



## UNIDAD 7

---

# Ejemplos de aplicaciones de Python con módulos.



# NumPy

Es una biblioteca de Python, muy popular para aplicaciones informáticas científicas, y es un acrónimo de “Numerical Python”.

<https://numpy.org/>

Las operaciones de NumPy se dividen en tres categorías principales:

- Transformada de Fourier y manipulación de formas.
  - Operaciones matemáticas y lógicas.
  - Álgebra lineal y generación de números aleatorios.
- 
- NumPy es extremadamente rápido en comparación con el núcleo de Python gracias a su uso intensivo de extensiones C.
  - Muchas bibliotecas avanzadas de Python, como Scikit-Learn, Scipy y Keras, hacen un uso extensivo de la biblioteca NumPy.
  - NumPy viene con una variedad de funcionalidades integradas, que en el núcleo de Python requerirían un poco de código personalizado.

```
import numpy as np  
nums = np.array([2, 3, 4, 5, 6])  
nums2 = nums + 2
```



## Instalación

Para instalar el paquete NumPy, puede usar el instalador pip:

```
$ pip install numpy
```

Si se está ejecutando Python a través de la distribución Anaconda:

```
$ conda install numpy
```

## Algunas funcionalidades

Cargar datos de un archivo.txt con delimitador de coma.

```
1 | archivoDatos=np.loadtxt('C:\ruta\archivoDatos.txt', delimiter = ',')
```



## Arreglos

### Arrays con Numpy

- **ndarray.ndim** -> Proporciona el número de dimensiones de nuestro array.
- **ndarray.dtype** -> Es un objeto que describe el tipo de elementos del array.
- **ndarray.shape** -> Devuelve la dimensión del array, es decir, una tupla de enteros indicando el tamaño del array en cada dimensión. Para una matriz de n filas y m columnas obtendremos (n,m).
- **ndarray.data** -> El buffer contiene los elementos actuales del array.
- **ndarray.itemsize** -> devuelve el tamaño del array en bytes.
- **ndarray.size** -> Es el número total de elementos del array.

```
1 import numpy as np # Importamos numpy como el alias np
2 miArray = np.arange(10) # Creamos un array de 0 a 9 separados de uno en uno
3 print(type(miArray))
4 numdim= miArray.ndim
5 dim=miArray.shape
6 tam= miArray.size
7 byte=miArray.itemsize
```



**identity(n,dtype)** →Devuelve la matriz identidad, es decir, una matriz cuadrada nula excepto en su diagonal principal que es unitaria. n es el número de filas (y columnas) que tendrá la matriz y dtype es el tipo de dato. Este argumento es opcional. Si no se establece, se toma por defecto como flotante.

**ones(shape,dtype)** →Crea un array de 1 compuesto de shape elementos.

**zeros(shape, dtype)** →Crea un array de 0 compuesto de “shape” elementos”.

**linspace(start,stop,num,endpoint=True,retstep=False)** →Crea un array con valor inicial start, valor final stop y num elementos.

**empty(shape, dtype)** →Crea un array de ceros compuesto de “shape” elementos” sin entradas.

**meshgrid(x,y)** →Genera una malla a partir de dos los arrays x, y.

**eye(N, M, k, dtype)** →Crea un array bidimensional con unos en la diagonal k y ceros en el resto. Es similar a identity. Todos los argumentos son opcionales. N es el número de filas, M el de columnas y k es el índice de la diagonal. Cuando k=0 nos referimos a la diagonal principal y por tanto eye es similar a identity.

**arange([start,]stop[,step,],dtype=None)** →Crea un array con valores distanciados step entre el valor inicial star y el valor final stop. Si no se establece step python establecerá uno por defecto.

```
1
2 import numpy as np # Importamos numpy como el alias np
3 g=np.zeros( (3,4) )
4 print(g)
5 k=np.linspace( 1, 4, 9 )
6 print(k)
7 X,Y=np.meshgrid([1,2,3],[7,9,34])
8 print(X)
   print(Y)
```

## Matrices con Numpy

Suma: se realiza elemento a elemento ya sea arrays o matrices.

```
1
2 import numpy as np # Importamos numpy como el alias np
3 a = np.array([[8, 2], [8, 4]])
4 b=a+a
5 print(b)
6 c=a*b
   print(c)
```

