



Entornos virtuales de Python

Los entornos virtuales se pueden describir como directorios de instalación aislados. Este aislamiento te permite localizar la instalación de las dependencias de tu proyecto, sin obligarte a instalarlas en todo el sistema.

Imagina que tienes dos aplicaciones, App1 y App2. Ambos usan el paquete Pak, pero requieren versiones diferentes. Si instala Pak versión 2.3 para App1, no podrá ejecutar App2 porque requiere la versión 3.1.

Aquí es donde los entornos virtuales son útiles.

Beneficios:

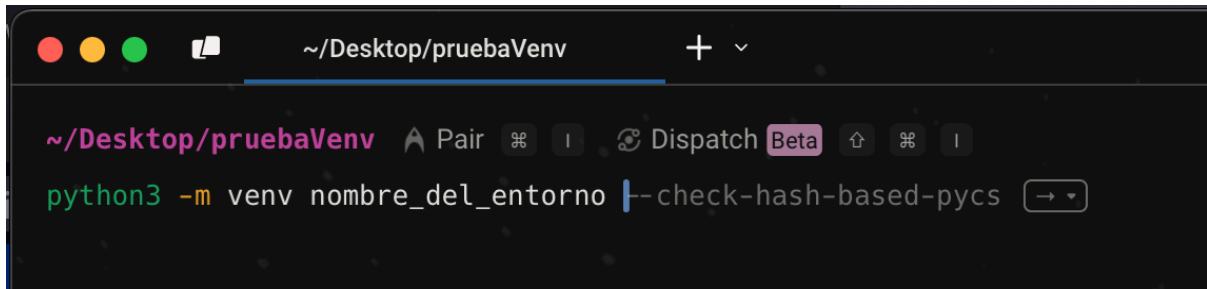
- Puedes tener varios entornos, con varios conjuntos de paquetes, sin conflictos entre ellos. De esta manera, los requisitos de diferentes proyectos se pueden satisfacer al mismo tiempo.
- Puedes lanzar fácilmente tu proyecto con sus propios módulos dependientes.

Vamos a ver cómo crearlos y utilizarlos:



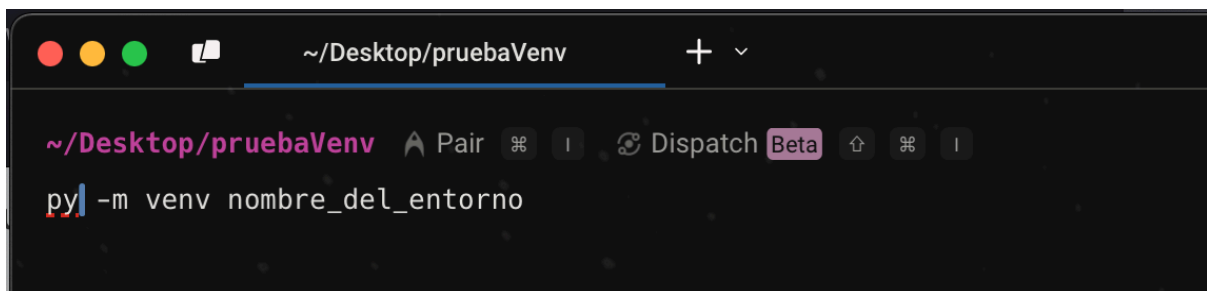
VENV

1. Creamos un entorno virtual utilizando el módulo `venv` de python, utilizando el comando `python3 -m venv nombre_del_entorno` (esto aplica a usuarios de Linux y Mac)



```
~/Desktop/pruebaVenv +  
~/Desktop/pruebaVenv Pair Dispatch Beta  
python3 -m venv nombre_del_entorno --check-hash-based-pycs
```

Para Windows sería:

