
Ambientes virtuales en Python

Gabriel Ruiz Martinez
Subcoordinación de Posgrado y Educación Continua, IMTA

2025

Contenido

1	¿Qué es un ambiente virtual en Python?	1
2	Creación y activación de un ambiente virtual en Windows	1
2.1	Activación del ambiente	4
2.2	Instalación de paquetes en el ambiente virtual	5
3	Desactivación del ambiente virtual	6
4	Ambientes virtuales en la distribución de Python Conda (Anaconda)	6
5	Ambientes virtuales en PyCharm	7
6	Manejo de ambientes virtuales en Spyder	9
7	Manejo de ambientes virtuales en Visual Studio Code, (VSC)	10
8	Comentario final	12

1. ¿Qué es un ambiente virtual en Python?

Debemos identificar que cuando usamos Python estamos empleando una "instalación del Python", la cual se encuentra integrada por un intérprete y módulos; a su vez, éstos últimos provienen de la librería estándar y si hay instalados, de paquetes por terceros. Esta estructura proporciona los componentes esenciales que requerimos para ejecutar un programa de Python, es decir, ejecutamos un script o programa en un ambiente de Python (figura 1).

Los ambientes virtuales son entornos simplificados que comparten el intérprete y la librería estándar pero como una instalación completa. Un ambiente se emplea para instalar paquetes de terceros para un proyecto o aplicación específicos, manteniendo el entorno global de Python sin modificaciones (Jolowicz, 2024).

La creación de ambientes virtuales en Python nos permite manejar de forma separada las dependencias que pueden existir entre los paquetes para un determinado proyecto; es decir, que fueron concebidos para evitar y prevenir conflictos entre paquetes con diferentes versiones, así como facilitar el mantenimiento de las configuraciones que requiere el proyecto.

2. Creación y activación de un ambiente virtual en Windows

En Python, con el módulo `venv` podemos generar ambientes independientes o distintos, donde se pueden usar determinados paquetes o versiones de éstos. Cabe resaltar que este módulo pertenece a la librería

estándar de Python y por ello, no se requiere instalarlo como otro paquete adicional.

La sintaxis para crear un ambiente en Python es:

```
python -m venv nombre_ambiente
```

La primera palabra de la instrucción anterior, le indica al sistema operativo que deberá ejecutar el programa de Python (python), para acceder al módulo (-m) que genera ambientes virtuales (venv) y este módulo creará un ambiente con el nombre especificado (nombre_ambiente).

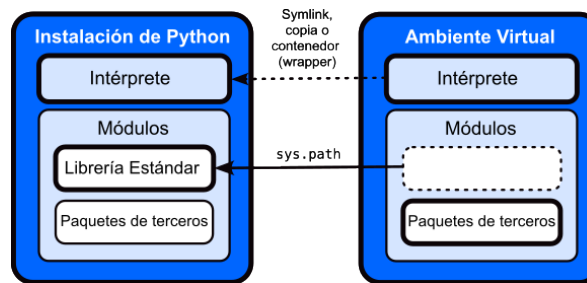


Figura 1. Los ambientes de Python poseen un intérprete y módulos. Los ambientes virtuales comparten el intérprete y la librería estándar con el ambiente global del Python. Tomado y modificado de Jolowicz (2024).

Antes de comenzar la creación de un nuevo ambiente de Python, se recomienda, ampliamente, que se genere una carpeta que almacene subcarpetas, que a su vez, corresponderán a los distintos ambientes que puedan crearse. Para este ejemplo, `envir_gar` será la carpeta donde se guardará la subcarpeta con el nuevo ambiente (figura 2).

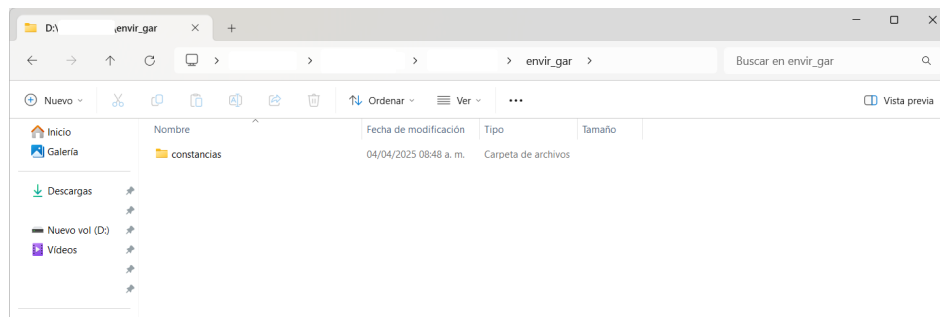


Figura 2. Carpeta donde se almacenarán las subcarpetas correspondientes a los ambientes virtuales.

Para crear un ambiente, abrimos una ventana de comandos de Python (Windows) y usando el comando `cd` nos ubicaremos dentro de la carpeta que se mencionó en el párrafo anterior (figuras 3 y 4).



Figura 3. En la ventana de comandos del sistema (Python), usamos el comando `cd` para ubicarnos en la carpeta que almacena los ambientes.

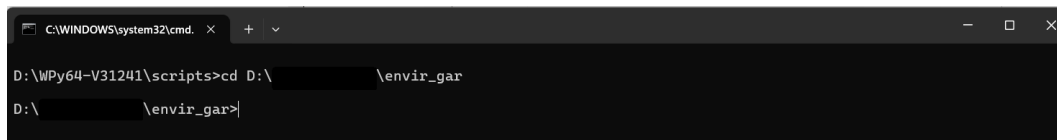


Figura 4. En la ventana de comandos del sistema (Python), nos ubicamos en la carpeta que almacena los ambientes.

