1. Instalar Python y VSCode

1.1. Instalación de Python

Python es un lenguaje de programación/*Scripting* muy popular y utilizando en la industria para automatizar procesos, crear aplicaciones de escritorio, web y muchas otras cosas más. Es Software Libre y mantenido por su comunidad, la cual está formada por un comité y grupo de usuarios que toman decisiones sobre sus aspectos y funcionalidades. Para más información puede ingresar en la página de PyAr (*Python Argentina)*.

1.1.1. GNU/Linux

Lo más probable es que ya tenga Python instalado en sus sistema, ya que innumerables herramientas lo utilizan. Pero es importante asegurarse de que se encuentre el paquete `python3-venv` en las distribuciones basadas en debian o `python3-virtualenv` en las basadas en fedora. Si no es así, su instalación es muy sencilla:

Instalación de virtualenv en debain/ubuntu/mint:

```
sudo apt install python3-venv -y
```

Instalación de virtualenv en fedora/redhat/centos:

```
sudo dnf install python3-virtualenv
```

1.1.2. MS Windows 10/11

Para instalar la última versión de Python en MS Windows se debe descargar de la página oficial `https://python.org/` el instalador y ejecutarlo en modo administrador. A continuación se facilita un video con el proceso completo.

`https://youtu.be/WxdlqFeb5IY?si=MJkfBq3BuWuqWhBh`

1.2. Instalar VSCode

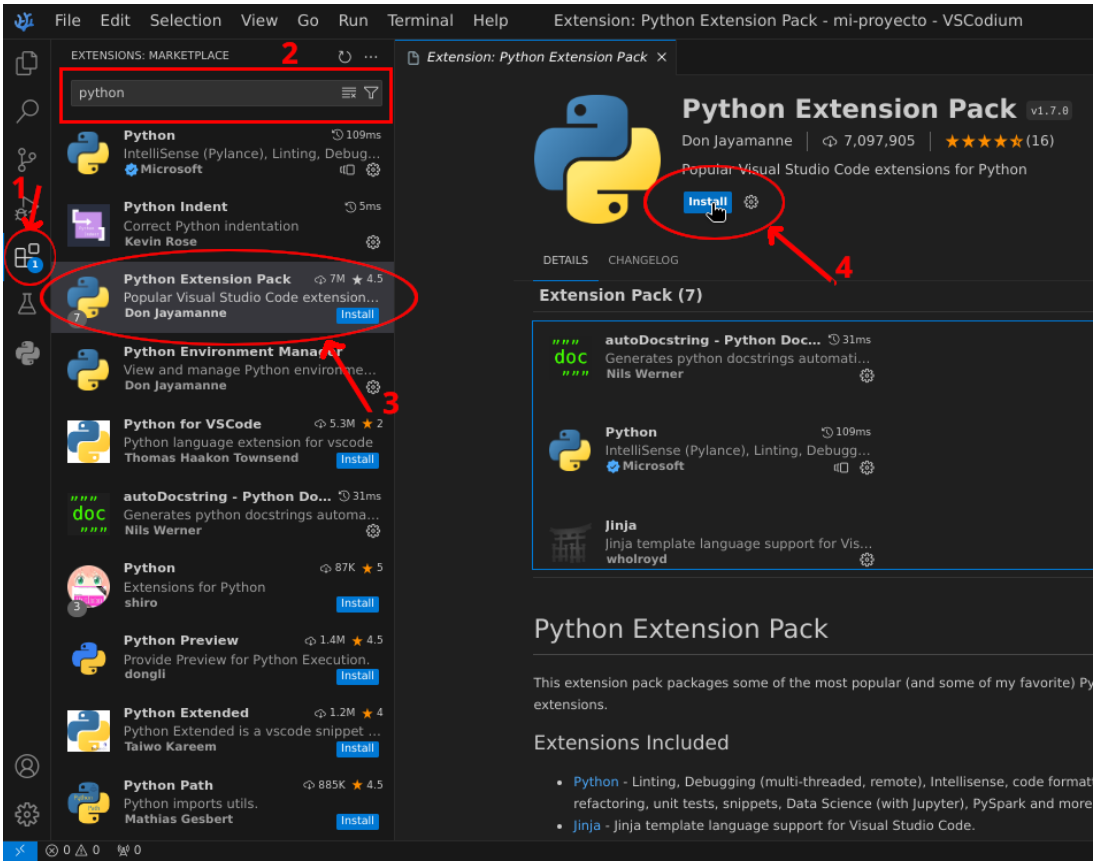
Visual Studio Code (VSCode o simplemente *code*) es un editor de texto extensible que se ha popularizado en los últimos años por su versatilidad. Sus herramientas logran un

