MAKE PYTHON DEVEX

How a 45+ year old tool can MAKE Python developer experience easier

Colin Dean, Lead Engineer

`@colindean@mastodon.social`

WHO IS THIS GUY?

Colin Dean, who wears many hats

Software engineer and community builder since 2002


Pittsburgh

DEVELOPER EXPERIENCE

AGENDA

Problem	Getting Python developer experience right is hard
Diagnosis	Python project definitions leave too much undefined
Remedy	Standardized Makefile for Python dev

Please send me your questions!

Grab a piece of paper so you can write them down.

OPEN THOUGHTS

tech.target.com/blog/make-python-devex

OPEN SOURCE

github.com/target/make-python-devex

PROBLEMS

- What Python version should I use for development?
- What Python version will be used for production?
- How do I install Python at a specific version?
- How do I ensure that my project works on a diverse development *and* production installation base?
- How do I install dependencies?

WHICH PYTHON?

DIAGNOSIS

SYSTEM PYTHON UNRELIABLE, INFLEXIBLE

```
$ date && sw_vers && uname -sm && /usr/bin/python3 --version
Fri May  6 12:39:51 EDT 2022
ProductName:    macOS
ProductVersion: 12.3.1
BuildVersion:   21E258
Darwin arm64
Python 3.8.9
```

INCONSISTENT INSTALLATION METHODS AND SOURCE

- ~~Operating System-provided packaging?~~
- Homebrew?
- **Pyenv? Anaconda?**
- …MacPorts? …Nix?

INCONSISTENT PYTHON VERSIONS

- Tech Debt Risk: dependencies stop supporting older versions
  - Python 3.8 and older, EOL
  - Python 3.9 and 3.10 in security-only phase
    - EOL October 2025 & 2026
  - Python 3.11, 3.12, 3.13 supported
    - Final non-security release in April: 2025-2027
    - EOL October: 2027-2029

NO ONE CAN REMEMBER COMMANDS

- …or wants to type them
- …or wants to use the correct, full-length command
  - `poetry run pytest`
- Neither enables build actions outside of their domain

PYTHON ECOSYSTEM MAY NECESSITATE COMPILING WITH OPTIONS

- Apple M1 processor, ARM64 m-arch., ca. June 2020
- `macos_{11...15}` `arm64` binary avail. growing
- Passing compiler flags is not ergonomic:

```
$ LDFLAGS="-L/opt/homebrew/Cellar/unixodbc/2.3.9_1/lib -lodbc
-liodbc -liodbcinst -ldl" CPPFLAGS="-
I/opt/homebrew/Cellar/unixodbc/2.3.9_1/include -
I/usr/include" HDF5_DIR=/opt/homebrew/opt/hdf5 poetry install
```

SCRIPTS GROW HARD TO MANAGE

- A collection of scripts that gets copied around
- Specialization for the repo difficult once shared
- Documentation often left to READMEs, not tooling

