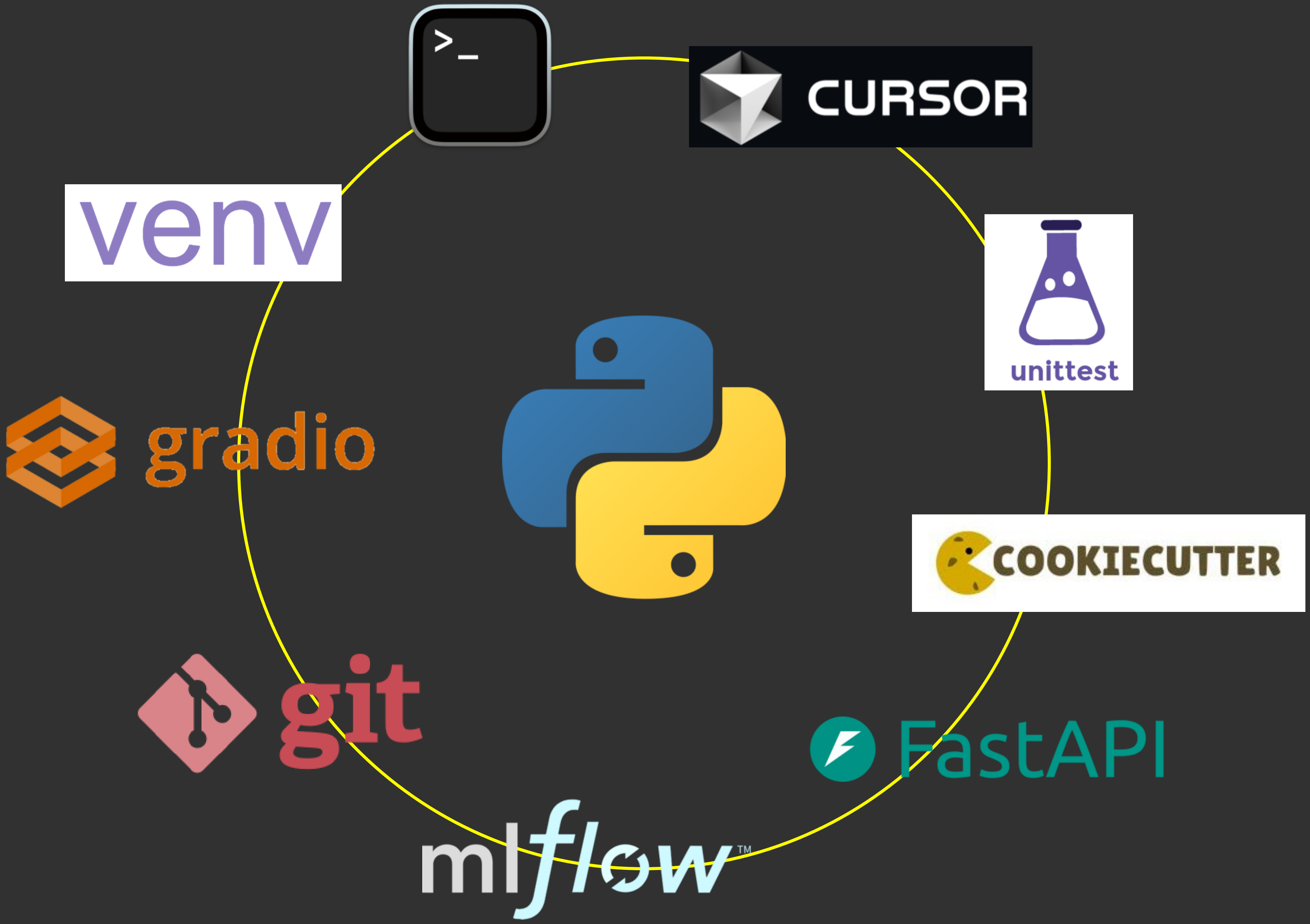
DAPT 619

ANALYTICS COMPUTING

Vishal Patel

Fall 2025


>_
CURSOR
unittest
COOKIECUTTER
FastAPI
mlflow™
git
gradio
venv

Beyond Python: Why Ecosystem Tools Matter

1. Team Collaboration & Version Control

Coding is a team sport.

2. Structured Projects & Environments

Avoid “works on my machine” problems

3. Quality Assurance & Testing

Test early, fail less.

4. Deployment & Accessibility

Share your code as apps or services.

5. Reproducibility & Automation

Make your work traceable and scalable.

6. AI-Era Skillset

Knowing *how to code* is less important now than *how to work with* code.

Command Line Interface (CLI)

Operator's Console
(for an IBM 701)

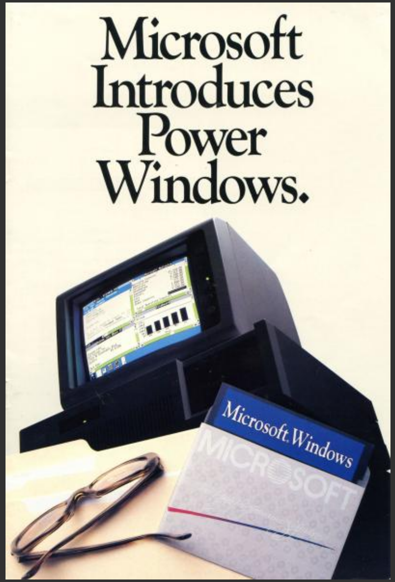
1950s

The First IBM PC
(IBM 5150)

1981

The First Commercial PC with
GUI (Lisa by Apple)

1983



Microsoft
Window 1.0

1985

1984 Super Bowl Apple Macintosh Ad

Point & Click

GUI

1. A software mechanism used to interact with your operating system using a mouse.
2. You can use a GUI to visually navigate and click on icons and images to make things work.
3. However, a GUI is inefficient for system administration tasks, especially if the environment is virtual or remote.