aumento en la productividad de los desarrolladores y su extensibilidad la facilidad de integrar diferentes herramientas y lenguajes.

Su instalación es muy sencilla, incluso con versiones portables, para todos los sistemas operativos *mainstream* (GNU/Linux, Windows y Mac). Estos instaladores o paquetes instalables se pueden obtener de su página oficial `https://code.visualstudio.com/download`.

1.2.1. Instalar Extensiones para Python

Una vez instalado el VSCode, es recomendable instalar las extensiones de desarrollo para Python. Esto se hace haciendo clic en la barra lateral izquierda sobre el icono de extensiones y buscando `Python Extension Pack`. El cual instala un paquete de extensiones para trabajar.



1. Clic en `'extentions'`
2. Buscar `'python'`

3. Clic en **'Python Extension Pack'**

4. Clic en **'install'**

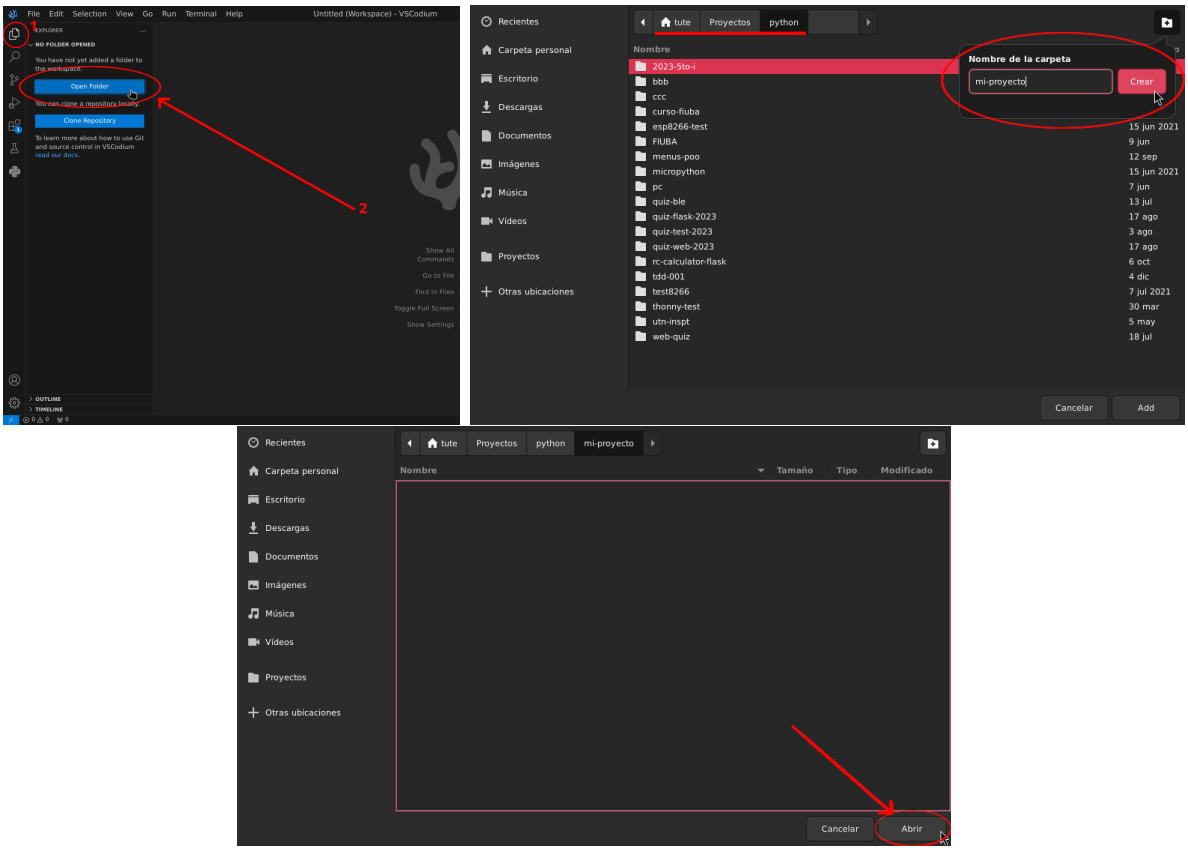
2. Crear un proyecto nuevo

Acá se expondrá una metodología de trabajo básico, donde por simplificación se trabajará cada proyecto en una carpeta y colocando los scripts (`.py`) en ella.

Es recomendable tener una carpeta general que contenga todos los proyectos a realizar, para su fácil ubicación. Es decir una carpeta que contenga a las demás.

2.1. Crear una carpeta para contener el proyecto

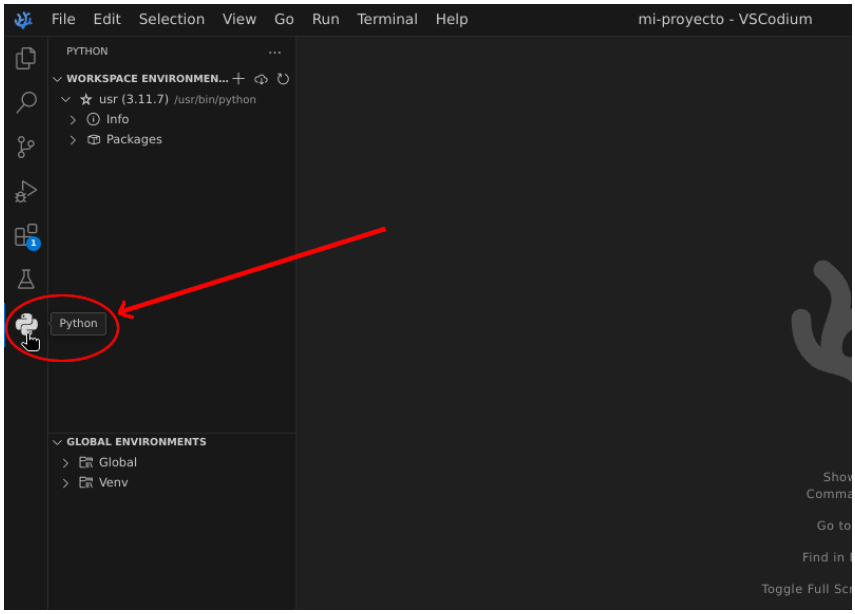
Para empezar, *abriremos* una carpeta en el VSCode. Ubicaremos la carpeta contenedora de proyectos y allí crearemos una nueva para contener nuestro primer proyecto.



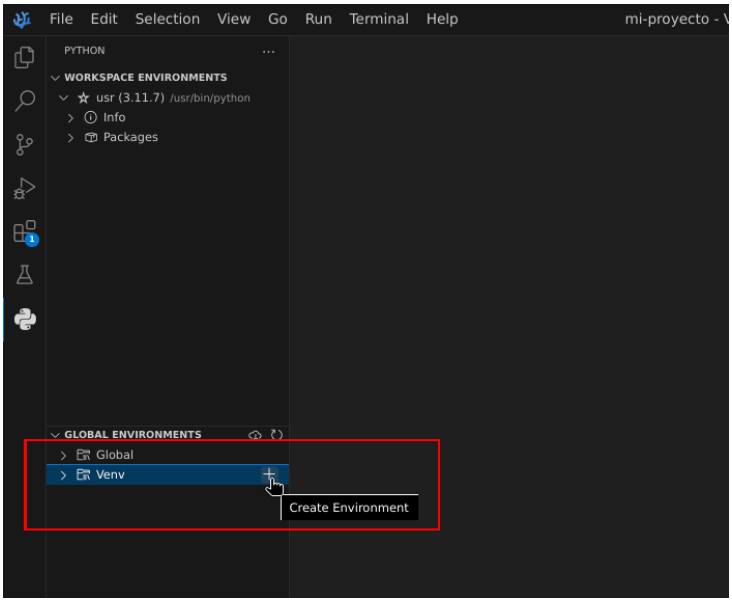
En este caso dentro de la carpeta personal del usuario se crea la carpeta `mi-proyecto` dentro de `Proyectos/python`. Luego de creada se accede a la misma y se abre.