REMEDY

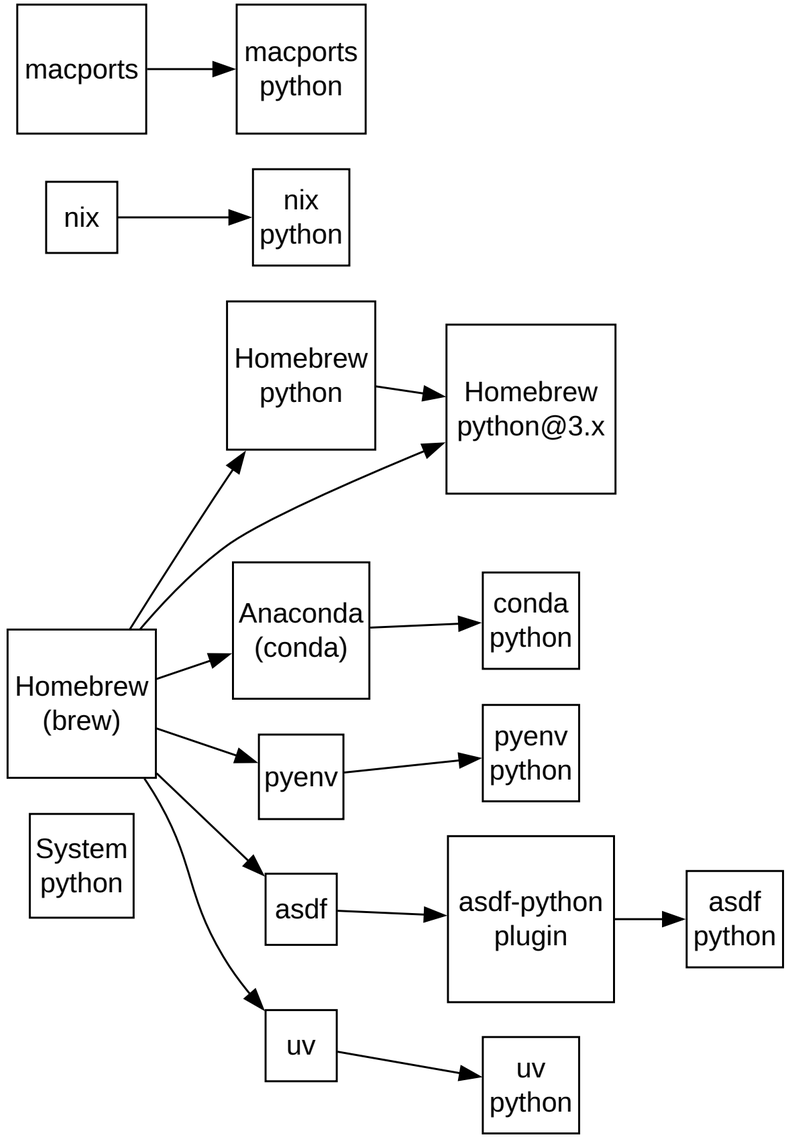
AUTOMATE

We need a **task runner** for onboarding *and* resync
instead of *just* better documentation.

INSTALL

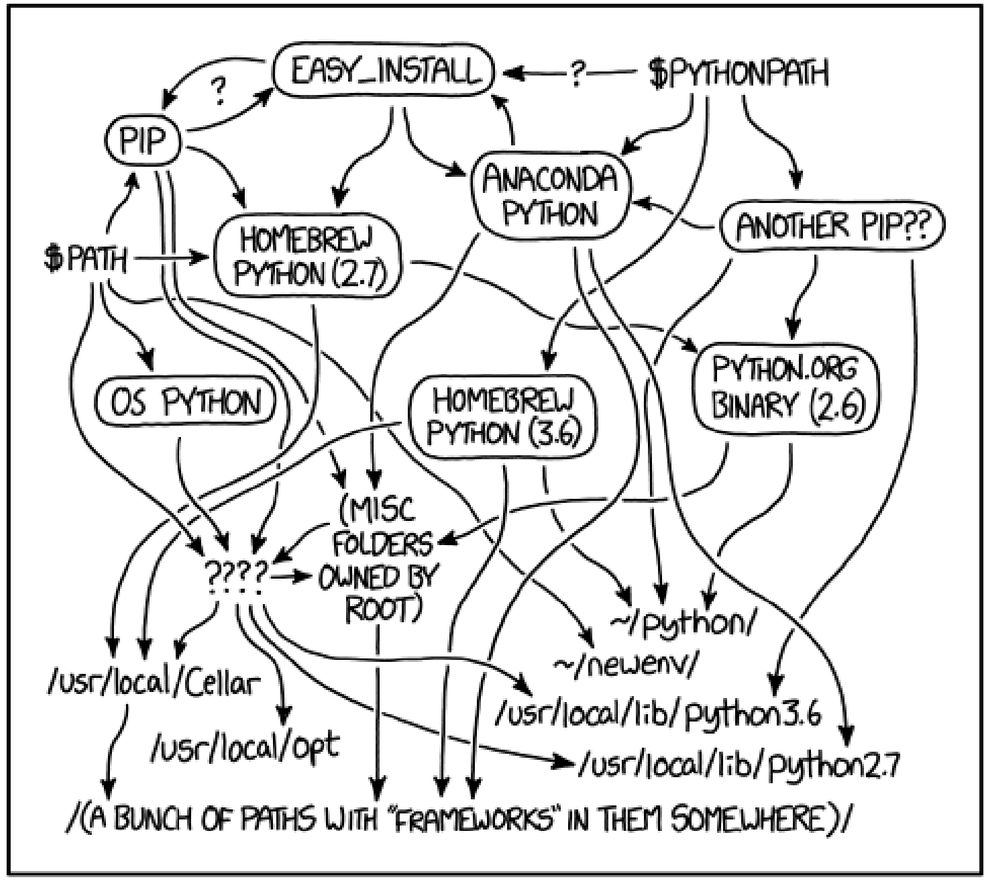
1. whatever installs Python
2. Python
3. whatever installs Python dependencies' dependencies
4. whatever installs Python dependencies
5. Python dependencies' dependencies
6. Python dependencies
7. everything else needed somewhere in there

PYTHON INSTALLATION METHODS



AGAIN: WHICH PYTHON?