Type & Enter

CLI

1. A software mechanism you use to interact with your operating system using your keyboard.
2. You can enter text commands to configure, navigate, or run programs on any server or computer system.
3. All operating systems—including Linux, macOS, and Windows—provide a CLI for faster system interaction.

Source: What is CLI? (https://aws.amazon.com/what-is/cli/)

Why Command Line?

1. It makes you more **flexible**, **faster** and more **efficient**.
2. It’s less resource intensive. It’s **scalable**.
3. The command line is **ubiquitous**.
4. The skills are **highly valued**.
5. It makes you look like a **cool hacker** ☺
6. It’s **easier** than you might think!

Mac OS

Terminal

Unix-based!

Windows

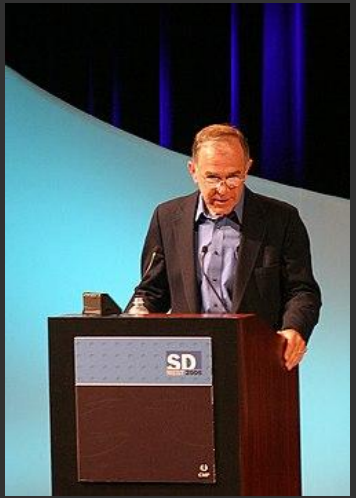
Command Prompt

PowerShell

WSL

git bash

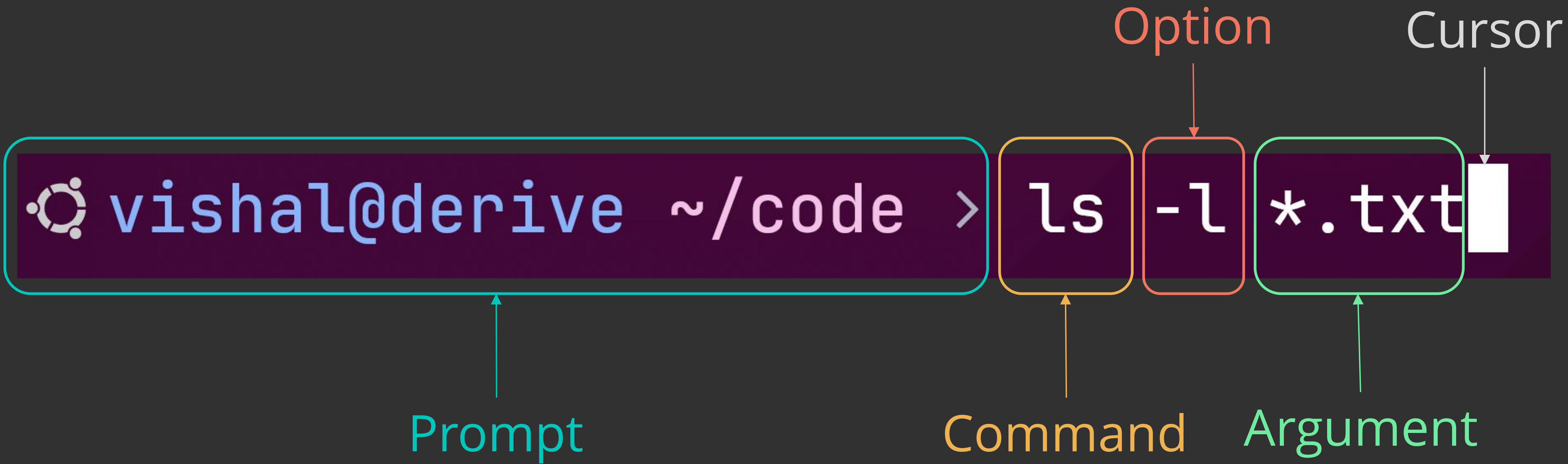
A software designed to execute on a Unix style command-line environment.



Steve Bourne, creator of the Bourne shell (**sh**) in 1976 at Bell Labs.

Shell: A terminal application used to interact with an OS through written commands.

Anatomy of a Command Line



```
vishal@derive ~/code > ls -l *.txt
```

Option

Cursor

Prompt

Command

Argument

Command Line Interface (CLI)

What does pwd print?

A. Your username
B. Your current working directory
C. The last command you ran
D. The parent directory

Fill in the blank:

`cd` ____ moves you one directory up from where you are.

You run:

```
$ cd /Users/vishal/projects
$ pwd
```

What will `pwd` output?

A. `projects`

B. `/Users/projects`

C. `/Users`

D. `/Users/vishal/projects`

True or False: cd with no arguments takes you to your home directory.

You want to go into a folder named `Project Files` inside your home directory. Which one works?

A. `cd ~/Project Files`

B. `cd ~/"Project Files"`

C. `cd ~/'Project Files'`

D. `B or C`

Given this directory tree:

```
/home/student
├─ notes
│   └─ week1
└─ demos
    └─ cli
```

Starting from `/home/student`, write the exact command to move into the `cli` folder.

Predict the output:

Assume you’re in `/home/student/course/week1` and you run:

```
$ cd ../../..
$ pwd
```

What prints?

You run:

```
$ cd /var/log
$ cd -
```

Where do you end up?

