Research Software Summer School: Environments & Dependencies

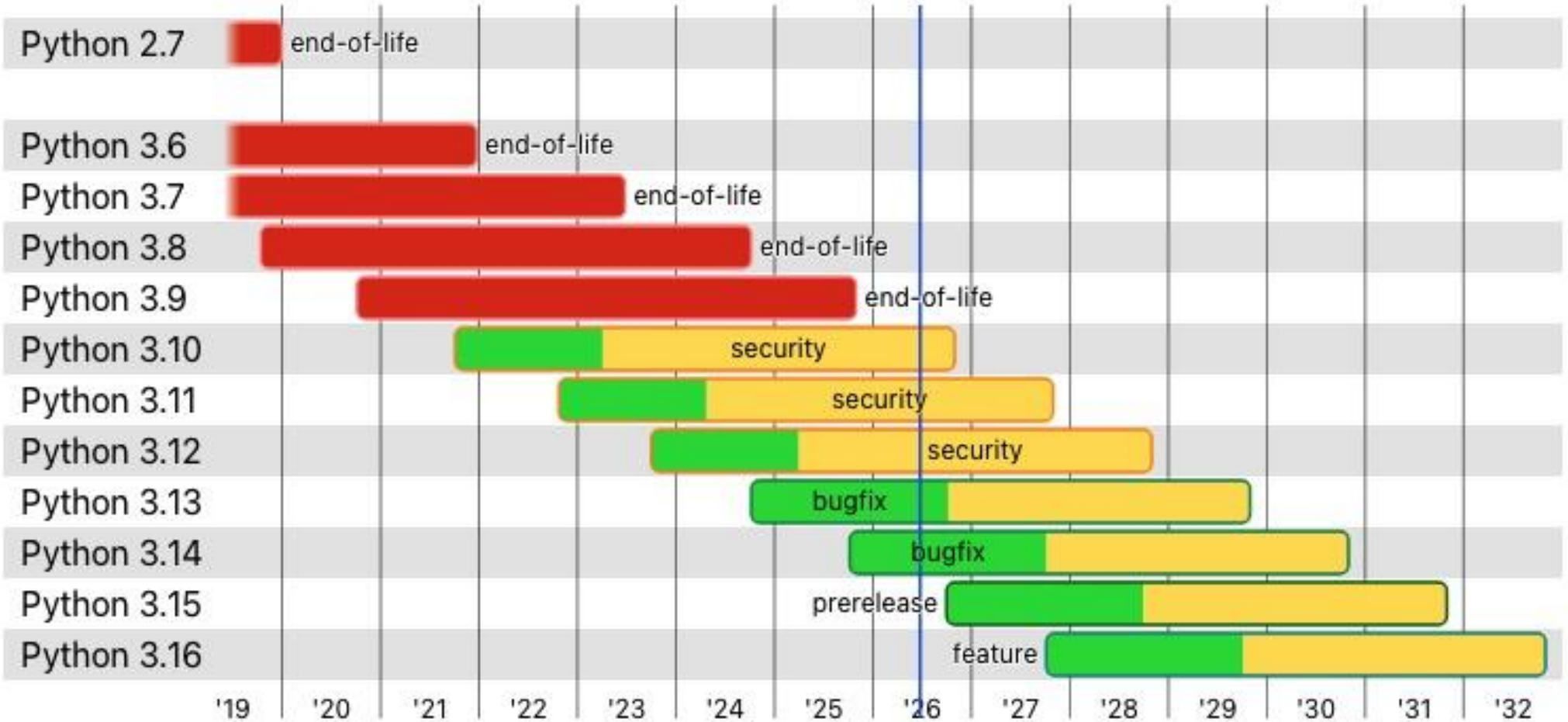
Jelte van Boheemen
Research Software Engineer

Reproducible code

- We want **reproducible** code
  - For others
  - For ourselves
  - For science
- More on this tomorrow!

