

ESCUELA DE TALENTO

DIGITAL

- 100% ONLINE ■ MENTORIZACIÓN PERMANENTE
- ORIENTADO A LA EMPLEABILIDAD ■ GRATUITO
- CONEXIÓN CON EL MERCADO

NTT DATA FOUNDATION

ESCUELA DE TALENTO DIGITAL

NTT DATA FOUNDATION

LIBRERÍAS

ÍNDICE

1. ¿QUÉ SON?	3
2. LIBRERÍAS PREINSTALADAS	3
3. REPOSITORIOS.....	4
3.1. PIP	4
3.2. CONDA-FORGE.....	6
4. LIBRERÍAS BÁSICAS.....	6
5. ENTORNOS VIRTUALES	7
5.1. ¿Qué son?.....	7
5.2. Creación	8
5.3. Activación	8
5.4. Desactivación	9
5.5. Listar	9

1. ¿QUÉ SON?

Las **librerías** son conjuntos de módulos que nos permiten utilizar **funcionalidades ya programadas** para facilitarnos cualquier tipo de programación. Estos conjuntos de módulos no son más que archivos de código que ya tienen definidas funciones, clases y variables que podemos utilizar desde otros programas que estemos realizando.

Python tiene dos tipos de librerías, las que ya están preinstaladas y las que deberán ser instaladas por el usuario, en función de sus necesidades, que se denominan repositorios. Vamos a ver ambos tipos en detalle.

2. LIBRERÍAS PREINSTALADAS

Para poder utilizar una librería ya preinstalada, tendremos que importarla en nuestro fichero de Python. Esto se hará de la siguiente manera:

```
import nombre de la librería
```

Por ejemplo, si queremos importar la librería `math` en un programa que estemos creando con Python, es tan fácil como incluir al principio del mismo la sentencia:

```
import math
```

Una vez que lo hemos importado podremos utilizar los métodos, funciones, o variables que contiene utilizando el formato:

```
Nombre de la librería.nombre del elemento
```

Por ejemplo, si queremos utilizar la función `sqrt` que está dentro de la librería `math` y que calcula la raíz cuadrada de un número que le indiquemos como parámetro, lo haríamos así:

```
math.sqrt (4)
```

Lo que devolvería el número `2.0`.

En este link podrás consultar todas las librerías preinstaladas en Python, así como todos los métodos, funciones, variables, etc. que tiene cada una: <https://docs.python.org/es/3/library/index.html>

Cuando trabajamos con librerías, es posible importar solo un módulo, una función o una variable, sin ser necesario importar la librería completa. De hecho, en ciertas librerías es necesario hacerlo así. La estructura para hacerlo es la siguiente:

```
from nombre de la librería import nombre de la función, módulo o variable
```

Por ejemplo, si para calcular la raíz cuadrada de un número no queremos importar la librería `math` al completo, y solo importamos la función `sqrt`, lo haríamos así:

```
from math import sqrt
```

Pero para ejecutarlo lo utilizaríamos igual:

```
print (math.sqrt (4))
```

que mostraría por pantalla 2.0.

Cuando los nombres de las librerías son muy largos, nos cuesta recordarlos, o preferimos trabajar con un nombre que nos sea más cómodo, podemos darles otro nombre en nuestro programa, es decir, podemos darles un alias que utilizaremos mientras estemos trabajando en nuestro código. Para ello se utiliza la estructura:

```
import nombre de la librería as nombre que queremos utilizar
```

Por ejemplo, la librería `statistics` contiene funciones estadísticas matemáticas, pero si nos resulta más cómodo llamarla `estadistica`, porque nos resulte más fácil escribir esta palabra en castellano, podremos renombrarla para usar este segundo nombre en nuestro programa. Entonces, si queremos calcular la media aritmética de una lista de números, podemos usar la función `mean ()` de la librería `statistics`, y hacerlo así:

```
import statistics as estadistica

lista = [2, 4, 6, 8]

media = estadistica.mean(lista)

print (media)
```

Lo que nos mostrará por pantalla el valor 5.

3. REPOSITARIOS

Los repositorios son archivos en los que se almacenan recursos digitales que nosotros podemos utilizar. Son, básicamente, “almacenes” de código reutilizable y librerías. Como no están preinstalados hay que instalarnos cuando los necesitemos y desinstalarlos cuando no los vayamos a utilizar.

