Setup Guide: VS Code + UV for Python

Jesús Villota Miranda

Data Science for Economics
CEMFI

Welcome to the setup guide for the course "Data Science for Economics" at CEMFI! This guide will help you set up everything you need to run Python code on your computer using Visual Studio Code (VS Code) as your code editor and UV as your tool for managing dependencies (like libraries). **Important note**: You won't need to install Python separately because UV can handle that for you.

We'll cover the steps for Windows, macOS, and Linux where they differ. Follow the sections for your operating system (OS). If you're unsure what OS you have, check your computer's settings (e.g., on Windows, search for "About your PC"; on macOS, click the Apple menu > About This Mac).

Step 1: Install VS Code

VS Code is a free code editor where you'll write and run your Python code.

1. Go to the official website: `https://code.visualstudio.com/download`.
2. Download the installer for your OS (it will auto-detect, but select Windows, Mac, or Linux if needed).
3. Run the downloaded file and follow the on-screen instructions to install. Accept the defaults if you're unsure.
4. Once installed, open VS Code to make sure it launches. It should look like a blank window with a menu bar at the top.

Step 2: Install UV

UV is a fast tool for managing Python environments and dependencies. It doesn't require Python to be installed beforehand—it's a standalone program.

For Windows:

1. Open PowerShell (search for "PowerShell" in your start menu and run it as administrator by right-clicking > Run as administrator).
2. Copy and paste this command, then press Enter:

   ```
   powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
   ```

3. Follow any prompts. It may ask you to restart PowerShell or your computer.
4. To verify: Reopen PowerShell and run `uv --version`. You should see a version number (e.g., uv 0.x.x).

For macOS:

1. Open Terminal (press Command + Space, type "Terminal", and open it).
2. Copy and paste this command, then press Enter:

   ```
   curl -LsSf https://astral.sh/uv/install.sh | sh
   ```

3. If you don't have curl, use: `wget -qO- https://astral.sh/uv/install.sh | sh` (but curl is usually pre-installed).
4. Follow any prompts. It may add UV to your PATH (where commands are found).
5. To verify: In Terminal, run `uv --version`. You should see a version number.

For Linux:

1. Open your Terminal (search for "Terminal" in your applications menu).
2. Copy and paste this command, then press Enter:

   ```
   curl -LsSf https://astral.sh/uv/install.sh | sh
   ```

3. If you don't have curl, install it first (e.g., on Ubuntu: `sudo apt update && sudo apt install curl`).
4. Follow any prompts.
5. To verify: In Terminal, run `uv --version`. You should see a version number.

If you run into issues (e.g., permission errors), search online for "install UV on [your OS]" or ask for help.

Step 3: Install Python Using UV

Now that UV is installed, use it to download and install Python. We'll install Python 3.11 (a stable, recent version).

1. Open your command line tool:
   - Windows: PowerShell.
   - macOS/Linux: Terminal.
2. Run this command:

   ```
   uv python install 3.11
   ```

3. UV will download and install Python automatically (it may take a few minutes depending on your internet).
4. To verify: Run `uv python list`. You should see Python 3.11 listed as installed.
5. Optionally, to make "python" command work globally, run: `uv python install --default`. But for now, we'll use it through VS Code.

Step 4: Set Up VS Code for Python

1. Open VS Code.
2. Install the Python extension (this adds Python support):
   - Click the Extensions icon on the left sidebar (it looks like four squares).
   - Search for "Python" (by Microsoft).
   - Click Install on the top result.
3. Create a new folder for your code:
   - In VS Code, go to File > Open Folder... and create/select an empty folder on your computer (e.g., name it "PythonProjects").
4. Create a test Python file:
   - In the Explorer sidebar (left side), right-click inside the folder > New File.
   - Name it `hello.py`.
   - Open the file and type:

     ```
     print("Hello, World!")
     ```

   - Save it (Ctrl+S or Command+S).
5. Select the Python interpreter:
   - At the bottom right of VS Code, click on the Python version (it might say "Select Interpreter" if none is chosen).
   - Search for the one installed by UV (it should show something like "Python 3.11 (uv)").
   - If it doesn't appear, click "Enter interpreter path..." and browse to where UV installed it (usually in your home directory under `.local/uv/python/` or similar—UV will tell you the path after installation).
6. Run the code:
   - Right-click in the `hello.py` file > Run Python File in Terminal.
   - Or press Ctrl+Shift+P (Command+Shift+P on Mac), type "Python: Run Python File", and select it.
   - You should see "Hello, World!" printed in the terminal at the bottom.

Step 5: Using UV for Dependencies (Basics)

For now, this setup lets you run plain Python. In the course, we will need libraries (dependencies) like pandas for data manipulation or numpy for numerical computations. UV makes managing these easy by creating isolated environments for your projects, preventing conflicts between different projects.

Here are the key UV commands for dependency management. Run these in the terminal inside VS Code (go to Terminal > New Terminal) while in your project folder.

- `uv init`: Initializes a new UV-managed project in the current directory. This creates a `pyproject.toml` file (which stores project metadata and dependencies) and sets up a virtual environment automatically if needed. Use this to start a new project.
- `uv add <package>`: Adds a dependency (library) to your project. For example, `uv add pandas` installs the pandas library and adds it to your `pyproject.toml`. UV will download and install the package into your project's virtual environment. You can specify versions, e.g., `uv add pandas==2.0.0`.

- `uv remove <package>`: Removes a dependency from your project. For example, `uv remove pandas` uninstalls the package and updates `pyproject.toml`.
- `uv sync`: Synchronizes your environment with the dependencies listed in `pyproject.toml`. This installs any missing packages or updates them to match the specified versions. Run this after adding/removing dependencies or when pulling changes from a shared project.
- `uv lock`: Generates or updates a `uv.lock` file, which pins exact versions of dependencies for reproducibility. This ensures everyone on the team uses the same package versions.
- `uv run <command>`: Runs a command or script in the project's virtual environment. For example, `uv run python hello.py` executes your script with the project's Python and dependencies. This is useful for ensuring isolation.
- `uv venv`: Creates a new virtual environment manually (e.g., `uv venv .venv`). UV often handles this automatically with `uv init`, but you can use this for custom setups.
- `uv list`: Lists all installed packages in the current environment, showing versions and details.

These commands create and manage virtual environments automatically, so you don't need to activate/deactivate them manually like with older tools.

If you get stuck, search for error messages online or ask in the session. This minimal setup should take 10-20 minutes. See you in class!