What command prints just your **home directory path** without moving you?

You're in `/tmp`. In one command, jump to your home directory's `Downloads` folder using `~`.

Git

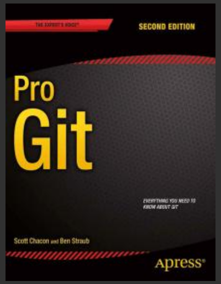
Version Control System: Save As

```
my_code.py
my_code_sept2025.py
my_code_sep2025_v2.py
my_code_sep2025_v2_final.py
my_code_sep2025_v2_final_edited.py
my_code_sep2025_v2_final_edited_FINAL.py
```

With the rise of “vibe coding” (aka AI-assisted coding), version control is more useful than ever.

Version Control System

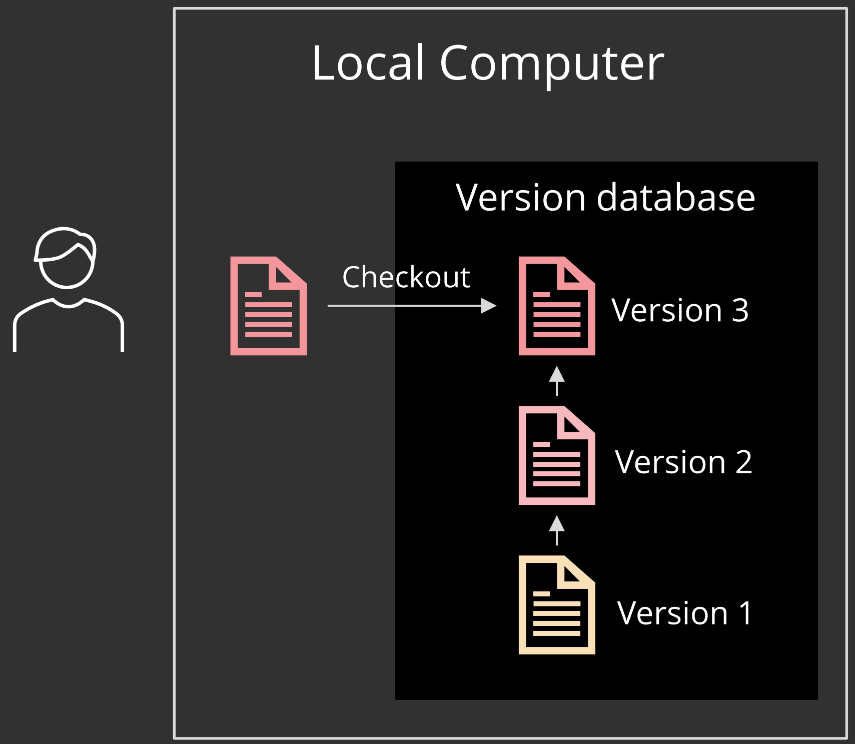
A system that records changes to a file or set of files over time so that you can recall specific versions later.



Version Control Systems

1. Local
2. Centralized
3. Distributed

1 Local



Local Computer

Version database

Checkout

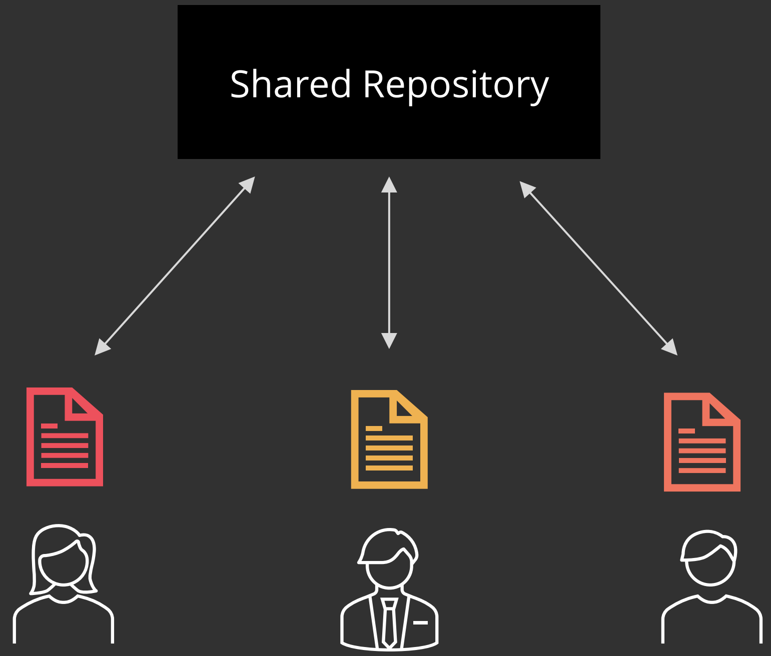
Version 3

Version 2

Version 1

The entire history of the project in a single place!

