

# Next Steps in Python: *Virtual Environments*

Aaron Geller

(with help from Scotty Coughlin)

Research Computing and Data Services

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](https://bit.ly/rcdsconsult).

# See many other upcoming workshops :

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

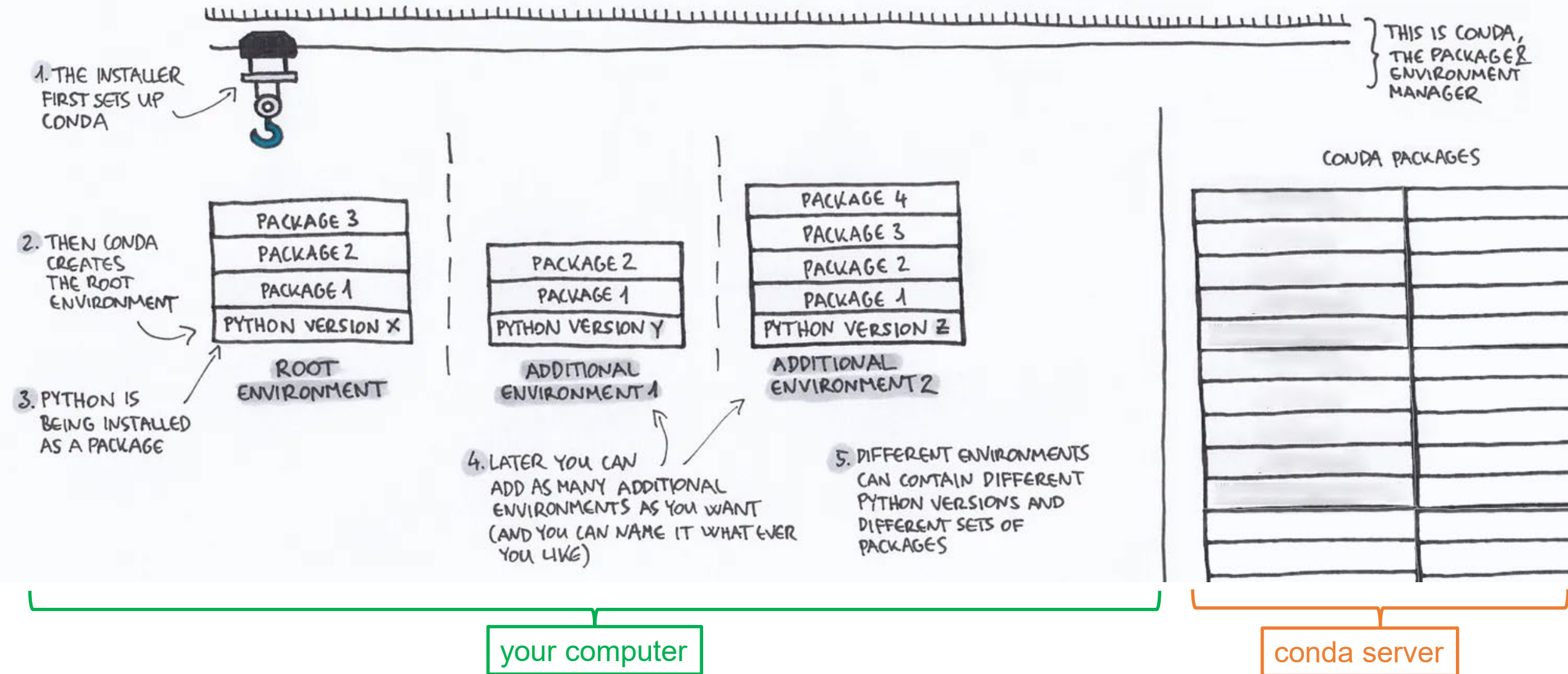
# What is a virtual environment?

And why should you use them?

# 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!



# 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 using `sudo` or having 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 as a YAML 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 ...

pre-compiling and 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 `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 `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)

