HERRAMIENTAS DE TRABAJO

MÁSTER EN INTELIGENCIA ARTIFICIAL

2021

¿QUÉ ES ANACONDA?

Para ejecutarse, muchos paquetes científicos dependen de versiones específicas de otros paquetes. Los científicos de datos a menudo usan múltiples versiones de muchos paquetes y, a su vez, utilizan múltiples entornos para separar estas diferentes versiones. Esto ayuda a los científicos de datos a garantizar que cada versión de cada paquete tenga todas las dependencias que requiere y funcione correctamente.

Anaconda es un **administrador de paquetes**, un **administrador de entornos**, una **distribución de data/science de Python/R** y una **colección de más de 7500 paquetes** de código abierto. La lista de paquetes disponibles en la última versión de Anaconda se puede consultar en el siguiente enlace: Anaconda package lists — Anaconda documentation. Algunos de estos paquetes ya vienen instalados, mientras que otros no vienen instalados pero están disponibles para su instalación con un sencillo comando que veremos más adelante.

Más allá de esta colección de paquetes de código abierto, pueden instalarse más de 1,5K paquetes del repositorio público de Anaconda y más de 20k paquetes de canales (repositorios de paquetes, sitios privados de nuestro ordenador o lugares públicos de la web donde se encuentran los paquetes disponibles para ser descargados y/o instalados) comunitarios, como Conda-forge y bioconda.

Aunque a veces se confunden ambos términos, en programación un **módulo** se define simplemente como una porción de un programa, mientras que una **librería** (también llamada biblioteca, del inglés "library") se define como un conjunto de implementaciones funcionales. Es decir, una librería **no** es un programa que se ejecute de manera autónoma, es código que lleva a cabo ciertas funcionalidades muy concretas para simplificar tareas complejas con el fin de servir a otros programas que la invoquen.

¿QUÉ ES MINICONDA?

Miniconda es una **alternativa a Anaconda mucho más ligera**. Básicamente, Miniconda es Conda junto a sus dependencias. Es un instalador mucho más pequeño, que generalmente se usa con una conexión a Internet activa. Anaconda requiere 3 GB de espacio libre en el disco duro, mientras que Miniconda solo necesita 400 MB.

Comandos de conda vs Anaconda Navigator

Anaconda Individual Edition contiene *conda* (sistema de gestión de paquetes y de entornos virtuales) y *Anaconda Navigator*, además de Python y cientos de paquetes científicos. Al instalar Anaconda, estamos instalando todo esto.

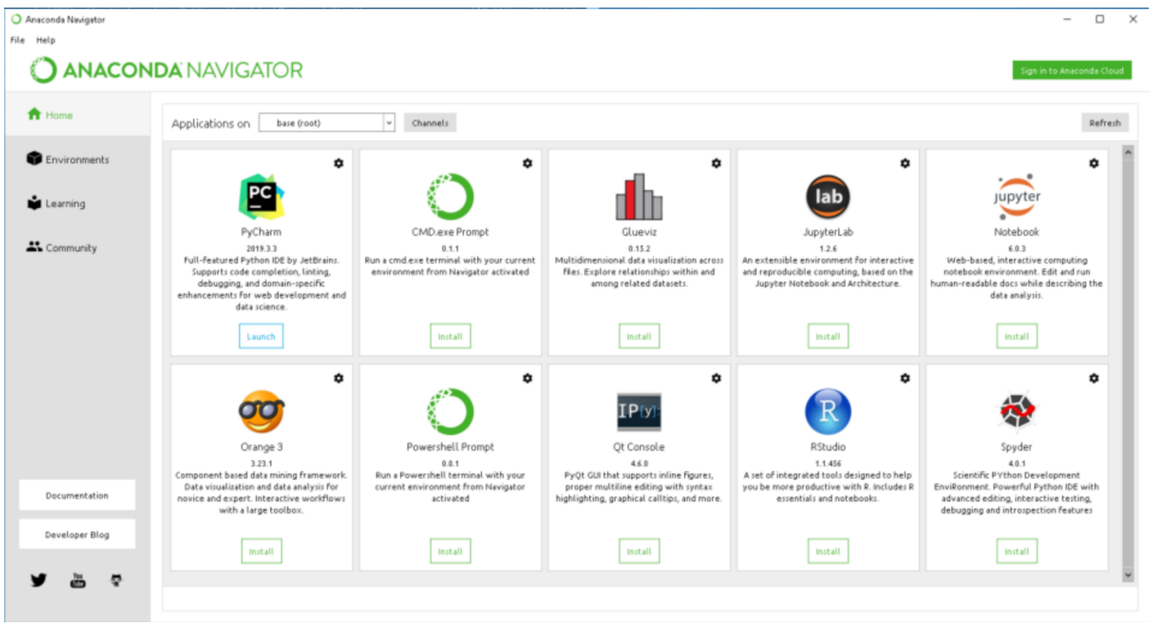
Conda se utiliza mediante comandos en la interfaz de línea de comandos "Anaconda Prompt" en Windows y en la terminal de macOs y Linux. Sin embargo, **Anaconda Navigator se incluye como una alternativa para poder usar conda sin comandos** de línea de comandos, permitiendo iniciar aplicaciones y administrar paquetes, entornos y canales de conda desde una sencilla interfaz gráfica de usuario (GUI).