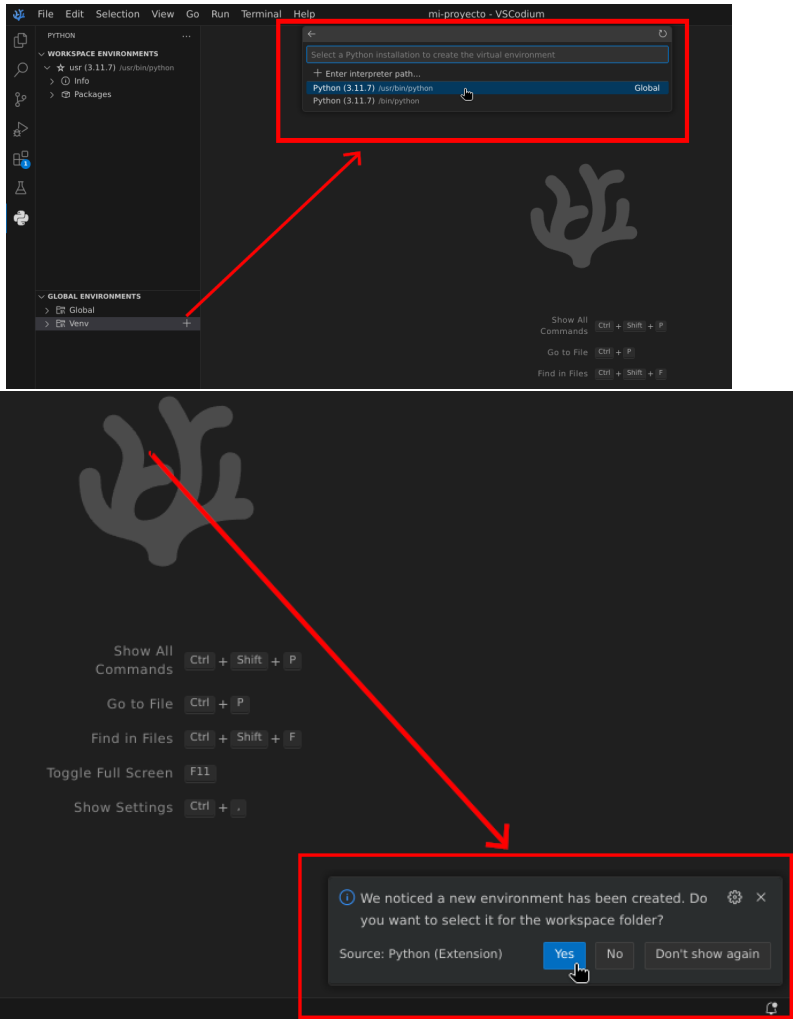
2.2. Crear un entorno virtual

Es **sumamente recomendable** trabajar cada proyecto en su propio "entorno virtual" de Python. Para poder crear uno, debemos dirigirnos al icono de la barra lateral izquierda de Python.