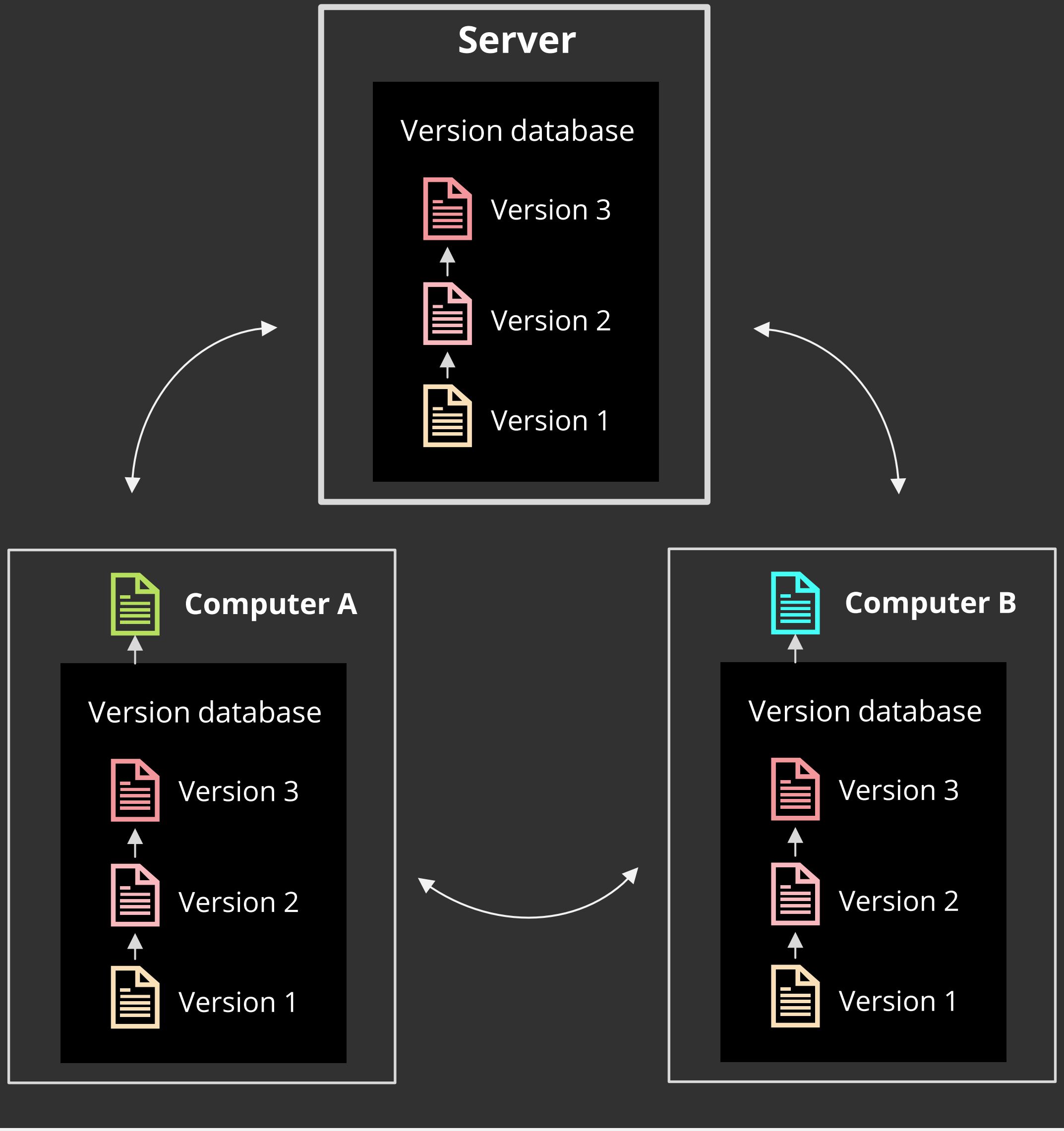
2 Centralized



Shared Repository

Single point of failure!

3 Distributed



- Clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.
- Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it.
- Every clone is really a full backup of all the data.

Goals of Git

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

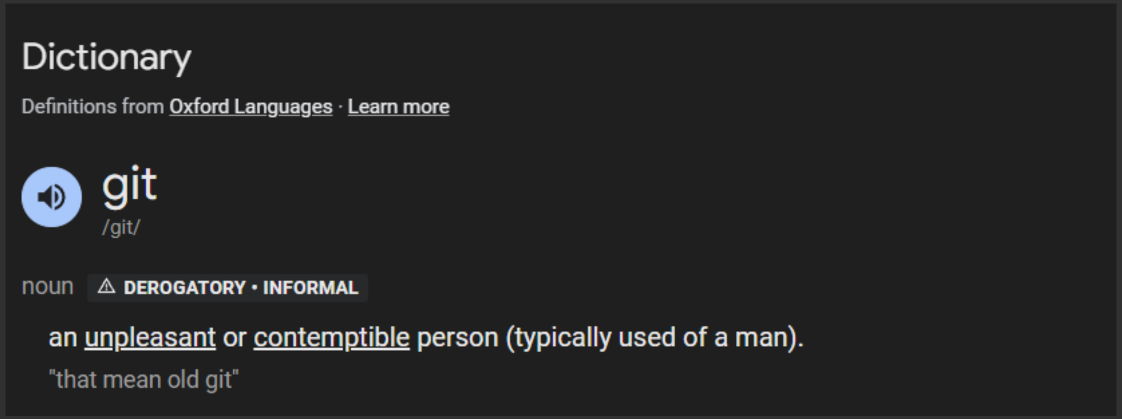
A Brief History of Git

Linus Torvalds, the creator of the Linux kernel (1991)

BitKeeper
Scalable Version Control

Git is the de facto version control system since 2005.

”I'm an egotistical bastard,
and I name all my projects after myself.
First 'Linux', now 'git’.”