Procedemos a crear un ambiente virtual de Python con el nombre de geonalisis, usando la instrucción antes mencionada (figura 5).

```
python -m venv geonalisis
```

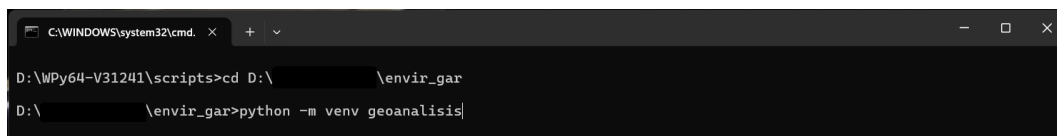


Figura 5. Creando el ambiente virtual de Python.

Una vez que proporcionamos la instrucción para crear el ambiente, esperamos un par de segundos y observaremos que el cursor de la ventana de comandos aparecerá de nuevo. Usando el Explorador de Windows podemos identificar que se generó una subcarpeta con el ambiente que ha sido creado unos momentos antes (figura 6).

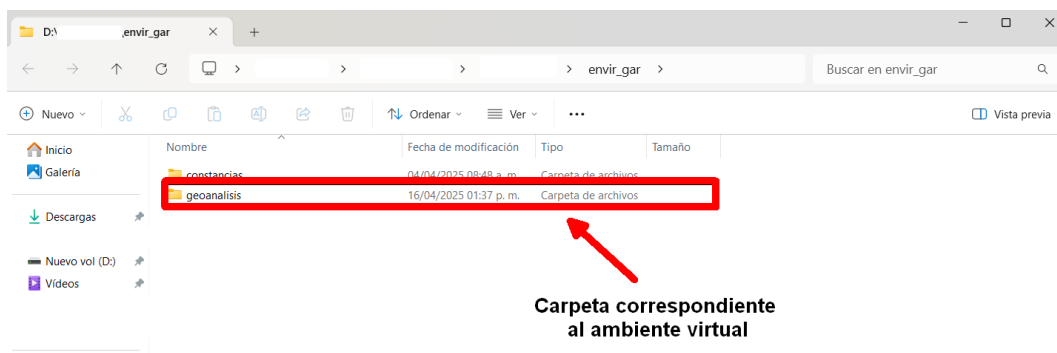


Figura 6. El proceso de creación del ambiente virtual generó una subcarpeta con el nombre del ambiente.

Dentro de la subcarpeta del ambiente, se encontrarán 3 subcarpetas (figura 7):

1. Subcarpeta Include.
2. Subcarpeta Lib.
3. Subcarpeta Scripts.

Nombre	Fecha de modificación	Tipo	Tamaño
Include	16/04/2025 01:37 p. m.	Carpeta de archivos	
Lib	16/04/2025 01:37 p. m.	Carpeta de archivos	
Scripts	16/04/2025 01:37 p. m.	Carpeta de archivos	
pyvenv.cfg	16/04/2025 01:37 p. m.	Archivo CFG	1 KB

Figura 7. Subcarpetas dentro de la carpeta del ambiente virtual, recién creado.

En la subcarpeta `Include` se guardarán los archivos de las librerías de C, si alguno de los paquetes del ambiente lo requiere; de lo contrario, esta subcarpeta se encontrará vacía.

Dentro de la subcarpeta `Lib` encontrarás la subcarpeta `site-packages`; en esta última carpeta se almacenarán todos los archivos relacionados a los paquetes instalados dentro de este ambiente.

En la subcarpeta `Scripts` se ubican 4 archivos que son importantes para usar el ambiente recién creado (figura 8):

1. `activate.bat`. Este archivo de lotes, debe ser invocado para hacer uso del ambiente.
2. `deactivate.bat`. Para dejar de usar o abandonar el ambiente este archivo deberá invocarse en la ventana de comandos.
3. `pip.exe`. Archivo ejecutable para instalar los paquetes que se emplearán en el ambiente.
4. `python.exe`. Archivo ejecutable para invocar el intérprete de Python que se usará dentro del ambiente.

Nombre	Fecha de modificación	Tipo	Tamaño
activate	16/04/2025 01:37 p. m.	Archivo	3 KB
activate.bat	16/04/2025 01:37 p. m.	Archivo por lotes ...	1 KB
Activate.ps1	16/04/2025 01:37 p. m.	Script de Windows...	26 KB
deactivate.bat	16/04/2025 01:37 p. m.	Archivo por lotes ...	1 KB
pip.exe	16/04/2025 01:37 p. m.	Aplicación	106 KB
pip3.12.exe	16/04/2025 01:37 p. m.	Aplicación	106 KB
python.exe	16/04/2025 01:37 p. m.	Aplicación	264 KB
pythonw.exe	16/04/2025 01:37 p. m.	Aplicación	253 KB

Figura 8. Archivos dentro de la carpeta `Scripts`.

2.1. Activación del ambiente

Para comenzar a usar el ambiente es necesario "activarlo" y para ello, en la ventana de comandos vamos a ejecutar el archivo `activate.bat` de la siguiente manera:

```
nombre_ambiente\Scripts\activate
```

para nuestro ejemplo, usamos:

```
geoanalisis\Scripts\activate
```

Identifica que antes de la ruta de acceso de la ventana de comandos, el sistema agregó el nombre del nuestro ambiente entre paréntesis; indicándonos que ya nos encontramos en el ambiente y podemos realizar nuestro proyecto con los paquetes que se requieran (figura 9).