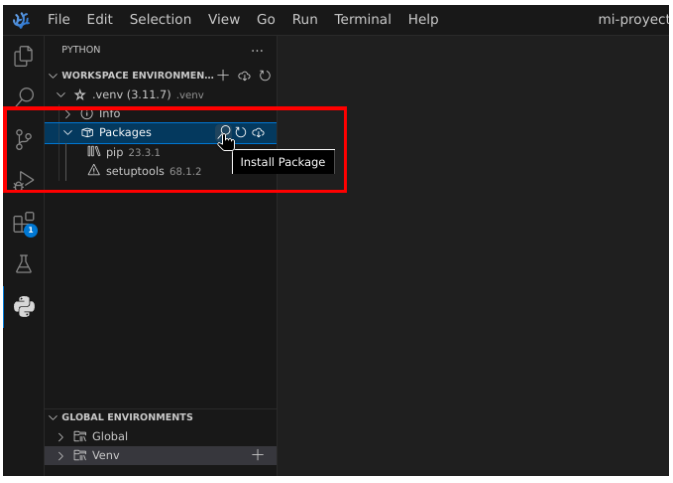
Allí debemos ir a la sección `'GLOBAL ENVIROMENTS'` y hacer clic en el `'+'` de la carpeta `Venv`. Luego seleccionar la versión de Python a utilizar para el entorno virtual y por último confirmar que deseamos utilizarlo para nuestro espacio de trabajo actual (*Workspace*).

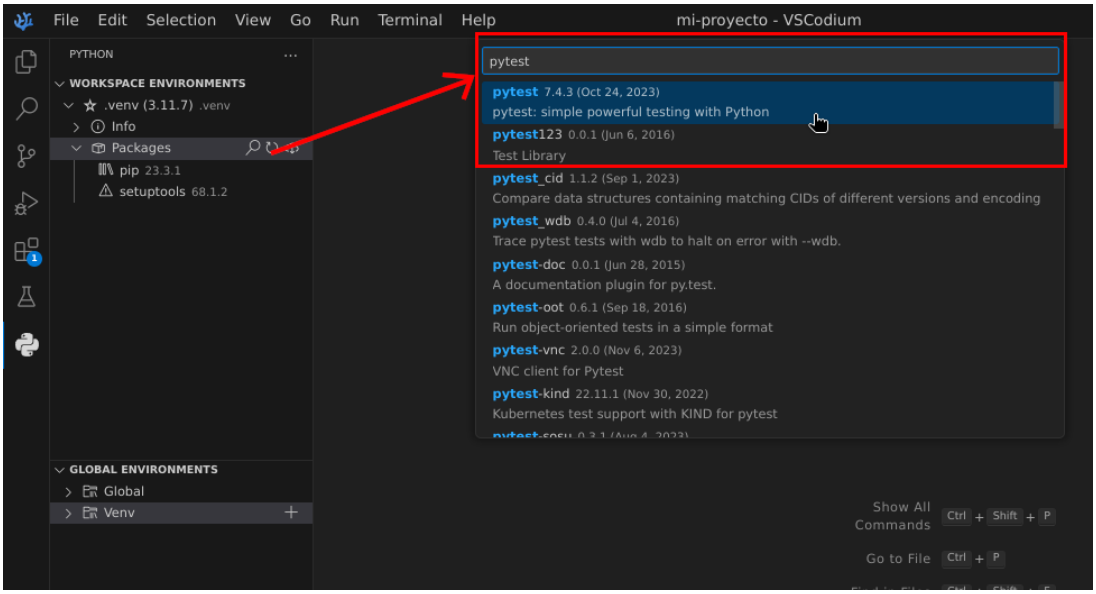




2.3. Instalar `pytest`

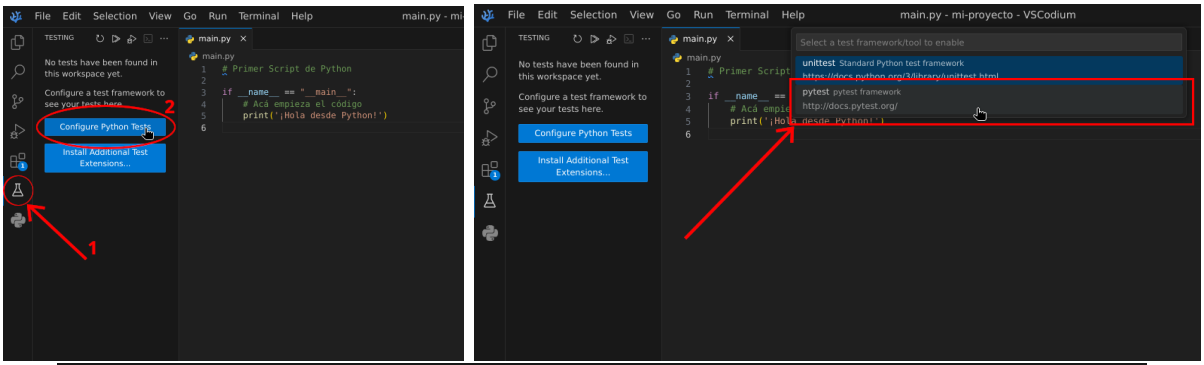
A continuación se instalará la herramienta de *testing* `pytest` dentro del entorno virtual recientemente creado. Para ello, debemos hacer clic en la lupa para buscar el paquete y escribir el nombre del mismo en el cuadro de búsqueda emergente.