Utilizar los comandos de conda o la sencilla interfaz de Anaconda Navigator es equivalente. La elección de una opción u otra depende de lo familiarizados que estemos con el uso de comandos. Siempre podemos cambiar de una a otra cuando queramos, ya que el trabajo que realizamos con una puede verse también con la otra.

Anaconda Navigator

Anaconda Navigator es una interfaz gráfica de usuario (GUI) de escritorio incluida en la distribución Anaconda que nos permite iniciar aplicaciones como Jupyter Notebook o Spyder, además de administrar fácilmente paquetes, entornos y canales de conda, todo ello sin usar comandos de línea de comandos.

Es decir, Navigator es una manera fácil de apuntar y hacer clic para trabajar con paquetes y entornos sin necesidad de escribir comandos conda en una ventana de terminal. Puede utilizarse para encontrar paquetes, instalarlos en un entorno, ejecutarlos y actualizarlos, todo dentro de Navigator.



Iniciar Navigator

Anaconda Navigator puede iniciarse de varias maneras:

- Windows
  Haciendo clic en el icono de inicio o bien abriendo la terminal de Anaconda (Anaconda prompt) y escribiendo el comando `anaconda-navigator`.

- Mac OS
  Abriendo Launchpad y después haciendo clic en el icono de Anaconda Navigator, o bien abriendo Launchpad, haciendo clic en el icono de terminal y, una vez abierta la terminal, escribiendo el comando `anaconda-navigator`.

- Linux
  Abriendo una terminal y escribiendo `anaconda-navigator`.

Por defecto, Anaconda Navigator incluye las siguientes aplicaciones:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

Las aplicaciones se instalan en el entorno activo. Para instalar una aplicación en un entorno específico, primero hay que seleccionar el entorno en la lista y después hacer clic en el botón “Instalar” de la aplicación. También se puede crear un nuevo entorno en la pestaña Entornos y después volver a la pestaña Inicio para instalar paquetes en el nuevo entorno.

Iniciar Jupyter

La aplicación Jupyter también puede iniciarse de varias maneras:

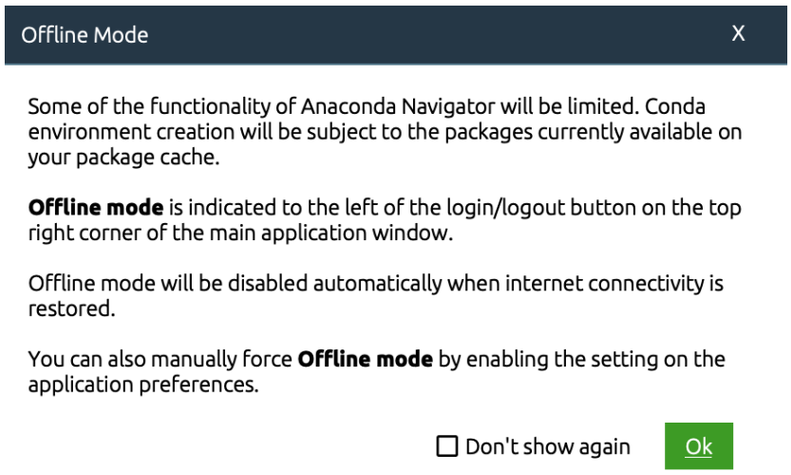
- Desde Anaconda Navigator, haciendo clic en el icono de Jupyter.
- Desde una terminal (terminal de Anaconda en Windows, o terminal habitual en Linux o macOS), escribiendo el comando `jupyter-notebook`.

Modos online y offline

Normalmente, Navigator se usa **en línea (modo online, con conexión a Internet)** para poder descargar e instalar paquetes. Navigator debe poder acceder a estos sitios, por lo que puede que sea necesario incluirlos en la lista blanca en la configuración del firewall de nuestra red:

- https://repo.anaconda.com
- https://conda.anaconda.org para conda-forge y otros canales en Anaconda Cloud (anaconda.org)
- google-public-dns-a.google.com (8.8.8.8:53) para verificar la conectividad a internet con Google Public DNS

Si Navigator detecta que el acceso a Internet no está disponible, habilita automáticamente el **modo fuera de línea (modo offline, sin conexión a Internet)** y muestra este mensaje:



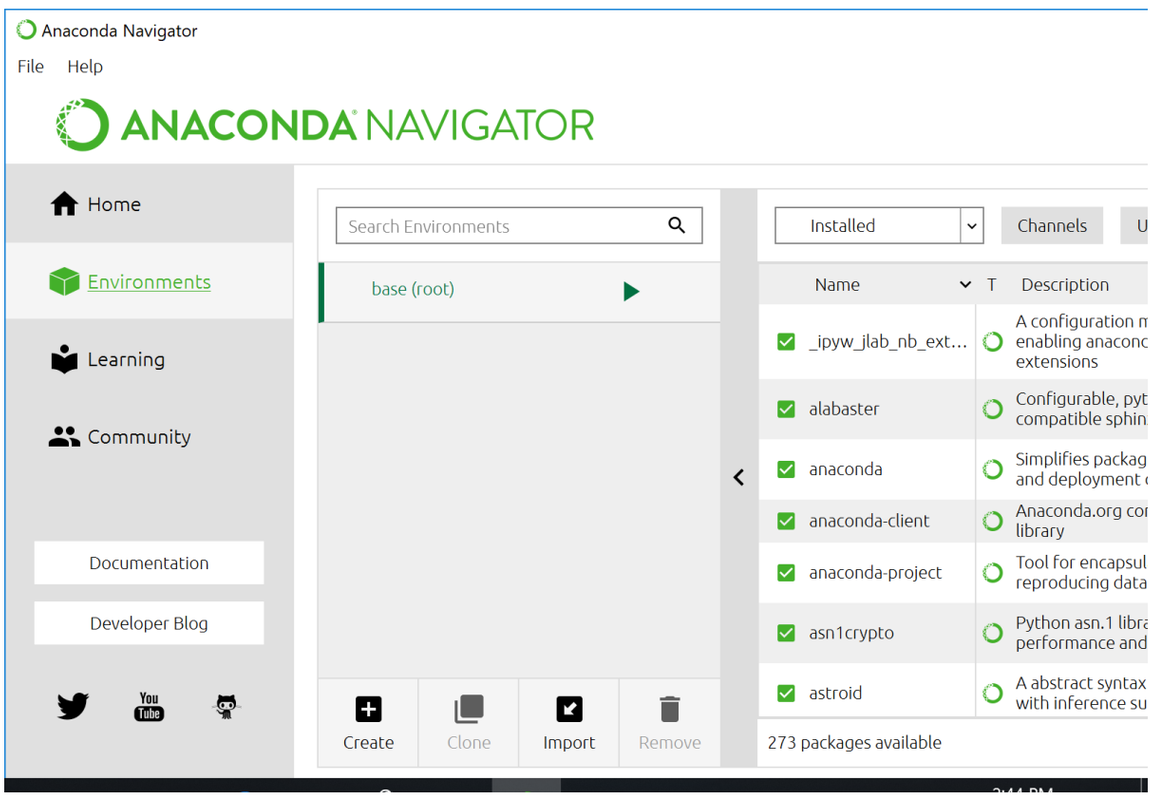
Además, si queremos utilizar el modo offline aunque el acceso a Internet esté disponible, podemos hacerlo seleccionando "Habilitar el modo fuera de línea" en el cuadro de diálogo Preferencias.
El uso de Anaconda Navigator en el modo fuera de línea es equivalente a usar los comandos de conda create, install, remove, y update con la bandera --offline, de manera que Conda no se conecte a internet.

Gestionar entornos

Anaconda permite crear entornos virtuales separados que contienen archivos, paquetes y dependencias que no interactuarán con otros entornos.

Con Navigator, como con conda, es posible crear, exportar, enumerar, eliminar y actualizar entornos que tengan instaladas diferentes versiones de Python y/o paquetes. Cambiar o moverse entre entornos se llama *activar el entorno*. No puede haber más de un entorno activo a la vez.

Dentro de Anaconda Navigator, la pestaña “Entornos” permite administrar entornos, paquetes y canales.

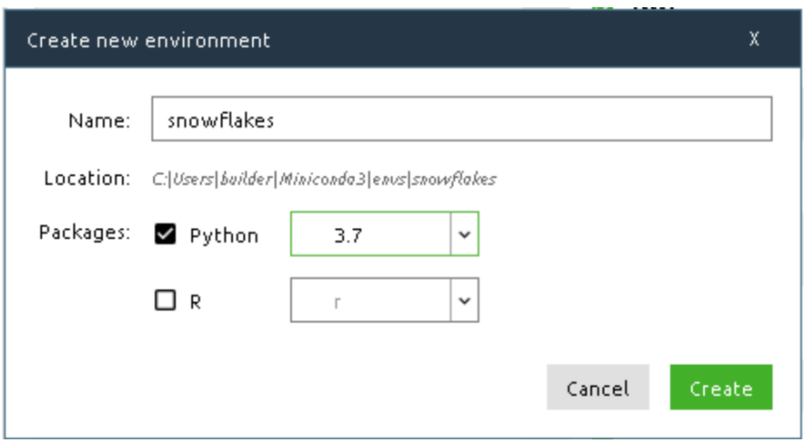


La columna izquierda enumera los entornos que tenemos creados. Basta con hacer clic en uno de ellos para activarlo. La columna de la derecha enumera los paquetes en el entorno actual.