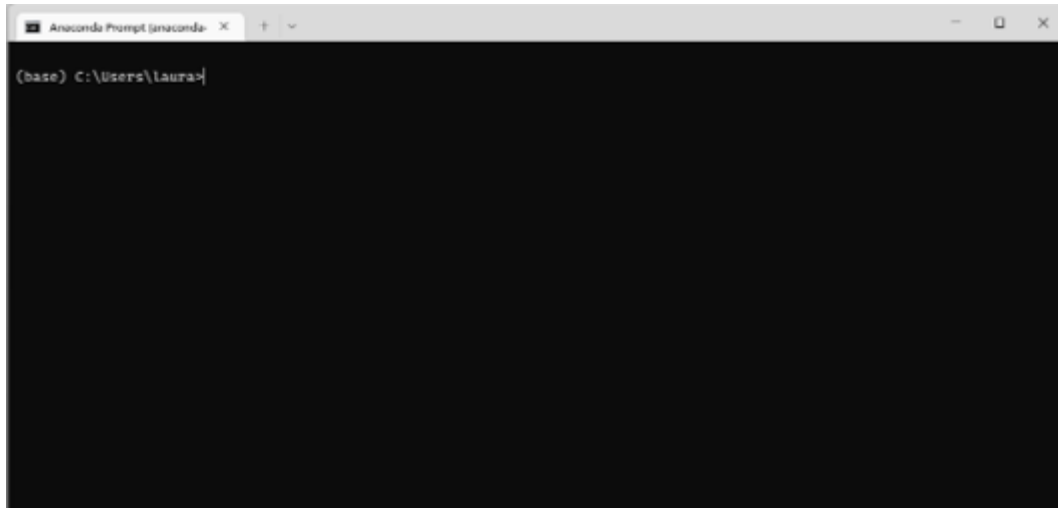
Podemos trabajar con repositorios de distintas maneras. Vamos a ver dos de ellas.

3.1. PIP

Python facilita la instalación de librerías no preinstaladas, localizadas en repositorios, a través del administrador de paquetes PIP, que contiene paquetes de software programados en dicho lenguaje.

Este administrador de paquetes permite instalar, desinstalar y actualizar librerías de forma sencilla. Además, desde Anaconda, utilizando Anaconda Prompt, podremos hacerlo directamente. Veamos cómo.

Busca directamente en la barra de búsqueda de tu ordenador Anaconda Prompt, y haz clic sobre el icono que aparece. Esto te abrirá una ventana de navegación como esta:



Para instalar una librería desde PIP utilizaremos la siguiente estructura:

```
pip install nombre de la librería a instalar
```

Si queremos instalar, por ejemplo, la librería `tensorflow`, que es una biblioteca de software gratuita especializada, principalmente, en el entrenamiento de redes neuronales, desde Anaconda Prompt ejecutaremos la sentencia:

```
pip install tensorflow
```

Esto nos descargará todos los paquetes de esa librería para que podamos trabajar con ellos.

Desde Jupyter Notebook también podemos instalar cualquiera de las librerías de PIP, lo único que, antes de la sentencia, tenemos que incluir el símbolo de porcentaje, `%`. Es decir, en una de las líneas de nuestro notebook pondremos:

```
%pip install tensorflow
```

Y la librería se instalará igual que lo hace desde Anaconda Prompt.

Una vez finalice el proceso de instalación, podremos comprobar que todo ha ido bien, ejecutando, desde Anaconda Prompt, la sentencia:

```
pip show tensorflow
```

Que nos mostrará un resumen de los datos principales de la librería, como la versión instalada, el autor, la licencia, etc.

También lo podemos consultar desde Jupyter Notebook, al igual que hemos hecho con la instalación, utilizando `%pip show tensorflow`.

Cuando ya no necesitemos una librería podremos desinstalarla ejecutando, en Anaconda Prompt, la sentencia:

```
pip uninstall tensorflow
```

Esto nos preguntará si estamos seguros con la frase:

Proceed (Y/n)?

Simplemente escribiendo **Y** y pulsando Enter, comenzará la desinstalación.

Estas instrucciones funcionan con todas las librerías de PIP.

Puedes ampliar la información sobre PIP en este enlace a su página oficial:

<https://pip.pypa.io/en/stable/getting-started/>

3.2. CONDA-FORGE

Conda-forge es una organización propiedad de Anaconda dedicada al mantenimiento y la distribución del gestor de paquetes conda.

La instalación desde conda no es tan sencilla como desde PIP, ya que puede haber varias opciones o diferentes parámetros a introducir en el comando, en función del elemento que queramos instalar. Con este repositorio es recomendable visitar la web oficial de conda-forge en Anaconda, o incluso la web del paquete que vamos a instalar, y utilizar los comandos recomendados.

La web oficial con el listado de paquetes que hay en conda-forge es esta:

<https://anaconda.org/conda-forge/>

Para trabajar con estos paquetes es recomendable usar siempre Anaconda Prompt.

4. LIBRERÍAS BÁSICAS

Para trabajar en el ámbito de la ciencia de datos hay una gran variedad de librerías y recursos, pero hay algunas que son esenciales para el análisis de datos. Son las siguientes:



Pandas. Es una librería para trabajar con conjuntos de datos en forma de tablas. Te permitirá hacer consultas y filtrar datos, entre otras operaciones relacionadas con el análisis. Su página oficial es <https://pandas.pydata.org/>



NumPy. Es una librería que proporciona soporte para trabajar con grandes series y matrices de números. A su vez, contiene una gran cantidad de funciones matemáticas para trabajar con dichos elementos. Su página oficial es <https://numpy.org/>



SciPy. Es una librería que proporciona una gran variedad de funciones matemáticas y científicas. Contiene algoritmos más avanzados y especializados que NumPy. Su página oficial es <https://scipy.org/>



Scikit-learn. Es una librería especializada en machine learning que ofrece herramientas para evaluar y optimizar modelos. Su página oficial es <https://scikit-learn.org/stable/>



Matplotlib. Es una librería que se utiliza para crear gráficos y visualizaciones de datos en forma de líneas, barras, gráficos de dispersión, etc. Es muy flexible y permite crear gráficos de alta calidad personalizados. Su página oficial es <https://matplotlib.org/>



NLTK. Es una librería que proporciona herramientas y recursos para el procesamiento del lenguaje natural. Con ella podremos reconocer la gramática, extraer información, analizar el significado o el sentimiento. Su página oficial es <https://www.nltk.org/>

5. ENTORNOS VIRTUALES

5.1. ¿Qué son?

