pyenv Python version switching

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.

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

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 root`
Print the pyenv root directory

`pyenv commands`
List all pyenv commands

`pyenv help <command>`
Help for one command

SETUP (cont.)

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

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

CHOOSE A VERSION (cont.)

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

`pyenv global system`
Use the system Python

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

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

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

`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 whence <command>`
Which versions provide a command

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

`pyenv rehash`
Regenerate shims

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
```

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