Next Steps in Python:
Virtual Environments

Aaron Geller

(with help from Scotty Coughlin)

Research Computing and Data Services

Follow @Northwestern_IT on Instagram for the latest workshop and bootcamp updates.

Northwestern | INFORMATION TECHNOLOGY

This workshop is brought to you by:

Northwestern IT
Research Computing and Data Services

Need help?

- AI, Machine Learning, Data Science
- Statistics
- Visualization
- Collecting web data (scraping, APIs), text analysis, extracting information from text
- Cleaning, transforming, reformatting, and wrangling data
- Automating repetitive research tasks
- Research reproducibility and replicability
- Programming, computing, data management, etc.
- R, Python, SQL, MATLAB, Stata, SPSS, SAS, etc.

Request a **FREE** consultation at bit.ly/rcdsconsult.

See many other upcoming workshops :

https://www.it.northwestern.edu/departments/it-services-support/research/research-events.html

What is a virtual environment?
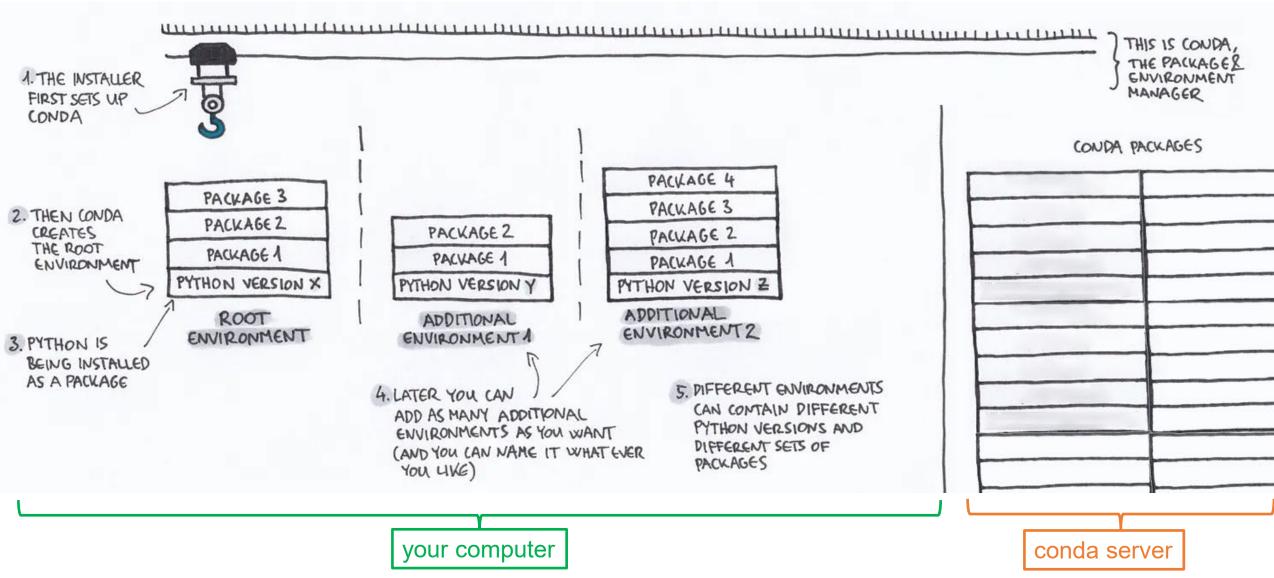
And why should you use them?

What is a virtual environment?

A virtual environment is an isolated directory on your computer that allows you to manage and install specific versions of Python and its packages without affecting the system-wide Python installation or other projects. By using virtual environments, you can maintain separate dependencies for different projects, ensuring compatibility and avoiding conflicts.

Have you ever...

- Had an issue where installing one Python package caused another previously working Python package to stop working?

- Needed to install a Python package on a system in which you do not have administrative privileges?

- Had trouble sharing or reproducing the environment in which you ran code successfully with other researchers?

- Had issues installing a Python package?

