Setting Up Python Natively On Machine

Totally fair to be confused here— `conda` and `venv` live in slightly different worlds, and Anaconda makes a lot of decisions for you behind the scenes.
Let's do three things:

1. Explain **when to use conda vs venv** (so you're not fighting your tools)
2. Show you **how to install "plain" Python** alongside Anaconda
3. Show you **how to create + use a venv** for our AI/agent projects

I'll keep it concrete and OS-agnostic, then you can follow the bits that match your machine.

1️⃣ Should you use `venv` instead of `conda`?

Short version:

- If you're doing **heavy scientific computing** (NumPy, SciPy, GPU, weird C libraries) and like Anaconda → **conda envs are fine**.
- If you're building **modern Python apps / agent frameworks / web backends** that use `pip`, `uv`, `poetry`, etc. → you're usually better off learning **venv + pip**.

For this **AI agents + tooling** track, I'd recommend:
✅ Keep Anaconda for your existing work, ✅ But for this new project, start using **"plain" Python + venv** so your environments look like what most open-source AI repos expect.
You can have both on the same machine with no conflict.

2️⃣ Installing "plain" Python alongside Anaconda

A. General idea

1. Install **Python from python.org** (or via your OS package manager)
2. Make sure that `python` / `python3` you use for this project is **the non-conda one**
3. Use that to create `venv` environments

I'll outline both Windows and macOS/Linux. Just follow the one that applies to you.

🪟 Windows: Python + venv next to Anaconda

1. **Download Python**

- Go to the official Python site (in your browser): Search: "Download Python" → choose the latest stable 3.x from python.org.
- Download the **Windows installer (64-bit)**.

2. **Run the installer**
   - On the first screen:
     - ✅ Check: **"Add Python 3.x to PATH"** (important!)
     - Then click **"Install Now"** (or "Customize installation" if you like, defaults are fine).
3. **Confirm it's installed**
   - Open **Command Prompt** or **PowerShell**, *not* the Anaconda Prompt.
   - Run:
     or
   - You should see something like `Python 3.12.x`.
4. **Avoid conda in this terminal**
   - Make sure your prompt does **not** show a conda env like `(base)`.
   - If it does, you're in an Anaconda-managed shell—close it and reopen a normal Command Prompt.

From there, you'll be using this `python` to create venvs.

🍎 macOS: Python + venv next to Anaconda

You have two good options:

Option 1: Official installer

1. Download macOS installer from python.org.
2. Run it → it'll install something like `/usr/local/bin/python3`.
3. In Terminal, run:
4. That `python3` is what you'll use with `-m venv`.

Option 2: Homebrew (my usual recommendation)

If you're comfortable with Homebrew:

```
brew install python
python3 --version
```

This gives you a nice, up-to-date Python at `/opt/homebrew/bin/python3` (Apple Silicon) or `/usr/local/bin/python3` (Intel).

🐧 Linux (Ubuntu / Debian-ish)

You probably already have Python 3, but make sure:

```
python3 --version
```

If needed, install:

```
sudo apt-get update
sudo apt-get install python3 python3-venv python3-pip
```

3️⃣ Using `venv` step-by-step (concrete workflow)

Let's say you want a project called `agent-playground`.

1. Create a project folder

```
mkdir agent-playground
cd agent-playground
```

2. Create a virtual environment

- On Windows:
- On macOS/Linux:

This creates a folder `.venv/` with an isolated Python + pip.

3. Activate the venv

- **Windows (Command Prompt):**
- **Windows (PowerShell):**
- **macOS / Linux:**

You'll know it worked if your prompt changes to something like:

```
(.venv) C:\Users\you\agent-playground>
```

or

```
(.venv) you@machine agent-playground %
```

4. Install packages with pip

Now any `pip install` goes *into this venv only*:

```
pip install pydantic aiohttp numpy
```

To see what's installed:

```
pip list
```

5. Freeze dependencies (for reproducibility)

```
pip freeze > requirements.txt
```

Later, if you (or someone else) want to recreate the environment:

```
python -m venv .venv
source .venv/bin/activate # or Windows equivalent
pip install -r requirements.txt
```

6. Deactivate when you're done

```
deactivate
```

4️⃣ VS Code + venv (so it "just works")

Since you're using VS Code, here's how to make it respect your venv:

1. Open **VS Code** in your project folder:
2. Install the **Python extension** (if you haven't already).
3. At the bottom status bar, click the **Python interpreter version** (or run "Python: Select Interpreter" in the Command Palette).
4. Choose the interpreter that looks like:
   - `./.venv/bin/python` (macOS/Linux)
   - `.venv\Scripts\python.exe` (Windows)

Now:

- Running code in VS Code uses that venv
- IntelliSense + linting sees the packages from that venv

5️⃣ Where this fits into Module 0.2

For our purposes in this course, your **Module 0.2 "success state"** looks like:

- You can:

    - create a new folder
    - `python -m venv .venv` inside it
    - activate it
    - `pip install` something
    - run a simple Python file in that venv

If you'd like, next step can be a **hands-on check**:
You tell me (in words or a terminal transcript):

- Your OS (Windows/macOS/Linux)
- What happens when you run:
    - `python --version` or `python3 --version`
    - `pip --version`

And I'll help you:

- verify you're using the non-conda Python
- write a minimal **"Module 0.2 test project"** with a venv and a `requirements.txt`.

Or, if you prefer, I can give you a tiny **exercise**:
"Create a venv, install `pydantic`, write a 10-line script using it, and send me the folder structure + code."
You don't need to decide; if you're already mid-install, just tell me what you see and we'll adapt.

1️⃣ Quick fix: use `python` or `py` instead of `python3`

In your VS Code terminal (Command Prompt), in your project folder, try:

```
python --version
```

If that prints something like `Python 3.13.x`, great. Then create your venv with:

```
python -m venv .venv
```

Alternative that works on most Windows setups:

```
py -3 -m venv .venv
```

Either of those is the "Windows way" of saying "run Python 3".

2️⃣ If you still see the Microsoft Store message

Then Windows is still using the **Microsoft Store alias** instead of your real Python install.
You can fix that:

927. Open **Settings**
928. Go to **Apps → Advanced app settings → App execution aliases**
929. Find entries like:
    - `python.exe`
    - `python3.exe`
930. Turn **both of those OFF** for the Microsoft Store version.

Then:

1146. Close VS Code completely
1147. Reopen VS Code
1148. Open a fresh terminal
1149. Try again:

If that now works, create the venv:

```
python -m venv .venv
```

You diagnosed the situation **correctly** — this error almost always means:
Python is installed, but not added to your system PATH.
That can be fixed **without reinstalling** Python, although reinstalling with the PATH checkbox is also fine. I'll walk you through both options, and you choose the one you prefer.

✅ Option A — Fix PATH (without reinstalling Python)

This works as long as Python is installed from python.org.

1️⃣ First, find where Python actually is

Open File Explorer and check these common install locations:

```
C:\Users\<your-username>\AppData\Local\Programs\Python\
C:\Program Files\Python3x\
C:\Program Files (x86)\Python3x\
```

Inside one of those, you should see folders like:

```
Python312\
python.exe
pythonw.exe
```

If you find `python.exe`, copy that folder path.
Example path (yours will differ):

```
C:\Users\Alex\AppData\Local\Programs\Python\Python312\
```

2️⃣ Add Python to your PATH manually

1. Press **Windows Key → type "environment variables" → open "Edit the system environment variables"**
2. Click **Environment Variables**
3. Under **User variables**, select **Path**
4. Click **Edit**
5. Click **New** and add:
   - The folder containing the Python executable (e.g.):
   - And also add the Scripts folder:
6. Click OK → OK → OK

3️⃣ Restart VS Code and test:

```
python --version
```

or:

```
py --version
```

If successful, you'll see something like:

```
Python 3.12.2
```

Then try your venv command again:

```
python -m venv .venv
```

✅ Option B — Reinstall Python (fastest for beginners)

If you want the simplest guaranteed fix:

1️⃣ Run the Python installer again

Go back to python.org → Download Windows Installer.

2️⃣ VERY IMPORTANT:

On the first page, click the checkbox:

✔️ **"Add Python 3.x to PATH"**

3️⃣ Complete installation.

4️⃣ Restart VS Code → new terminal → test:

```
python --version
```

This usually resolves everything instantly.

⚠️ IMPORTANT: Disable Microsoft Store Python Alias

Otherwise Windows may still intercept `python` commands.

1. Open **Settings**
2. Go to **Apps** → **Advanced app settings** → **App execution aliases**
3. Turn **OFF**:

- `python.exe`
- `python3.exe`

This prevents Windows from redirecting you to the Microsoft Store instead of your real Python install.

🌱 After PATH is fixed

Try again:

```
python -m venv .venv
.venv\Scripts\activate
```

Then verify the interpreter:

```
where python
```

You should see:

```
<project>\.venv\Scripts\python.exe
```

That means your venv is correctly activated.

If you want, you can tell me:

1. **Which folder contains your python.exe**
2. The output of
3. The output of

And I'll guide you precisely through the shortest path to getting venv working.
No judgment——this is one of the *most* common hurdles when moving from Anaconda to real Python environments. You're doing everything right by working through it.