Dictionary

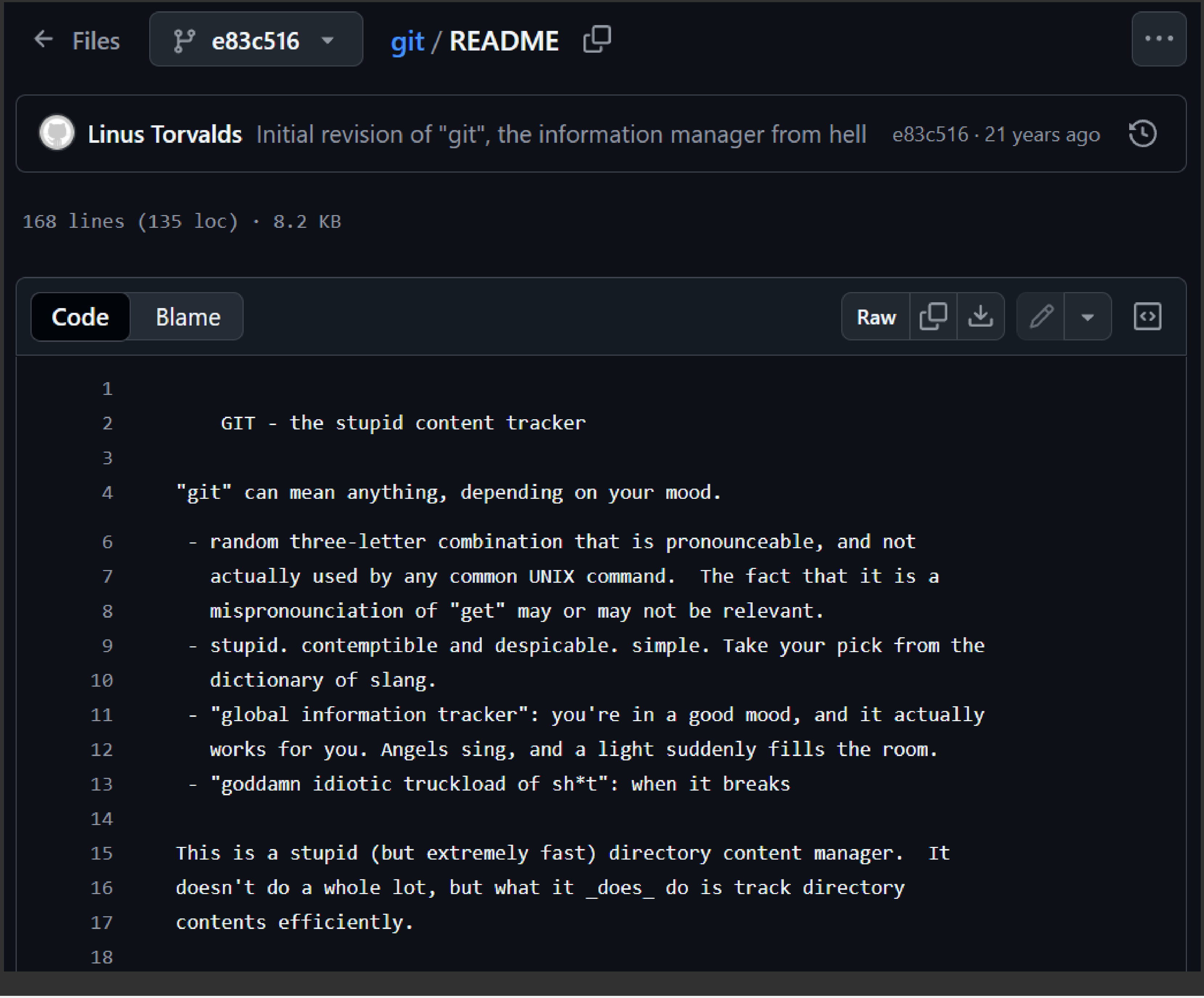
Definitions from Oxford Languages · Learn more

git
/git/

noun ⚠ DEROGATORY • INFORMAL

an unpleasant or contemptible person (typically used of a man).

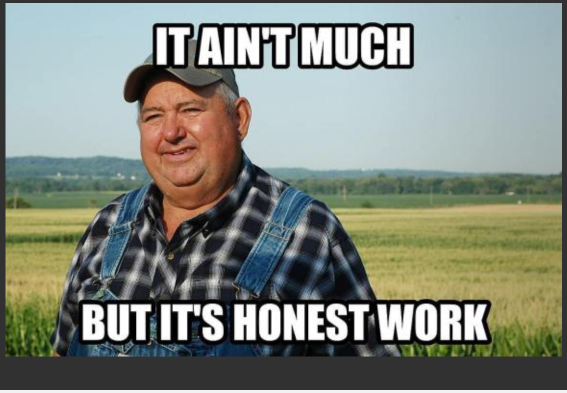
"that mean old git"

