

«Talento Tech»

# Data Analytics con Python

Clase 07





# Clase N° 7 | Conceptos básicos

## Temario

- Aprenderemos a visualizar datos en Python.
- Incorporaremos dos nuevas librerías: matplotlib y seaborn.
- Crearemos distintos gráficos a partir de dataframes.
  - Gráficos de barras
  - Histogramas
  - Gráficos de torta
  - Gráficos de línea y dispersión.



## Visualización de datos

En clases anteriores hemos visto cómo analizar datos utilizando la librería Pandas. La misma nos permitía disponer de una nueva estructura de datos: los dataframes. En general los datos que vayamos a analizar pueden estar en un formato de tablas, como .csv o excel. Los dataframe nos permitían crear una estructura de tablas, con filas y columnas. A su vez, desde un mismo dataframe podíamos filtrar datos y generarnos nuevos datos..

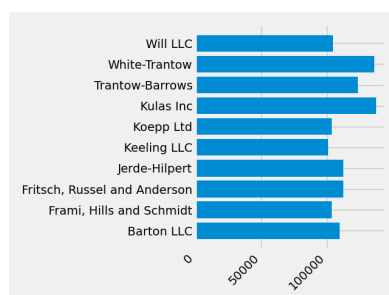
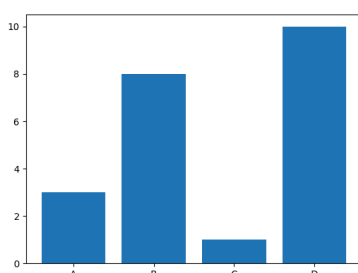
Luego de que hayamos realizado un análisis exploratorio de nuestros datos, vamos a querer graficar algunos resultados. En esta clase veremos cómo hacer una visualización de datos utilizando dos librerías nuevas: [matplotlib](#) y [seaborn](#).

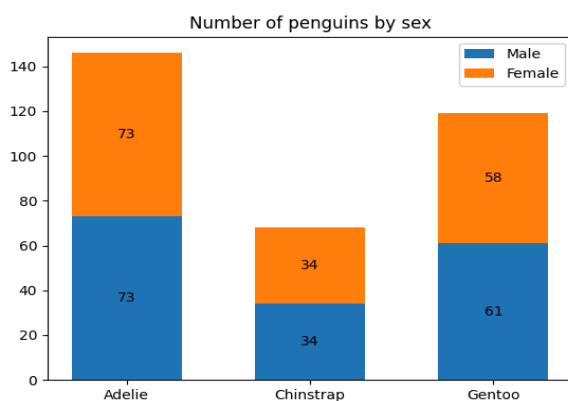
### Graficando con Matplotlib.



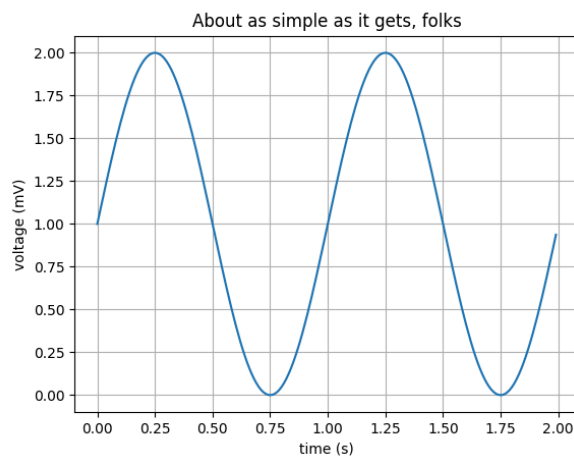
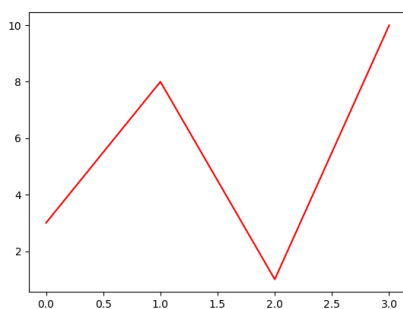
Esta librería nos permite realizar diversos tipos de gráficos.

#### Gráficos de barras:





### Gráficos de línea.



### Gráficos de dispersión (puntos)

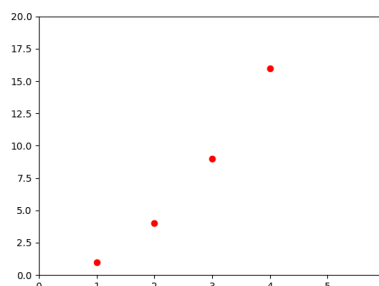
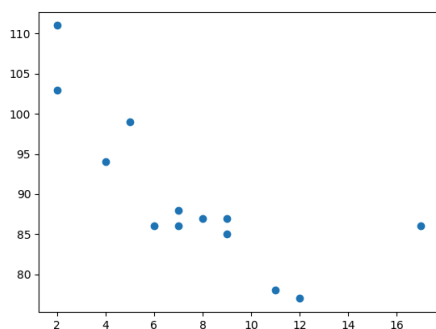
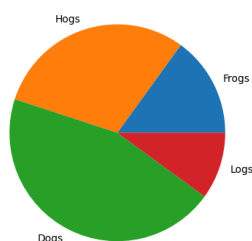
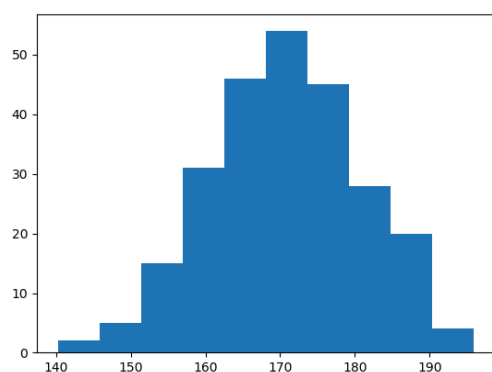


Gráfico de torta.