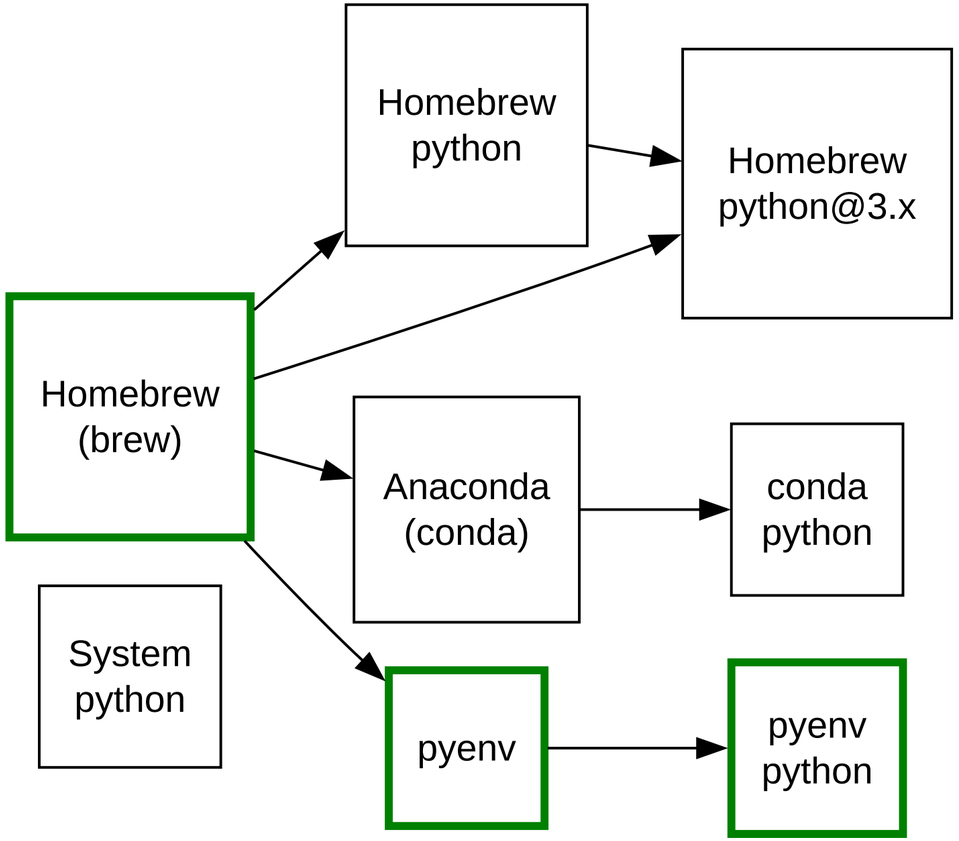
“This one, and you need not care.”