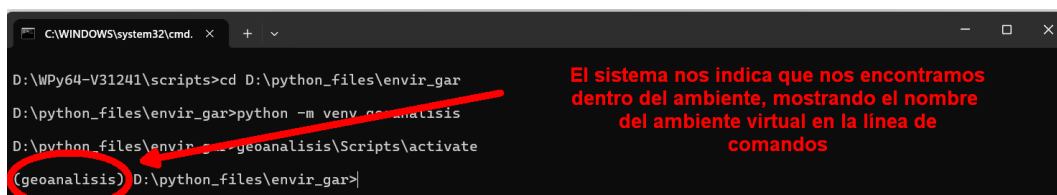


Figura 9. Ambiente virtual activado.

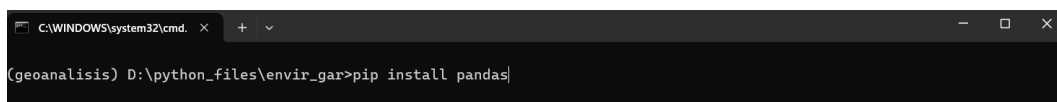
2.2. Instalación de paquetes en el ambiente virtual

Ya que el ambiente ha sido recién creado, si verificamos los paquetes que se encuentran instalados con el comando `pip freeze` o `pip list`, no se visualizará ningún paquete.

Recordemos que uno de los objetivos en los ambientes virtuales es prevenir conflictos entre versiones de paquetes, por tal razón, inicialmente, el ambiente no cuenta con ninguno y es por ello, debemos instalar los paquetes que requiere nuestro proyecto de Python. La instalación se realiza usando el comando `pip`. Te recomiendo que consultes las páginas oficiales del paquete para conocer las instrucciones que debes seguir para la instalación de dicho paquete.

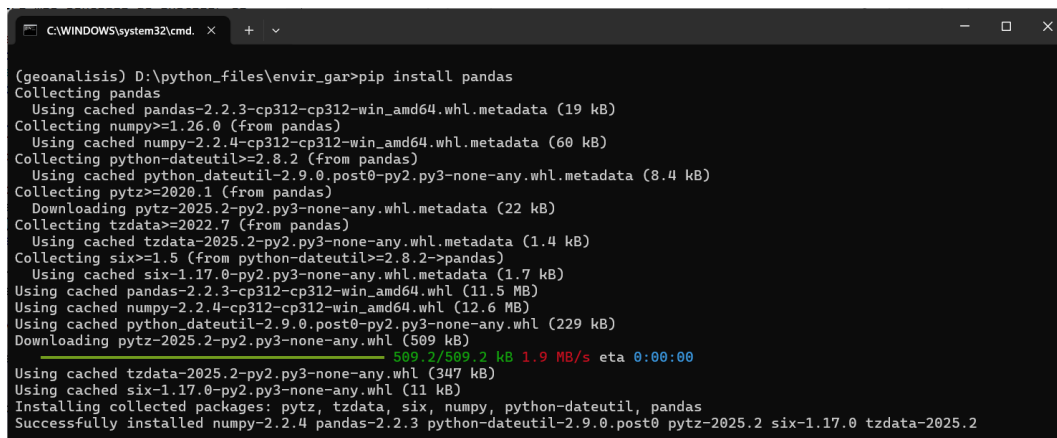
Para este ejemplo, vamos a instalar el paquete de Pandas en nuestro ambiente. Si consultamos la [página web de instalación de pandas](#) nos indican que la manera más sencilla de instalar el paquete es por medio de la instrucción (figuras 10 y 11):

```
pip install pandas
```



```
C:\WINDOWS\system32\cmd. x + v
(gеоanalysis) D:\python_files\envir_gar>pip install pandas|
```

Figura 10. Instalando el paquete de Pandas en el ambiente virtual activado.

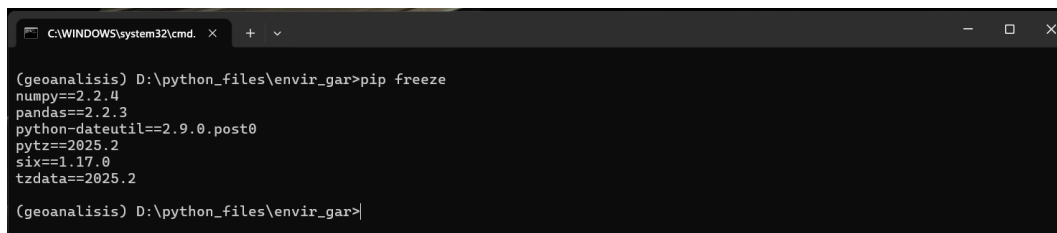


```
C:\WINDOWS\system32\cmd. x + v
(geoanalysis) D:\python_files\envir_gar>pip install pandas
Collecting pandas
  Using cached pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
Collecting numpy>=1.26.0 (from pandas)
  Using cached numpy-2.2.4-cp312-cp312-win_amd64.whl.metadata (60 kB)
Collecting python-dateutil>=2.8.2 (from pandas)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.8.2->pandas)
  Using cached six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Using cached pandas-2.2.3-cp312-cp312-win_amd64.whl (11.5 MB)
Using cached numpy-2.2.4-cp312-cp312-win_amd64.whl (12.6 MB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
509.2/509.2 kB 1.9 MB/s eta 0:00:00
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-dateutil, pandas
Successfully installed numpy-2.2.4 pandas-2.2.3 python-dateutil-2.9.0.post0 pytz-2025.2 six-1.17.0 tzdata-2025.2
```

Figura 11. Instalación finalizada del paquete.