Part 1:
Python Environments

Python versions



Python installation

- You have a *global* version of Python installed
- Check your version
  - **python --version**

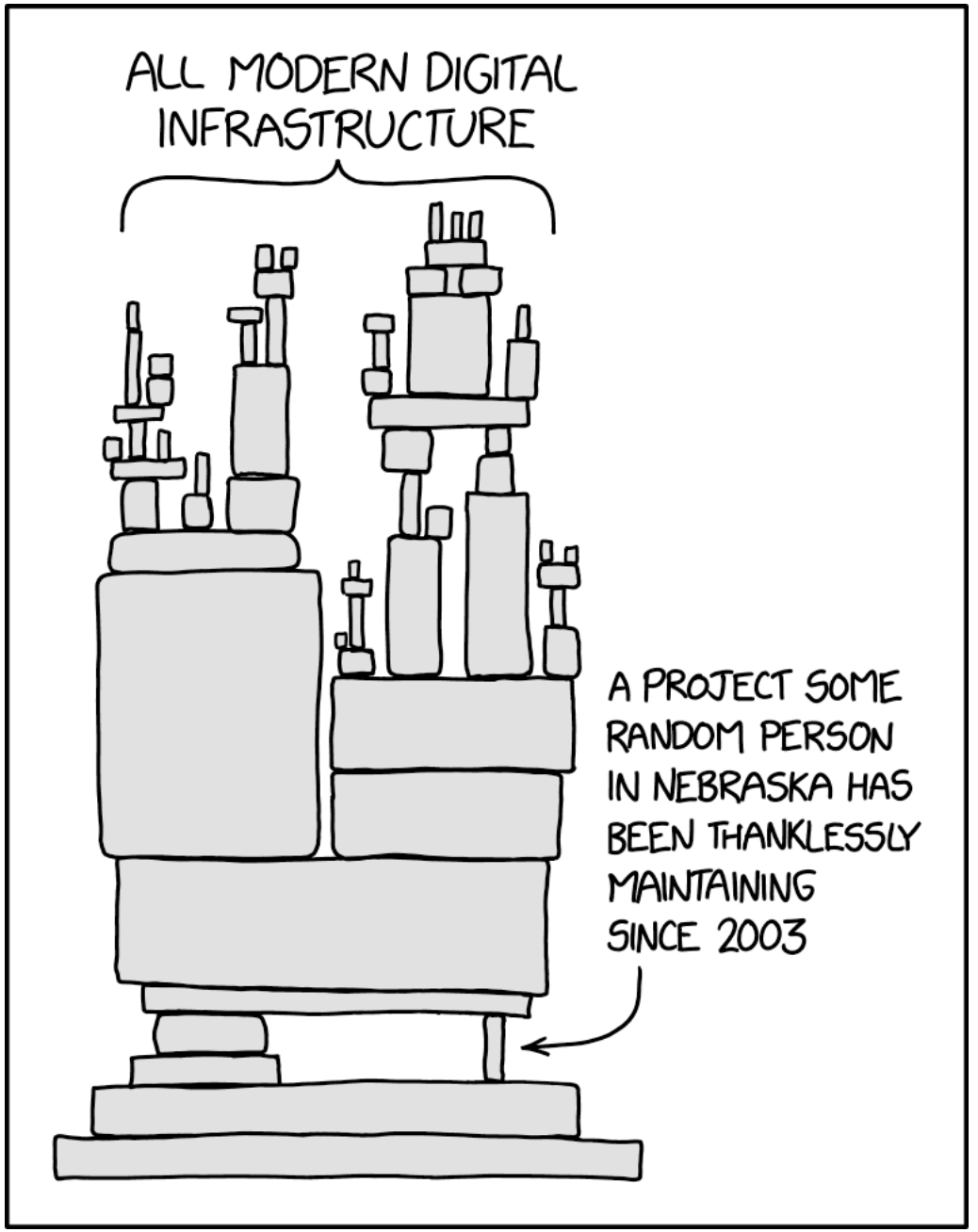
- You can install other versions of Python
- Why would you do that?