Traspuestas, conjugado, inversa, cálculos de determinantes ecuaciones lineales, autovalores

```
1
2 import numpy as np # Importamos numpy como el alias np
3 g=np.matrix( [[3,4,-9], [7,4,-5] ,[1,3,9]] )
4 print(g)
5 b=np.matrix( [[-9], [-5] ,[9]] )
6 print(b)
7 c=g*b
8 print(c)
9 bt=b.T #traspuestas
10 print(bt)
11 bh=b.H #traspuestas y conjugada
12 print(bh)
13 gi=g.I #inversa
14 print(gi)
15 detgi=np.linalg.det(gi) #calculo del determinante
   tragi=np.trace(gi) #calculo de la traza
```

## Producto matricial y producto elemento a elemento:

```
1
2 import numpy as np # Importamos numpy como el alias np
3 a = np.array([[8, 2], [8, 4]])
4 b=np.dot(a,a)
5 print(b)
6 c=a*a
7 print(c)
8 d=np.multiply(a, a)
   print(d)
```

## Producto vectorial y producto exterior:

```
1
2 import numpy as np # Importamos numpy como el alias np
3 a = np.array([[8, 1, 4]])
4 b= np.array([[3, 7, 4]])
5 c= np.cross(a, b) # Producto vectorial
6 print(c)
7 d=np.outer(a, b) # Producto exterior
   print(d)
```

## Funciones trigonométricas y la transformada de Fourier:

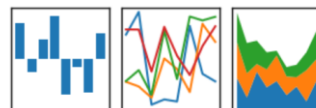
```
1
2 import numpy as np # Importamos numpy como el alias np
3 x=np.linspace(0,1,100)
4 y=np.sin(x)
   print (np.fft.fft(y))
```



## Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.

### Principales características:

- Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Realiza todas estas operaciones de manera muy eficiente.





## Estructuras de datos

- Series: Estructura de una dimensión.
- DataFrame: Estructura de dos dimensiones (tablas).
- Panel: Estructura de tres dimensiones (cubos).

Estas estructuras se construyen a partir de arrays de la librería NumPy, añadiendo nuevas funcionalidades.

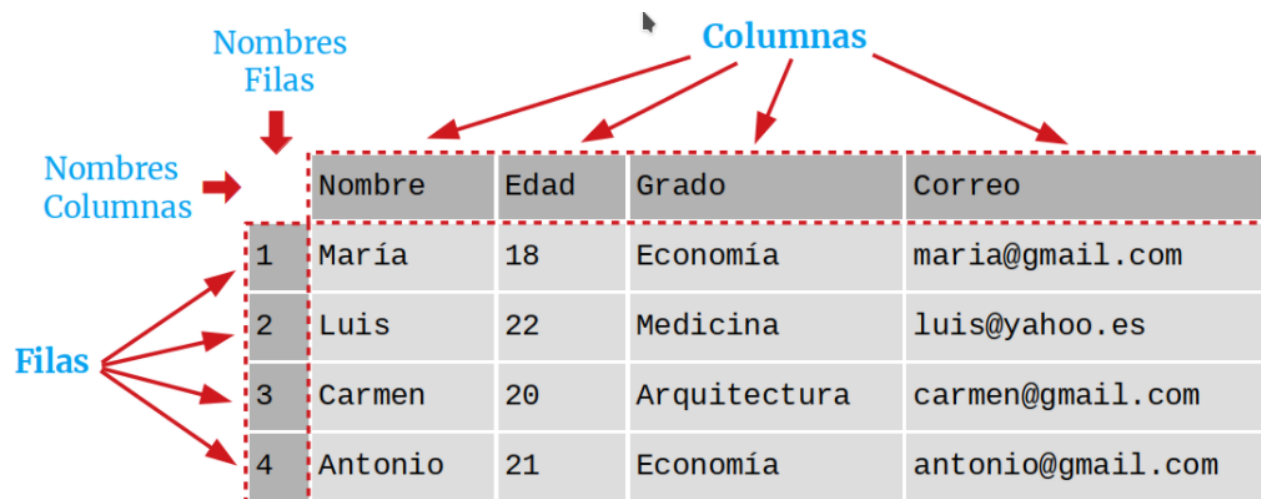
### Manejo de series – dataframes - panels.

Series(data=lista, index=indices, dtype=tipo) : Devuelve un objeto de tipo Series con los datos de la lista lista, las filas especificados en la lista indices y el tipo de datos indicado en tipo.

Índice →	A1	A2	A3	A4
Valores →	Matemáticas	Economía	Programación	Inglés

```
>>> import pandas as pd
>>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})
>>> print(s)
Matemáticas    6.0
Economía       4.5
Programación   8.5
dtype: float64
```

Un DataFrame contiene dos índices, uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los nombres de las filas y las columnas.