Histogramas.



## Instalación Matplotlib

Para utilizar la librería matplotlib en primer lugar debemos instalarla.

Código

```
##  
!pip install matplotlib  
##
```

Una vez instalada la librería la vamos a importar la librería y su módulo pyplot. Por convención se suele importar con el alias plt. Como vamos a graficar datos que provengan de un dataframe recuerda también importar pandas.

Código

```
##  
import pandas as pd # importamos pandas como pd  
import matplotlib.pyplot as plt # importamos matplotlib  
##
```

Luego necesitamos el dataframe con los datos que queremos graficar. Por el momento usaremos un dataframe con dos columnas: temperatura y humedad.

Código

```
##  
temperaturas = pd.Series([20, 25, 21, 24, 27, 27, 25])  
humedad = pd.Series([45, 65, 30, 70, 80, 65, 75])  
  
df = pd.DataFrame() # Creamos un dataframe vacío  
  
df['temperatura'] = temperaturas # creamos una columna llamada temperatura  
df['humedad'] = humedad # creamos una columna llamada humedad  
df  
##
```

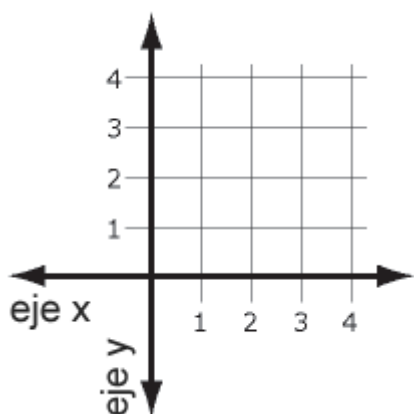
Lo que vemos en pantalla es:

```
##
```

```

temperatura  humedad
0           20      45
1           25      65
2           21      30
3           24      70
4           27      80
5           27      65
6           25      75
##
    
```

Como vemos tenemos definido un dataframe con dos columnas, las columnas contienen datos de tipo numérico, por lo cual podemos realizar gráficos. ¿Recordás las clases de matemáticas cuando realizan gráficos? Teníamos dos ejes, el x y el y, esto es exactamente igual. Vamos a hacer gráficos como los harías en la escuela. Nuestra famosa 'tablita' de la que hacíamos en clase es el dataframe.



Tener en cuenta que si utilizamos un gráfico lo que veremos son representaciones numéricas en el eje **y** (el vertical), vamos a graficar números.

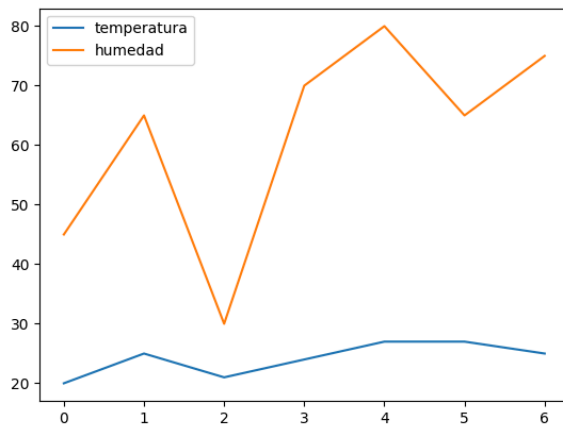
Podría pasar que nuestros datos de una columna sean de tipo categórico (una categoría de cosas), por ejemplo países. No podemos graficar cosas categóricas, pero podemos realizar algún proceso previo. Por ejemplo podríamos graficar cuantas veces se repite cada país en nuestro dataframe. Volveremos con esto más adelante. Por ahora nos limitaremos a datos numéricos.

### Gráfico de líneas.

Para hacer un gráfico de líneas, debemos usar la función `.plot()`. Como estamos usando un dataframe le aplicaremos al dataframe la función.

La línea `plt.show()` nos permite ver el gráfico generado.

```
##
df.plot() # realizamos un gráfico de líneas
plt.show() # mostramos el gráfico.
##
```



Obtenemos un gráfico de líneas con ambas columnas del dataframe. Es decir grafica todas las columnas que sean de tipo numéricas. Además nos aparece una leyenda, que nos dice que es cada color:



¿Qué pasaría si solo queremos graficar la temperatura? ¿Se te ocurre alguna manera de hacerlo?

Si te acordás, cuando queríamos solo una columna del dataframe lo que hacíamos era indicar la columna entre corchetes. ¿Qué tal si probamos eso y aplicamos el `.plot()`?

```
Código
##
df['temperatura'].plot()
##
```

¡Perfecto! Vamos entendiendo la lógica de cómo hacer gráficos. Nota que desapareció la leyenda, ya que en el gráfico solo tenemos una columna.



Otra forma es no usar `.plot()` aplicado al dataframe. Sino usando directamente la librería `matplotlib`.

Para esto necesitamos crear dos objetos de la librería: `Figure` y `Axes`. Esto se hace con la función `subplots()`. La función nos crea un objeto `Figure` y un `Axes`, vamos a llamarlos con convención `fig` y `ax`.