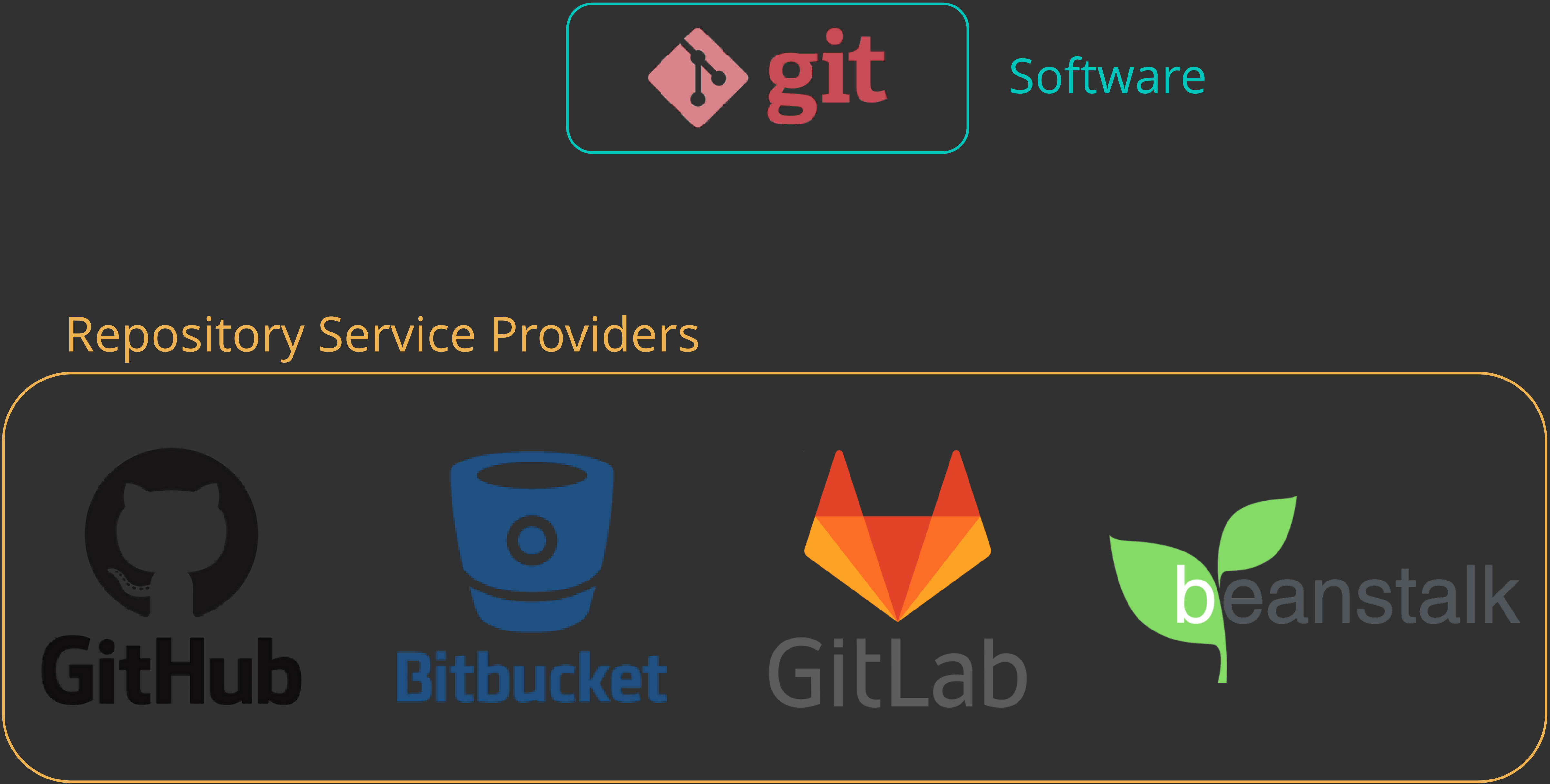


Linus Torvalds Initial revision of "git", the information manager from hell e83c516 · 21 years ago

168 lines (135 loc) · 8.2 KB

```

  GIT - the stupid content tracker

"git" can mean anything, depending on your mood.
 - random three-letter combination that is pronounceable, and not
   actually used by any common UNIX command.  The fact that it is a
   mispronounciation of "get" may or may not be relevant.
 - stupid. contemptible and despicable. simple. Take your pick from the
   dictionary of slang.
 - "global information tracker": you're in a good mood, and it actually
   works for you. Angels sing, and a light suddenly fills the room.
 - "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager.  It
doesn't do a whole lot, but what it _does_ do is track directory
contents efficiently.