The diagram illustrates the structure of a DataFrame. It shows a table with 4 rows and 4 columns. The columns are labeled 'Nombre', 'Edad', 'Grado', and 'Correo'. The rows are indexed 1, 2, 3, and 4. Red arrows point from the labels to the corresponding parts of the table: 'Nombres Filas' points to the row indices, 'Nombres Columnas' points to the column headers, 'Filas' points to the entire row set, and 'Columnas' points to the entire column set.

	Nombre	Edad	Grado	Correo
1	María	18	Economía	maria@gmail.com
2	Luis	22	Medicina	luis@yahoo.es
3	Carmen	20	Arquitectura	carmen@gmail.com
4	Antonio	21	Economía	antonio@gmail.com

```
>>> import pandas as pd
>>> datos = {'nombre': ['María', 'Luis', 'Carmen', 'Antonio'],
... 'edad': [18, 22, 20, 21],
... 'grado': ['Economía', 'Medicina', 'Arquitectura', 'Economía'],
... 'correo': ['maria@gmail.com', 'luis@yahoo.es', 'carmen@gmail.com', 'antonio@gmail.com'],
... }
>>> df = pd.DataFrame(datos)
>>> print(df)
   nombre  edad  grado  correo
0  María   18   Economía maria@gmail.com
1   Luis   22   Medicina luis@yahoo.es
2  Carmen   20  Arquitectura carmen@gmail.com
3 Antonio   21   Economía antonio@gmail.com
```



## Matplotlib



Es una librería de Python especializada en la creación de gráficos en dos dimensiones.

Permite crear y personalizar los tipos de gráficos más comunes, entre ellos:

- Diagramas de barras
- Histograma
- Diagramas de sectores
- Diagramas de caja y bigotes
- Diagramas de violín
- Diagramas de dispersión o puntos
- Diagramas de líneas
- Diagramas de áreas
- Diagramas de contorno
- Mapas de color

y combinaciones de todos ellos.

