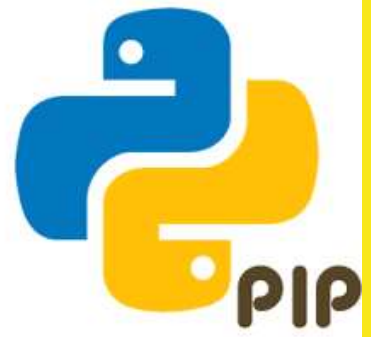


Grupo 6: Introducción a PIP y Bibliotecas Populares (Numpy-Pandas-Matplotlib)

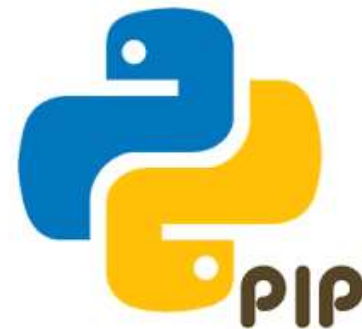
FACTORÍA F5 - GRUPO 6



Uso del gestor de paquetes PIP

Python

Qué es PIP?



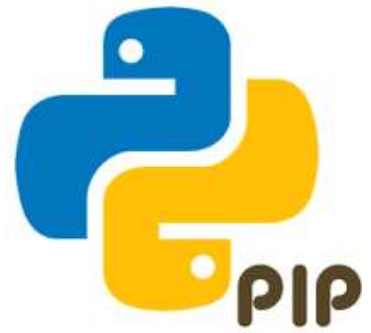
Pip es un gestor de paquetes Python, que nos permite instalar, actualizar, eliminar y buscar paquetes y librerías (bibliotecas) de Python, y nos proporciona una forma conveniente de gestionar dependencias y versiones entre diversas librerías de Python que tengamos en nuestro sistema. Pip puede ser utilizado en Linux, macOS y Windows. Es el acrónimo de «Pip Installs Packages» y es una herramienta muy útil.

Los paquetes Python instalados mediante pip proceden del siguiente repositorio:

<https://pypi.org/>

Aclaración previa: Antes de instalar PIP tenemos que tener instalado Python (Ver cómo instalar python en presentación del Grupo 1 o [aquí](#)). macOS suele incluir Python 2.7 pre-instalado en versiones 10.8 y 12.3.

Instalar el gestor de paquetes PIP En GNU-Linux



Según el gestor de paquetes que se posee se utiliza un comando u otro:

- 1) Para instalar pip en una distribución que use el gestor de paquetes apt deberán ejecutar el siguiente comando en la terminal:

```
sudo apt install python3-pip
```

- 1) Si por lo contrario usan el gestor de paquetes dnf deberán ejecutar el siguiente comando:

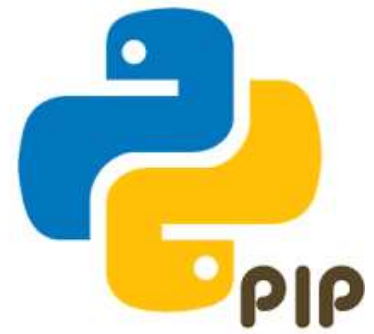
```
sudo dnf install python3 python3-wheel
```

- 1) Si usan Pacman:

```
pacman -S python-pip
```

- 1) Finalmente si usan Zipper deberán ejecutar el siguiente comando:

```
sudo zypper install python3-pip python3-setuptools python3-wheel
```



Instalar el gestor de paquetes PIP en macOS

Sigue estos pasos:

1. Descarga el script de instalación de get-pip.py: Puedes descargar este script usando curl, ejecuta el siguiente comando en la Terminal:

curl <https://bootstrap.pypa.io/get-pip.py> o <https://bootstrap.pypa.io/pip/2.7/get-pip.py> (en caso de contar con versión 2.7 de python)

2. Ejecuta el script: Usa el comando python para ejecutar el script que descargaste en el paso anterior. Esto instalará pip y setuptools. Ejecuta el siguiente comando en la Terminal:

python get-pip.py o

Nota: Si tienes Python 3 instalado, tal vez necesites usar python3 en lugar de python. **python3 get-pip.py**

3. Verifica la instalación: Ejecuta el siguiente comando en la Terminal:

pip --version

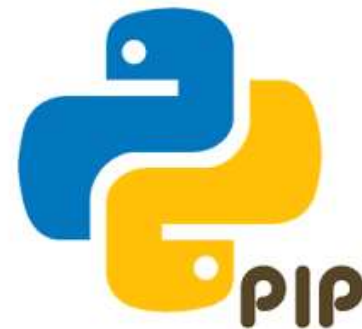
Deberías ver la versión de pip y su ubicación en la salida.

Ahora deberías tener pip instalado en tu macOS y estar listo para usarlo para instalar paquetes de Python.

Recuerda que si tienes problemas con permisos, puedes usar el modificador sudo para ejecutar los comandos como administrador:

sudo python get-pip.py

Sin embargo, ten en cuenta que usar sudo conlleva riesgos y debe usarse con precaución.