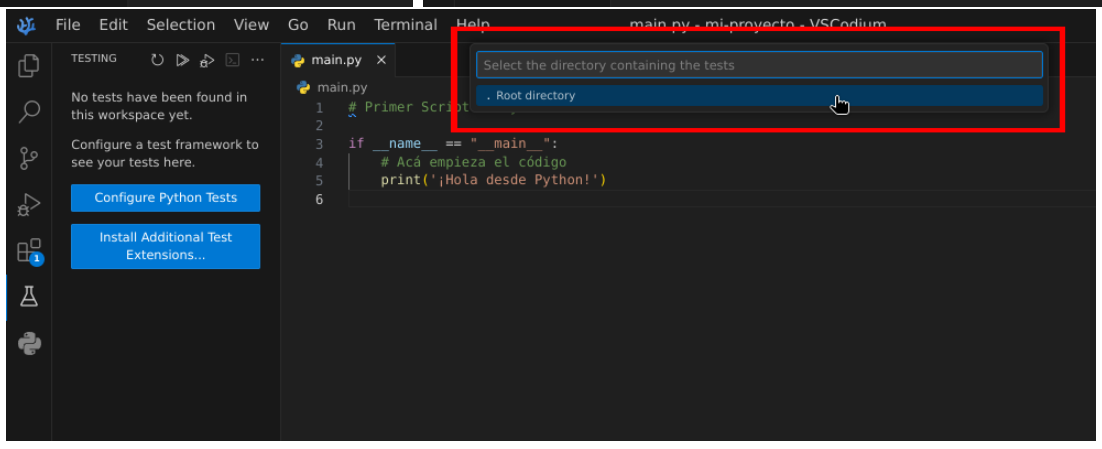




2.4. Configurar `pytest`

Para poder ejecutar las pruebas unitarias debemos configurarlo en el VSCode. Para ello se ingresa en la barra de herramienta de la izquierda en el apartado de *testing* y allí hacer clic para configurar.





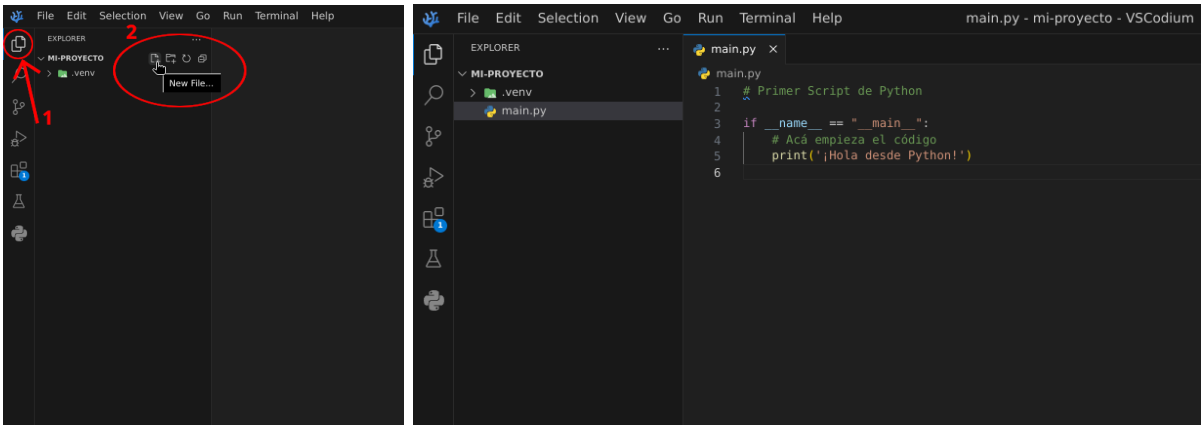
3. Probar el entorno

Para verificar que el entorno está listo para su utilización, se propone hacer un *script* y un *test* y ejecutarlos.

