DAY – 30

9 September 2025

You can create a Python virtual environment in VS Code using either the built-in command in the editor or the integrated terminal. Using a project-specific environment is a best practice for managing dependencies.

Method 1: Using the Command Palette (Recommended)

This method is the simplest as VS Code automatically handles environment creation and selection.

- **Open the Command Palette:** Press Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS).
- **Run the Create Environment command:** Type "Python: Create Environment" in the command palette search bar and select it.
- **Choose the environment type:** Select Venv (or Conda if you are using an Anaconda distribution).
- **Select an interpreter:** Choose the base Python interpreter you want to use for the new environment from the list.
- **Wait for creation:** A notification will show the progress. The environment folder (typically named .venv or env) will appear in your workspace.
- **Confirmation:** VS Code will automatically detect and select this new environment for your workspace. The selected interpreter version will be visible in the bottom right of the status bar.

Method 2: Using the Integrated Terminal

You can also create the environment manually in the terminal and then select it in VS Code.

- **Open the Terminal:** Go to the top menu and select Terminal > New Terminal, or use the shortcut Ctrl+Shift+ (backtick).
- **Navigate to your project folder:** Use cd commands to ensure you are in the root directory of your project.

- **Run the creation command:**

On Windows: python -m venv .venv

On macOS/Linux: python3 -m venv .venv
This creates a directory named .venv containing the environment files.

- **Activate the environment in the current terminal (optional for VS Code, but useful for command-line use):**

On Windows (Command Prompt): .\\.venv\\Scripts\\activate

On Windows (PowerShell): .\\.venv\\Scripts\\Activate.ps1

On macOS/Linux: source .venv/bin/activate
Once activated, you'll see the environment name (e.g., (.venv)) at the beginning of your terminal prompt.

- **Select the interpreter in VS Code:** Open the Command Palette (Ctrl+Shift+P) and run the Python: Select Interpreter command. Select the path to the Python executable within your new .venv folder (e.g., .venv/Scripts/python.exe on Windows or .venv/bin/python on Linux/macOS).

Any new terminal you open in VS Code will now automatically use the selected virtual environment.

Python environments in VS Code

An "environment" in Python is the context in which a Python program runs that consists of an interpreter and any number of installed packages.

Types of Python environments

Global environments

By default, any Python interpreter installed runs in its own global environment. For example, if you just run python, python3, or py at a new terminal (depending on how you installed Python), you're running in that interpreter's global environment. Any packages that you install or uninstall affect the global environment and all programs that you run within it.

Tip: In Python, it is best practice to create a workspace-specific environment, for example, by using a local environment.

Local environments

There are two types of environments that you can create for your workspace: virtual and conda. These environments allow you to install packages without affecting other environments, isolating your workspace's package installations.

Virtual environments

A virtual environment is a built-in way to create an environment. A virtual environment creates a folder that contains a copy (or symlink) to a specific interpreter. When you install packages into a virtual environment it will end up in this new folder, and thus isolated from other packages used by other workspaces.

Note: While it's possible to open a virtual environment folder as a workspace, doing so is not recommended and might cause issues with using the Python extension.

Conda environments

A conda environment is a Python environment that's managed using the conda package manager (see Getting started with conda).Choosing between conda and virtual environments depends on your packaging needs, team standards, etc.

Python environment tools

The following table lists the various tools involved with Python environments:

Expand table

Tool	Definition and Purpose
pip	The Python package manager that installs and updates packages. It's installed with Python 3.9+ by default (unless you are on a Debian-based OS; Install python3-pip in that case).
venv	Allows you to manage separate package installations for different projects and is installed with Python 3 by default (unless you are on a Debian-based OS; install python3-venv in that case)

conda	Installed with Miniconda. It can be used to manage both packages and virtual environments. Generally used for data science projects.