Instalar el gestor de paquetes PIP en Windows

Para instalar PIP en Windows tenemos que proceder del siguiente modo:

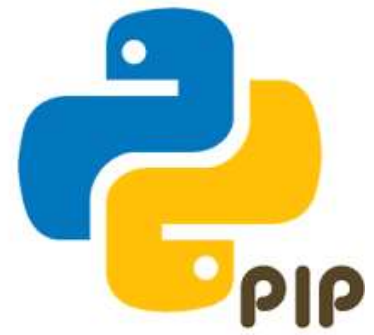
- 1) Abrir el navegador web y acceder a <https://bootstrap.pypa.io/get-pip.py> para descargar el fichero `get-pip.py`.
- 2) Abrir la consola de windows e ir a la ubicación donde hemos descargado el fichero `get-pip.py`.
- 3) Finalmente ejecutamos el siguiente comando para que se instale el gestor de paquetes PIP.

```
python3 get-pip.py
```

- 1) Para verificar que PIP se ha instalado correctamente tan solo hay que ejecutar el siguiente comando:

```
pip --version
```

Si aparece información sobre la versión de PIP que has instalado, entonces la instalación ha sido exitosa.



Operaciones que se pueden realizar con Pip(parte 1)

`pip install nombre_paquete`: Para instalar un programa o librería.

`pip install --pre nombre_paquete`: Instalar la versión beta de un programa o librería.

`pip download nombre_paquete`: Para descargar un paquete y la totalidad de sus dependencias sin instalar nada en el sistema operativo.

`pip uninstall nombre_paquete`: Para desinstalar un programa o librería.

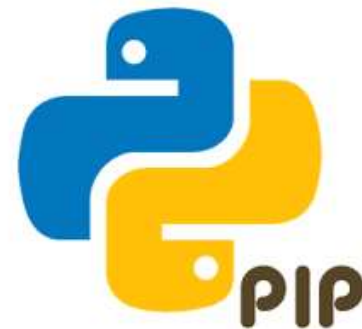
`pip freeze`: Listar la totalidad de paquetes python instalados con pip indicando su versión y siguiendo siempre un formato determinado.

`pip list`: Muestra los mismos resultados que el comando `pip freeze`. Los resultados son más fáciles de leer, pero no siguen un formato tan rígido como `pip freeze`.

`pip list --outdated`: Listar la totalidad de paquetes obsoletos y que por lo tanto tienen una versión más reciente disponible.

`pip show nombre_paquete`: Obtener información sobre un paquete instalado mediante pip.

`pip check`: Para verificar que los paquetes instalados están bien instalados y no tienen problemas de dependencias.



Operaciones que se pueden realizar con Pip(parte 2)

`pip install --upgrade nombre_paquete`: Para actualizar un paquete específico a la última versión.

`pip freeze | grep -v '^\\-e' | cut -d = -f 1 | xargs -n1 pip install -U`: Actualizar la totalidad de paquetes a la última versión.

`pip freeze --local | grep -v '^\\-e' | cut -d = -f 1 | xargs -n1 pip install -U`: Actualizar la totalidad de paquetes a la última versión de únicamente los paquetes instalados dentro del entorno virtual de desarrollo activo.

`pip freeze | xargs pip uninstall -y`: Para eliminar la totalidad de paquetes Python instalados.

`pip cache dir`: Ver la ubicación del directorio que almacena copias en forma de cache de los paquetes que se descargan de los repositorios de PyPI (Python Package Index).

`pip cache purge`: Limpiar la caché de pip.

`pip --version`: Mostrar la versión de pip instalada en el sistema operativo.

`pip --help`: Mostrar ayuda para usar el comando pip.



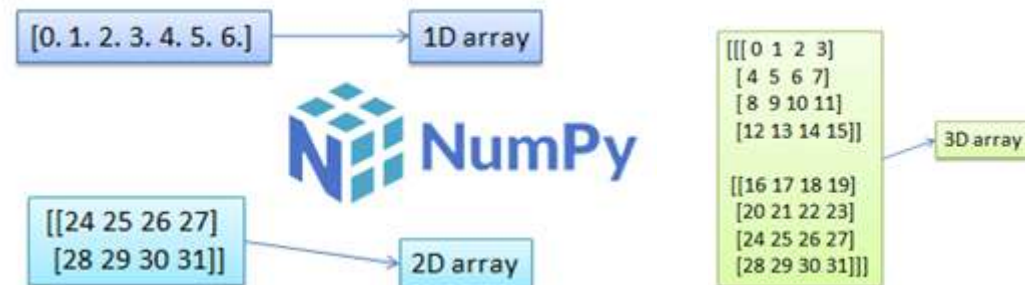
NumPy

Introducción a la manipulación de arrays

¿Qué es?

NumPy (Numerical Python) es una biblioteca de Python que proporciona soporte para trabajar con arreglos multidimensionales, matrices y funciones matemáticas de alto nivel.