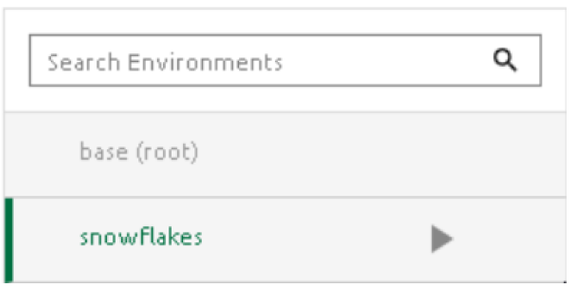
Los canales son ubicaciones donde Navigator o conda buscan paquetes. Para abrir el administrador de canales hay que hacer clic en el botón “Canales”.

Para crear un nuevo entorno con nombre *snowflakes* e instalar un paquete en él:

- En Navigator, hacer clic en la pestaña Entornos, luego hacer clic en el botón “Crear”.
- En el campo *Nombre del entorno*, escribir un nombre descriptivo para el entorno.



- Hacer clic en “Crear”. Navigator crea el nuevo entorno y lo activa.



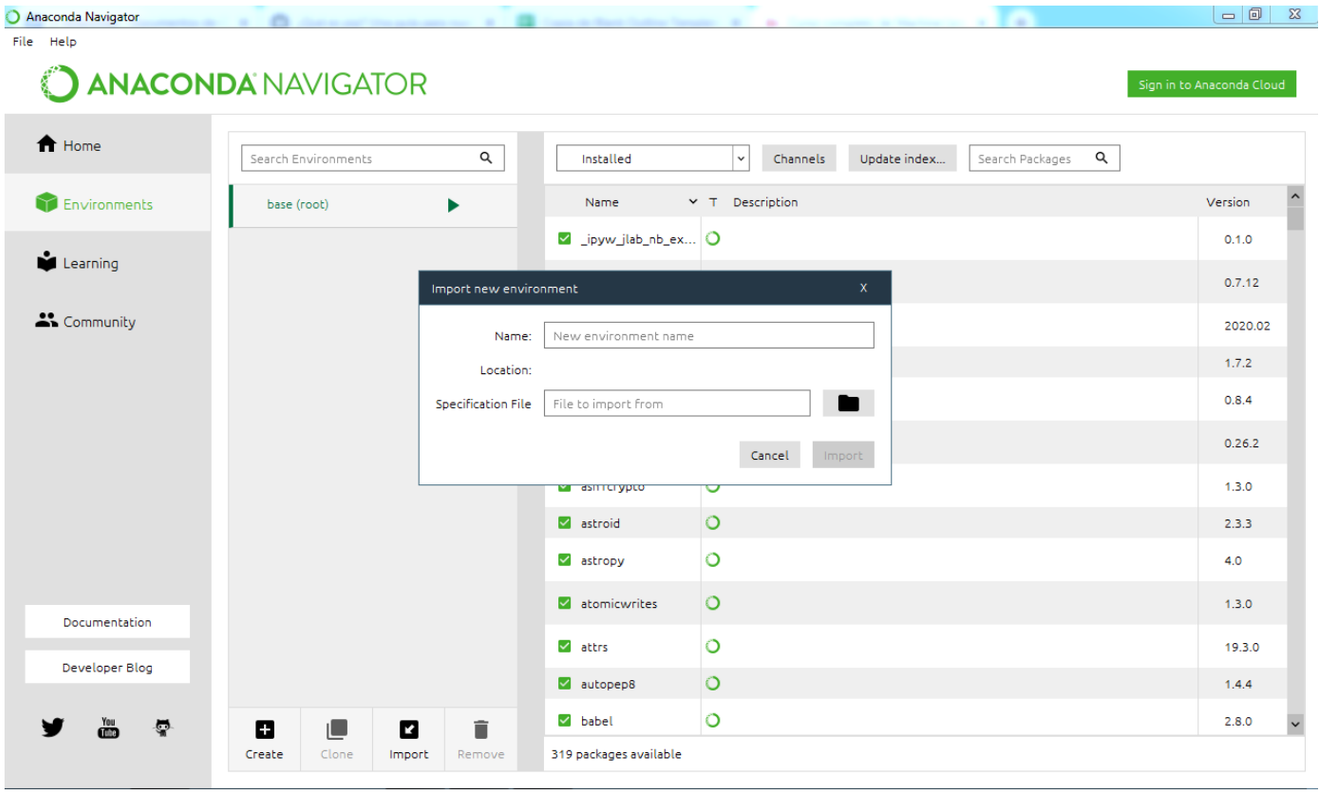
Ahora tenemos dos entornos, el entorno predeterminado *base (root)* y *snowflakes*.