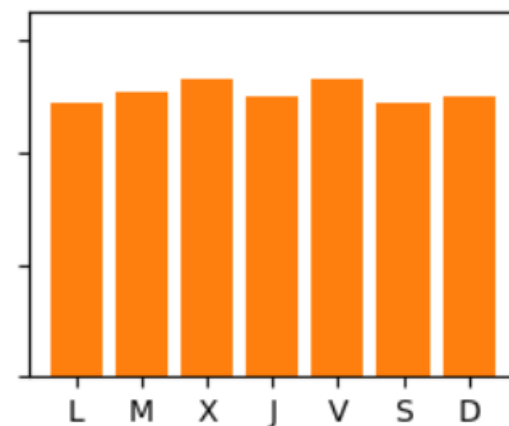
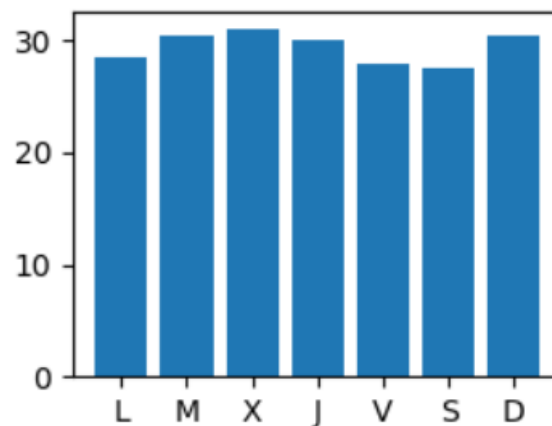
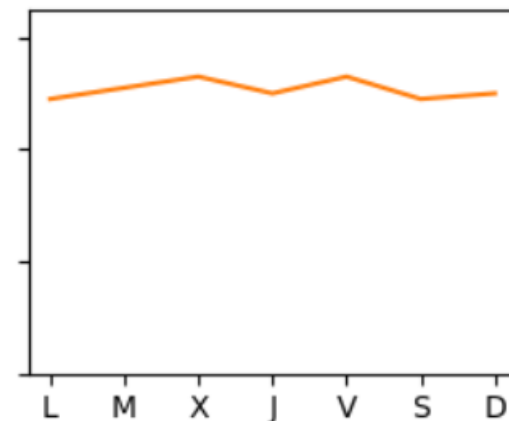
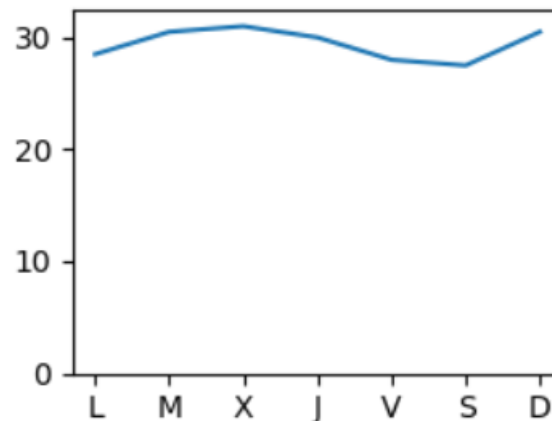
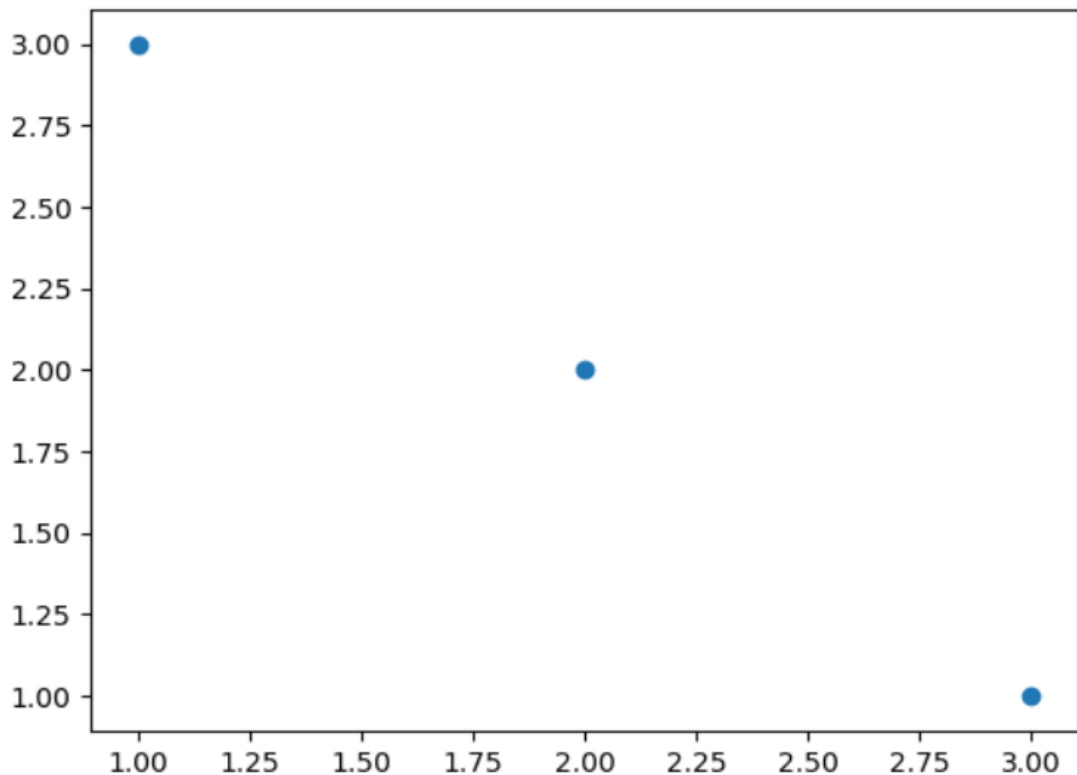


Para crear un gráfico con matplotlib es habitual seguir los siguientes pasos:

- Importar el módulo pyplot.
- Definir la figura que contendrá el gráfico, que es la region (ventana o página) donde se dibujará y los ejes sobre los que se dibujarán los datos. Para ello se utiliza la función `subplots()`.
- Dibujar los datos sobre los ejes. Para ello se utilizan distintas funciones dependiendo del tipo de gráfico que se quiera.
- Personalizar el gráfico. Para ello existen multitud de funciones que permiten añadir un título, una leyenda, una rejilla, cambiar colores o personalizar los ejes.
- Guardar el gráfico. Para ello se utiliza la función `savefig()`.
- Mostrar el gráfico. Para ello se utiliza la función `show()`.