Más información: [Sitio Oficial Numpy](https://numpy.org)



¿Qué ofrece?

Proporciona estructuras de datos eficientes y optimizadas para cálculos matemáticos y ofrece una amplia gama de funciones matemáticas.

Algunas de las aplicaciones de NumPy incluyen:

- Álgebra lineal
- Análisis estadístico
- Generación de números aleatorios
- Operaciones con matrices

¿ Por qué utilizar Numpy?

- **Homogeneidad de tipos de datos**

Todos los elementos del array deben ser del mismo tipo de datos.

- **Rendimiento**

Los arrays de NumPy están diseñados para un rendimiento óptimo en operaciones matemáticas y de manipulación de datos.

- **Operaciones matemáticas y de álgebra lineal**

Los arrays de NumPy permiten operaciones elemento por elemento, broadcasting y otras características que facilitan las operaciones matemáticas.

- **Tamaño**

Tienen un tamaño fijo en el momento de la creación, lo que significa que no puedes agregar ni eliminar elementos del array sin crear un nuevo array.

- **Memoria**

¿ Qué es ndarray ?

Es una estructura de datos eficiente para trabajar con matrices multidimensionales en NumPy.

- **ndim**: Número de dimensiones (ejes) de la matriz.
- **shape**: Tupla que indica el tamaño de la matriz en cada dimensión (filas, columnas, etc.).
- **size**: Número total de elementos en la matriz.
- **dtype**: Tipo de datos de los elementos (**int32**, **float64**).
- **itemsize**: Tamaño en bytes de cada elemento.
- **data**: Búfer que contiene los datos sin procesar de la matriz.

Creación De Arrays

FUNCIÓN	DESCRIPCIÓN	EJEMPLO
<code>np.array()</code>	Crear un array	<pre>>>>a = np.array([1, 2, 3])</pre>
<code>np.zeros()</code>	Crear un array de ceros	<pre>>>> np.zeros(2) array([0., 0.])</pre>
<code>np.ones()</code>	Crear un array de unos	<pre>>>> np.ones(2) array([1., 1.])</pre>
<code>np.arange()</code>	Crea un array con valores espaciados uniformemente entre un rango dado	<pre>>>>np.arange(2, 9, 2) array([2, 4, 6, 8])</pre>
<code>np.linspace()</code>	Crea un array con un número específico de puntos espaciados uniformemente entre un rango	<pre>>>> linspace(0, 10, num=5) array([0., 2.5, 5., 7.5, 10.])</pre>
<code>np.random.rand()</code>	crear un array con valores aleatorios uniformemente distribuidos entre 0 y 1	<pre>>>> np.random.rand(2, 3)</pre>

Indexación y Selección

La indexación y selección en NumPy permite acceder a elementos específicos, filas, columnas o secciones de un array.

Ejemplos:

- Indexar y dividir matrices

```
>>> data = np.array([1, 2, 3])  
  
>>> data[1]  
2  
  
>>> data[0:2]  
array([1, 2])  
  
>>> data[1:]  
array([2, 3])  
  
>>> data[-2:]  
array([2, 3])
```

- **Seleccionar valores de su matriz que cumplan ciertas condiciones**

```
>>> a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
```

```
>>> divisible_by_2 = a[a%2==0]  
>>> print(divisible_by_2)  
[ 2  4  6  8 10 12]
```

```
>>> c = a[(a > 2) & (a < 11)]  
>>> print(c)  
[ 3  4  5  6  7  8  9 10]
```

Manipulación de Arrays

NumPy ofrece varias funciones para cambiar la forma, redimensionar y aplanar arrays.

FUNCIÓN	DESCRIPCIÓN	EJEMPLO
<code>reshape()</code>	Remodela la matriz sin cambiar los datos	<pre>>>> arr_1d = np.array([1, 2, 3, 4, 5, 6]) >>> arr_2d = arr_1d.reshape(2, 3)</pre>
<code>T</code>	Cambiar los ejes de una matriz	<pre>>>> arr_2d.T</pre>
<code>flatten()</code>	Aplanar la matriz en una matriz 1D	<pre>>>> arr_2d.flatten() [1 2 3 4 5 6]</pre>
<code>np.newaxis</code>	Aumenta las dimensiones de la matriz en una dimensión	<pre>>>> arr_1d[:, np.newaxis] [[1] [2]]</pre>
<code>np.resize()</code>	Remodela la matriz pero puede cambiar los datos	<pre>>>> arr_resized = np.resize(arr_1d, (2, 4)) [[1 2 3 4] [5 6 1 2]]</pre>
<code>np.stack()</code>	Concatena dos arrays	<pre>>>> arr_1 = np.array([1, 2, 3]) >>> arr_2 = np.array([4, 5, 6]) >>> arr_concatenated = np.stack((arr_1, arr_2))</pre>
<code>np.sort()</code>	Ordena los elementos de una matriz	<pre>>>> np.sort(np.array([5, 2, 9, 1, 7, 3])) [1 2 3 5 7 9]</pre>

Operaciones

1. Aritméticas

Numpy permite realizar operaciones aritméticas elemento a elemento en arrays, como suma '+', resta '-', multiplicación '*' y división '/'.

