

Entornos virtuales para Python



¿Por qué usar entornos virtuales?

- Son cajas aisladas para los proyectos.
- Cada entorno puede tener sus propias dependencias (librerías).
- Los entornos no interfieren entre sí ni con la instalación global de Python.
- Mantiene el sistema limpio.

venv

Viene incluido con Python 3 por lo que no es necesario instalarlo.

Crea un entorno:

```
python3 -m venv nombre_del_entorno
```

Activa el entorno en Linux/macOS:

```
source nombre_del_entorno/bin/activate
```

Activa el entorno en Windows:

```
nombre_del_entorno\Scripts\activate
```

Instala paquetes:

```
pip install nombre_del_paquete
```

Desactiva el entorno:

```
nombre_del_entorno\Scripts\activate
```

conda / miniconda

Descarga: <https://www.anaconda.com/>

Miniconda es una versión más ligera de Anaconda que solo incluye el comando conda y sus dependencias.



ANACONDA®

Crea un entorno:

```
conda create --name nombre_del_entorno
```

Activa el entorno en Linux/macOS:

```
conda activate nombre_del_entorno
```

Activa el entorno en Windows:

```
activate nombre_del_entorno
```

Instala paquetes:

```
conda install nombre_del_paquete
```

Desactiva el entorno:

```
conda deactivate
```

Lista los entornos virtuales:

```
conda env list
```

poetry

Crea automáticamente un entorno virtual para cada proyecto.

Instalación (hay que instalar primero pipx):

```
pipx install poetry
```



Crea un proyecto y lo activa:

```
poetry new nombre_del_proyecto
```

Esto crea un nuevo directorio con la estructura básica de un proyecto de Python y un archivo `pyproject.toml` donde se gestionan las dependencias.

Activa un proyecto que no está activado:

```
poetry shell
```

Desactiva el entorno:

```
exit
```

Instala paquetes:

```
poetry add nombre_del_paquete
```

Lista los entornos virtuales:

```
poetry env list
```