Virtual environments (e.g. conda) can help!

1. THE INSTALLER FIRST SETS UP CONDA

THIS IS CONDA, THE PACKAGE & ENVIRONMENT MANAGER

CONDA PACKAGES

2. THEN CONDA CREATES THE ROOT ENVIRONMENT

PACKAGE 3
PACKAGE 2
PACKAGE 1
PYTHON VERSION X

ROOT ENVIRONMENT

3. PYTHON IS BEING INSTALLED AS A PACKAGE

PACKAGE 2
PACKAGE 1
PYTHON VERSION Y

ADDITIONAL ENVIRONMENT 1

PACKAGE 4
PACKAGE 3
PACKAGE 2
PACKAGE 1
PYTHON VERSION Z

ADDITIONAL ENVIRONMENT 2

4. LATER YOU CAN ADD AS MANY ADDITIONAL ENVIRONMENTS AS YOU WANT (AND YOU CAN NAME IT WHATEVER YOU LIKE)

5. DIFFERENT ENVIRONMENTS CAN CONTAIN DIFFERENT PYTHON VERSIONS AND DIFFERENT SETS OF PACKAGES

your computer

conda server

Virtual environments can solve this ...

Had an issue where installing one Python package caused another previously working Python package to stop working.

by ...

resolving dependencies for all packages and notifying you of any conflicts between packages (before causing errors in existing installations).

Virtual environments can solve this ...

Needed to install a Python package on a system in which you do not have administrative privileges.

by ...

allowing you to install packages without having/using administrative privileges.

Virtual environments can solve this ...

Had trouble sharing or reproducing the environment in which you ran code successfully with other researchers.

by ...

allowing you to export your environment to a file which you can send to someone else so they can reproduce your environment.

Virtual environments can solve this ...

Had issues installing a Python package.

by ...

validating that all the dependencies of the application in question work for your Operating System (for packages available through conda/mamba).

Do not mess with your system Python!

- Avoid using administrative privileges, e.g, to not use
  `sudo pip install XXX`

- If you break the system Python it will be hard to recover.

- Instead, a virtual environment provides you with an isolated location/directory on your computer, where nothing bad will happen if you mess it up. (As a last resort, you can always delete that directory, reinstall Python+conda, and start fresh.)

Creating virtual environments via conda/mamba

Two different ways

(Ana)conda

- There are a few ways to create Python virtual environments. We will focus on using `conda`/`mamba`.

- Regardless of the install method, the main components include
  - The executable file(s) `conda` or `mamba` (on your computer)
  - Folder/directory containing all your environments and packages (on your computer)
  - "Channels" with pre-tested packages (on a separate server); I will explain these later.

Anaconda Distribution using the Navigator

Pro:

- Easy installation

- No command-line needed

- Point and click

Con:

- Slower to load, lots of bloat

- Confusing recent changes to their Terms of Service (TOS)

- TOS says: Anaconda cannot be used for commercial purposes at Northwestern
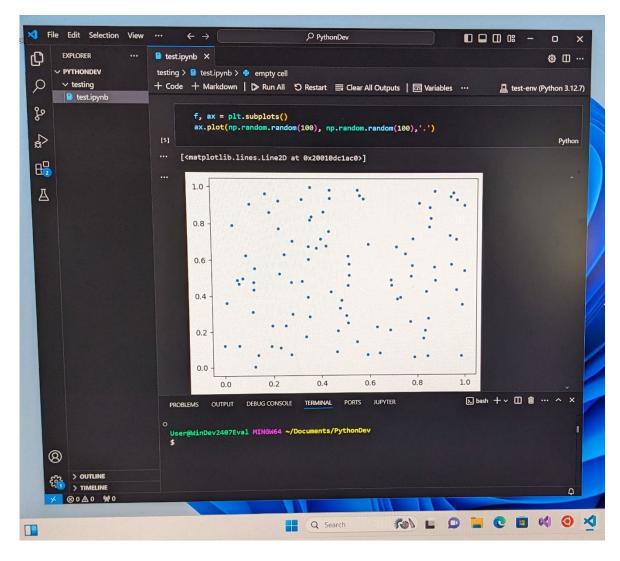
- **WARNING** : their TOS may continue to change