- Para cambiar entre ellos (*activar* y *desactivar entornos*) hay que hacer clic en el nombre del entorno que se quiere usar. El entorno activo es el que tiene la flecha al lado de su nombre.

¡Importante! El entorno activo es el que tiene la flecha al lado de su nombre.

Para regresar al otro entorno, basta con hacer clic en su nombre.

A veces, necesitaremos crear un entorno con una serie de paquetes con versiones concretas. En estos casos, lo ideal es utilizar un archivo con extensión .yml que incluya todos los paquetes a instalar con sus versiones concretas correctamente especificadas. Desde Anaconda Navigator es posible crear un entorno a partir de un archivo .yml desde la pestaña “Entornos” (“Environments”) pulsando en el botón “Import”:



Entre los recursos de esta sección, se incluye un archivo .yml para crear un entorno idéntico al que se está utilizando durante el curso.

Gestionar Python

Muchos paquetes científicos requieren una versión específica de Python para ejecutarse, pero es difícil mantener varias instalaciones distintas de Python en nuestro ordenador. **Anaconda se encarga de gestionar múltiples instalaciones de diferentes versiones de Python sin que interfieran entre ellas.**

Cuando se crea un nuevo entorno, Navigator instala la misma versión de Python que se utilizó cuando al descargar e instalar Anaconda. Si se desea utilizar una versión diferente de Python, por ejemplo Python 3.5, simplemente hay que crear un nuevo entorno y especificar la versión de Python que se desea en ese entorno.

Para crear un nuevo entorno llamado "snakes" que contenga Python 3.5:

- En Navigator, hacer clic en la pestaña Entornos y después hacer clic en el botón “Crear”.
  Aparece el cuadro de diálogo *Crear nuevo entorno*.
- En el campo *Nombre del entorno*, escribir el nombre "snakes" y seleccionar la versión de Python que se desea usar en el cuadro Paquetes de Python (3.8, 3.7, 3.6, 3.5 o 2.7). Seleccionar una versión diferente de Python de las que hay en los otros entornos que se crearon anteriormente (en los ejemplos de este manual, los otros entornos son *base* y *snowflakes*).
- Hacer clic en el botón “Crear”.
- Activar la versión de Python que se desea usar haciendo clic en el nombre de ese entorno.

Gestionar paquetes

Con Anaconda Navigator, es posible comprobar qué paquetes se han instalado y cuáles están disponibles, buscar un paquete específico e instalarlo.

- Para buscar un paquete que ya se haya instalado, hacer clic en el nombre del entorno en que se desea buscar. Los paquetes instalados se muestran en el panel derecho.

- Se puede cambiar la selección de paquetes que se muestran en el panel derecho en cualquier momento haciendo clic en el cuadro desplegable que se encuentra sobre él y seleccionando *Instalado, No instalado, Actualizable, Seleccionado* o *Todo*.
- Para verificar si un paquete que no se ha instalado, por ejemplo "beautifulsoup4", está disponible en el repositorio de Anaconda, es necesario estar conectado a Internet. En la pestaña Entornos, en el cuadro *Buscar paquetes*, escribir "beautifulsoup4" y, en el cuadro Buscar subconjunto, seleccionar *Todo* o *No instalado*.
- Para instalar el paquete en el entorno actual, marcar la casilla de verificación junto al nombre del paquete, luego hacer clic en el botón ”Aplicar” inferior. El programa recién instalado se muestra en la lista de programas instalados.

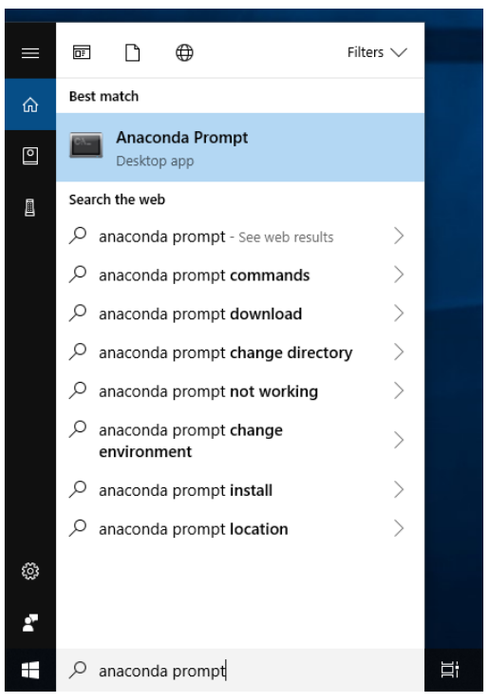
Conda

Conda es un sistema de gestión de paquetes y de entornos que se utiliza con comandos de línea de comandos en la terminal de Anaconda para Windows, o en una ventana de terminal simple para macOS o Linux.