1. Matemáticas y trigonométricas

Numpy proporciona funciones como exponentes '*np.exp(g)*', logaritmos '*np.log(g)*', raíces cuadradas '*np.sqrt(g)*' y funciones trigonométricas '*np.sin(g)*'.

Funciones Universales (ufunc)



- Son funciones que operan sobre arrays de NumPy elemento por elemento.
- Incluyen operaciones matemáticas simples, como suma, resta, multiplicación y división, y funciones trigonométricas, exponenciales y logarítmicas
- Son útiles para realizar operaciones matemáticas y lógicas en arrays sin necesidad de utilizar bucles explícitos.
- Ejemplos:

```
arr1 = np.array([1, 2, 3, 4])  
arr2 = np.array([5, 6, 7, 8])
```

```
sum_result = np.add(arr1, arr2)  
print("Suma:", sum_result)
```

```
quotient_result = np.divide(arr1, arr2)  
print("Cociente:", quotient_result)  
# Output: Cociente: [0.2 0.33333333  
0.42857143 0.5]
```

Pandas: Manejo de datos tabulares.

¿Qué es Pandas?



Es una biblioteca para el análisis y manipulación de datos.

Especialmente útil para trabajar con datos tabulares, como hojas de cálculo, bases de datos y almacenes de datos o data warehouses.

Fácil de usar para usuarios familiarizados con SQL y Python.

Es una herramienta esencial para analistas, científicos de datos e ingenieros de datos que trabajan con datos estructurados.

Características Principales

Facilidad de uso: Pandas ofrece una sintaxis intuitiva y sencilla de aprender, lo que la hace accesible para usuarios de todos los niveles.

Versatilidad: Pandas puede trabajar con una amplia variedad de tipos de datos, incluyendo números, cadenas de texto, fechas y valores booleanos.

Rendimiento: Pandas está optimizada para el manejo de grandes conjuntos de datos, lo que la hace ideal para tareas de análisis de datos a gran escala.

Integración con otras bibliotecas: Pandas se integra a la perfección con otras bibliotecas populares de Python para el análisis de datos, como NumPy, Matplotlib y scikit-learn.

Casos de uso

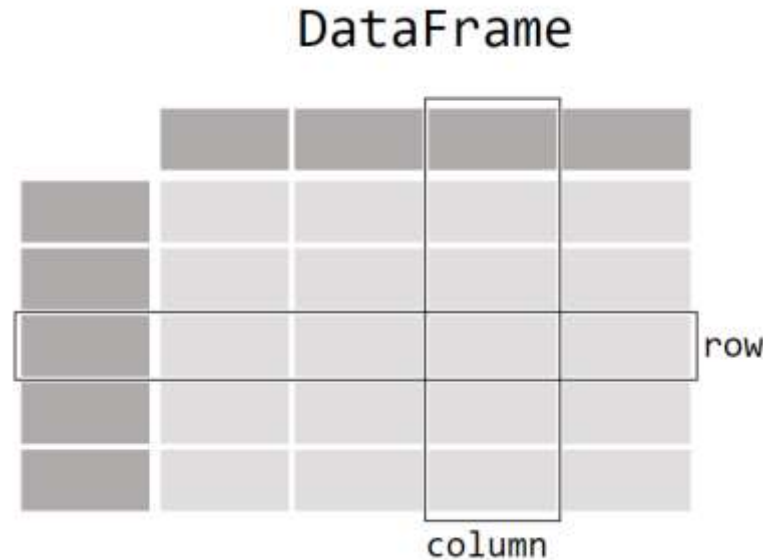
Limpieza y preparación de datos: Pandas proporciona herramientas para limpiar y preparar datos para su análisis, como eliminar valores faltantes, corregir errores y normalizar formatos.

Análisis exploratorio de datos: Pandas permite explorar y visualizar datos para identificar patrones, tendencias y anomalías.

Manipulación de datos: Pandas facilita la manipulación de datos, como agregar, eliminar y modificar filas y columnas, unir conjuntos de datos y realizar cálculos estadísticos.

Visualización de datos: Pandas también ofrece funciones integradas para crear gráficos y visualizaciones de datos, como histogramas, gráficos de dispersión y gráficos de líneas.

Estructuras de datos principales



Series:

Son un arreglo unidimensional y etiquetado en el que se pueden almacenar datos de cualquier tipo.

DataFrames:

Estructura de dos dimensiones que almacena información como un arreglo bidimensional o como una tabla con renglones y columnas.