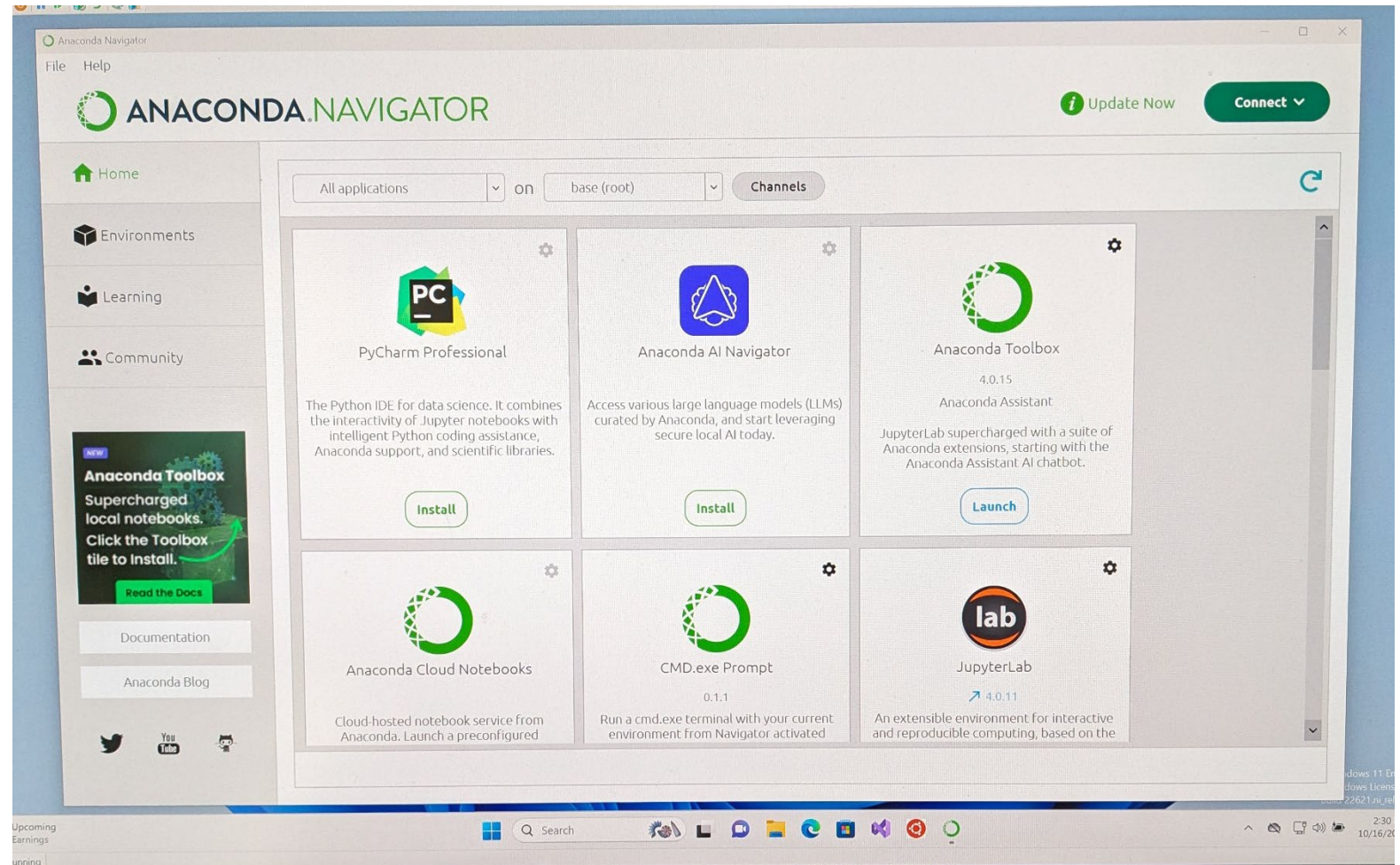
# Anaconda Distribution using the Navigator

## Pro:

- Easy installation
- No command-line needed
- Point and click

## Con:

- Slower to load, lots of bloat
- Confusing changes to TOS recently
- Cannot be used for commercial purposes at Northwestern