De hecho en la página web antes mencionada, nos recomiendan que la instalación del paquete se realice en ambientes virtuales.

Si verificamos de nuevo que paquetes se encuentran instalados en el ambiente, usamos el comando `pip freeze` y obtendremos un resultado similar al mostrado en la figura 12.



```
C:\WINDOWS\system32\cmd. x + v
(geoanalysis) D:\python_files\envir_gar>pip freeze
numpy==2.2.4
pandas==2.2.3
python-dateutil==2.9.0.post0
pytz==2025.2
six==1.17.0
tzdata==2025.2
(geoanalysis) D:\python_files\envir_gar>
```

Figura 12. Paquetes instalados en el ambiente virtual.

Puede existir el caso donde sea necesario compartir tus programas y estos se encuentran en un ambiente con determinados paquetes. Cuando ocurra esta situación, es recomendable hacer un archivo de requisitos para la instalación de los paquetes.

En la ventana de comandos del ambiente activo, escribimos la siguiente línea de comandos para generar el archivo de requisitos:

```
pip freeze > listapaquetes.txt
```

A las personas que les compartes la lista de los paquetes que tiene tu ambiente, les indicarás que para realizar la instalación con pip deben usar el parámetro `-r` que precede al nombre del archivo de requisitos, es decir:

```
pip install -r listapaquetes.txt
```

3. Desactivación del ambiente virtual

Cuando hemos concluido la programación, desarrollo o ejecución del proyecto relacionado al ambiente, se recomienda que se desactive el ambiente antes de cerrar la ventana de comandos. Para desactivar el ambiente virtual, en la línea de comandos escribimos la siguiente instrucción:

```
deactivate
```

Cuando se desactiva el ambiente virtual, el nombre del éste desaparecerá de la línea de comandos (figura 13).



Figura 13. Desactivando el ambiente virtual.

4. Ambientes virtuales en la distribución de Python Conda (Anaconda)

Tal como se mencionó anteriormente, se recomienda que tu ubicas dentro de la carpeta donde almacenarás todos tus ambientes y crear subfolder para cada uno de ellos.

En una ventana de Anaconda Prompt, usamos la siguiente instrucción para crear ambientes virtuales con la distribución de Conda:

```
conda create -n nombre_ambiente
```

Y para activar el ambiente, escribimos en la línea de comandos:

```
conda activate nombre_ambiente
```

Para desactivar el ambiente, deberás emplear el siguiente comando:

```
conda deactivate
```

Conda permite realizar la creación de un ambiente, así como la instalación de paquetes mediante un archivo yaml (*.yaml). En la primera línea se proporciona el nombre del ambiente, en la segunda línea escribimos la palabra "dependencies" y en las siguientes líneas se escriben los nombres de los paquetes que se instalarán. Para nuestro ejemplo, la estructura del este archivo es la siguiente:

```
name: nombre_ambiente
dependencies:
  - nombre_paquete
```

Un archivo yaml puede ser creado en cualquier editor de texto, solamente debes de guardar el archivo con la extensión .yaml. Para nuestro ejemplo, el archivo tendrá el nombre de geoana.yaml y el contenido del archivo *.yaml sería:

```
name: geoanalisis
dependencies:
  - pandas
```

Usando este archivo, en Conda Prompt creamos el ambiente con la siguiente instrucción:

```
conda env create -f geoana.yaml
```

Te recomiendo que consultes las indicaciones que se proporcionan en la pagina web oficial relacionada al tema, que ofrece Conda a los usuarios de esta distribución. La dirección web es:

<https://docs.conda.io/projects/conda/en/stable/user-guide/tasks/manage-environments.html>

5. Ambientes virtuales en PyCharm

Debido a la interfase gráfica de usuario que ofrece PyCharm, la creación de ambientes se realiza en el momento que se crea un proyecto nuevo y por tal razón, no es necesario que proporciones de manera manual alguna instrucción o comando para generar un ambiente virtual.

En la ventana que se abre para generar el proyecto (figura 14), 1) en "Location" debes proporcionar la ruta de acceso donde se ubicara la carpeta del proyecto en tu disco duro; 2) en el tipo de intérprete, deberás presionar el botón "Project venv" y 3) en "Python version" indicas la ruta de acceso del archivo donde se encuentra ejecutable de Python.

Si PyCharm detecta que en la computadora existe la distribución de Anaconda instalada, entonces en el tipo de intérprete aparece el botón de "Base Conda". Si eliges esta opción, en "Path to Conda" por default PyCharm coloca la ruta de acceso del intérprete de Python de Anaconda. Identifica que en esta opción no se crea ningún ambiente.

Mediante el botón "Custom environment" (figura 15), PyCharm te permite la manera de indicar cómo definir el ambiente. Cuando oprimas el botón de personalizar el ambiente, en "Environment" aparecen dos opciones: en la primera opción se genera un nuevo ("Generate new") y en la segunda, se indica el uso de un ambiente, previamente creado "Select existing".