`fig` será el contenedor de nuestro gráfico y `ax` serán los ejes donde se dibujan los gráficos.

Luego sobre `ax` realizamos el `plot()`. Ahora dentro del `plot` indicamos cuál es la columna que queremos graficar.

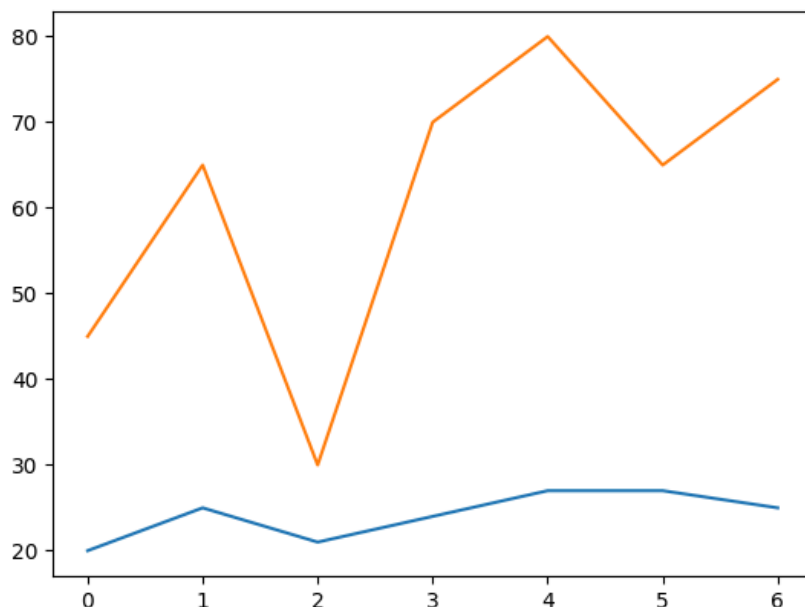
Código

```
##  
fig, ax = plt.subplots() # creamos la figura y los ejes  
ax.plot(df['temperatura'])  
plt.show()  
##
```

Si quisiéramos en el mismo gráfico ambas columnas, debemos hacer dos `plots`:

Código

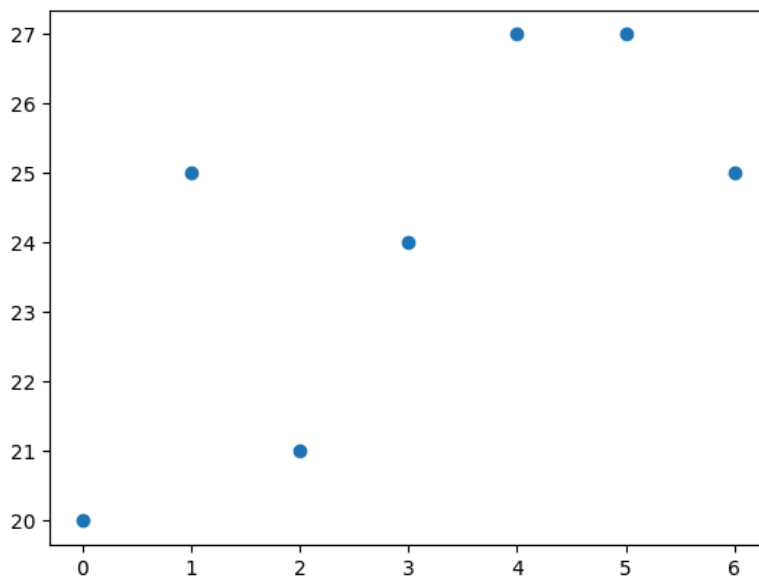
```
##  
fig, ax = plt.subplots()  
ax.plot(df['temperatura'])  
ax.plot(df['humedad'])  
plt.show()  
##
```



### Gráfico de puntos.

Ya vimos cómo realizar el gráfico de líneas. Si en lugar de que los valores se unan en una línea, necesitamos solamente los puntos podemos hacer un gráfico de dispersión. Para esto usamos la función `scatter()`, dentro debemos aclarar cuáles son los valores que van en el eje x y cuáles en el y: `ax.scatter(x, y)`. Nuestros valores en el eje x serán los índices de nuestro dataframe: `df.index`

```
##
fig, ax = plt.subplots()
ax.scatter(df.index, df['temperatura'])
plt.show()
##
```



### Gráfico de barras y torta

Para los gráficos de tipo línea o dispersión necesitábamos tener variables numéricas en ambos ejes. Si observas, en ambos, en el eje x se graficaron los índices del dataframe.

¿Cómo haríamos si tengo variables categóricas? Supongamos que tenemos un dataframe con dos columnas. En una tenemos los cursos disponibles y en otra la cantidad de estudiantes en ese curso.

