pyenv Python version switching

START HERE

`pyenv install -l`
List installable versions

`pyenv local <version>`
Pin this directory

`pyenv install <version>`
Build a Python version

`pyenv shell <version>`
Pin this shell only

`pyenv versions`
Installed versions

`pyenv version`
Active version and why

`pyenv global <version>`
Set the default

`pyenv which <command>`
Real path of a command

python shim in PATH → asks **pyenv** → **Resolve version** env, file, global → picks → **~/.pyenv/versions** <version>/bin/python → runs → **Your command** right interpreter

CONCEPTS

pyenv builds each Python from source under ~/.pyenv/versions and switches between them with shims; it does not manage packages or virtualenvs by itself.

Version resolution (first hit wins)

Source	Set by
`PYENV_VERSION`	`pyenv shell`
`.python-version here`	`pyenv local`
`.python-version in a parent`	`pyenv local`
`$(pyenv root)/version`	`pyenv global`
`system Python`	`fallback if none`

Run pyenv version to see which source won; it prints the version and the file or variable it came from.

SETUP

`export PYENV_ROOT="$HOME/.pyenv"`
Root directory (default ~/.pyenv)

`export PATH="$PYENV_ROOT/bin:$PATH"`
Put the pyenv command on PATH

`eval "$(pyenv init - bash)"`
Enable shims in bash (~/.bashrc)

`eval "$(pyenv init - zsh)"`
Enable shims in zsh (~/.zshrc)

`pyenv init - fish | source`
Enable shims in fish (config.fish)

`pyenv init --path`
Print only the shims PATH setup

`pyenv root`
Print the pyenv root directory

`pyenv commands`
List all pyenv commands

`pyenv help <command>`
Help for one command

Login shells need the init lines too: for bash also in ~/.profile or ~/.bash_profile, for zsh in ~/.zprofile.

Edit shell startup files, then open a new shell. pyenv shell does nothing without the init eval line.

INSTALL VERSIONS

`pyenv install -l`
List all installable versions

`pyenv install <version>`
Build and install a version

`pyenv install 3.12`
Prefix: newest 3.12.x known to pyenv
A partial version resolves to the latest matching patch release.

`pyenv install`
Install what .python-version names

`pyenv install -s <version>`
Skip if already installed

`pyenv install -f <version>`
Reinstall even if present

`pyenv install -v <version>`
Show the compile output

`pyenv install -k <version>`
Keep the source tree after build

`pyenv install -g <version>`
Build a debug version

`pyenv install <version>:<alias>`
Install under a custom name

`pyenv uninstall <version>`
Remove a version (asks first)

`pyenv uninstall -f <version>`
Remove without confirming

`pyenv latest <prefix>`
Latest installed match for a prefix

`pyenv latest -k <prefix>`
Latest known (installable) match

Install fails with missing zlib, ssl or ffi? Install the distro build dependencies first (see Build dependencies).

CHOOSE A VERSION

`pyenv versions`
List installed, active one marked *

`pyenv version`
Active version and where it came from

`pyenv version --bare`
Print only the version name

`pyenv global <version>`
Default for the user (~/.pyenv/version)

`pyenv global <v1> <v2>`
Several versions, searched in order

`pyenv global system`
Use the system Python

CHOOSE A VERSION (cont.)

`pyenv local <version>`
Write .python-version here

`pyenv local <v1> <v2>`
Several versions in one file

`pyenv local --unset`
Remove .python-version

`pyenv local -f <version>`
Set even if not installed

`pyenv shell <version>`
Set PYENV_VERSION in this shell

`pyenv shell --unset`
Clear the shell-level version

`PYENV_VERSION=<version> <command>`
Override for a single command

.python-version holds one version per line; lines starting with # are ignored. pyenv searches parent directories too.

FIND AND RUN

`pyenv which <command>`
Full path pyenv would run

`pyenv which --nosystem <command>`
Ignore the system Python

`pyenv whence <command>`
Which versions provide a command

`pyenv prefix [<version>]`
Install directory of a version

`pyenv exec <command>`
Run with the selected version first in PATH

`pyenv rehash`
Regenerate shims
Needed after installing a package that adds a command; pip installs usually trigger it for you, other installers may not.

`pyenv shims`
List existing shims

A command not found by pyenv's selected versions falls through to the next match on PATH, usually the system one.

BUILD AND ENVIRONMENT

`PYENV_ROOT`
Where versions and shims live

`PYENV_VERSION`
Version to use; beats files

`PYENV_DIR`
Directory to start the .python-version search

BUILD AND ENVIRONMENT (cont.)

`PYENV_DEBUG=1 pyenv <command>`
Trace what pyenv does

`rm -rf $(pyenv root)`
Uninstall pyenv and all versions

`brew upgrade pyenv`
Upgrade a Homebrew install

`pyenv update`
Upgrade a git install (needs pyenv-update)

BUILD DEPENDENCIES

Debian / Ubuntu

```
sudo apt update
sudo apt install make build-essential libssl-dev zlib1g-dev \
  libbz2-dev libreadline-dev libsqlite3-dev curl git \
  libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev \
  libffi-dev liblzma-dev libzstd-dev
```

Fedora

```
sudo dnf install make gcc patch zlib-devel bzip2 bzip2-devel \
  readline-devel sqlite sqlite-devel openssl-devel tk-devel \
  libffi-devel xz-devel libuuid-devel gdbm-libs libnsl2
```

macOS (Homebrew)

```
brew install openssl@3 readline sqlite3 xz tcl-tk@8 zstd zlib pkgconfig
```

A build that finishes with 'optional modules not built' still installs; missing modules (ssl, tkinter) show up later at import time.

RECIPES

Fresh machine, new project

```
pyenv install 3.12
cd myproj
pyenv local 3.12
python -m venv .venv
pyenv version
```

Test against several versions

```
pyenv install -s 3.11 && pyenv install -s 3.12
pyenv local 3.12 3.11
python3.11 -m pytest
python3.12 -m pytest
```

Debug a missing command

```
pyenv version
pyenv which <command>
pyenv whence <command>
pyenv rehash
```

Install the newest patch release

```
pyenv install -l | grep -E '^ *3\.12'
pyenv install 3.12
pyenv global $(pyenv latest 3.12)
```