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

XKCD 1987

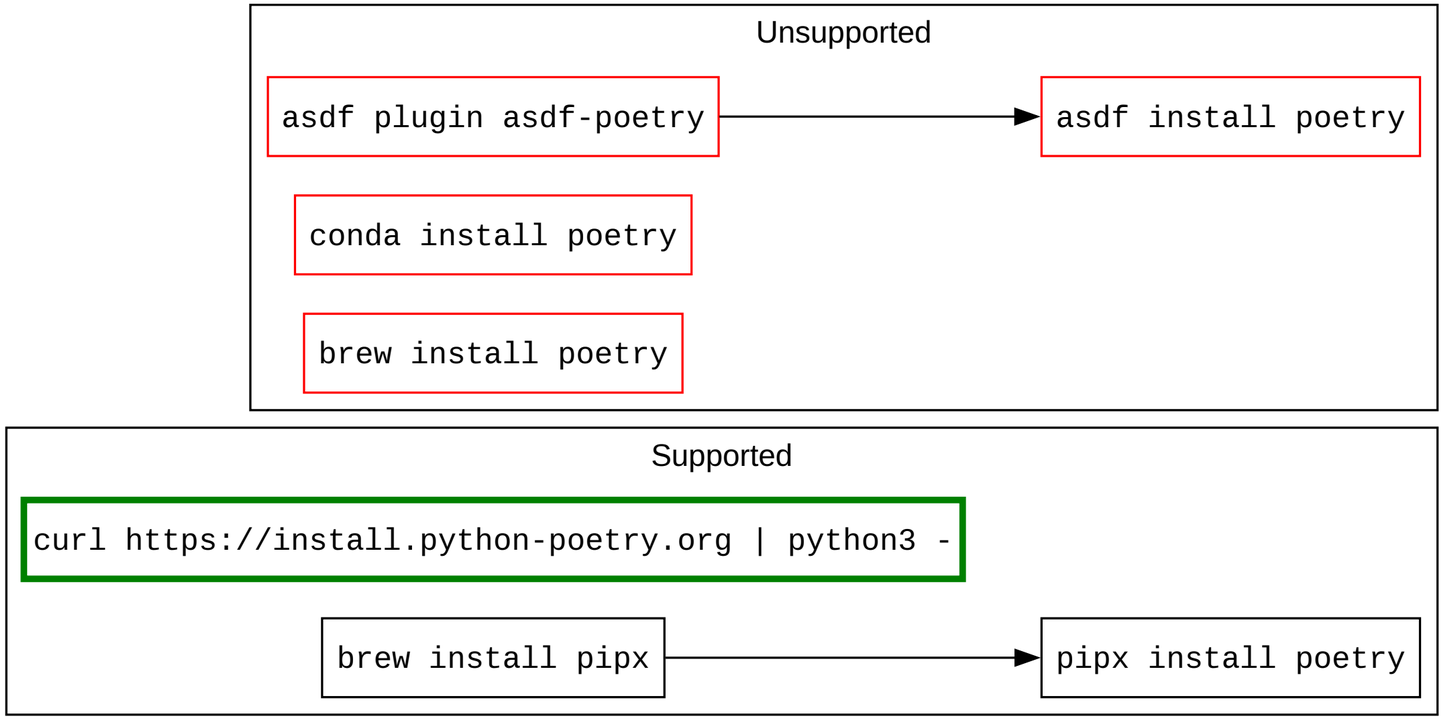
HOMEBREW + PYENV

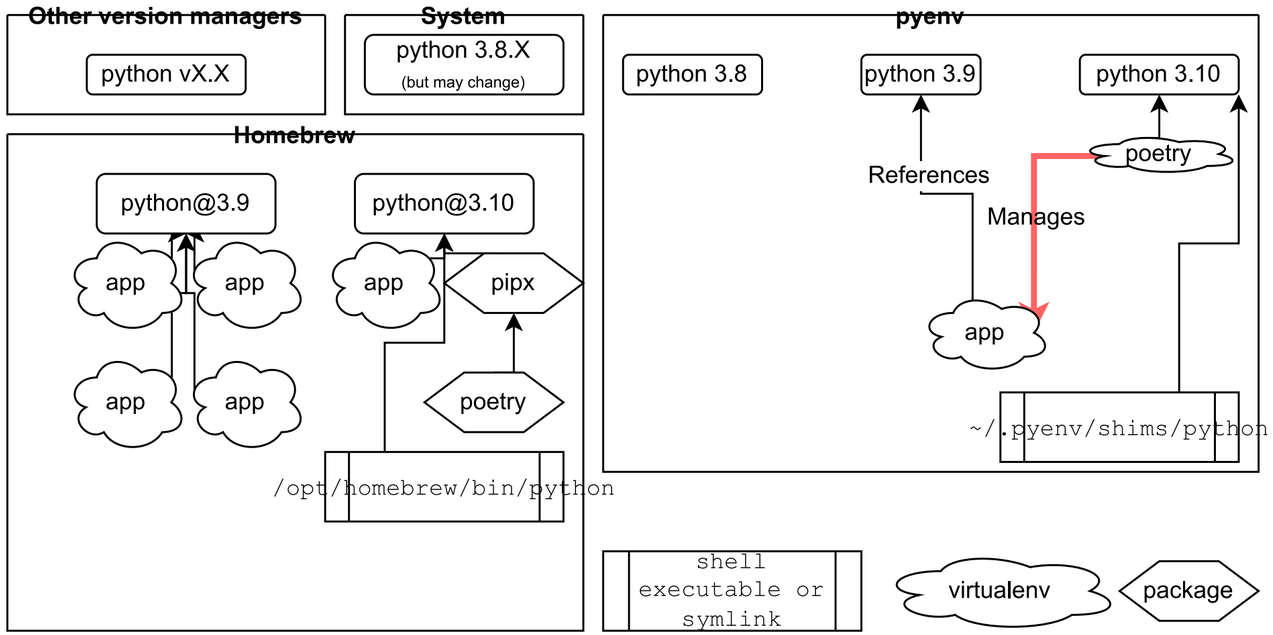


POETRY FOR PYTHON DEPENDENCIES AND PACKAGING

- `pyproject.toml` standardization
- Separately-managed virtualenv
- Modern dependency mgmt with locking
- Builds packages