Métodos principales (lectura y manipulación)

Método	Descripción
<code>read_csv()</code>	Lee datos desde un archivo CSV en un DataFrame.
<code>read_json()</code>	Lee datos desde un archivo JSON en un DataFrame.
<code>head()</code>	Muestra las primeras N filas de un DataFrame.
<code>tail()</code>	Muestra las últimas N filas de un DataFrame.
<code>info()</code>	Muestra información sobre la estructura y los datos de un DataFrame.
<code>describe()</code>	Calcula estadísticas descriptivas sobre los datos numéricos de un DataFrame.
<code>loc[]</code>	Selecciona filas y/o columnas por índice o etiqueta.
<code>iloc[]</code>	Selecciona filas y/o columnas por posición.
<code>dropna()</code>	Elimina filas con valores faltantes.
<code>fillna()</code>	Rellena valores faltantes con un valor especificado.

Métodos principales (agrupar, ordenar y combinar)

Método	Descripción
<code>groupby()</code>	Agrupar datos por una o más columnas y realiza operaciones agregadas.
<code>sort_values()</code>	Ordena un DataFrame por una o más columnas.
<code>merge()</code>	Combina dos DataFrames en uno nuevo.
<code>append()</code>	Añade filas a un DataFrame existente.
<code>drop()</code>	Elimina filas o columnas de un DataFrame.
<code>reset_index()</code>	Convierte el índice de un DataFrame en una columna.
<code>set_index()</code>	Convierte una columna en el índice de un DataFrame.
<code>concat()</code>	Combina dos o más DataFrames en uno nuevo.
<code>astype()</code>	Convierte el tipo de datos de una columna o un DataFrame.
<code>rename()</code>	Cambia el nombre de columnas o índices en un DataFrame.

Métodos principales (visualización y análisis)

Método	Descripción
<code>plot()</code>	Crea gráficos para visualizar datos.
<code>hist()</code>	Crea un histograma para una columna de datos numéricos.
<code>bar()</code>	Crea un gráfico de barras para una o más columnas de datos.
<code>boxplot()</code>	Crea un diagrama de cajas para una o más columnas de datos.
<code>scatter()</code>	Crea un diagrama de dispersión para dos columnas de datos.
<code>corr()</code>	Calcula el coeficiente de correlación entre dos columnas de datos.
<code>get_dummies()</code>	Crea variables binarias a partir de variables categóricas.
<code>factorize()</code>	Convierte variables categóricas en índices y valores numéricos.
<code>pd.cut()</code>	Crea categorías a partir de valores numéricos.
<code>pd.qcut()</code>	Crea categorías a partir de valores numéricos utilizando cuantiles.

Ejemplos



✓ Ejercicio No.1 - Crear una Serie desde una lista y visualizarla

```
import pandas as pd

data = ['Ana', 'Juan', 'Carlos', 'Laura']
serie = pd.Series(data)

#1.1 Desplegar la columna creada junto con su índice intrínseco
print('*** 1.1 ***')
print(serie)

#1.2 Desplegar el segundo elemento
print('*** 1.2 ***')
print(serie[1])
```

```
*** 1.1 ***
0      Ana
1      Juan
2    Carlos
3      Laura
dtype: object
*** 1.2 ***
Juan
```


Ejemplos

✓ Ejercicio No.2 - Crear un DataFrame y visualizarlo con condiciones


```
[2] import pandas as pd

data = {'Nombres': ['Ana', 'Juan', 'Carlos', 'Laura'],
        'Edades': [25, 32, 18, 21]}
df = pd.DataFrame(data)

#2.1 Desplegar el DataFrame df
print('*** 2.1 ***')
print(df)

#2.2 Desplegar únicamente el renglón correspondiente a 'Carlos'
print('*** 2.2 ***')
print(df[df['Nombres'] == 'Carlos'])

#2.3 Desplegar los renglones cuyas edad es menor o igual a 25
print('*** 2.3 ***')
print(df[df['Edades'] <= 25])
```



```
*** 2.1 ***
  Nombres  Edades
0     Ana      25
1     Juan      32
2    Carlos      18
3     Laura      21

*** 2.2 ***
  Nombres  Edades
2    Carlos      18

*** 2.3 ***
  Nombres  Edades
0     Ana      25
2    Carlos      18
3     Laura      21
```



