Blog

# ENTORNOS VIRTUALES EN PYTHON Y ANACONDA

## ¿Por qué necesito Entornos Virtuales?

Las aplicaciones de Python a menudo usan paquetes y módulos que no vienen como parte de la biblioteca estándar. En la mayoría de ocasiones, las aplicaciones necesitarán una versión específica de una librería, ya que la aplicación puede requerir que se corrija un error en particular o que la aplicación se escriba usando una versión obsoleta de la interfaz de la biblioteca.

Esto significa que puede que no sea posible que una instalación de Python cumpla con los requisitos de cada aplicación. Si la aplicación A necesita la versión 1.0 de un módulo en particular, pero la aplicación B necesita la versión 2.0, entonces los requisitos están en conflicto y la instalación de la versión 1.0 o 2.0 dejará una aplicación incapaz de ejecutarse.

La solución para este problema es crear un **entorno virtual**, un árbol de directorios autocontenido que contenga una instalación de Python para una versión particular.

Diferentes aplicaciones pueden utilizar diferentes entornos virtuales. Para resolver el ejemplo anterior de requisitos en conflicto, la aplicación A puede tener su propio entorno virtual con la versión 1.0 instalada, mientras que la aplicación B tiene otro entorno virtual con la versión 2.0. Si la aplicación B requiere que una biblioteca se actualice a la versión 3.0, esto no afectará el entorno de la aplicación A.

# Usando Entornos Virtuales

Si no estas utilizando Python 3 podemos instalar **Entornos Virtuales** con la herramienta pip:

```
1. pip install virtualenv
```

Si está utilizando Python 3, entonces ya debería tener instalado el módulo venv (https://docs.python.org/3/library/venv.html) de la biblioteca estándar.

Comenzamos por hacer un nuevo directorio para trabajar:

```
1. mkdir python-virtual-environments && cd python-virtual-environments
```

Para crear el nuevo entorno virtual

Para python 2:

```
1. virtualenv env
```

Para python 3:

```
1. python3 -m venv env
```

El enfoque de Python 3 venv tiene la ventaja de obligarlo a elegir una versión específica del intérprete de Python 3 que se debe utilizar para crear el entorno virtual. Esto evita cualquier confusión en cuanto a en qué instalación de Python se basa el nuevo entorno.

De Python 3.3 a 3.4, la forma recomendada de crear un entorno virtual era usar la herramienta de línea de comandos pyvenv que también viene incluida con su instalación de Python 3 de forma predeterminada. Pero en 3.6 y superiores, python3 -m venv es el camino a seguir.

Con esta tenemos nuestro ambiente virtual creado con la siguiente esctructura:

```
1. env/
```

```
  bin/
  include/
  lib/
    site-packages/
```

En el directorio `bin/` se encuentran los ejecutables necesarios para interactuar con el entorno virtual. En el directorio `include/` se encuentran algunos archivos de cabecera de C (cuya extensión es `*.h`) necesarios para compilar algunas librerías de *Python*.

Finalmente, en el directorio `lib/` se encuentra una copia de la instalación de *Python* así como un directorio llamado `site-packages/` en donde se almacenan los paquetes *Python* instalados en el entorno virtual.

Ahora lo vamos a activar nuestro entorno virtual donde en un futuro instalaremos todos los módulos y librerías que necesitemos para nuestro proyecto. Lo activamos desde el directorio `bin`

```
source bin/activate
```

Y ahora nos aparece entre paréntesis nuestro entono virtual de la forma:

```
(env) nuestra ruta:
```

Ahora ya estamos listo para instalar aquí todos los requerimientos que necesitmeos para nuestro proyecto.

Cuando necesitemos salir tan solo debemos escribir lo siguiente y saldremos de nuestro entorno

```
deactivate
```

# Creando ambientes virtuales en Anaconda

Para crear nuestro ambientes virtuales ejecutamos en nuestro cmd:

```
conda create --name myenv
```