Si creamos un nuevo ambiente, PyCharm nos pide que indiquemos a partir de que tipo ("Type") vamos a generar el archivo, ya sea con el uso de venv o Conda (figura 15). Si deseamos usar venv, de la lista de tipos seleccionamos "Virtualenv" (figura 16). Debemos indicar el intérprete base de Python para nuestro ambiente (ver figura 1), así como la ruta de acceso donde se encuentra el archivo ejecutable de Python. En la ubicación del ambiente puedes reconocer que es la misma ruta de acceso del proyecto, solamente que se agrega la carpeta "venv". También, la interfase gráfica te ofrece: a) la posibilidad de los paquetes del intérprete base se hereden al nuestro ambiente y b) que el ambiente recién creado se encuentre disponible para todos los proyectos.

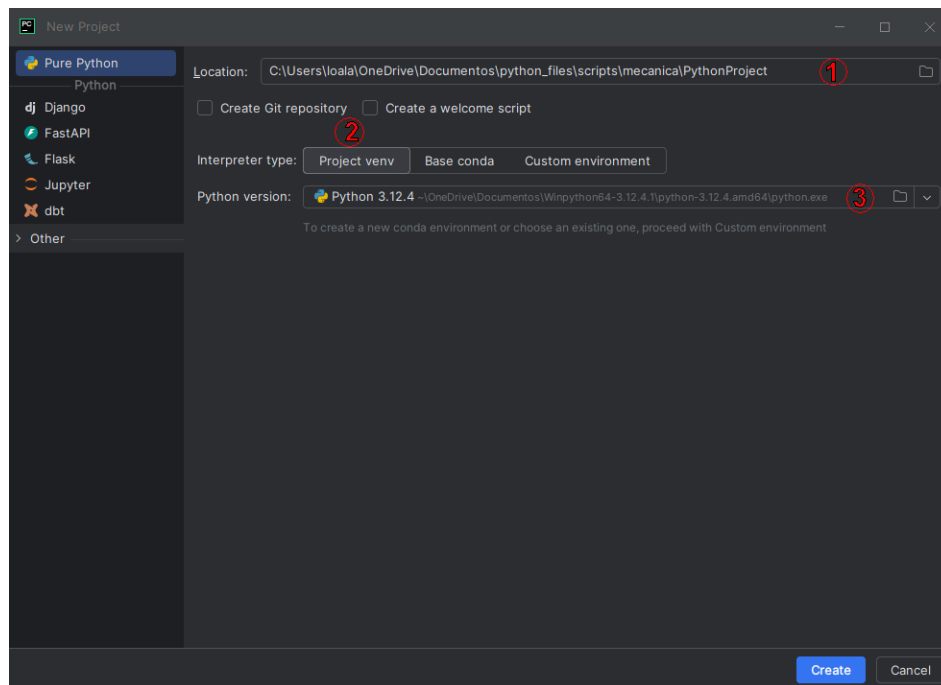


Figura 14. Ventana de PyCharm para generar un nuevo proyecto.

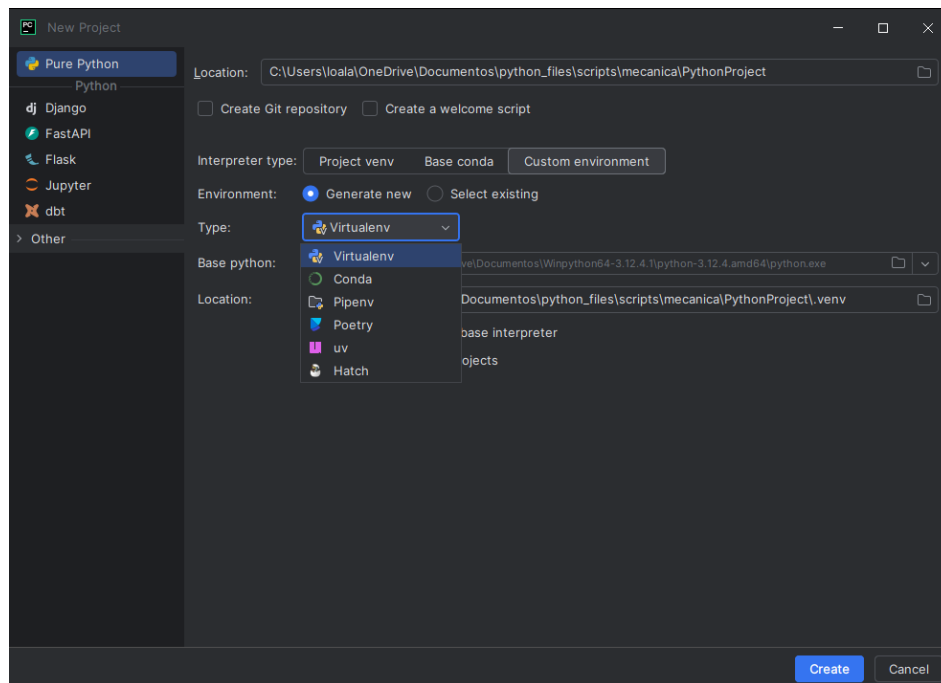


Figura 15. Ventana de PyCharm para generar un nuevo proyecto, personalizando la creación del ambiente.