## Unidad 7: Ejemplos de aplicaciones de Python con módulos.

```
# Importar el módulo pyplot con el alias plt
import matplotlib.pyplot as plt
# Crear la figura y los ejes
fig, ax = plt.subplots()
# Dibujar puntos
ax.scatter(x = [1, 2, 3], y = [3, 2, 1])
# Guardar el gráfico en formato png
plt.savefig('diagrama-dispersion.png')
# Mostrar el gráfico
plt.show()
```





## API REST

Application Programming Interface o Interfaz de Programación de Aplicaciones.

Agrupar al conjunto de reglas, códigos y especificaciones que las aplicaciones deben seguir para que otro software las consuma y se comuniquen entre ellos.

REST (Representational State Transfer o Transferencia de Estado Representacional)

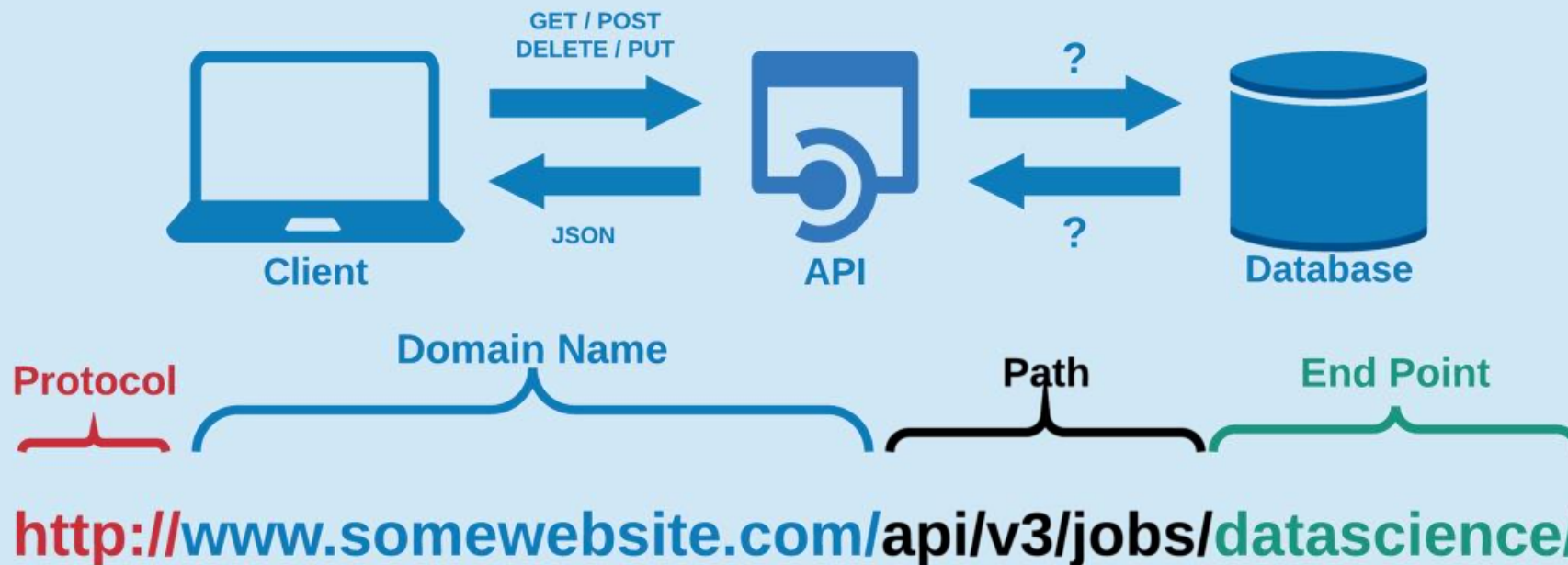
Es un estilo arquitectónico y enfoque de comunicaciones utilizado en el desarrollo de servicios web

REST es cualquier interfaz entre sistemas que use HTTP para el transporte de datos haciendo uso también de los métodos de este protocolo para comunicarse (GET, POST, PUT, PATCH y DELETE). Permite trabajar con múltiples formatos de datos como XML, JSON, datos binarios y texto plano. Esto hace tomar ventaja a este enfoque, pues en el caso de SOAP (Simple Object Access Protocol) solo admite el formato XML.

Se considera a una API de tipo REST cuando su arquitectura se ajusta a las reglas y restricciones de esta interfaz.

En el campo de las APIs, REST se ha vuelto el estándar más lógico, flexible, eficiente y habitual.

# Develop an API using Flask and Python3



Flask permite desarrollar aplicaciones web con el lenguaje Python, de forma sencilla.



ESCUELA  
POLITÉCNICA  
NACIONAL



ESCUELA  
POLITÉCNICA  
NACIONAL



python™