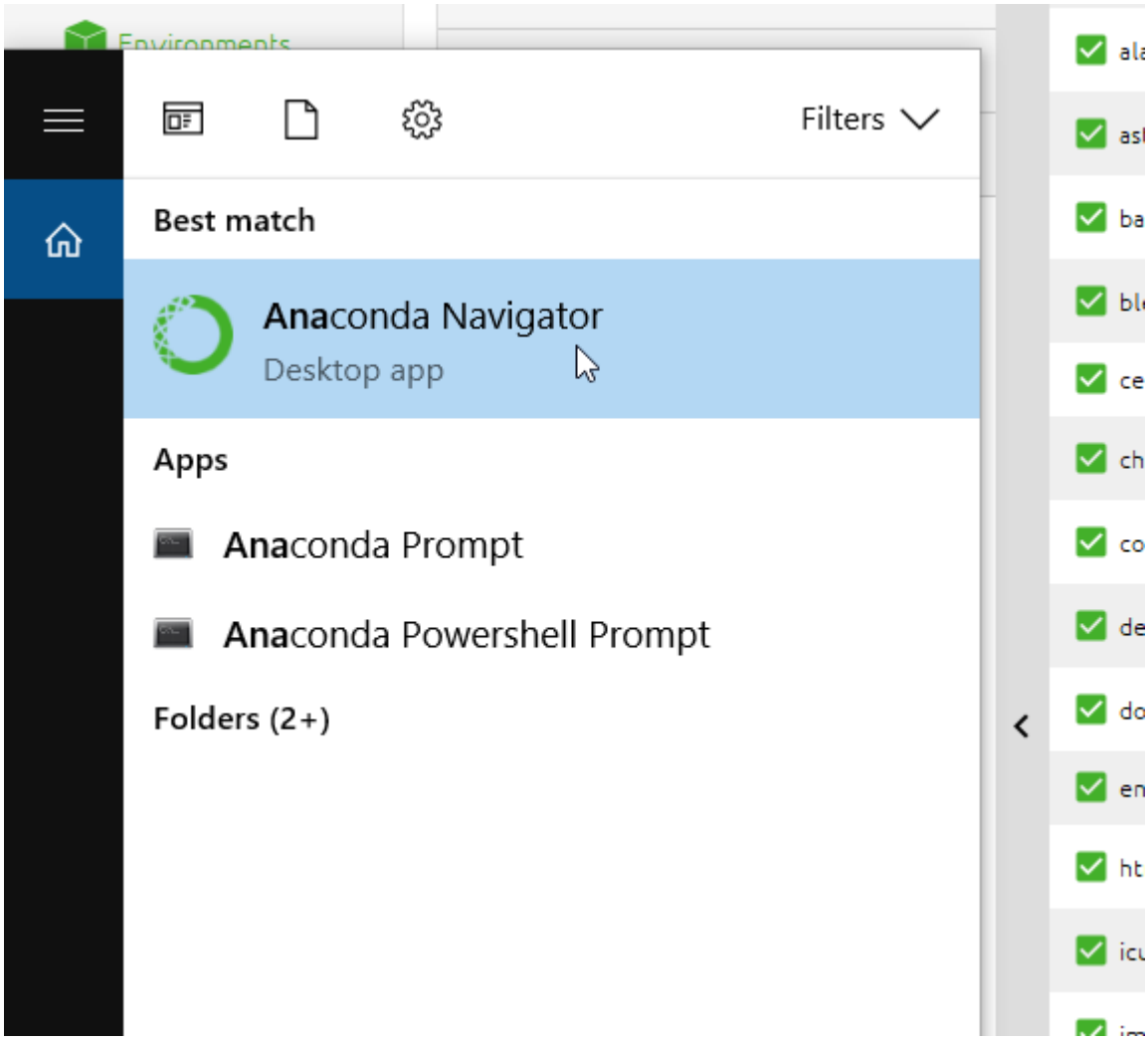
Sustituimos myenv por el nombre que queramos darle y cuando pregunte “proceed ([y]/n)?” marcar la “y” (yes) con eso ya tendremos creado nuestro ambiente virtual. Para verificarlo podemos hacerlo desde 2 formas.