Código:

```
##
df = pd.DataFrame({'curso': ['python', 'IA', 'videojuegos'], 'estudiantes': [32, 43, 37]})
df
##
```

Vemos el siguiente dataframe

```
##
      curso  estudiantes
0   python           32
1      IA            43
2 videojuegos        37
##
```

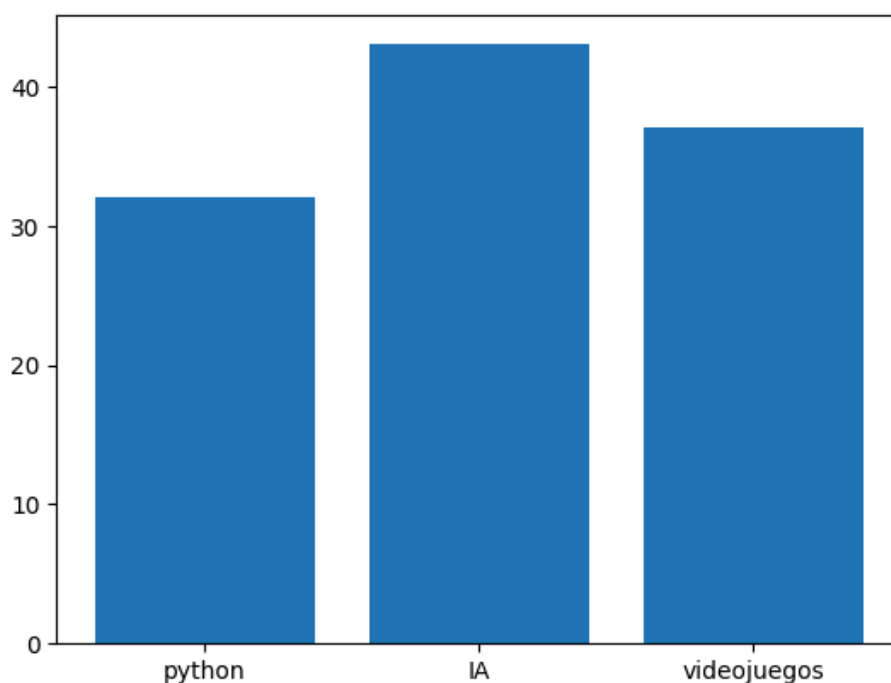


Ahora lo que queremos es realizar un gráfico con esos datos. Podríamos realizar dos tipos de gráfico. Un gráfico de barra, en la que cada barra es un curso, o un gráfico de torta. Comencemos con el gráfico de barras.

Para realizarlo debemos usar la función `.bar()` y pasarle dos parámetros. En la primera posición lo que queremos ver en el eje x, y en la segunda posición el eje y.

```
##
fig, ax = plt.subplots()
ax.bar(df['curso'], df['estudiantes'])
plt.show()
##
```

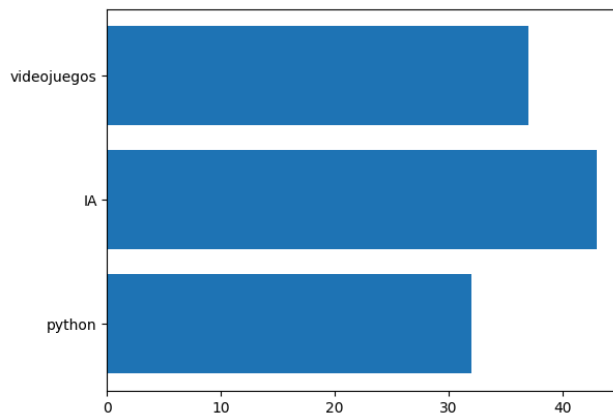
Obtenemos:



También podemos cambiar la disposición, para que las barras sean horizontales. En lugar de la función `bar()` usamos `barh()`.

Código:

```
##
fig, ax = plt.subplots()
ax.barh(df['curso'], df['estudiantes'])
plt.show()
##
```

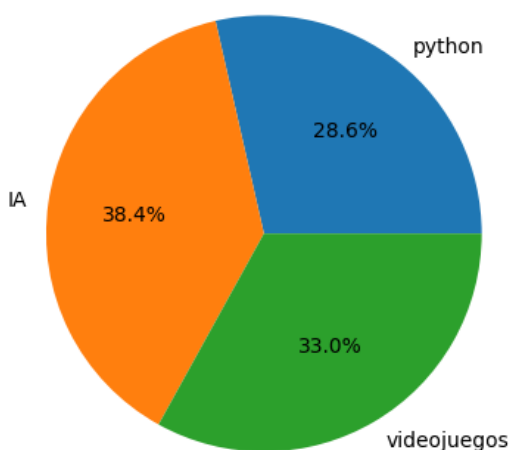


Usemos el mismo dataframe para crear un gráfico de torta.

Aquí usaremos la función `pie()`. Como parámetros debemos pasar el valor a graficar en primer lugar (la columna con la cantidad de estudiantes). Un parámetro `labels`, que serán las etiquetas (en este caso la columna 'cursos'). Podemos agregar que nos muestre los porcentajes con el parámetro `autopct='%0.1f%%'`.

Código:

```
##
fig, ax = plt.subplots()
ax.pie(df['estudiantes'], labels=df['curso'], autopct='%0.1f%%')
plt.show()
##
```



### Histogramas.

Los histogramas son un tipo de gráfico en el que vemos la cantidad de veces que se repite un dato, es decir, la frecuencia.

Tomemos un nuevo dataframe. Aquí tenemos un dataframe con estudiantes y sus notas.