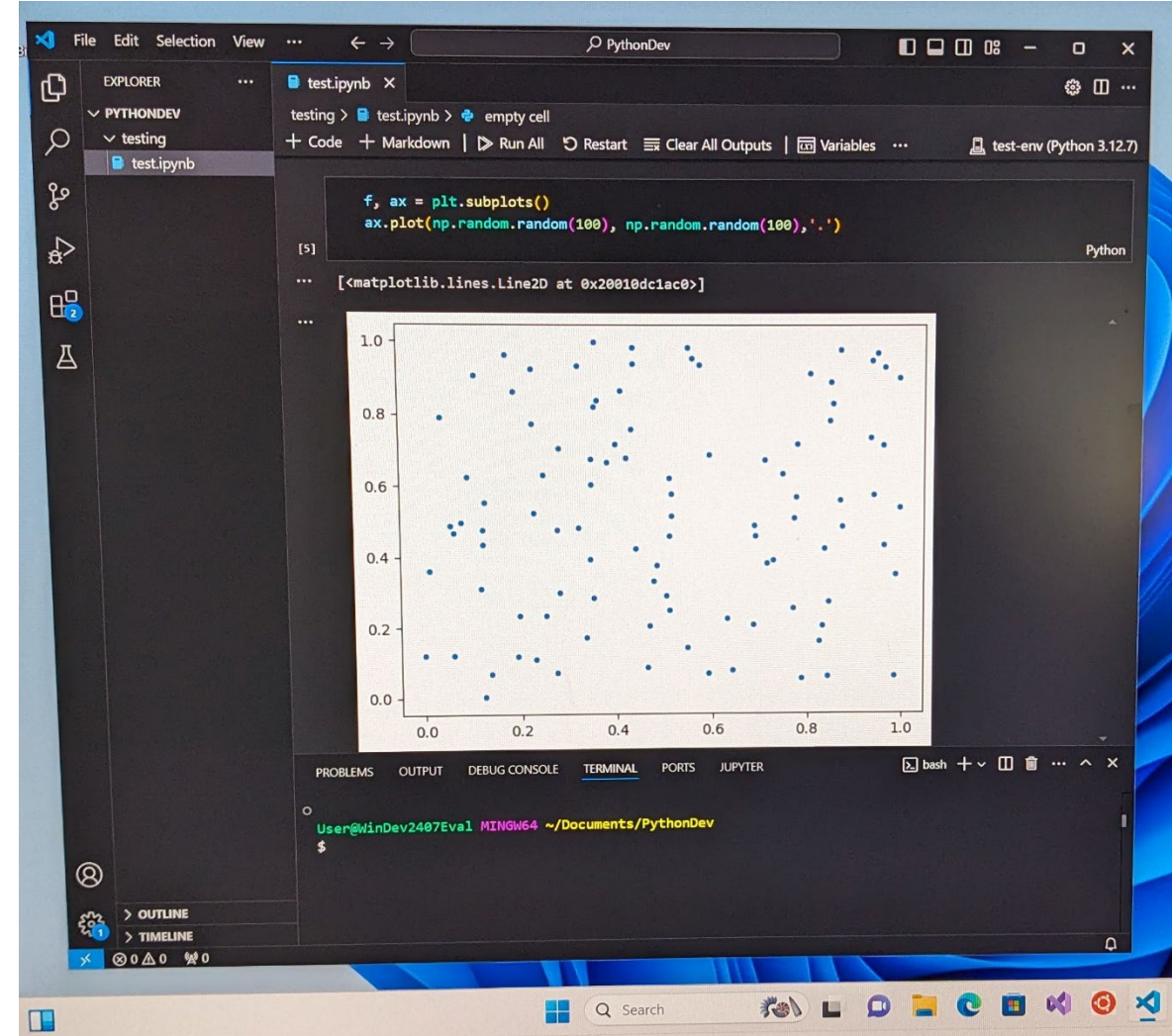
# Open source version using VS Code and miniforge

## Pro:

- VS Code is a very powerful general-purpose IDE
- No restrictions on use
- Great integration with GitHub

## Con:

- Need to use some command-line (to create envs)
- More installation steps (I installed VS Code, Git Bash and miniforge)
- Steeper learning curve than Anaconda Navigator



# Channels

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

## conda-forge : community driven

The screenshot shows the Anaconda.org page for the `conda-forge / packages / numpy 2.1.2` package. The page includes a description: "The fundamental package for scientific computing with Python." and a note that it was "copied from cf-staging / numpy". It features tabs for "Conda", "Files", "Labels", and "Badges". The "Conda" tab is active, displaying the license (BSD-3-Clause), home page (<http://numpy.org/>), development link (<https://github.com/numpy/numpy>), documentation link (<https://numpy.org/doc/stable/>), total downloads (83935869), and last upload time (5 days and 21 hours ago). Below this, the "Installers" section lists various platform-specific installers for version 2.1.2, including `linux-64`, `osx-arm64`, `osx-64`, `win-64`, `linux-ppc64le`, and `linux-aarch64`. A note states: "Info: This package contains files in non-standard labels." The "conda install" section provides a list of commands to install the package from different channels, such as `conda install conda-forge::numpy` and `conda install conda-forge/label/broken::numpy`.

## default/anaconda : “proprietary”

The screenshot shows the Anaconda.org page for the `anaconda / packages / numpy 2.1.1` package. The page includes a description: "Array processing for numbers, strings, records, and objects." It features tabs for "Conda", "Files", "Labels", and "Badges". The "Conda" tab is active, displaying the license (BSD-3-Clause), home page (<https://numpy.org/>), development link (<https://github.com/numpy/numpy>), documentation link (<https://numpy.org/doc/stable/reference/>), total downloads (5232468), and last upload time (1 day and 1 hour ago). Below this, the "Installers" section lists various platform-specific installers for version 2.1.1, including `linux-64`, `linux-aarch64`, `linux-s390x`, `osx-64`, `osx-arm64`, `win-64`, `linux-32`, `linux-ppc64le`, and `win-32`. The "conda install" section provides a single command to install the package: `conda install anaconda::numpy`. The "Description" section states: "NumPy is the fundamental package needed for scientific computing with Python."



# Common commands (if using command line)

<code>\$ mamba create</code>	Create a new environment (requires other arguments, e.g. name)
<code>\$ mamba install</code>	Install a package within an environment (add package name)
<code>\$ mamba activate</code>	Activate/enter an environment (add environment name)
<code>\$ mamba search</code>	Search for a particular package (add package name)
<code>\$ mamba list</code>	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:

```
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.

Questions?

Thank you!