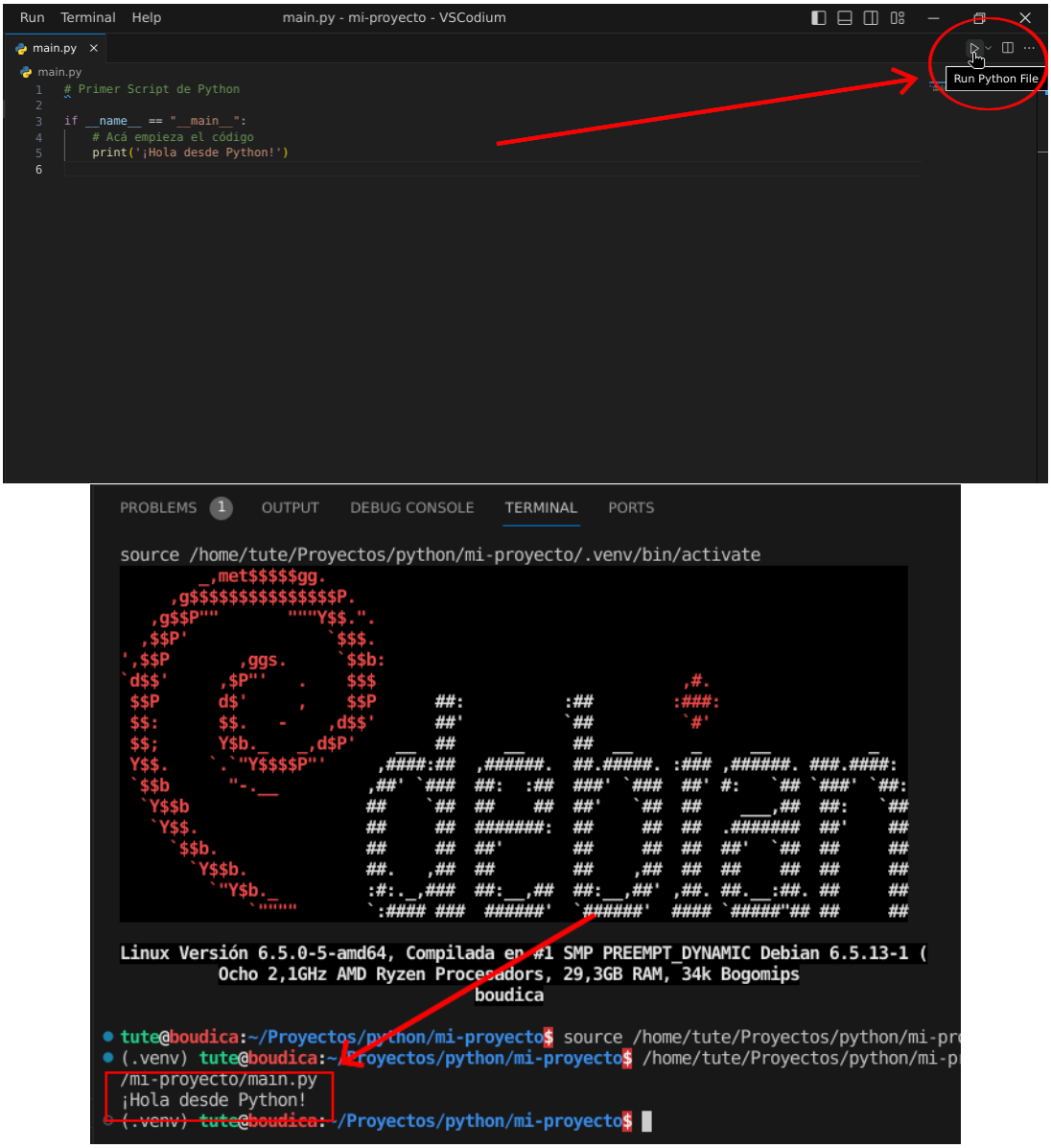
3.1. Crear y ejecutar un script de Python