Python packages

- (Almost) all Python project use 3rd-party packages
- View installed packages
  - **pip list**
- Install more using pip
  - **pip install <package_name>**
- Where do the installed packages end up?
- Packages often have multiple versions

Dependencies

- All the packages your project *depends* on
- Other projects *may* have similar dependencies
- Dependencies *may* have dependencies of their own



Chaos

- Multiple:
  - Python versions
  - Projects
  - Dependencies

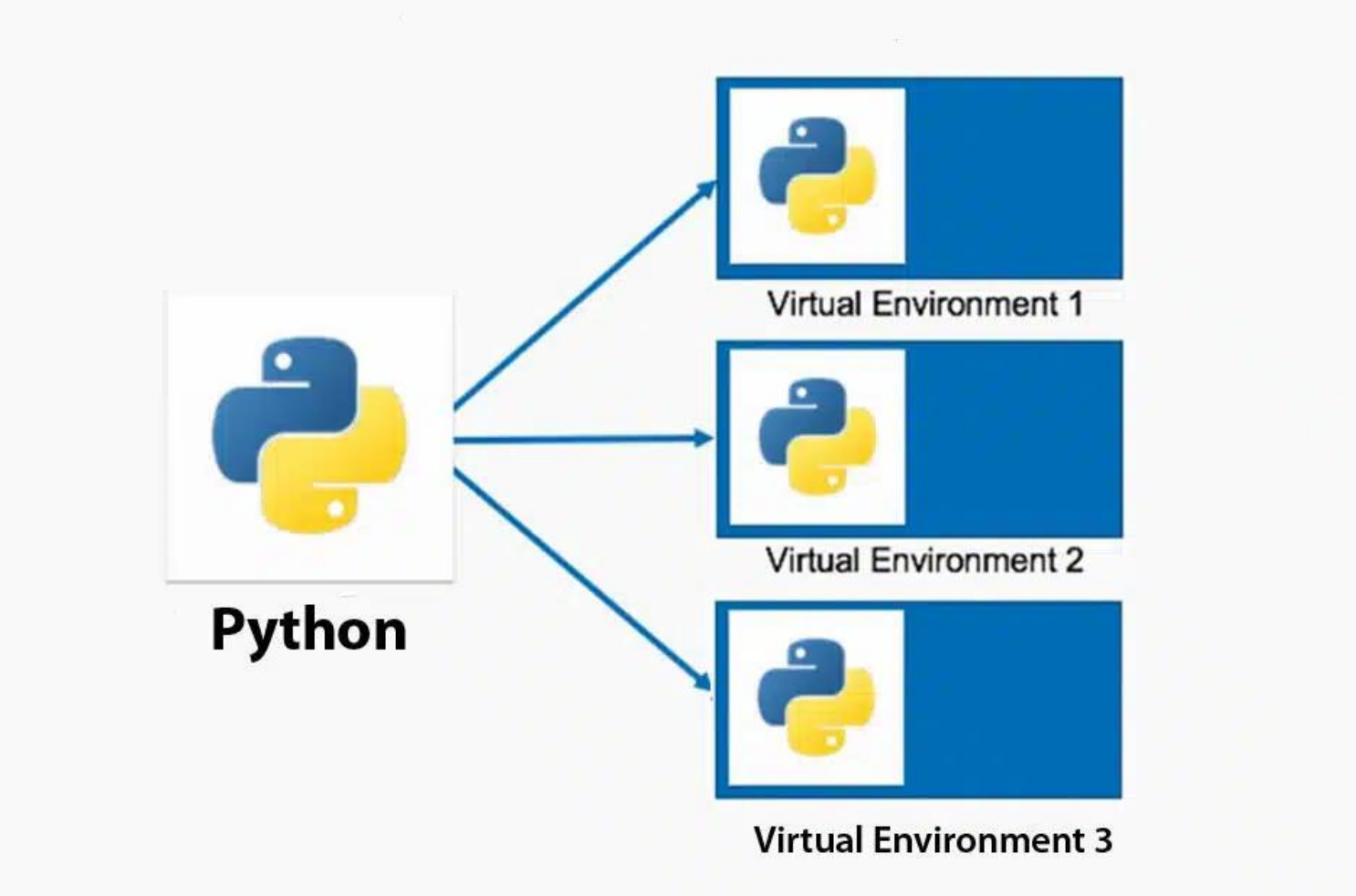
- How do you keep them isolated?