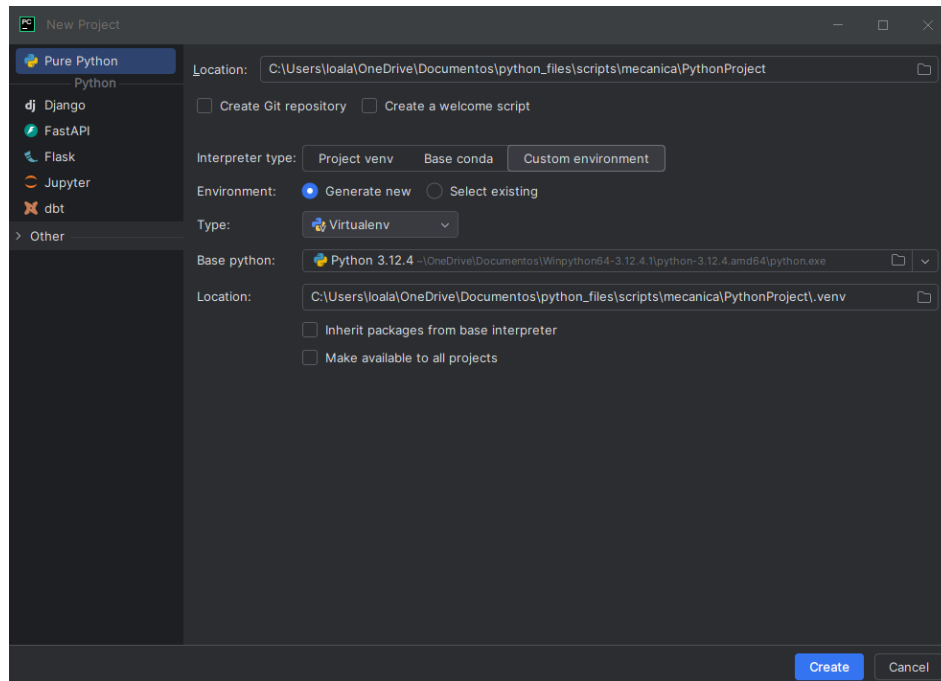


Figura 16. Ventana de PyCharm para personalizar el proceso de creación de un ambiente.

Por otra parte, al seleccionar un ambiente existente, debemos indicar la distribución del intérprete de Python, ya sea Python o Conda. Si elegimos Python, debemos indicar la ruta de acceso del archivo ejecutable de Python, el cual se localiza en la subcarpeta `Scripts` del ambiente (figura 7).

6. Manejo de ambientes virtuales en Spyder

En Spyder no es posible crear ambientes virtuales pero mediante el intérprete de Python, que se encuentra en la carpeta de `Scripts`, puedes hacer uso de los paquetes instalados.

En la barra de menús de Spyder, selecciona el menu Herramientas y posteriormente, Preferencias; en la ventana que se abre, en el panel izquierdo selecciona y abre Intérprete de Python. Activa la opción de "Usar el siguiente intérprete" y proporciona la ruta de acceso del archivo ejecutable de Python de la subcarpeta `Scripts` del ambiente. Finalizas, oprimiendo el botón de OK (figura 17).

A continuación, de la barra de menús elige Terminales y de las opciones que se despliegan, deberás elegir "Reiniciar el núcleo" (figura 18).

Si al efectuar las indicaciones anteriores, en la consola te aparece un mensaje de error mencionando que no es posible usar el núcleo porque no se encuentra el paquete "spyder-kernels", cierra Spyder y en la ventana de comandos de Python (Windows), activa el ambiente y procede a instalar el módulo que falta con la siguiente instrucción:

```
pip install spyder-kernels==2.5.*
```

Abre de nuevo Spyder, verifica que el intérprete de Python corresponda al ambiente y para cerciorarte de la disponibilidad de los paquetes, visualiza la lista de módulos con `pip freeze`.

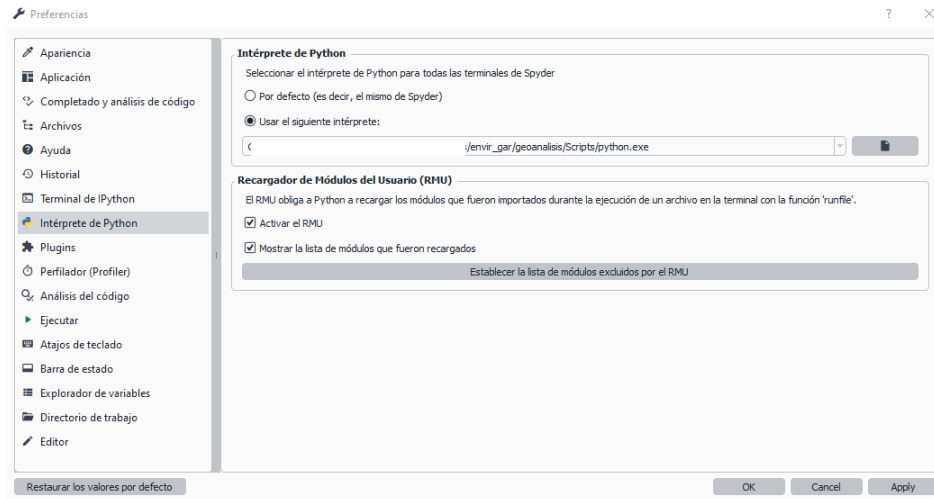


Figura 17. Ventana de Spyder para indicar el intérprete de Python.