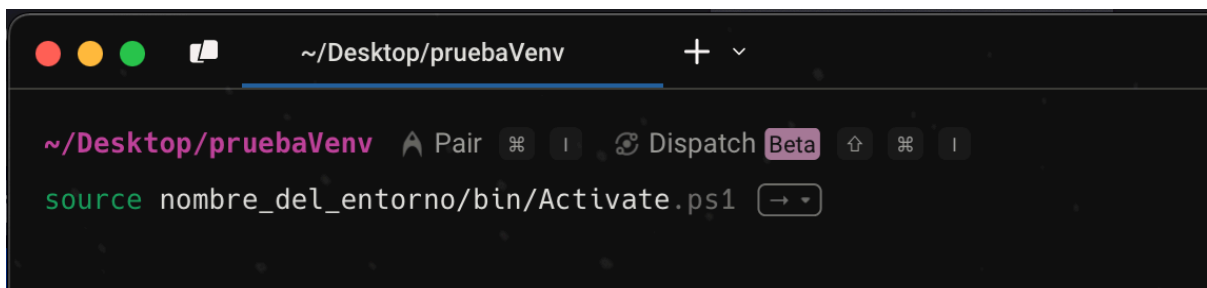


```
~/Desktop/pruebaVenv +  
~/Desktop/pruebaVenv Pair Dispatch Beta  
py -m venv nombre_del_entorno
```

2. Activamos el entorno virtual:

En Linux y Mac, lo hacemos con el comando:

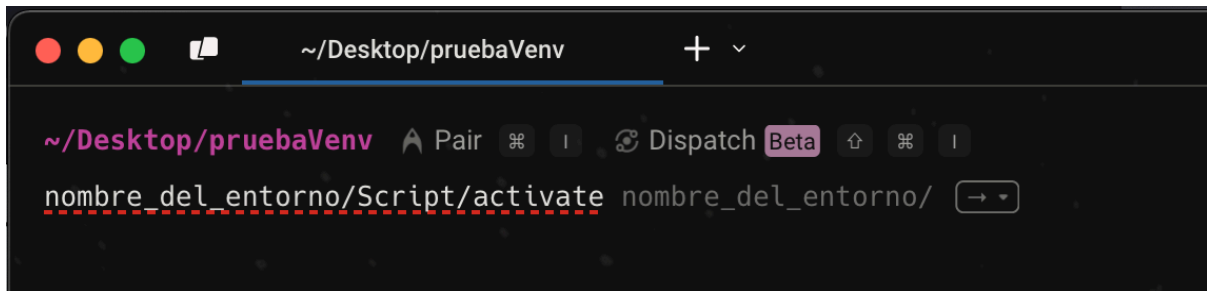
`source nombre_del_entorno/bin/Activate`



```
~/Desktop/pruebaVenv +  
~/Desktop/pruebaVenv Pair Dispatch Beta  
source nombre_del_entorno/bin/Activate.ps1
```