INSTALLING POETRY





Installed Poetry uses its own virtualenv and manages other virtualenvs externally

REJECTED SOLUTIONS

- **Containers** - too slow, IDE problems
- **Develop on cluster** - wildly slow, IDE problems
- **Rewrite in another language/eco system** - Complexity not worth considering

GLUE

STANDARDIZED `Makefile` FOR PYTHON + SHELL DEV

- Use Make, which is widely available.
- Minimize project-specific customizations in order to enable copy-paste to new projects.
- Use `make` tasks in CI builds to mirror dev environment.

BASIC TASK RUNNER

```
task: dependency-task ## Helpful explanation of the task
↦    command --that does --something
```

PRODUCE FILES

```
build: output.txt output.pdf ## Build the things

output.txt: input.txt
↦    command --that produces --something $@ --from $<
```

`make build` will only run the `output.txt` task when:

- `output.txt` doesn't exist
- `input.txt` has changed since the last time the task was run

↦ = tab

https://github.com/colindean/plaintextaccounting_workshop

https://makefiletutorial.com

WHY NOT X?

- CMake
- Gradle
- SBT
- XMake
- Just
- OK
- Meson
- Brazil
- npm
- Maven
- Ant

```
make deps check test build

make help clean
```

</rant>

snap back to reality

(ope, there goes gravity)

OUR PARTICULAR SETUP

WITH ONLINE HELP!

```
$ make help

Usage:
  make <task>

Utility
  help              Display this help
  version-python    Echos the version of Python in use
```

```
Dependency Setup
  deps               Installs all dependencies
  deps-brew          Installs dev dependencies from Homebrew
  deps-peru          Installs data dependencies using Peru
  deps-py            Installs Python dev and runtime deps
  deps-py-update     Update Poetry deps
```

```
Testing
  test                Runs tests

Building and Publishing
  build               Runs a build
  publish             Publish a build to the configured repo
```

```
Code Quality
  check             Runs linters and other important tools
  check-py          Checks only Python files
  check-py-ruff     Runs ruff linter
  check-py-flake8   Runs flake8 linter
  check-py-black    Runs black in check mode (no changes)
  check-py-mypy     Runs mypy
  check-sh          Run shellcheck on shell scripts
  fix-sh            Runs shellcheck & applies suggestions
  format-py         Runs black||ruff, may make changes
  format-shell      Runs shfmt on shell scripts & tests
```

CHALLENGES

SUPPORTING MULTIARCHITECTURE

- macOS x86_64 **&** arm64 for dev
- Linux x86_64 for prod
- Python ecosystem mixed readiness

Apple Silicon == M1 == arm64

```
FLAGS =
ifeq ($(shell uname -m), arm64)
ifeq ($(shell uname -s), Darwin)
HDF5_DIR = $(shell brew --prefix --installed hdf5)
LIBS = odbc libiodbc
F_LDFLAGS = LDFLAGS="$(shell pkg-config --libs $(LIBS))"
F_CPPFLAGS = CPPFLAGS="$(shell pkg-config --cflags $(LIBS))"
FLAGS = $(F_LDFLAGS) $(F_CPPFLAGS) HDF5_DIR=$(HDF5_DIR)
endif
endif

POETRY = $(FLAGS) poetry
```

macOS arm64

```
LDFLAGS="-L/opt/homebrew/Cellar/unixodbc/lib -lodbc -
libiodbc" \
CPPFLAGS="-I/opt/homebrew/Cellar/unixodbc/include" \
HDF5_DIR=/opt/homebrew/Cellar/hdf5
poetry
```

macOS x86_64 || Linux *

```
poetry
```

FUTURE WORK

IMPROVEMENTS

- More Make documentation in Makefile
- Streamline one-time setup experience
- Environment debugging
- uv version

FIN

COLOPHON

- Pandoc rendering
- Reveal.js presentation

Slides source: https://github.com/colindean/talks

THANK YOU

https://tech.target.com/blog/make-python-devex

https://github.com/target/make-python-devex