , como media, desviación estándar, entre otras, aplicadas a las columnas numéricas.
<code>df.head()</code>	Devuelve las primeras 5 filas de df .
<code>df.tail()</code>	Devuelve las últimas 5 filas de df .
<code>df.shape</code>	Devuelve una tupla con el tamaño de df , en el formato (filas, columnas)
<code>df.columns.tolist()</code>	Devuelve la lista de nombres de columnas de df .
<code>df["columna"].unique().tolist()</code>	Devuelve una lista con el conjunto de valores que toman los datos en la columna elegida del DataFrame, sin repetir

FILTRADO Y FUNCIONES

Comando	Concepto
<code>df_columna = df["columna"]</code>	Guarda en df_columna a la serie de valores correspondiente a la columna elegida del DataFrame df .
<code>df["columna"].mean()</code>	Devuelve el valor promedio de la columna .
<code>df["columna"].min()</code>	Valor mínimo de la columna .
<code>df["columna"].max()</code>	Valor máximo de la columna .
<code>df["columna"].count()</code>	Conteo del total de valores a lo largo de la columna . Por defecto ignorará los valores faltantes (NaN).
<code>df["columna"].value_counts()</code>	Conteo del total de valores en cada categoría presente en la columna . Por defecto ignorará los valores faltantes (NaN). La columna debe corresponder a una variable categórica.
<code>df_filtrado = df[df["columna"] == valor]</code>	Guarda en df_filtrado al DataFrame formado por las filas de df que cumplen tener un valor determinado para la columna elegida.
<code>filtro_a = df["columna1"] >= valor_a</code> <code>filtro_b = df["columna2"] != valor_b</code> <code>df_filtrado = df[(filtro_a) & (filtro_b)]</code>	Une varios filtros, incluso si provienen de columnas diferentes. conectores: & (and) (or)
<code>df.groupby(by="columna").funcion()</code>	Agrupar los datos de df en cada una de las distintas categorías en la columna elegida, y aplica la funcion de resumen deseada (por ejemplo, función podría ser mean, count, min o max).
<code>df.groupby(["col1", "col2"]).funcion()</code>	Agrupar los datos de df por orden jerárquico (primero " col1 ", luego " col2 ") finalmente aplica la función de resumen elegida (por ejemplo, función podría ser mean, count, min o max).

HOJA DE AYUDA

Matplotlib

[Link a la DOCUMENTACIÓN](#)

El módulo matplotlib.pyplot se usa para crear gráficos, diagramas, histogramas y más.

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

Comando	Concepto
<code>plt.plot(x, y, forma)</code>	Genera un gráfico de líneas utilizando los valores de x como coordenadas en el eje horizontal y los valores de y en el eje vertical.
<code>plt.title("Título del gráfico")</code>	Agrega un título al gráfico.
<code>plt.xlabel / plt.ylabel ("Título del eje x / y")</code>	Agrega etiquetas al eje x o al y, según corresponda.
<code>plt.show()</code>	Muestra el gráfico en pantalla.

Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

1

Initialize

```
import numpy as np
import matplotlib.pyplot as plt
```

2

Prepare

```
X = np.linspace(0, 10*np.pi, 1000)
Y = np.sin(X)
```

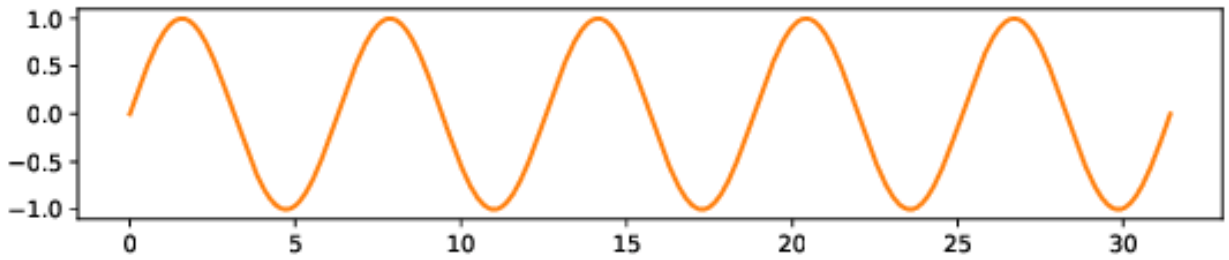
3

Render

```
fig, ax = plt.subplots()
ax.plot(X, Y)
plt.show()
```

4

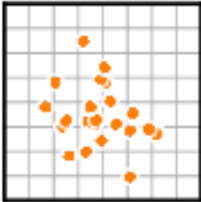
Observe



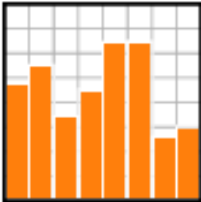
Choose

Matplotlib offers several kind of plots (see Gallery):


```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```




```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```



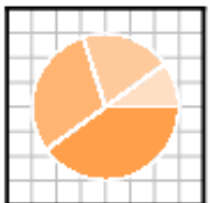
```
Z = np.random.uniform(0, 1, (8, 8))
ax.imshow(Z)
```



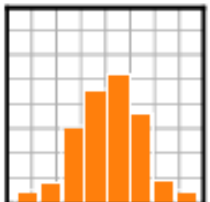
```
Z = np.random.uniform(0, 1, (8, 8))
ax.contourf(Z)
```



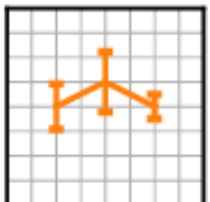
```
Z = np.random.uniform(0, 1, 4)
ax.pie(Z)
```



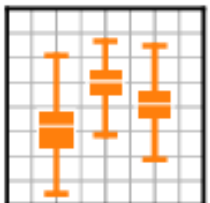
```
Z = np.random.normal(0, 1, 100)
ax.hist(Z)
```



```
X = np.arange(5)
Y = np.random.uniform(0, 1, 5)
ax.errorbar(X, Y, Y/4)
```



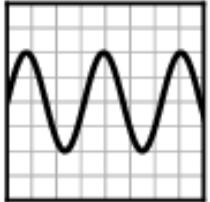
```
Z = np.random.normal(0, 1, (100, 3))
ax.boxplot(Z)
```



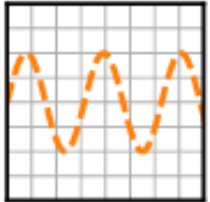
Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

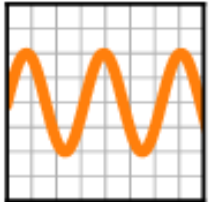
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```



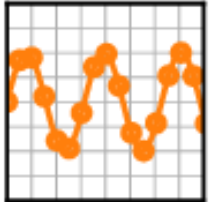
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```



```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```



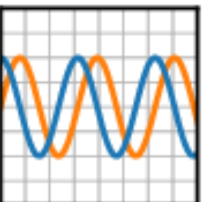
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```



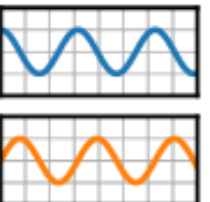
Organize

You can plot several data on the same figure, but you can also split a figure in several subplots (named Axes):

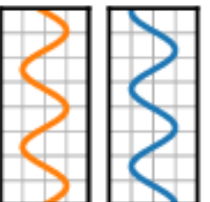
```
X = np.linspace(0, 10, 100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, X, Y2)
```



```
fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(X, Y1, color="C1")
ax2.plot(X, Y2, color="C0")
```

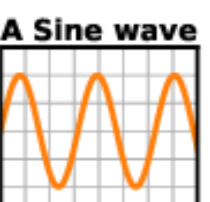


```
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```

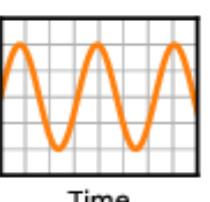


Label (everything)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```



```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```



Explore

Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```