Open-source version using miniforge (and optionally VS Code)

Pro:

- No restrictions on use

- Fast and streamlines

- VS Code is a very powerful general-purpose IDE

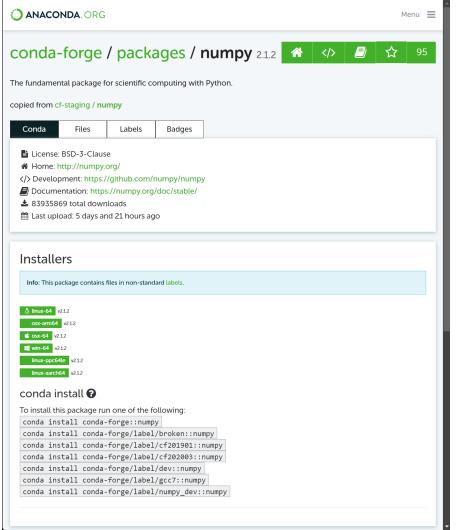- VS Code has great integration with GitHub

Con:

- Need to use some command-line (to create envs)

- More installation steps (I installed VS Code, Git Bash in addition to miniforge)

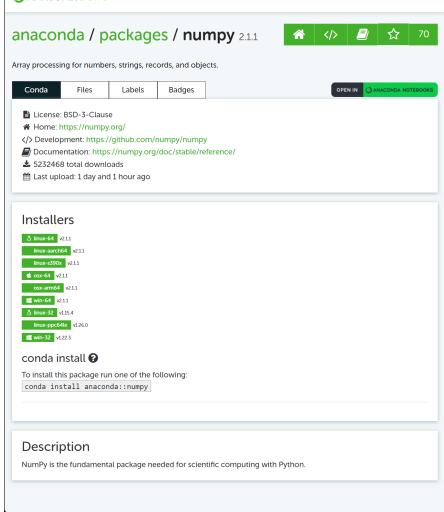- Steeper learning curve than Anaconda Navigator

Channels

- Servers containing pre-tested packages that work well together
- Many options

conda-forge : community driven

default/anaconda : "proprietary"

Common commands (if using command line)

$ `mamba create`   Create a new environment (requires other arguments, e.g. name)

$ `mamba install`   Install a package within an environment (add package name)

$ `mamba activate`   Activate/enter an environment (add environment name)

$ `mamba search`   Search for a particular package (add package name)

$ `mamba list`   List all packages installed in the active environment

(You can replace `mamba` with `conda` and get the same result; `mamba` is usually faster.)

[Link to the conda cheat sheet](#)

Exercises

Your turn!

Create two environments and test some code

1. Create two environments (I provide shell commands below, but if you are working in the Anaconda Navigator you can do this in the Navigator GUI via point-and-click and hand-selecting these packages):

    ```
    conda create --name numpy1-test python=3.12 numpy=1.26.4 jupyter

    conda create --name numpy2-test python=3.12 numpy=2.1.2 jupyter
    ```

2. Go to [the workshop's GitHub repo](), download the two test notebooks, and run them both in both environments. (One should only work in `numpy1-test`, and the other should only work in `numpy2-test`)

3. Experiment with creating your own virtual environment(s) and then running your own code. For example, you may want to install `python, numpy, pandas, jupyter, matplotlib`

Hint: In order to switch environments in the Anaconda Navigator, you need to enter the environments tab to activate a given environment. Once an environment is activated, you can then go back to the main screen of the Navigator and open Jupyter. If you are working from the terminal (i.e., you installed miniforge), you will want to use the `activate` and `deactivate` commands. If you are working inside VS Code, you can also switch your environment via the `Select Kernel` button in the upper right of your window when viewing either .ipynb file (this button may way the active environment's name instead of "Select Kernel").

Questions?