```

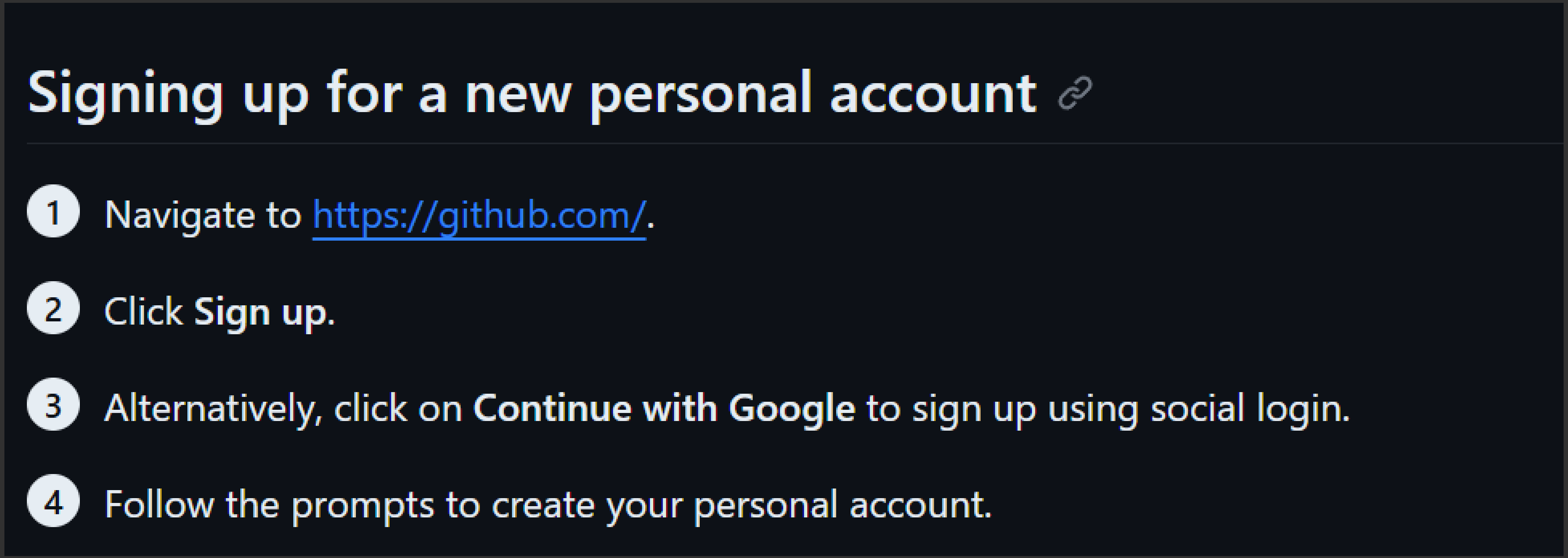




Software

Repository Service Providers

Create a personal account on github.com.



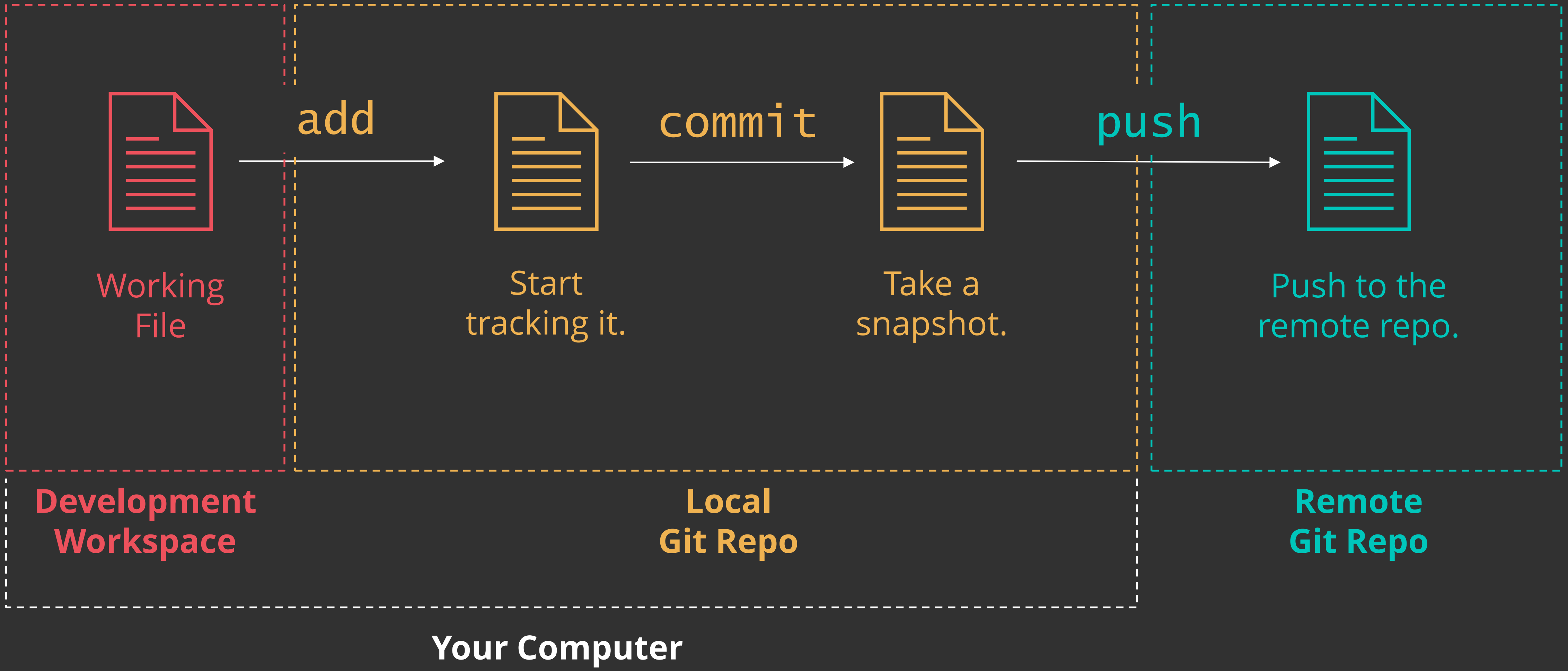
Signing up for a new personal account

1. Navigate to https://github.com/.
2. Click **Sign up**.
3. Alternatively, click on **Continue with Google** to sign up using social login.
4. Follow the prompts to create your personal account.