Código:

```
##
df_notas = pd.DataFrame({
    'Nombre': ['Lucas', 'Pablo', 'Líam', 'Abril', 'Esteban', 'Sofía',
              'Gustavo', 'Ana', 'Martina', 'Luciana'],
    'Calificación': [8, 9, 7, 6, 9, 10, 4, 3, 10, 8]})
df_notas
##
```

Lo que obtenemos es:

```
##
  Nombre  Calificación
0  Lucas             8
1  Pablo             9
2  Líam              7
3  Abril             6
```

```

4     Esteban      9
5     Sofía       10
6     Gustavo      4
7     Ana          3
8     Martina     10
9     Luciana      8
##
    
```

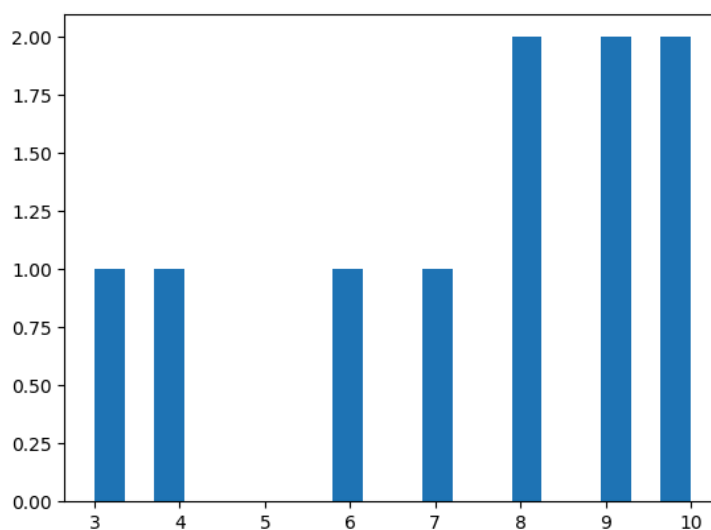
Realicemos el histograma de las calificaciones.

Código:

```

##
fig, ax = plt.subplots()
ax.hist(df_notas['Calificación'], bins=20)
plt.show()
##
    
```

Usamos la función `hist()`. Como parámetros indicamos cuál es la columna de nuestro dataframe, y el número de bins indica en cuántos intervalos se divide nuestro rango. Prueba modificando el valor de este parámetro cambiándolo por otros valores (deben ser enteros) y observa lo que ocurre.



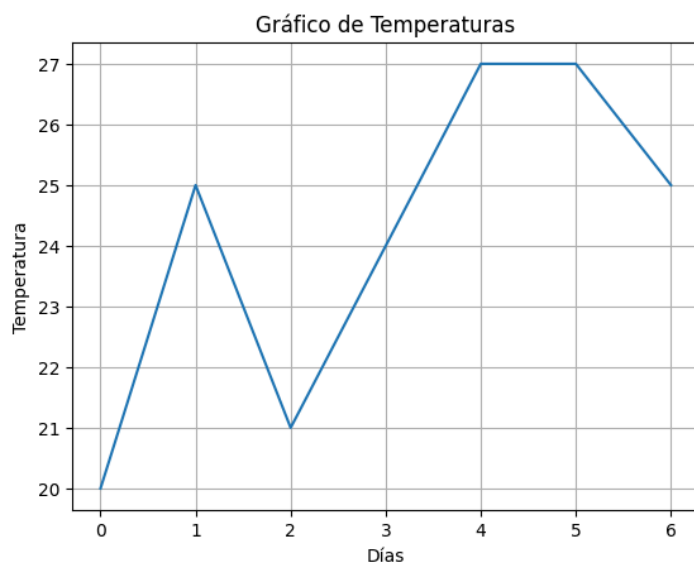
**Títulos, leyendas y estilos.**

A todos los gráficos anteriores quisiéramos agregarles estilos. Tomemos el primer gráfico que hicimos.

Código:

```
##
fig, ax = plt.subplots()
ax.plot(df['temperatura'])
ax.set_title('Gráfico de Temperaturas') # Agrego título
ax.set_xlabel('Días') # Nombre del eje x
ax.set_ylabel('Temperatura [°C]') # Nombre del eje y
ax.grid(True) # Agrego una grilla
plt.show()
##
```

Como verás, usamos `set_title()` para los títulos, `set_xlabel()` y `set_ylabel()` para los ejes. Además le agregamos una grilla.



Podemos también cambiar los colores del gráfico y agregar leyendas. Para cambiar el color agregamos el parámetro `color` al `plot`. En la siguiente página tienes la documentación sobre los colores: [matplotlib colores](#).

Código:

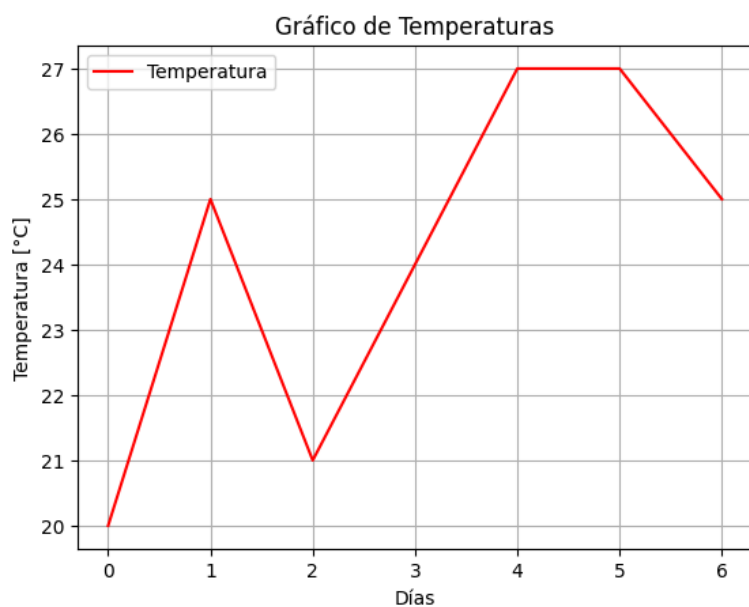
```
##
```



```

fig, ax = plt.subplots()
ax.plot(df['temperatura'], label='Temperatura', color='r') # Agregamos label que será la leyenda
                                                         # Cambiamos el color

ax.set_title('Gráfico de Temperaturas')
ax.set_xlabel('Días')
ax.set_ylabel('Temperatura [°C]')
ax.grid(True)
ax.legend() # Ver las leyendas
plt.show()
##
    
```



## Graficando con Matplotlib y Seaborn

Podemos complementar el uso de Matplotlib con la librería Seaborn. Esto nos permitirá hacer gráficos con menos líneas de código y estilos ya definidos.