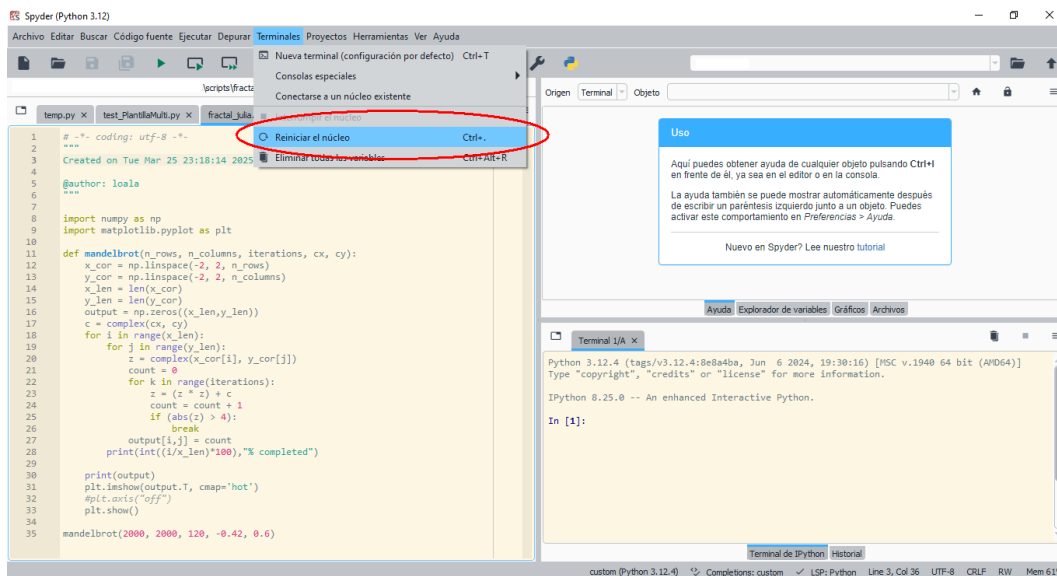


Figura 18. Reiniciando el núcleo de la consola de Spyder.

7. Manejo de ambientes virtuales en Visual Studio Code, (VSC)

De manera similar que en PyCharm, VSC ofrece usar virtualenv de Python y Conda para originar ambientes. Como primer paso vamos a establecer el "workspace" que requiere VSC para crear un ambiente, es por ello que en la barra de menús, elegimos el menú "File" y seleccionamos "Open Folder"; como resultado, la interfase abre una ventana donde ubicaremos la carpeta donde se creará el ambiente. Como ejemplo para este apartado, se creó una carpeta con el nombre de geoanalisisVSC y dicha carpeta fue la elegida como espacio de trabajo.

Una vez elegido la carpeta que almacenará nuestro ambiente, en la barra de menús desplegamos el menú de "View" y en la "Command Palette" buscamos y elegimos el comando "Python: Create Environment" (figura 19a). A continuación se desplegará una lista donde aparece la opción Venv y Conda (si éste se encuentra instalado en la computadora) (figura 19b). Al seleccionar Venv, visualizamos una lista donde se identifican los

intérpretes que podemos usar como base crear el nuevo ambiente (figura 20a). Cuando elegimos intérprete de nuestra elección, VSC comienza a generar de manera automática el ambiente, apareciendo en la parte inferior derecha una ventana donde nos notifican que el ambiente está siendo creado (figura 20b).

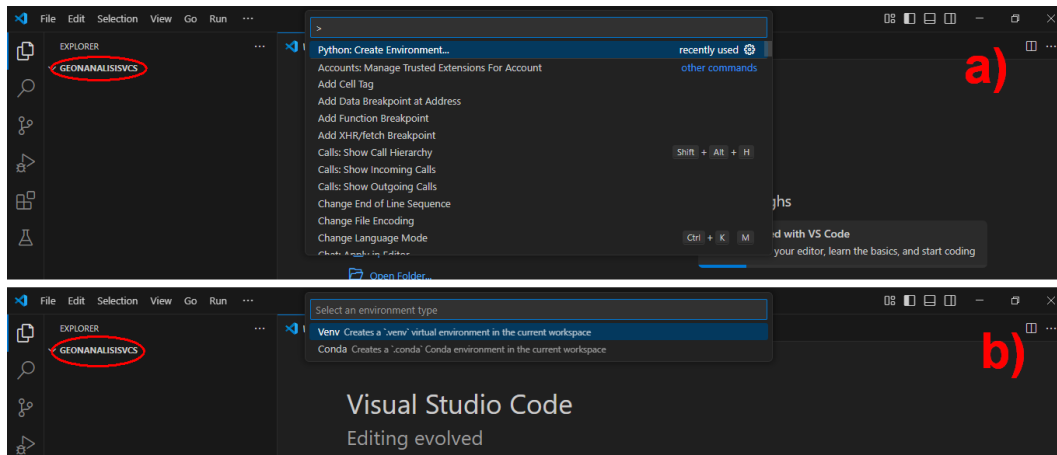


Figura 19. a) Paleta de Comandos y búsqueda de comando de creación de ambientes; b) Lista de intérpretes de Python para generar el ambiente.

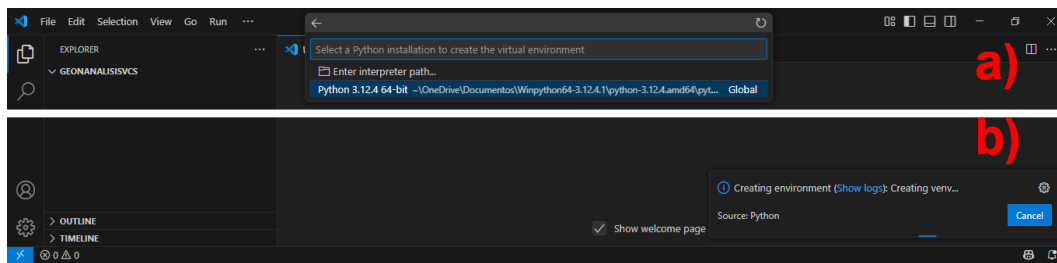


Figura 20. a) Paleta de Comandos y búsqueda de comando de creación de ambientes; b) Lista de intérpretes de Python para generar el ambiente.