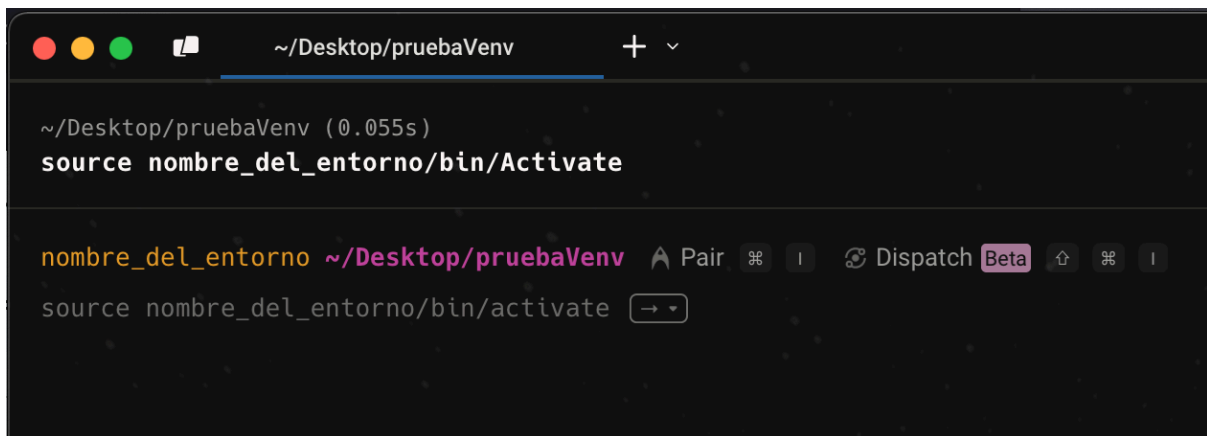


En Windows sería: `nombre_del_entorno/Script/activate`



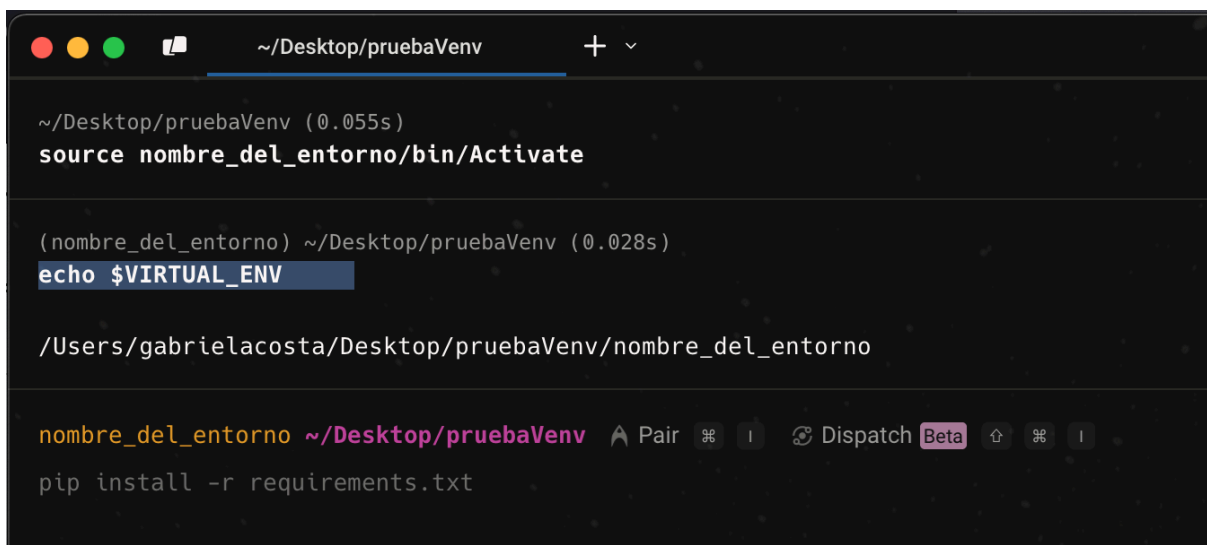
```
~/Desktop/pruebaVenv  
~/Desktop/pruebaVenv A Pair Dispatch Beta  
nombre_del_entorno/Script/activate nombre_del_entorno/
```

El resultado debería ser:



```
~/Desktop/pruebaVenv (0.055s)  
source nombre_del_entorno/bin/Activate  
nombre_del_entorno ~/Desktop/pruebaVenv A Pair Dispatch Beta  
source nombre_del_entorno/bin/activate
```

Cuando ingresamos a un entorno virtual de python, la misma terminal nos avisa que efectivamente se activó. Si no vemos cambios, podemos ingresar el comando: `echo $VIRTUAL_ENV`. Si el resultado es una ruta, significa que estás dentro de un entorno virtual. Si no imprime nada, entonces no estás en uno.