Conda instala, ejecuta y actualiza rápidamente los paquetes y sus dependencias. También crea, guarda, carga y cambia fácilmente entre entornos virtuales. Con solo unos pocos comandos, permite configurar un entorno totalmente separado para ejecutar una versión concreta de Python, mientras continuamos ejecutando otra versión diferente de Python en nuestro entorno normal.

¿Cómo se utiliza conda?

- En Windows, en el menú Inicio, buscar y abrir "Anaconda Prompt", que es la terminal propia de Anaconda.



  En Windows, todos los comandos de conda se escriben en esta terminal.

- En macOS, abrir Launchpad y luego hacer clic en el icono de terminal. En macOS, todos los comandos de conda se escriben en esta ventana de terminal.

- En Linux, abrir una ventana de terminal. En Linux, todos los comandos de conda se escriben en esta ventana de terminal.

Una vez abierta la terminal que corresponda según nuestro sistema operativo (Windows, macOS o Linux), para verificar que conda esté instalado y ejecutándose en nuestro sistema podemos escribir el comando:

```
conda --version
```

Si está instalado, conda mostrará el número de su versión, por ejemplo conda 4.7.12.

Para actualizar conda a la versión más actual, escribir:

```
conda update conda
```

Si hay disponible una versión más reciente de conda, escribir `y` (del inglés “yes”) para actualizarla:

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

Existen varios comandos de conda que es interesante conocer. Para ello, consulta la **hoja de trucos** incluida en este capítulo del curso.

Para abreviar, puedes simplificar muchas opciones de comando de uso frecuente que van precedidas por 2 guiones (--), escribiéndolas solo 1 guión delante, seguido de la primera letra de la opción. Por ejemplo, `--name` y `-n` son lo mismo. Lo mismo sucede con `--env` y `-e`, ambos son equivalentes.

Gestionar entornos

Conda permite crear entornos virtuales separados que contienen sus propios archivos, paquetes y dependencias, de tal forma que no interactúan con otros entornos.
Al comenzar a usar conda, ya existe un entorno predeterminado denominado *base*. Sin embargo, no es conveniente trabajar en este entorno. Lo ideal es crear entornos separados para mantener los programas que se vayan creando aislados unos de otros.

Para crear un nuevo entorno e instalar un paquete en él:

- Denominaremos el entorno *snowflakes* e instalaremos el paquete “BioPython”. En la terminal de Anaconda (en el caso de Windows) o en la ventana de terminal (Linux o macOS), escribir lo siguiente:

```
conda create --name snowflakes biopython
```

- Conda verifica qué paquetes adicionales ("dependencias") necesitará “BioPython” y pregunta si deseamos continuar:

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

- Escribir "y" (del inglés “yes”) y presionar la tecla Enter para continuar.

- Para usar o *activar* el nuevo entorno, escribir lo siguiente:

```
conda activate snowflakes
```

Para ver una lista de todos los entornos que hemos creado, escribir conda info --envs. Aparece una lista de entornos, similar a la siguiente:

conda environments:

base /home/username/Anaconda3

*snowflakes * /home/username/Anaconda3/envs/snowflakes*

¡Importante! El entorno activo es el que tiene un asterisco (*).

Para cambiar del entorno actual al predeterminado (*base*), simplemente escribir:

```
conda activate
```

Si se desea crear un entorno desde un archivo .yml:

```
conda env create -f environment.yml
```

Gestionar Python

Cuando se crea un nuevo entorno, conda instala la misma versión de Python que la que se utilizó al descargar e instalar Anaconda. Si se desea utilizar una versión diferente de Python, por ejemplo Python 3.5, simplemente hay que crear un nuevo entorno y especificar la versión de Python que corresponda.

Para crear un nuevo entorno llamado "snakes" que contenga Python 3.5, escribir en la terminal (la que corresponda según nuestro sistema operativo):

```
conda create --name snakes python=3.5
```

Cuando conda pregunte si deseamos continuar, escribir "y" y presionar Enter.

Para activar el nuevo entorno, escribir:

```
conda activate snakes
```

Para verificar que el entorno “snakes“ se haya agregado y esté activo:

```
conda info --envs
```

Se mostrará una lista similar a la siguiente:

conda environments:

base /home/username/anaconda3

*snakes * /home/username/anaconda3/envs/snakes*

snowflakes /home/username/anaconda3/envs/snowflakes

Para verificar qué versión de Python está instalada en el entorno actual:

```
python --version
```

Gestionar paquetes

Con conda, es posible verificar qué paquetes hay instalados, cuáles están disponibles y buscar un paquete específico para instalarlo.