Los entornos virtuales son entornos aislados en los que se pueden instalar paquetes de Python sin afectar a la configuración del sistema ni a otros entornos virtuales.

Son necesarios porque, en ocasiones, necesitamos aislar las librerías que utilizamos en proyectos diferentes, de modo que cada proyecto tenga sus librerías en las versiones que necesita.

Hay veces que las librerías, entre ellas, pueden causar conflictos, sobre todo dependiendo de la versión que tengamos instalada, ya que hay versiones de algunas librerías que no funcionan con algunas versiones de otras librerías. Utilizar entornos virtuales evita este tipo de problemas.

Se puede trabajar con entornos virtuales desde la línea de comandos de Anaconda Prompt, con conda o con Python, o directamente desde la línea de comandos de Windows, así que, veamos cómo gestionarlos.

5.2. Creación

Podemos crear un entorno virtual desde Anaconda Prompt con el comando:

```
conda create --name nombre del entorno a crear
```

Por ejemplo, si queremos crear el entorno virtual fundamentos, desde la línea de comandos de Anaconda Prompt ejecutaremos

```
conda create --name fundamentos
```

Entonces nos preguntará:

```
Proceed ([y]/n)?
```

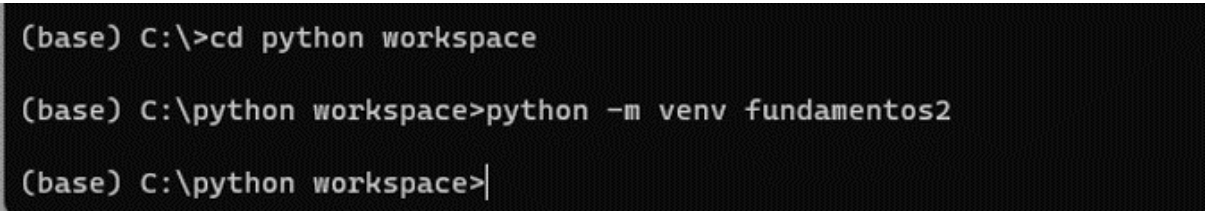
Si contestamos con **y**, procederá a crearlo.

También lo podemos crear desde Python, sin necesidad de Anaconda, utilizando Anaconda Prompt o la línea de comandos de Windows.

Para crearlo desde Python tenemos que situarnos en la carpeta que queramos utilizar y ejecutar

```
python -m venv nombre del entorno
```

Por ejemplo, si creamos en la raíz del disco duro de nuestro ordenador una carpeta llamada `python workspace`, accederemos a ella desde la línea de comandos y ejecutaremos la orden. Para acceder a la carpeta, si la hemos creado en la raíz (C:), utilizaremos `cd python workspace`, y tendremos algo como esto:



```
(base) C:\>cd python workspace  
(base) C:\python workspace>python -m venv fundamentos2  
(base) C:\python workspace>|
```

Que nos habrá creado el entorno virtual fundamentos2.

5.3. Activación

Para activar el entorno, en Anaconda, utilizaremos la siguiente estructura:

```
conda activate nombre del entorno
```

En nuestro caso:

```
conda activate fundamentos
```

Para activarlo en Python, tendremos que acceder a la carpeta `Scripts` que nos ha creado al crear el entorno, y una vez ahí, lanzar el comando `activate`.

Tendríamos entonces algo como esto:

```
(base) C:\python workspace>cd fundamentos2
(base) C:\python workspace\fundamentos2>cd scripts
(base) C:\python workspace\fundamentos2\Scripts>activate
(fundamentos2) (base) C:\python workspace\fundamentos2\Scripts>
```

Con `cd` hemos accedido primero a `fundamentos2`, y después a la carpeta `Scripts`, desde donde hemos activado el entorno.

5.4. Desactivación

Para desactivar el entorno, desde Anaconda, utilizaremos la siguiente estructura:

```
conda deactivate
```

Este comando desactiva el entorno actual y vuelve al entorno que hay por defecto.

Desde Python, se procede de la misma forma que para activarlo, se accede a la carpeta `Scripts` y desde ahí se utiliza el comando `deactivate`.

```
(fundamentos2) (base) C:\python workspace\fundamentos2\Scripts>deactivate
(base) C:\python workspace\fundamentos2\Scripts>
```

5.5. Listar

Si queremos saber cuáles son los entornos que tenemos en Anaconda en un momento dado, podemos listarlos con el comando:

```
conda env list
```