Virtual Environments

- Based on *global* Python
- Separate *local* version

- Separate installed packages

- “Walled garden”



Chaos (still)

- Multiple:
  - Python versions
  - tools to create/manage venvs
  - where do you put venvs?
  - environments to update

- Solution:
  - keep it simple
    - one Python version
    - use **python –m venv**
    - put venv in project directory
    - no **anaconda**

Exercise time!

- Day 1 > Virtual Environments > Python Environments
  - Exercise: create a virtual environment
  - Exercise: ignore a virtual environment
  - (if using VSCode): select interpreter

- If you get stuck
  - ask the helpers (put a pink sticky on your laptop)
  - ask your neighbour
  - adagium: being stuck a short while is learning, longer is wasting time

Part 2:
Reproducible Dependencies

Reproducible dependencies: why?

- ModuleNotFoundError: No module named ‘pandas’
- ”It works on my machine”

- Who has been here? (I have)

- We should have reproducible dependencies

Reproducible dependencies: how?

- Supply the the dependencies with the code - problem solved
  - E.g. include the virtual environment
  - Is this a good idea?

- Write down the dependencies
  - Give instructions on how to reproduce the environment
  - Is this a good idea?

Writing down dependencies

- Add them to the documentation
  - Upside: allows reproducing environment
  - Downside: only human-readable

- Provide a requirements file
  - Upside: human- and machine-readable
  - Downside: none, you’re doing it right

Requirements file

- Convention: **requirements.txt**
- Plaintext file
- Every line is a single requirement

- Example:

requirements.txt

```
1    requests
2    pandas
3    matplotlib
```

Working with requirements files

- Creating
  - Write down all your packages, one line per package

- Using
  - **pip install –r requirements.txt**
  - Installs all packages in requirements file

Alternatives to requirements.txt

- Alternatives:
  - setup.py
  - pyproject.toml

- Advice:
  - be aware
  - use requirements.txt

Chaos (once again)

- You created a requirements file
- Your colleague installed everything
- They run the code
- What went wrong?
- Been there? (I have)

Specifying versions

- Supply the **exact** version of dependency

requirements.txt

```
requests==2.32.0
pandas==2.23
matplotlib==3.10.1
```

- Creating pinned requirements
  - Install all your packages
  - **pip freeze > requirements.txt**

- Recreates the environment **exactly**
  - Or does it?

Semantic versioning

- Tell what changed **just** by looking at version number
- Pattern: **MAJOR.MINOR.PATCH**
  - **MAJOR**: breaking changes, not compatible
  - **MINOR**: new features, should be compatible
  - **PATCH**: bugfixes, always compatible

- Examples:

Old version	New version
requests==2.32.0	requests==2.33.0
matplotlib==1.5.7	matplotlib==3.10.1
pandas==2.23.1	pandas==2.23.12

Exercise time!

- Day 1 > Virtual Environments > Reproducible depencencies
  - Exercise: create a requirements file
  - Exercise: recreate an environment
  - Exercise: update requirements.txt

- If you get stuck
  - ask your neighbour
  - ask the helpers (put a pink sticky on your laptop)
  - adagium: being stuck a short while is learning, longer is wasting time

Utrecht University

Sharing science,
shaping tomorrow