Para encontrar un paquete que ya esté instalado, primero hay que activar el entorno en que se desea buscar. También se puede verificar si un paquete que aún no haya sido instalado, por ejemplo "beautifulsoup4", está disponible en el repositorio de Anaconda (es necesario estar conectados a Internet):

```
conda search beautifulsoup4
```

Conda muestra una lista de todos los paquetes con ese nombre en el repositorio de Anaconda, por lo que sabemos que está disponible. Para instalar este paquete en el entorno actual:

```
conda install beautifulsoup4
```

Para desinstalar un paquete en el entorno activo (por ejemplo, el paquete anterior):

```
conda uninstall beautifulsoup4
```

Para ver una lista de los paquetes instalados en este entorno:

```
conda list
```

Conda VS Pip

Conda y pip a menudo se consideran casi idénticos. Aunque algunas de las funcionalidades de estas dos herramientas se superponen, fueron diseñadas y deberían usarse para diferentes propósitos.

Pip es la herramienta recomendada por Python Packaging Authority para instalar paquetes desde Python Package Index, PyPI. **Aunque Anaconda cuenta con su propio sistema de gestión de paquetes (conda), también incluye pip**, debido a su enorme uso en toda la comunidad Python y a que, en ocasiones, puede utilizarse de forma complementaria a conda, como veremos más adelante.

Conda, como ya se indicó en la sección anterior, es un administrador de paquetes y entornos multiplataforma que instala y administra paquetes conda desde el Repositorio de Anaconda así como de la Anaconda Cloud. Los paquetes conda no se limitan al software Python, también pueden contener bibliotecas C o C ++, paquetes R o cualquier otro software.

Esto supone una diferencia clave entre conda y pip. **Pip instala paquetes de Python, mientras que conda instala paquetes que pueden contener software escrito en cualquier idioma**.

Otra diferencia clave entre las dos herramientas es que conda tiene la capacidad de crear entornos virtuales aislados que pueden contener diferentes versiones de Python y / o los paquetes instalados en ellos. Esto puede ser extremadamente útil cuando se trabaja con herramientas de ciencia de datos, ya que diferentes herramientas pueden contener requisitos conflictivos que podrían evitar que se instalen en un solo entorno. Pip no tiene soporte integrado para entornos, sino que depende de otras herramientas como virtualenv o venv para crear ambientes aislados. Herramientas como pipenv y poetry proporcionan un método unificado para trabajar con estos entornos.

Pip y conda también se distinguen por cómo se cumplen las relaciones de dependencia dentro de un entorno. Al instalar paquetes, **pip instala dependencias en un bucle recursivo en serie**. No se hace ningún esfuerzo para garantizar que las dependencias de todos los paquetes se cumplan simultáneamente. Esto puede conducir a entornos que se rompen de manera sutil, si los paquetes instalados en primer lugar tienen versiones de dependencia incompatibles en relación con los paquetes instalados más adelante. Por el contrario, **conda utiliza un "solucionador de satisfacción" (SAT) para verificar que se cumplen todos los requisitos de todos los paquetes** instalados en un entorno. Esta comprobación puede llevar más tiempo, pero ayuda a evitar la creación de entornos dañados.

Dadas las similitudes entre conda y pip, no es sorprendente que algunos intenten combinar estas herramientas para crear entornos de ciencia de datos. Una razón importante para combinar pip con conda es cuando uno o más paquetes solo están disponibles para instalar a través de pip. Más de 1.500 paquetes están disponibles en el repositorio de Anaconda, y además existen miles de paquetes adicionales disponibles en la nube de Anaconda desde los canales conda-forge y bioconda. A pesar de esta gran colección de paquetes, sigue siendo pequeña en comparación con los más de 150.000 paquetes disponibles en PyPI.

Ocasionalmente se necesita un paquete que no está disponible como paquete conda, pero está disponible en PyPI y se puede instalar con pip. En estos casos, tiene sentido tratar de usar tanto conda como pip. Sin embargo, es importante tener en cuenta que, algunas veces, pueden surgir problemas cuando conda y pip se utilizan juntos, y con el fin de evitarlos en la medida de lo posible existen recomendaciones de buenas prácticas que puedes consultar en este artículo.

La siguiente tabla resume las principales diferencias entre conda y pip:

	conda	**pip**
Gestiona paquete de tipo:	Binario	Wheel o source
Necesita compiladores	No	Sí
Tipos de paquetes	Cualquiera	Solo Python
Crean entornos	Sí	No, requiere *virtualenv* o *venv*
Comprueba dependencias	Sí	No

Gestionar paquetes con pip

En secciones anteriores, vimos cómo gestionar paquetes tanto con Anaconda Navigator como con conda. Además, también sabemos ya que es posible utilizar pip de manera alternativa a conda, o bien combinando ambos sistemas. Pero, ¿cómo se utiliza pip dentro de Anaconda?