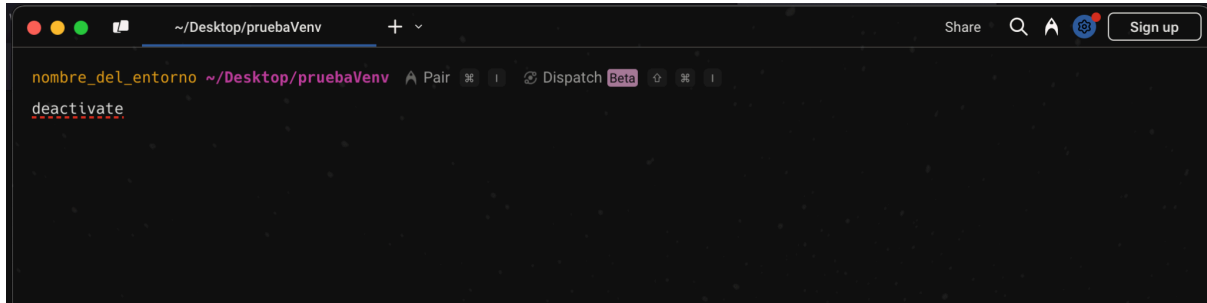


```
~/Desktop/pruebaVenv (0.055s)  
source nombre_del_entorno/bin/Activate  
(nombre_del_entorno) ~/Desktop/pruebaVenv (0.028s)  
echo $VIRTUAL_ENV  
/Users/gabrielacosta/Desktop/pruebaVenv/nombre_del_entorno  
nombre_del_entorno ~/Desktop/pruebaVenv A Pair Dispatch Beta  
pip install -r requirements.txt
```



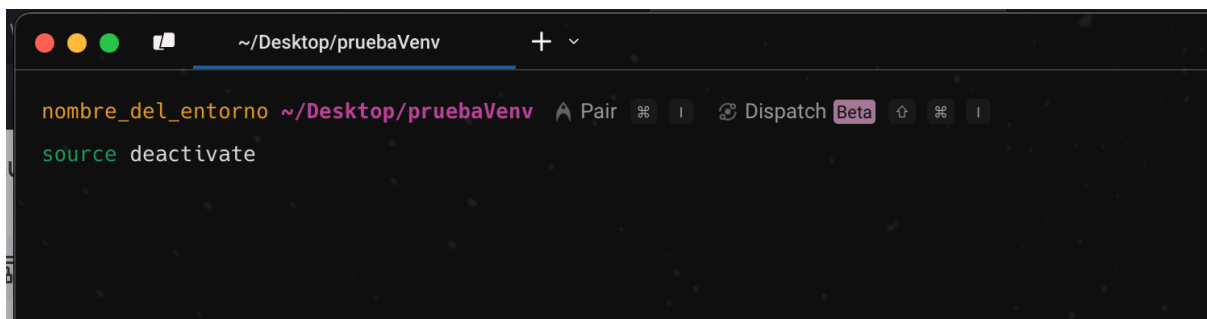
3. Salir de un entorno virtual:

Simplemente escribimos `deactivate` para salir del entorno.



```
nombre_del_entorno ~/Desktop/pruebaVenv +  
deactivate
```

Para Linux y Mac se utiliza nuevamente el comando `source`



```
nombre_del_entorno ~/Desktop/pruebaVenv +  
source deactivate
```

Instalar paquetes

Ahora, ya podemos instalar paquetes, librerías, frameworks en un espacio de trabajo aislado.

Para instalar algo, a partir de ahora vamos a usar **PIP**

Pip es el gestor de paquetes de Python. Se usa para instalar, actualizar y administrar bibliotecas y dependencias de Python desde el repositorio PyPI (Python Package Index): <https://pypi.org/>



Comandos básicos de pip:

Instalar una librería

```
nombre_del_entorno ~/Desktop/pruebaVenv + v
nombre_del_entorno ~/Desktop/pruebaVenv A Pair % I Dispatch Beta ^ % I
pip install nombre_paquete
```

Ver librerías instaladas:

```
nombre_del_entorno ~/Desktop/pruebaVenv + v
nombre_del_entorno ~/Desktop/pruebaVenv (1.709s)
pip list

Package Version
-----
pip      25.0

A Check for outdated packages. % Enter
nombre_del_entorno ~/Desktop/pruebaVenv A Pair % I Dispatch Beta ^ % I
```