1. Nos vamos a la siguinte ruta

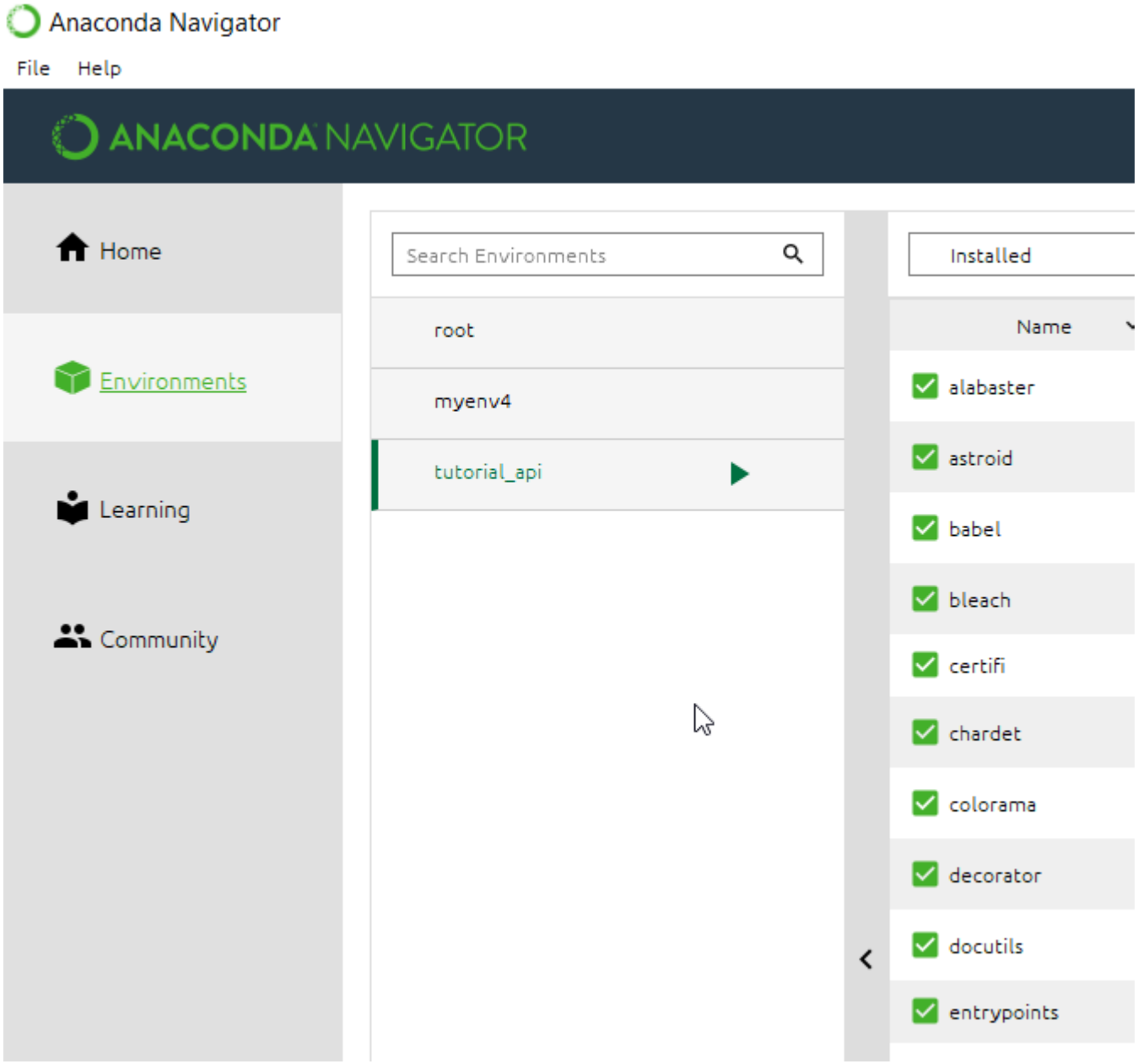
```
C:\Users\*tu usuario*\.conda
```

   Abrimos el archivo “environments.txt” y vemos los ambiertes virtuales que tenemos.

2. Ejecutamos Anaconda Navigator




(https://unipython.com/wp-content/uploads/2019/05/anaconda_navigator.png)
En anaconda navigator vamos a **enviroments** (debajo de home) y podemos ver los **enviroments** que tenemos, en mi caso yo tengo 3 root (raiz), myenv4 y tutorial_api como podemos ver en la siguiente imagen:




(https://unipython.com/wp-content/uploads/2019/05/anaconda_navigator2.png)
Si queremos crear un entorno con una versión específica de Python hacemos:

```
conda create -n myenv python=3.4
```

Para activar el entorno virtual e intalar librerias:

```
conda activate nombrequelehasdado
```

Para desactivarlo:

```
conda deactivate
```

## Ejecutar spyder desde tu entorno

Si quieres ejecutar spyder desde tu entorno virtual tan solo escribe spyder (asegurate de estar dentro de tu entorno virtual).