En primer lugar, hay que instalar conda en el entorno dentro del cual se desea trabajar (por ejemplo, “myenv”) y activarlo.

```
conda install -n myenv pip
```

```
conda activate myenv
```

Una vez activado, para comprobar que pip está instalado lanzar el comando:

```
pip --version
```

Si está instalado, la salida debería ser similar a la siguiente:

```
pip 18.1 from C:\Python37\lib\site-packages\pip (python 3.7)
```

Lanzando el comando `pip help`, se muestra una lista de los comandos disponibles, además de las opciones generales (aquí no incluidas):

```
$ pip help

Usage:

  pip <command> [options]

Commands:

  install                     Install packages.

  download                    Download packages.

  uninstall                   Uninstall packages.

   freeze                           Output installed packages in requirements
format.

  list                        List installed packages.

  show                        Show information about installed packages.

  check                       Verify installed packages have compatible

                              dependencies.

  config                      Manage local and global configuration.

  search                      Search PyPI for packages.

  wheel                       Build wheels from your requirements.

  hash                        Compute hashes of package archives.

  completion                  A helper command used for command completion.

  help                        Show help for commands.
```

Como se puede ver, pip cuenta con el comando *install* para instalar paquetes. Por ejemplo, para instalar el paquete “pandas”, habría que lanzar las siguientes instrucciones desde la terminal correspondiente (Anaconda prompt en Windows, y terminal en Linux o macOS):

```
pip install pandas
```

Para desinstalar un paquete, basta con lanzar el comando:

```
pip uninstall pandas
```

Si se desea ver una lista de los paquetes instalados en el entorno activo, el siguiente comando la mostrará:

```
pip list
```

Resumen de la sección

Tras esta sección, tenemos claros los siguientes conceptos:

- **Módulo**: a grandes rasgos, es una porción de un programa que puede ser importada por otros programas, es decir, es un conjunto de líneas de código que se usan para un propósito específico y se pueden usar en otros programas tal como están. Formalmente, un módulo de Python es un archivo con extensión .py que contiene definiciones y declaraciones de Python.

- **Paquete**: básicamente, es un directorio que contiene un conjunto de módulos, o de forma aún más simple, es una carpeta que contiene varios archivos. Más formalmente, es una colección de módulos bajo un espacio de nombres común.

- **Librería o biblioteca**: conjunto de implementaciones funcionales (conjunto de módulos y paquetes). Es decir, una librería no es un programa que se ejecute de manera autónoma, es código que lleva a cabo ciertas funcionalidades muy concretas para simplificar tareas complejas con el fin de servir a otros programas que la invoquen.

- **Entorno virtual**: "ambiente" bien delimitado creado en nuestro ordenador con el objetivo de aislar recursos (como librerías) del sistema principal o de otros entornos virtuales. Esto significa que en el mismo sistema (ordenador) es posible tener instaladas múltiples versiones de una misma librería sin crear ningún tipo de conflicto.

- **Anaconda**: es un administrador de paquetes, un administrador de entornos, una distribución de data/science de Python/R y una colección de más de 7500 paquetes de código abierto.

  Anaconda Individual Edition contiene *conda* (sistema de gestión de paquetes y de entornos virtuales) y *Anaconda Navigator*, además de Python y cientos de paquetes científicos. Al instalar Anaconda, estamos instalando todo esto.

- **Miniconda**: alternativa a Anaconda mucho más ligera.

- **Anaconda Navigator**: interfaz gráfica de usuario (GUI) de escritorio incluida en la distribución Anaconda que nos permite iniciar aplicaciones como Jupyter Notebook o Spyder, además de administrar fácilmente paquetes, entornos y canales de conda, todo ello sin usar comandos de línea de comandos.

- **Conda**: sistema de gestión de paquetes y de entornos que se utiliza con comandos de línea de comandos en la terminal de Anaconda para Windows, o en una ventana de terminal simple para macOS o Linux.

- **Pip**: herramienta recomendada por Python Packaging Authority para instalar paquetes desde Python Package Index, PyPI. Aunque Anaconda cuenta con su propio sistema de gestión de paquetes (conda), también incluye pip.

Además, hemos aprendido a utilizar la interfaz de Anaconda Navigator y los comandos de conda para gestionar entornos, administrar varias versiones de Python y gestionar paquetes. También hemos visto las diferencias entre conda y el gestor de paquetes pip, y la forma de utilizar este último con Anaconda.

Puedes encontrar más información acerca de la distribución Anaconda en
https://www.anaconda.com/