Volviendo a la estructura de archivos de nuestro proyecto, accediendo al icono superior de la barra lateral izquierda, crearemos un *script* llamado `main.py` y en este ingresará el siguiente código de prueba.

```
1  # Primer Script de Python
2
3  if __name__ == "__main__":
4      # Acá empieza el código
5      print("¡Hola desde Python!")
```



Para poder ejecutar el *script* disponemos de un icono de *play* en la esquina superior derecha del editor. Si se ejecuta con éxito, se verá en la parte inferior la salida de la terminal con el texto `¡Hola desde Python!`.



3.2. Crear una prueba y ejecutarla

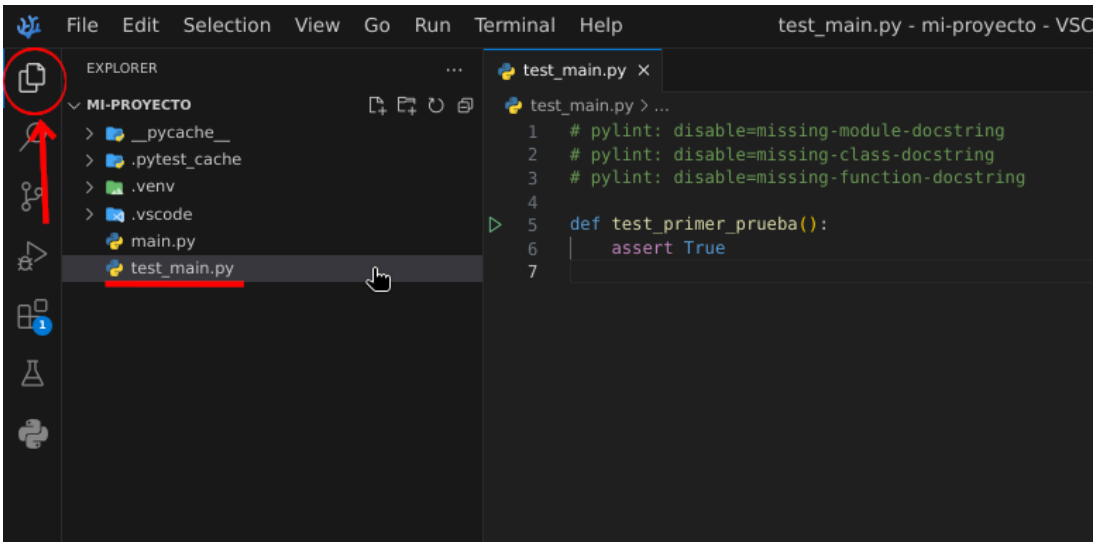
Las pruebas deben escribirse en un archivo que comience con la palabra `'test_'` seguida del nombre del modulo o clase objetivo del test. Por ejemplo, aquí se llamará `test_main.py`.

Los tests son funciones que deben verificar alguna condición, habitualmente con el comando `assert`, esto se estudiará a fondo en el próximo apunte que explica la metodología **TDD** (*Test-Driven Development*). En estos momentos simplemente se escribirá un código anémico con un test artificial que siempre da correcto. Los test unitarios son funciones que deben empezar con la palabra `'test_'`. Por una cuestión de comodidad se inhabilitarán

algunos *warnings* para evitar mensajes que en este momento no son importantes.

```python
# pylint: disable=missing-module-docstring
# pylint: disable=missing-class-docstring
# pylint: disable=missing-function-docstring

def test_primer_prueba():
    assert True
```



Para ejecutar las pruebas unitarias, total o parcialmente, se dirigirá en la barrá de herramientas lateral izquierda al apartado de *testing*, y allí puede darle *play* a el proyecto, a algún archivo de testing por separado o a un test único. Debemos ver como cada test obtiene una tilde verde si es que la prueba paso positivamente.