Matplotlib

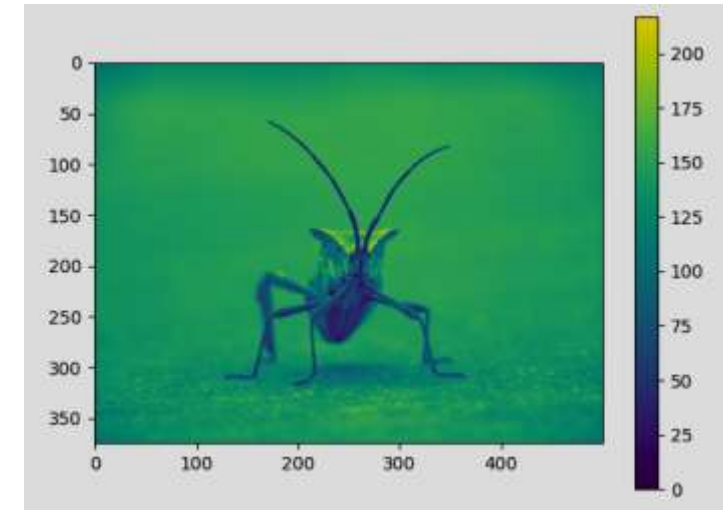
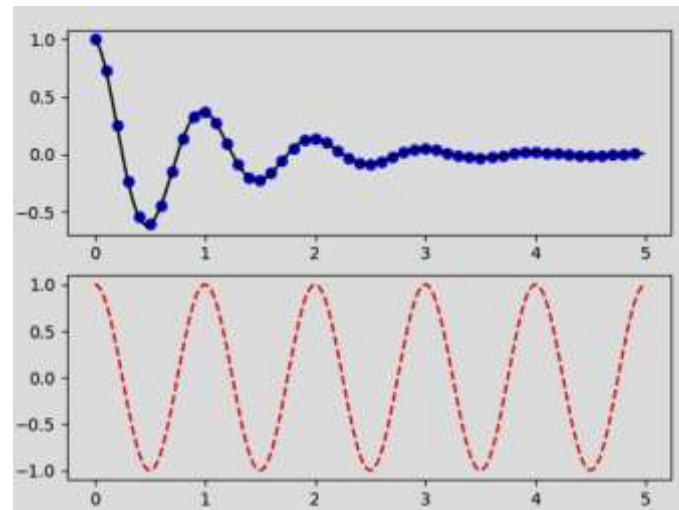
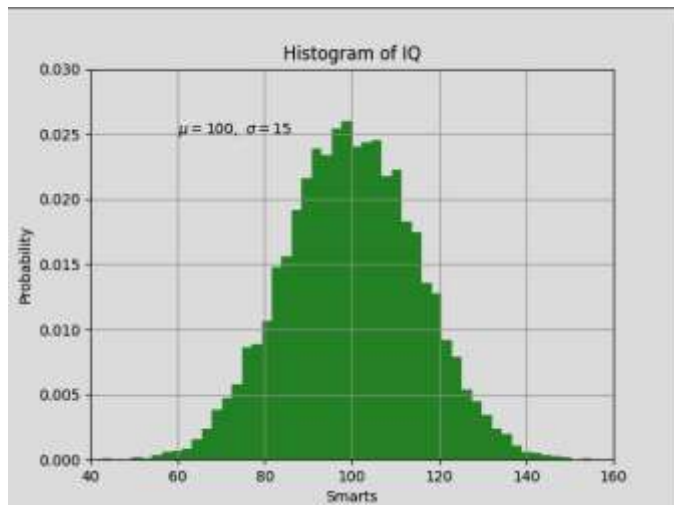


pyplot

¿Qué es Matplotlib?