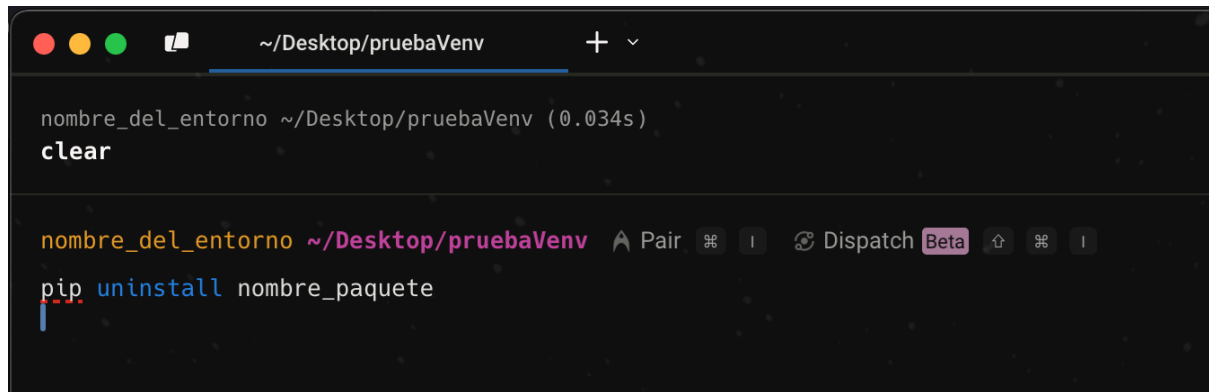
Actualizar una librería

```
nombre_del_entorno ~/Desktop/pruebaVenv + v
nombre_del_entorno ~/Desktop/pruebaVenv (0.034s)
clear

nombre_del_entorno ~/Desktop/pruebaVenv A Pair % I Dispatch Beta ^ % I
pip install --upgrade nombre_paquete
nombre_del_entorno/ → v
```



Desinstalar una librería



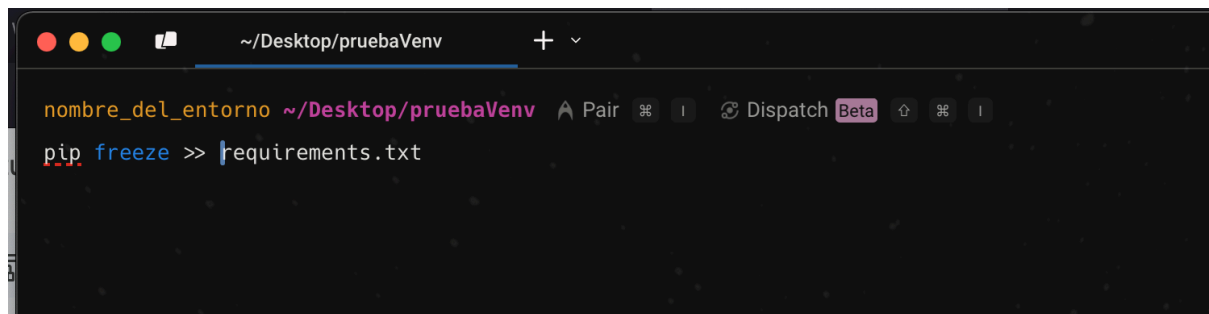
```
nombre_del_entorno ~/Desktop/pruebaVenv (0.034s)
clear

nombre_del_entorno ~/Desktop/pruebaVenv A Pair % I Dispatch Beta
pip uninstall nombre_paquete
```

Crear un requirements.txt

El archivo **requirements.txt** es una lista de las dependencias (librerías) que tu proyecto necesita para funcionar. Es muy útil para compartir tu proyecto con otros, o para instalar las mismas dependencias en otro entorno virtual.