Hagamos algunos gráficos usando el archivo .csv de películas de [Amazon Prime](#).

Primero debemos instalar la librería.

Código:

```
##  
!pip install seaborn  
##
```

En nuestro .csv tenemos la distintos directores, queremos graficar los 10 directores con mayor número de producciones.

Elegimos la columna que contiene los directores: 'director', borramos los datos nulos de la columna usando dropna(). Luego usamos .value\_counts() para que nos cuente cuantas veces aparece cada director. Como queremos los primeros 10 aplicamos head(10).

Guardamos todo en un nuevo dataframe que lo llamamos top\_10\_directores.

Código:

```
##  
top_10_directores = df['director'].dropna().value_counts().head(10)  
top_10_directores  
##
```

Lo que vemos es:

```
##  
director  
Mark Knight
```



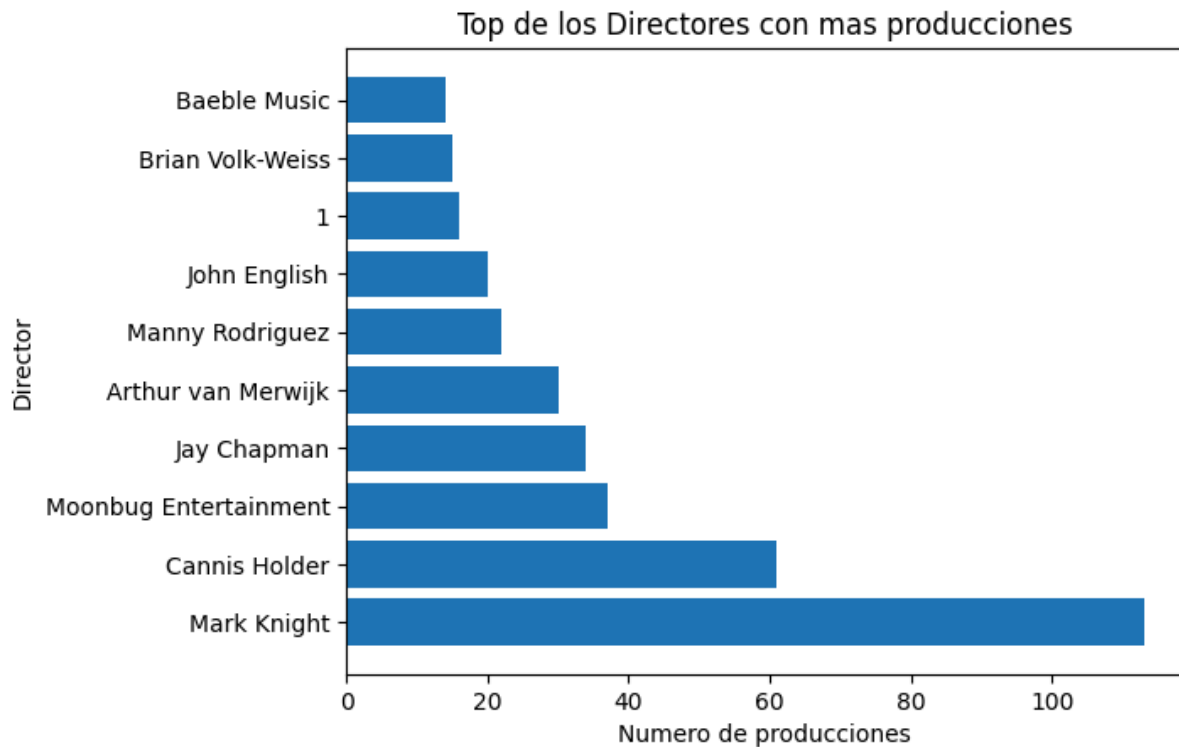
Cannis Holder	61
Moonbug Entertainment	37
Jay Chapman	34
Arthur van Merwijk	30
Manny Rodriguez	22
John English	20
1	16
Brian Volk-Weiss	15
Baeble Music	14
##	

En este nuevo dataframe 'director' no es una columna, sino los índices.

Si lo hacemos usando solamente matplotlib tendríamos lo siguiente:

Código:

```
##
fig, ax = plt.subplots()
ax.barh(top_10_directores.index, top_10_directores.values)
ax.set_title('Top de los Directores con mas producciones')
ax.set_xlabel('Numero de producciones')
ax.set_ylabel('Director')
plt.show()
##
```



Ahora veamos la diferencia si usamos Seaborn.

Importamos primero ambas librerías. Seaborn por convención la importamos como sns.