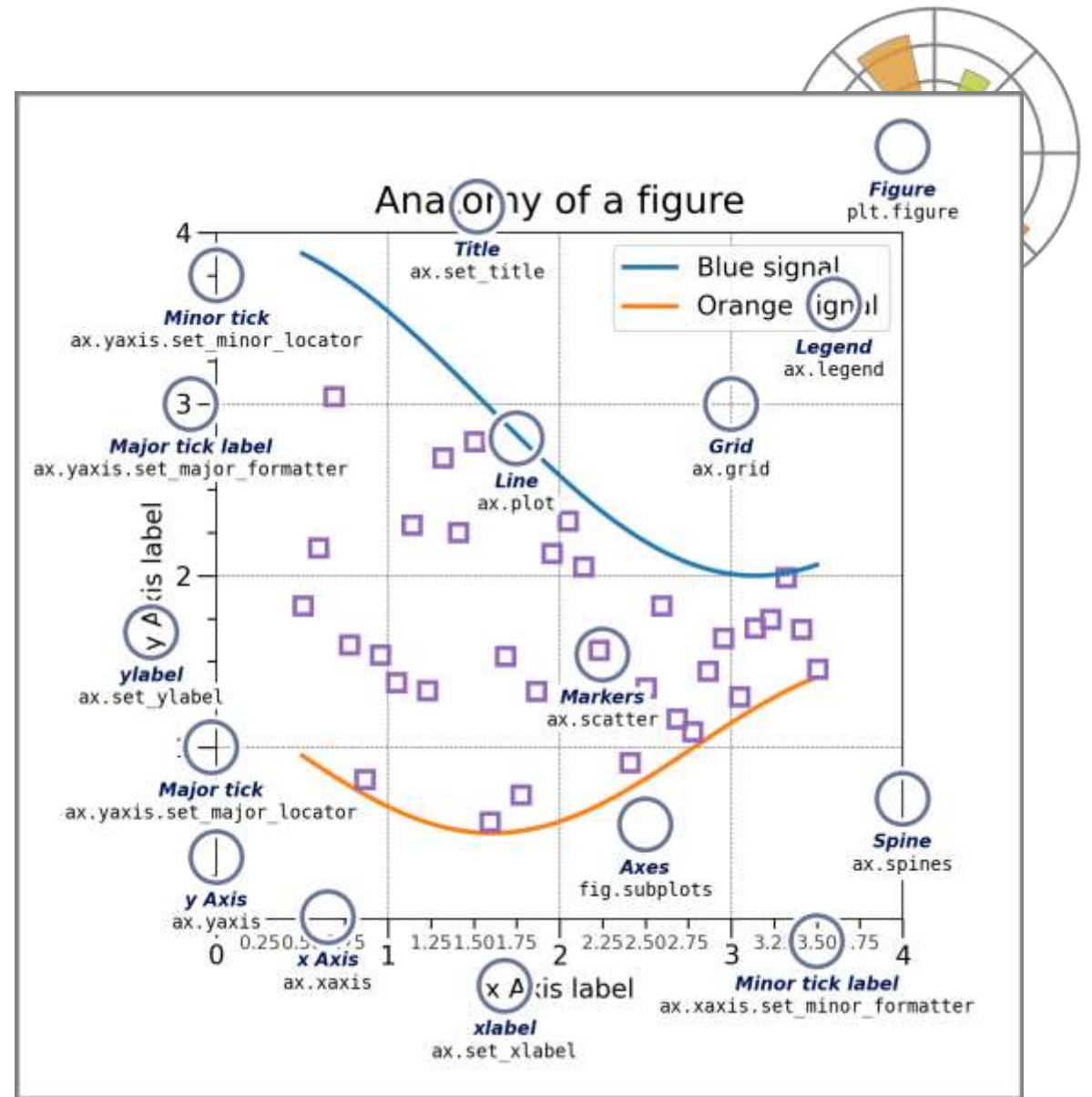


Matplotlib es una biblioteca open source de python que permite generar gráficos (estaticos, animados e interactivos) para la visualización de datos. El paquete `matplotlib` es una extensa biblioteca de funciones para generar gráficos 2D y 3D.



Partes de una Figure

- **Figure**, plano, región donde se encuentran los axes.
- **Axes**, gráficos.
- **Axis**, ejes del gráfico.
- **Artist**, todo lo visible de una Figure es un Artist. (Title, xlabel, Figure, Axes, Spine, etc)



Formas de usar Matplotlib



Hay dos formas principales de generar imágenes.

- Usando las funciones del módulo pyplot.

```
import matplotlib.pyplot as plt

# Generando imagen con funciones pyplot

x = np.linspace(0, 2, 100)

plt.figure(figsize=(5, 2.7), layout='constrained')
plt.plot(x, x, label='linear') # Plot datos en Axes imp
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label') # Agregar label del axes-x.
plt.ylabel('y label') # Agregar label del axes-y.
plt.title("Simple Plot") # Agregar titulo.
plt.legend() # Agregar leyenda.
```

- Creando explícitamente Figures y Axes (Estilo P00)

```
import matplotlib.pyplot as plt

# Generando imagen con estilo P00 (Creando Figure y Axes)

x = np.linspace(0, 2, 100)

# Crea el objeto Figure y el objeto Axes
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear') # Línea 1
ax.plot(x, x**2, label='quadratic') # Línea 2
ax.plot(x, x**3, label='cubic') # Línea 3
ax.set_xlabel('x label') # Agregar label del axes-x.
ax.set_ylabel('y label') # Agregar label del axes-y.
ax.set_title("Simple Plot") # Agregar titulo.
ax.legend() # Agregar leyenda.
```


Pyplot

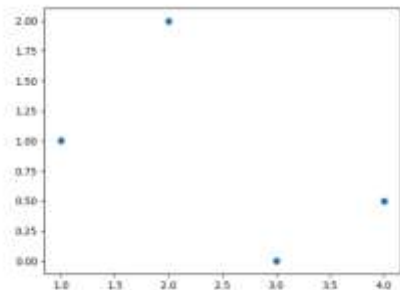


Dentro del paquete matplotlib destaca el módulo pyplot. Este módulo oculta muchas de las funcionalidades de bajo nivel de la biblioteca, permitiendo el uso de sencillas funciones para los elementos gráficos más habituales, tales como creación de figuras, trazado de líneas, visualización de imágenes, inserción de texto, etc.

Diagrama de puntos: `scatter(x, y)`:

Dibuja un diagrama de puntos con las coordenadas de la lista `x` en el eje X y las coordenadas de la lista `y` en el eje Y.

