Anaconda environments

# Anaconda

Aunque siempre se puede instalar Python más o menos a pelo (es decir, por sí solo), hemos optado por hacerlo a través de [Anaconda](https://docs.anaconda.com/) que es una distribución de Python ampliamente utilizada, que comprende además un gestor de paquetes y de entornos.

# Configurando nuestro entorno

Por defecto, cualquier paquete o librería que instalemos en Anaconda lo haremos sobre el entorno *root*, este entorno es el único existente al principio, es decir, según terminamos de instalar Anaconda. Todos los notebooks que creemos en un principio lo haremos sobre el. Ahora bien, siempre tenemos la opción de crear (modificar y eliminar) entornos adicionales.

* Un entorno nos permite configurar la versión de Python que queremos manejar, así como los paquetes que queremos utilizar en el mismo y sus versiones. De esta manera podemos tener nuestros proyectos organizados y preservar diversos entornos con diversos paquetes, diversas versiones de ellos e incluso de Python.

Para ver hasta qué punto disponemos de versiones de Python para elegir, haremos una prueba

* Lo primero es abrir el Anaconda Prompt (muy posiblemente tengamos el icono con forma de una pantallita negra en nuestro escritorio), una vez abierto ejecutaremos el comando conda search “^python$” (*conda search* nos permite buscar paquetes disponibles y las versiones de los mismos, en este caso le pasamos una expresión regular indicando que queremos ver aquellos paquetes que solo contengan la palabra python; ^ indica inicio de línea y $ indica final de línea, estos y otros muchos caracteres nos permiten construir lo que se denominan *expresiones regulares*). Es importante entrecomillar, como se indica, la expresión.

Vamos a optar por crear un entorno basado en Python 3.5, a la hora de poner el nombre es importante que sea descriptivo para cuando vayamos a acceder a él, tengamos una idea del propósito con el que fue creado y de qué contiene. En este caso y por ser un entorno de prueba (que seguramente se eliminará al termina de leer esto) nos permitiremos una licencia:

* En el terminal ejecutad conda create --name datahack\_bootcamp python=3.5 jupyter

Fijaos que después de indicar la versión indicamos aquellos paquetes que queremos que se instalen junto con el entorno, esto se puede hacer en el momento de crear el entorno, o bien se puede proceder a la instalación de los mismos una vez se haya creado el entorno. Lo siguiente que os aparecerá es un mensaje informándoos de los paquetes básicos que se instalarán en nuestro nuevo entorno, si estamos de acuerdo tendremos que responder con y y enter (o simplemente enter ya que por defecto está seleccionado y). Habrá que esperar un momento a que se terminen de descargar e instalar todos los paquetes (el tiempo es variable, hay que tener en cuenta que cada paquete es descargado), una vez completada la instalación se nos informará de cómo podemos activar este entorno:

* Para activar nuestro entorno bastará con ejecutar activate datahack\_bootcamp

De este modo, el entorno *root* dejará de estar activo y cualquier instalación que hagamos, notebook que abramos o código que ejecutemos, lo haremos contra el entorno que activo). Veremos que el prompt de nuestro terminal cambia para indicar que ahora tenemos activado el entorno *datahack\_bootcamp*. Desde aquí podemos verificar, por ejemplo, que la versión de Python que estamos usando es la que indicamos al instalar el entorno:

* Para verificar esto ejecutaremos python --version

Es muy recomendable instalar en cada nuevo entorno que creemos el paquete nb\_conda. Este paquete permite integrar Anaconda con los notebook de Jupyter, incluso nos permite cambiar de un entorno a otro desde el interfaz de Jupyter:

* Ejecutando conda install nb\_conda

Aseguraos de que antes de instalar cualquier cosa, tenéis activo el entorno sobre el cual queréis instalarlo. Parece de perogrullo, pero en el fragor de las instalaciones…se puede olvidar.

Es interesante además recordar que, si queremos abrir nuestro servidor de Jupyter desde un directorio en concreto a partir del Anaconda Prompt, procederemos de la siguiente manera:

* Dentro del Anaconda Prompt: cd <directorio\_destino>
* Esto nos llevará al directorio destino (acordaos que el comando cd significa “change directory”)
* Desde aquí ejecutaremos jupyter notebook (sin guion ni nada) para abrir el servidor de Jupyter. Es importante aclarar que, al igual que cuando abrimos el servidor de Jupyter desde su acceso directo, solamente podremos descender en el árbol de directorios con respecto al directorio inicial, no podremos ir al directorio padre del directorio inicial.

De vuelta al Anaconda Prompt, otro comando que nos será útil cuando tengamos varios entornos creados ya que nos permite listar no solo su nombre sino su ubicación física es:

* conda env list (ó también conda info --envs)

Además, nos indica con un asterisco el entorno activo. Es importante recordar que si abrimos un nuevo Anaconda prompt el entorno por defecto será de nuevo el *root*.

Seguramente durante el proceso de creación del entorno, al ver los paquetes que se instalaron por defecto, se nos planteó la duda de cómo podemos instalar nosotros mismos paquetes adicionales en nuestro entorno a medida que los vayamos necesitando:

* Si no tenemos ningún entorno activado, podremos instalar paquetes a través de conda, ejecutando conda install --name <nombre\_entorno> <nombre\_paquete>
* Si tenemos algún entorno activado, podremos instalar paquetes en ese mismo entorno a través de conda, ejecutando conda install <nombre\_paquete>
* Igualmente podremos listar los paquetes del entorno actualmente activo junto con sus versiones mediante conda list (igualmente, si no tenemos ningún entorno activo, podremos listar sus paquetes y las versiones de los mismos mediante conda list --name <nombre\_entorno>)
* Nos encontraremos con que en ocasiones conda no será suficiente para instalar todos los paquetes que en un momento dado requiramos; aquí es cuando entra en juego pip (**P**ip **I**nstalls **P**ackages) que es el package manager oficial de Python, encargado mayormente de instalar aquellos paquetes publicados en [Python Package Index (PyPI)](https://pypi.python.org/pypi).

A veces es posible que necesitemos exportar nuestro entorno de manera que un compañero o nosotros mismos en otra máquina, podamos replicar ese mismo entorno sin tener que crearlo desde cero e ir instalando paquete a paquete. Para exportar un entorno, haremos lo siguiente:

* Primero activamos el entorno a exportar: activate <nombre\_entorno>
* Después exportamos el entorno: conda env export > <nombre\_fichero>.yml

El fichero resultante está en formato [YAML](https://en.wikipedia.org/wiki/YAML), podéis abrirlo y leerlo sin problema. Pero más que leerlo, lo que seguramente queramos es usar ese fichero para recrear el entorno:

* conda env create --file=<nombre\_fichero>.yml o también conda env create –f <nombre\_fichero>.yml

Finalmente, si queremos borrar un entorno haremos lo siguiente:

* conda remove --name <nombre\_entorno> --all