Para poder crearlo escribimos el siguiente comando:



```
nombre_del_entorno ~/Desktop/pruebaVenv A Pair % I Dispatch Beta
pip freeze >> requirements.txt
```

Esto nos dejará como resultado un txt con todas las dependencias y sus versiones específicas utilizadas en nuestro proyecto

```
requirements.txt
1 appnope==0.1.4
2 asttokens==3.0.0
3 comm==0.2.2
4 debugpy==1.8.13
5 decorator==5.2.1
6 executing==2.2.0
7 ipykernel==6.29.5
8 ipython==9.0.2
9 ipython_pygments_lexers==1.1.1
10 jedi==0.19.2
11 jupyter_client==8.6.3
12 jupyter_core==5.7.2
13 markdown-it-py==3.0.0
14 matplotlib-inline==0.1.7
15 mdurl==0.1.2
16 nest-asyncio==1.6.0
17 packaging==24.2
18 parso==0.8.4
19 pexpect==4.9.0
20 platformdirs==4.3.7
21 prompt_toolkit==3.0.50
22 psutil==7.0.0
23 ptyprocess==0.7.0
24 pure_eval==0.2.3
25 Pygments==2.19.1
26 python-dateutil==2.9.0.post0
27 pyzmq==26.3.0
28 rich==13.9.4
29 six==1.17.0
30 stack-data==0.6.3
31 tornado==6.4.2
32 traitlets==5.14.3
33 wcwidth==0.2.13
34
```

Instalación de la librería RICH:

https://rich-readthedocs-io.translate.goog/en/stable/introduction.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

1. Vamos a instalar rich, para eso usamos el comando: pip install rich

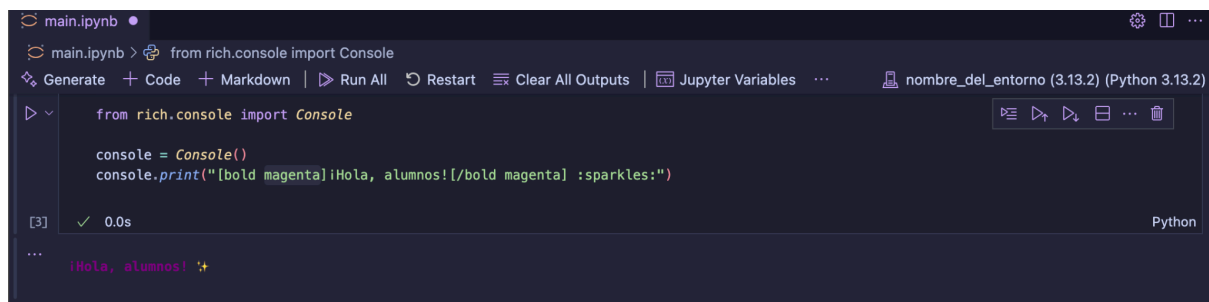
```
~/Desktop/pruebaVenv
nombre_del_entorno ~/Desktop/pruebaVenv Pair Dispatch Beta
pip install rich
```

2. Ahora vamos a probar este ejemplo de código en una celda de jupyter notebook:

```
from rich.console import Console

console = Console()
console.print("[bold magenta];Hola, alumnos![/bold magenta] :sparkles:")
```

Este es el resultado:



```
main.ipynb
main.ipynb > from rich.console import Console
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables ... nombre_del_entorno (3.13.2) (Python 3.13.2)
from rich.console import Console
console = Console()
console.print("[bold magenta];Hola, alumnos![/bold magenta] :sparkles:")
[3] ✓ 0.0s Python
... ;Hola, alumnos! :sparkles:
```

Conclusión

Los entornos virtuales son una herramienta fundamental en Python que permite aislar dependencias de proyectos, evitando conflictos entre diferentes versiones de librerías. Se crean con venv, una herramienta incluida en Python, y se activan o desactivan según sea necesario.

Beneficios clave:

- Evitan conflictos entre proyectos con diferentes versiones de paquetes.
- Mantienen limpio el sistema principal, sin instalar paquetes de manera global.
- Facilitan la portabilidad y colaboración en proyectos.