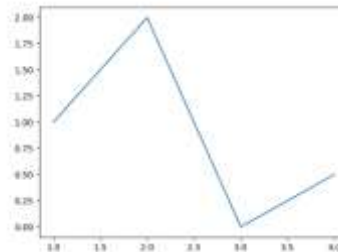
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.scatter([1, 2, 3, 4], [1, 2, 0, 0.5])
plt.show()
```



Diagramas de líneas

`plot(x, y)`: Dibuja un polígono con los vértices dados por las coordenadas de la lista `x` en el eje X y las coordenadas de la lista `y` en el eje Y.

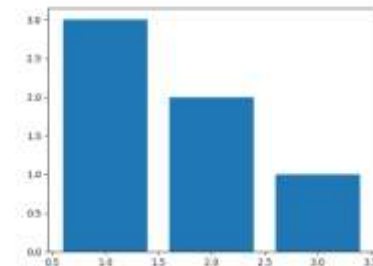
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 2, 0, 0.5])
plt.show()
```



Diagramas de barras verticales

`bar(x, y)`: Dibuja un diagrama de barras verticales donde `x` es una lista con la posición de las barras en el eje X, e `y` es una lista con la altura de las barras en el eje Y.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.bar([1, 2, 3], [3, 2, 1])
plt.show()
```



Pyplot



Ejemplos de Uso de Pyplot

Diagrama de dispersión o puntos: Los diagramas de dispersión son herramientas visuales poderosas para analizar la relación entre dos variables cuantitativas.

Gráfico de Líneas:

- **Tendencias en el tiempo:** Los gráficos de líneas son ideales para mostrar cómo cambian los datos a lo largo del tiempo. Se utilizan comúnmente para visualizar series temporales, como las fluctuaciones en los precios, el crecimiento de la población, el cambio de temperatura, etc.

.

- **Comparación de series temporales:** Pueden superponerse varias series en el mismo gráfico para comparar diferentes conjuntos de datos a lo largo del tiempo.

Gráfico de Barras:

- **Comparación de categorías:** Los gráficos de barras son útiles para comparar diferentes categorías entre sí. Pueden mostrar las diferencias en valores absolutos entre categorías distintas.

.

- **Distribución de frecuencias:** Pueden utilizarse para mostrar la frecuencia de diferentes valores o categorías.

Pyplot



Gráfico de Dispersión

Relación entre dos variables: Los gráficos de dispersión son ideales para mostrar la relación entre dos variables. Cada punto en el gráfico representa una observación.

- **Detección de patrones:** Pueden ayudar a identificar patrones, tendencias o correlaciones en los datos.
- **Detección de valores atípicos:** Pueden revelar valores atípicos que no siguen el patrón general de los datos.

Gráfico de Sectores (Pie Chart)

- **Proporción de partes de un todo:** Los gráficos de sectores se utilizan para mostrar la proporción de cada categoría en relación al total. Son ideales para representar distribuciones porcentuales.
- **Visualización de partes de un conjunto:** Pueden ayudar a entender cómo se divide un conjunto en partes y cuál es la contribución de cada parte.

Histograma

- **Distribución de datos:** Los histogramas son útiles para mostrar la distribución de un conjunto de datos continuos. Dividen el rango de datos en intervalos (bins) y cuentan el número de observaciones en cada intervalo.
- **Frecuencia de valores:** Ayudan a identificar la frecuencia de diferentes valores o rangos de valores en un conjunto de datos.
- **Forma de la distribución:** Pueden revelar la forma de la distribución, como si es normal, sesgada, bimodal, etc.

Personalización de gráficos.



Personalización de Gráficos

- **Añadir Leyendas**

Las leyendas se utilizan para describir cada serie de datos en el gráfico. Puedes añadir una leyenda usando `plt.legend()`.

- **Añadir Anotaciones**

Las anotaciones se utilizan para destacar puntos específicos en el gráfico. Puedes añadir anotaciones usando `plt.annotate()`.

- **Guardar Gráficos**

Puedes guardar los gráficos en diferentes formatos de archivo (como PNG, PDF, etc.) utilizando el método `savefig`.

**Veamos algunos ejemplos en
Google Colab...**

Ejemplos prácticos...

Referencias...



NumPy



pandas



Video Introducción Numpy: <https://youtu.be/Vpgwb8WNIqY>

Documentación oficial de Pandas: <https://pandas.pydata.org/docs/>

Tutorial de pandas de W3Schools: <https://www.w3schools.com/python/pandas/default.asp>

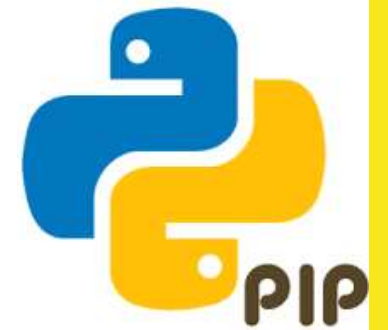
Python Charts: https://python-charts.com/es/matplotlib/textos/?utm_content=cmp-true

Tutorial de Python de W3Schools: <https://www.w3schools.com/python/>

Ciencia de datos: <https://aprendepython.es/pypi/datascience/>

Documentación oficial de Matplotlib: <https://matplotlib.org/stable/>

Muchas Gracias!!!



FACTORÍA F5 - GRUPO 6