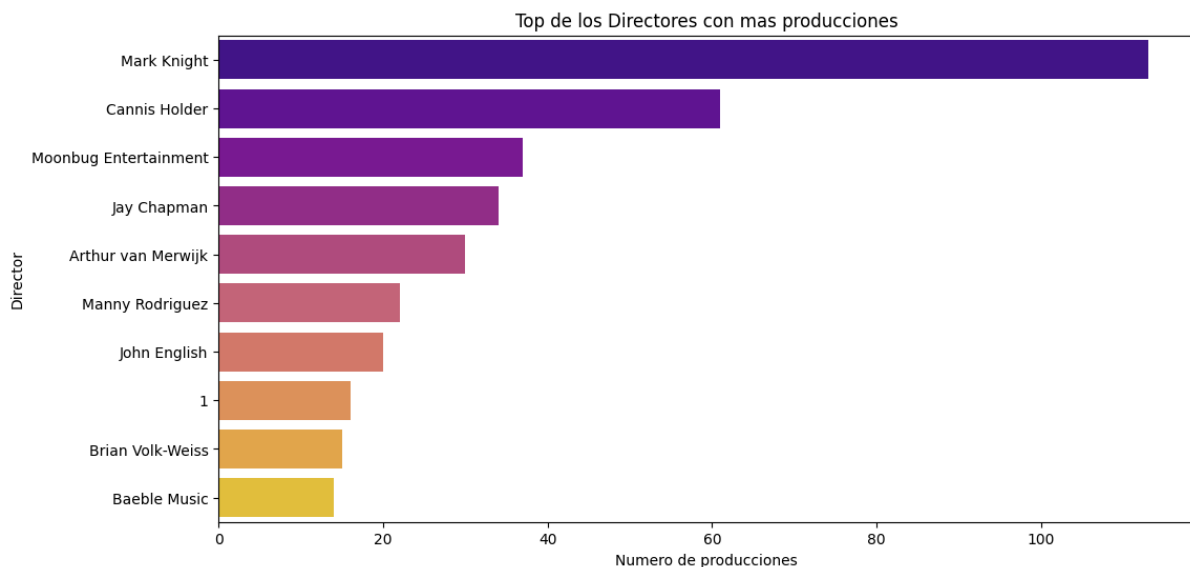
Código:

```
##
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6)) # figsize nos permite ajustar el tamaño
sns.barplot(x=top_10_directores.values, y=top_10_directores.index,
palette='plasma') # gráfico de barras
plt.title('Top de los Directores con mas producciones') # Agregó título
plt.xlabel('Número de producciones') # eje x
plt.ylabel('Director')
plt.show()
##
```

En la línea `plt.figure(figsize=(12, 6))` ajustamos el tamaño de nuestro gráfico. Para el gráfico de barras usamos: `sns.barplot(x=top_10_directores.values, y=top_10_directores.index, palette='plasma')`. Estamos usando la librería seaborn, y de la misma la función `barplot`. Dentro debemos indicarle que irá en el eje x y que irá en el eje y. Como el gráfico es horizontal, el nombre de los directores será el eje y, y la cantidad de producciones irá en el eje x. Además hay un parámetro `palette`, el mismo viene de la librería `matplotlib` y nos cambia la paleta de colores. Puedes ver más paletas [aquí](#)

El gráfico obtenido es:



Ya se ven un poco mejor nuestros gráficos.

Un caso en que resulta mejor usar Seaborn es si queremos un gráfico de barras que nos cuente cuántas veces aparece una variable, pero separados por un tipo de dato. En nuestro dataset tenemos una columna 'tipo', que puede ser "Movie" o "TV Show". Hagamos un gráfico del "Rating", pero diferenciando los tipos.

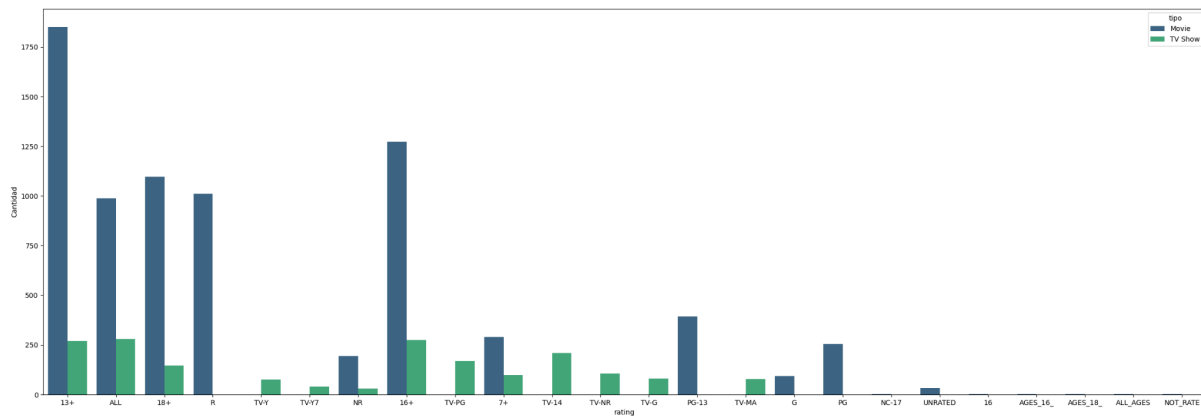
Código:

```

##
plt.figure(figsize=(30,10))
sns.countplot(x= df['rating'],data=df,hue = 'tipo', palette='viridis')
plt.ylabel('Cantidad')
plt.show()
    
```

##

Usamos la función `countplot`, dentro debemos indicar el eje x (parámetro `x`), cuál es el dataframe que estamos usando (parámetro `data`), con respecto a que columna queremos diferenciar las barras (parámetro `hue`) y la paleta si lo deseamos. Obtendremos este gráfico:



## Descargar los gráficos

Todos los gráficos que creamos los podemos descargar para utilizarlos en otros archivos. Para esto debemos agregar una línea luego de generar nuestro gráfico.

