Dependency and package management in Python A hands-on approach

Introduction

- I finished a script using python
- It works in my machine™
- I want to share it with others
- I want it to share across other projects
- I want to modify it without breaking existing implementations

What this workshop is not about

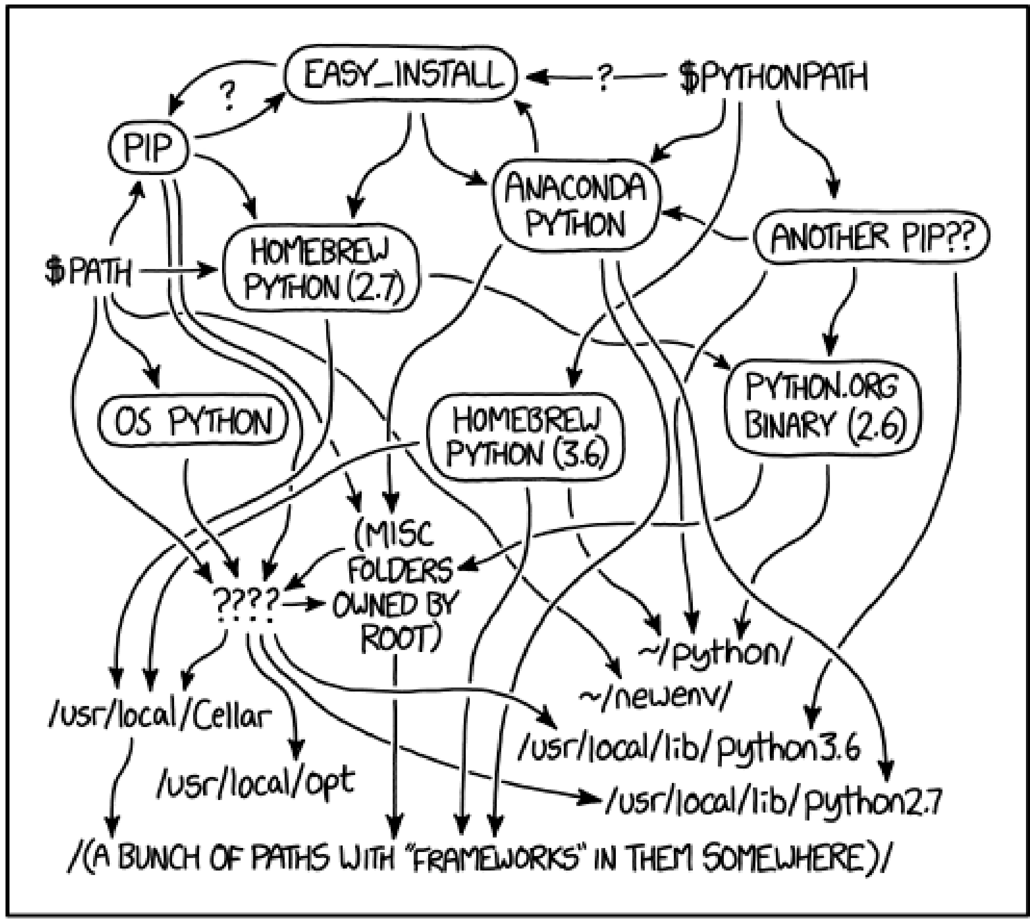
- It’s not about bundling code into a .exe
- It’s not about writing performant code
- Touches some edges about code quality but it’s not about code quality
- It’s not about learning docker

Things you will need

- docker from https://docs.docker.com/get-docker/
  - note some differences exist between docker for windows and docker
- But I don’t want to/I can’t install docker
  - Create an account at https://labs.play-with-docker.com
  - Some features are not guaranteed
  - Some times availability is capped
- A pypi account at https://pypi.org/ (optional)
- A dockerhub account at https://hub.docker.com/ (optional)
- Visual studio code with python and remote development extensions
  - https://code.visualstudio.com/docs/python/
  - https://code.visualstudio.com/docs/devcontainers/attach-container

The gist of working with python

- Never do things on your default python installation
- Always work on a virtual environment
- Know and let others know what version of Python your code is using
- Know at all times the location of your python version



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

Sharpen your axe

> If I had to chop down a tree in eight hours, I would spend six hours sharpening my axe.

This saying is *usually* attributed to Abraham Lincoln, who seemed very good at chopping down trees. Taking care of your tools will make your craft easier. This is true for any craft and specially of software development.

Pyenv

- https://github.com/pyenv/pyenv
- Let pyenv manage versions of python.
- e.g. Change between python 3.8 and 3.10
- Useful for cross-version testing (e.g. with tox)
- Lets you manage conda environments, although it’s a bit buggy. Learn how to manage the two at the same machine.
- Let pyenv manage virtual environments
- Using pyenv extension https://github.com/pyenv/pyenv-virtualenv

Pipx https://github.com/pypa/pipx It’s pip but for system-wide CLI applications poetry sqlacodegen ruff Anything you want to use globally but don’t want to install at every virtual environment Pipx uses its own virtual environment We will use it to install Poetry

https://pypa.github.io/pipx/

Poetry https://github.com/python-poetry python dependency and environment manager pip install cowsay ≈ poetry add cowsay poetry checks that cowsay version installed is compatible with other libraries installed in the project we’ll talk about compatibility of dependencies later

Docker Containerization technology Virtual machine ≇ Container Tackles the works on my machine™ problem Leveraged by kubernetes and docker swarm Can’t think of a better workshop about containers than this Other alternatives exist: podman, lxc A fair/good investment of your time https://explainxkcd.com/wiki/index.php/1988:_Containers

Levels overview

First level

Installing everything by hand

First level: installing everything by hand

- Start a docker container with python installed
- Install pyenv inside
- Install pipx inside
- Install poetry inside
- Play a bit with these new tools
- write a small CLI application

Second level: Publish your package

- Build and publish your package
- Install it and use it from another repository

Third level: automate the installation of tools in Docker

- Write a Dockerfile
- Build a docker image with your package
- Play a bit with your new image

Fourth level: Publish your package outside of your local machine

- Start a local pypi repository using docker compose
- Configure poetry to connect to this repository
- Publish your package to this repository

Fifth level: Automate what should be automated

- Automate a development environment using docker compose and devcontainers
- Automate releases of your package using Github Actions
- Automate publishing new images to a container registry using Github Actions