How to use pyenv to manage Python versions

[A person wearing glasses

Description automatically generated with medium confidence](https://blog.teclado.com/author/jose/)

[Jose Salvatierra](https://blog.teclado.com/author/jose/)

3 min read · Nov 28

<https://blog.teclado.com/how-to-use-pyenv-manage-python-versions/?ref=The+Teclado+Blog-newsletter>



As a Python developer, I install new Python versions when they come out. We get new features and performance improvements, but we don't always want to use the latest version for everything. Some projects use older Python versions, and other projects must use a specific Python version.

If you need to install multiple Python versions, going the ol' installer route isn't the best idea. You end up with multiple Python executables in your PATH, such as python, python2, python3, python3.7, and so on. Also it becomes really tricky to distinguish minor versions, such as python3.10.3 from python3.10.4.

So, [pyenv](https://github.com/pyenv/pyenv) to the rescue! This command-line tool will handle installing and running specific Python versions!

Their README file actually explains [how it works](https://github.com/pyenv/pyenv#how-it-works) and [how to install it](https://github.com/pyenv/pyenv#installation) really well. Give it a read!

As a note, for Windows I've often use the [pyenv-win project](https://github.com/pyenv-win/pyenv-win), which has worked well for me.

To install pyenv on MacOS, I've just gone for Homebrew:

brew update

brew install pyenv

After installing you'll have to add a few things to your shell, to configure it. Detailed instructions are in [the README](https://github.com/pyenv/pyenv#set-up-your-shell-environment-for-pyenv).

**How I actually use**pyenv

The great thing about pyenv for me is that it just works, and I only use it at the very beginning of a project, to create a virtual environment.

Once I've created a virtual environment using a specific Python version (which I get from pyenv), from then on I just activate the virtual environment to get that version, and I don't have to faff around with pyenv any more.

So, how do you do this?

First, see if the version of Python you want is available (you may have to update pyenv to see recent versions, which I do with brew update && brew upgrade pyenv):

pyenv install --list

This will show you a long output, which may contain things like the following:

...

3.10.1

3.10.2

3.10.3

3.10.4

3.10.5

3.10.6

3.10.7

3.11.0rc2

3.11-dev

3.12-dev

...

So now you know, you can install from these versions (and again, reminder to update to see the most recently-added versions).

To install a Python version:

pyenv install 3.10.7

**Selecting a Python version**

You can see the currently-selected Python version with this command:

pyenv version

And you can see all versions (including the currently selected one) with:

pyenv versions

You can have a globally selected Python version, and then locally selected Python versions. The global version is used if no local version is selected.

You can select a local version with:

pyenv local 3.10.7

This creates a .python-version file in your current folder, which tells pyenv to use that Python version.

If you want to change the global Python version:

pyenv global 3.10.7

**Using the selected Python version**

Now, let's create a virtual environment using pyenv:

pyenv exec python -m venv .venv

This uses pyenv exec to run the python command, which will use Python 3.10.7 in our case. The -m venv .venv argument passed to python tells it to run the venv module, and gives it the name of .venv.

This will create a virtual environment in a folder called .venv.

And to be honest, this is how I use pyenv. That's it, nothing more!

If you want to test your applications in multiple Python versions, you can also do so easily. Real Python has a great in-depth [blog post](https://realpython.com/intro-to-pyenv/#activating-multiple-versions-simultaneously) that covers that, so feel free to check it out.

That's everything! I hope you've found this interesting, and you've learned something new. If you want to delve deeper into Python and everything it has to offer, please consider joining our [Complete Python Course](https://go.tecla.do/complete-python-sale). This mega-course covers many Python topics, and it's currently on sale!

Thanks for reading, and I'll see you next time.