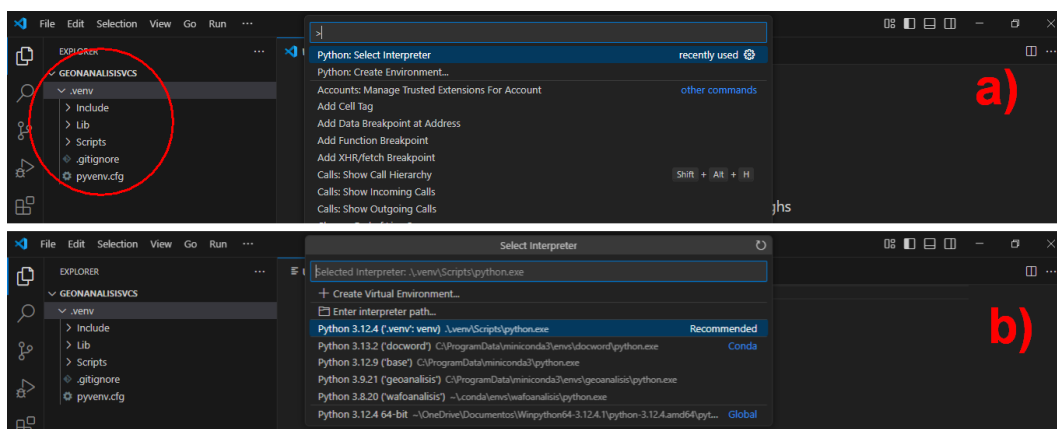


Figura 21. a) Seleccionando el intérprete del ambiente, también se resaltan las carpetas recién creadas; b) Elijiendo la opción de búsqueda manual de intérprete de Python.

En la figura 21a se muestra que VSC generó las carpetas Include, Lib, Scripts antes mencionadas. El siguiente paso a seguir será establecer el intérprete del ambiente, que en este caso se ubica en la carpeta de Scripts.

De nuevo, abrimos la "Command Palette" y buscamos el comando "Python: select interpreter"; si tenemos más ambientes, una lista con los intérpretes se abrirá, pero en nuestro caso seleccionamos la opción "Enter interpreter path" (figura 21b) y se desplegará el comando "Find... Browse your file system to find a Python interpreter" lo seleccionamos y se abrirá un ventana del sistema para la carpeta con el archivo ejecutable del intérprete; para nuestro ejemplo, este archivo se encontrará en la carpeta .venv, subcarpeta Scripts, elegimos python.exe y se oprime el botón de "Select interpreter".

Si abrimos un archivo *.py en VSC, en la parte inferior derecha de la interfase se identifica el intérprete que fue seleccionado en el paso anterior (figura 22).

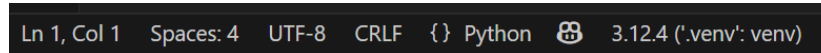


Figura 22. En la barra de estado de VCS se identifica el intérprete Python que se encuentra en uso.

8. Comentario final

En esta nota se expone lo que es un ambiente virtual en Python y la manera en la cual se pueden implementar usando diferentes alternativas. Debido a la problemática que se presenta por la dependencia de ciertas versiones de paquetes que requieren los programas, es muy importante para el programador usar los ambientes para eludir conflictos originados por el manejo de diferentes versiones de módulos.

Si deseas tener información técnica acerca de cómo funcionan los ambientes virtuales te recomiendo consultes el capítulo 2 del libro de Jolowicz (2024), donde encontrarás con más detalle la administración que realiza Python para el manejo de los ambientes.

Referencias

Anaconda, I. (2017). Managing environments, user guide. <https://docs.conda.io/projects/conda/en/stable/user-guide/tasks/manage-environments.html>. Recuperado el 17 de abril del 2025.

Jolowicz, C. (2024). *Hypermodern Python tooling*. O'Reilly Media, Inc., Sebastopol, CA. 268 p.

Mayer, C. (2012). PEP 405 – Python virtual environments. <https://peps.python.org/pep-0405/>. Recuperado el 18 de abril del 2025.

Microsoft (2025). Python environments in VS code. <https://code.visualstudio.com/docs/python/environments>. Recuperado 18 de abril del 2025.

Rohowsky, M. (2023). Pycharm virtual environments (venv) explained! 10 min updated 2023. [video]. youtube. https://youtu.be/2P30W3TN4nI?si=y_1fJbyXGPs0fUZE.

The Python Software, F. (2025a). venv — creation of virtual environments, Python documentation. <https://docs.python.org/3/library/venv.html>. Recuperado el 16 de abril del 2025.

The Python Software, F. (2025b). Virtual environments and packages, Python tutorial. <https://docs.python.org/3/tutorial/venv.html>. Recuperado el 16 de abril del 2025.

Ambientes virtuales en Python © 2025 por Gabriel Ruiz Martínez tiene la licencia CC BY-NC-SA 4.0. Para ver una copia de esta licencia, visita <https://creativecommons.org/licenses/by-nc-sa/4.0/>

Los ejemplos fueron desarrollados con WinPython 3.12.4.1, MiniConda 25.3.0, Spyder 5.5.5, PyCharm Pro 2025.1, Visual Studio Code 1.99.3. La nota fue creada en \LaTeX (TeX Live 2023, TexStudio 4.8.6, compilado con PDFLaTeX) usando la fuente Source Sans Pro; gráficos con Inkscape v.1.4.