Código:

##

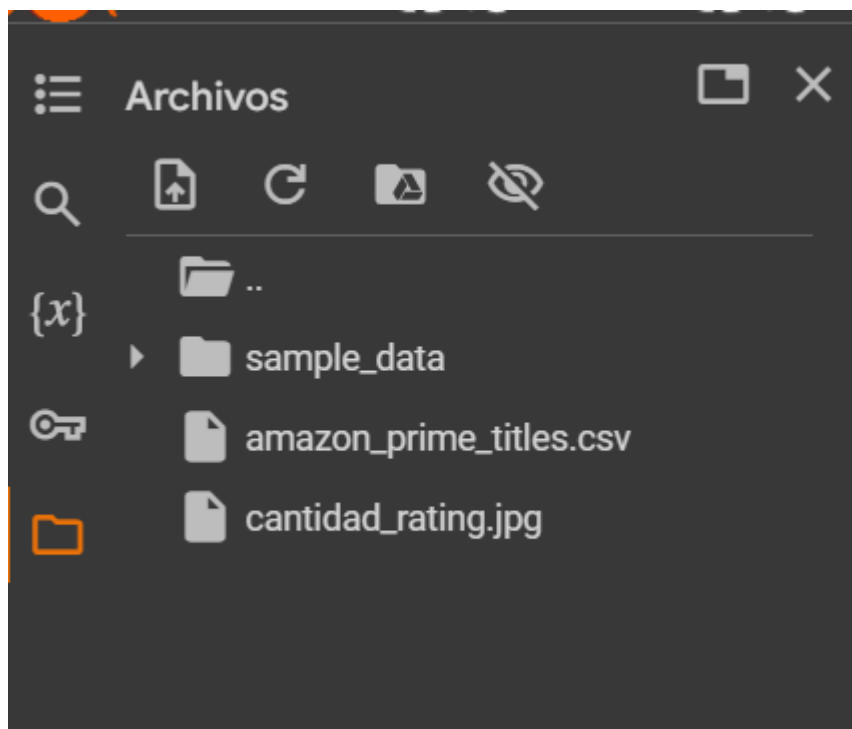
```

plt.figure(figsize=(30,10))
sns.countplot(x= df['rating'],data=df,hue = 'tipo',palette='viridis')
plt.ylabel('Cantidad')
plt.title('Cantidad de producciones según el público')
plt.show()
plt.savefig('cantidad_rating.jpg') # guardamos en formato .jpg
##
    
```

Dentro de `plt.savefig()` colocamos el nombre con el cual queremos descargar nuestro archivo y su formato (.jpg).



En el Colab, en la carpeta de la barra de la izquierda encontrarás tu gráfico y podrás descargarlo.



Te invito a que explores creando tus gráficos. Puedes utilizar ambas librerías o solo matplotlib si lo prefieres.

Piensa qué datos puedes graficar y qué gráfico es adecuado para cada caso. No te olvides de agregar los nombres a los ejes, títulos y leyendas según lo requieras.

En la próxima clase seguiremos realizando gráficos pero con una nueva herramienta: Tableau. Aprenderás a crear tus propios tableros con los cuales presentar tus resultados.

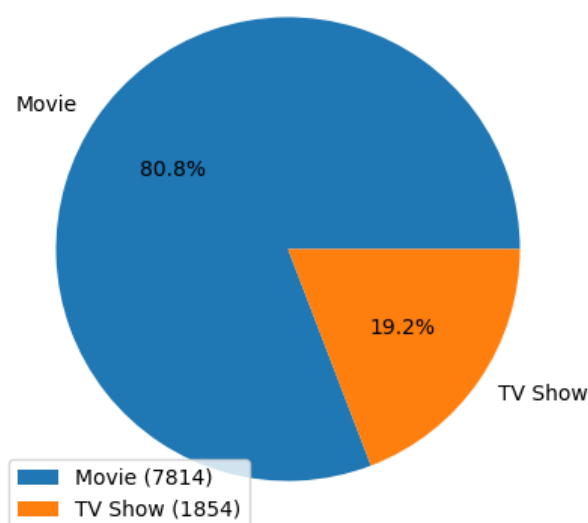
¡Allá vamos!

## Desafío N° 7:


Seguiremos usando el dataset de producciones de Amazon. Importa el .csv y crea un dataframe con los datos. Utiliza la librería matplotlib para crear un gráfico de torta en el que se vean los porcentajes de producciones según el tipo: Movie y Show TV Show. Muestra en el gráfico el porcentaje de cada tipo. Agrega el título y las leyendas. Descarga el archivo con el nombre 'grafico\_producciones' en formato .jpg.

Debes obtener algo como el siguiente gráfico:

Cantidad de películas y TV Shows



### [Colab del Desafío 7](#)

 **Haz una copia de este Colab antes de comenzar tu código.**

Ve a Archivo ➡ Guardar una copia en Drive.  
Luego escribe el código en esa copia





**Buenos Aires**  
*aprende*  
Agencia de Actividades para el Futuro

**BA** Buenos  
Aires  
Ciudad