Why is My Python Module Not Found in My Virtual Environment?

I've reinstalled it, checked my requirements.txt, and still no luck! Here's the real fix.

Ever been coding away in your Python virtual environment (venv), only to be hit with the dreaded **"ModuleNotFoundError"**? You're sure you installed it, but Python just can't seem to find it.

I've been there *way* too many times, and it's a frustrating roadblock, especially when you're ready to play with exciting new modules. Thankfully, I've discovered a simple solution that's become my holy grail for taming unruly venvs.

The key is knowing *exactly* where your Python and pip are pointing.

More often than not, they're not looking where you think they are! This confusion often leads to installing packages in the wrong place, leaving your venv empty-handed.

The Magic Words: which

Two simple commands will reveal the truth:

```
which python
which pip
```

These commands will print the path to the python and pip executables currently being used. If these paths aren't pointing inside your virtual environment's bin directory (e.g., ./my_venv/bin), then you've found the culprit! This is especially common when using tools like Homebrew

(Macbook folks raise your hands), which can sometimes override your system's default Python.

Rescuing Your Venv:

Once you've confirmed the problem, the fix is straightforward:

Activate your venv
The easiest way is to open your project in VS Code, locate the activate script within your venv's bin directory (e.g., ./my_venv/bin/activate), and drag it directly into your terminal. Then, hit enter. This executes the script and properly activates your environment.

(Re)install your packages
Now that your venv is active, use pip to install or reinstall the necessary packages. If you have a requirements.txt file, use:

```
pip install -r requirements.txt
```

Otherwise, install individual packages with:

```
pip install <package_name>
```

Putting it all together

Let's say your venv is called my_env.

Check the paths

```
which python  # Might show /usr/local/bin/python (or something outside your venv)
which pip     # Might show /usr/local/bin/pip
```

Activate the venv: Drag and drop the **activate** file (usually in ./my_env/bin/activate) into your terminal and press Enter.

Check again

```
which python # Should now show ./my_env/bin/python which pip
which pip # Should now show ./my_env/bin/pip
```

Install packages (again):

```
pip install -r requirements.txt  # Or pip install <package_name>
```

That's it!

This quick check and reset will save you from countless headaches and get your Python projects back on track.

Trust me, your future self will thank you.

By Carl Kho on October 3, 2024.

Canonical link

Exported from Medium on October 31, 2025.