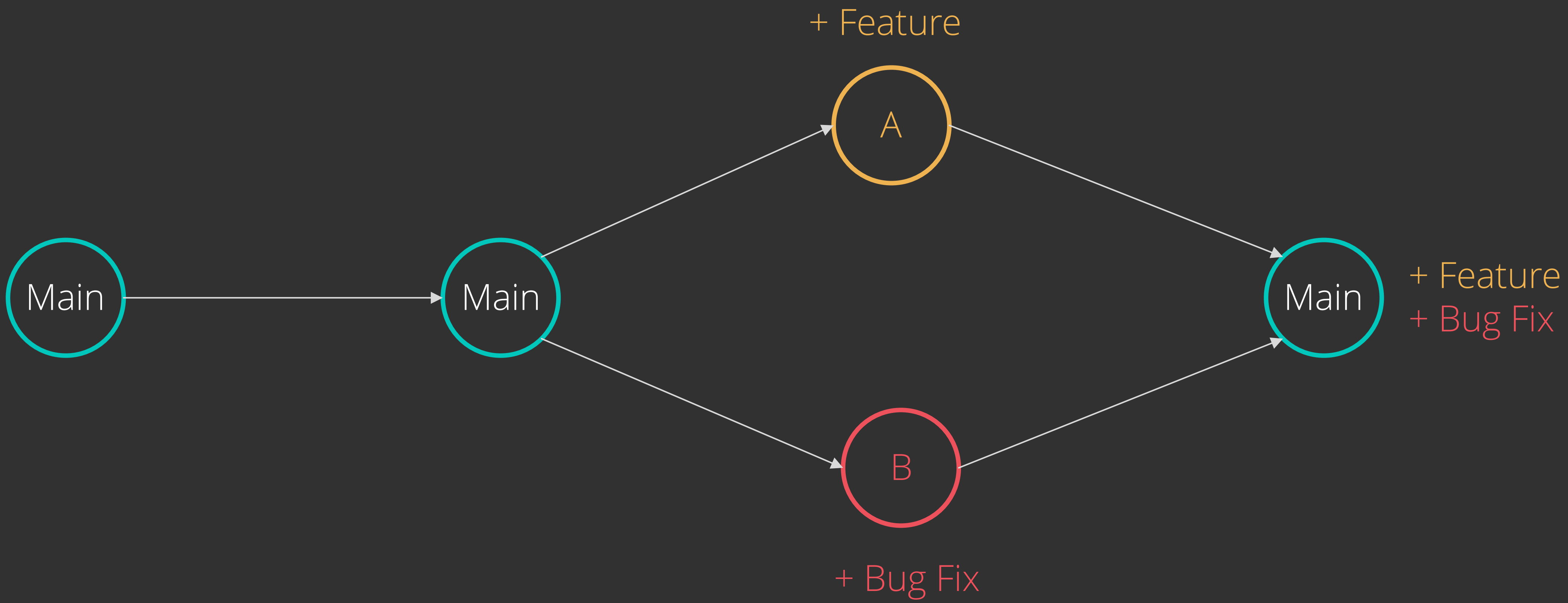
https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

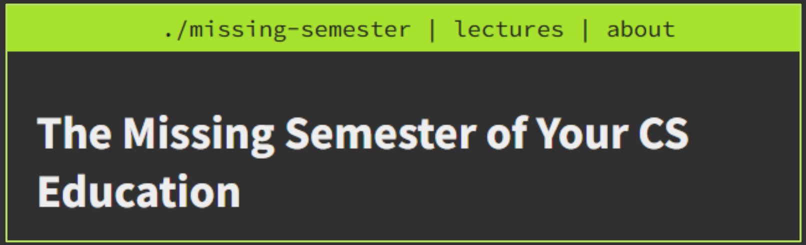
Git Process: Add, Commit, Push



Git Branching



Git



The Missing Semester of Your CS Education

What does `git clone <URL>` do?

A. Copies a remote repo to your machine.
B. Creates a blank repo on GitHub.
C. Downloads only the latest commit.
D. Copies your current folder into the remote.

Put these in the most common order for first-time work on a repo:

```
commit, push, clone, add
```

You just ran:

```
echo "hello" > notes.txt
git status
```

Which state is `notes.txt` in?

A. Untracked

B. Staged

C. Committed

D. Pushed

You created two files: `a.py`, `b.py`. You only want to stage `a.py`.

Which command you should use to do this?

True of False: `git push` sends your local commits to the remote repository.

To create a new branch named `feature-login`:

A. `git branch feature-login`

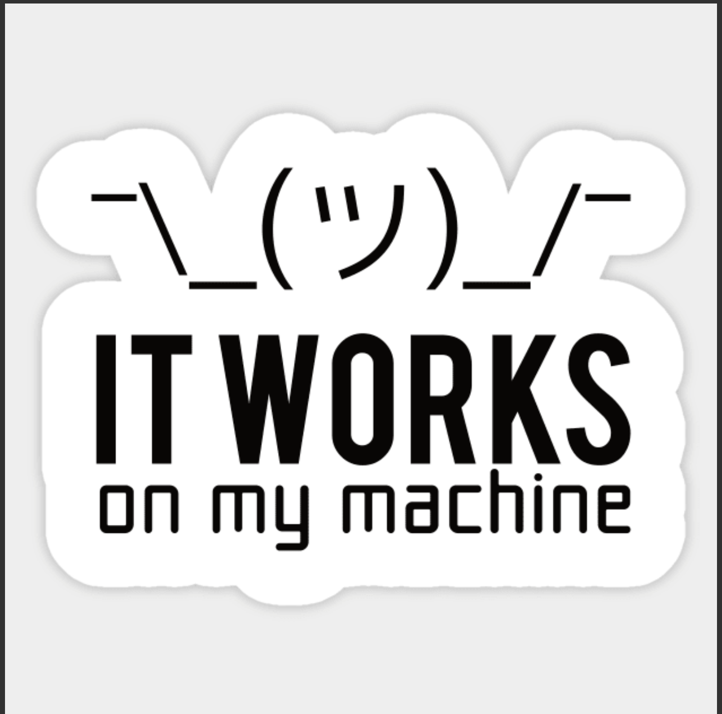
B. `git -b feature-login`

C. `git branch --new feature-login`

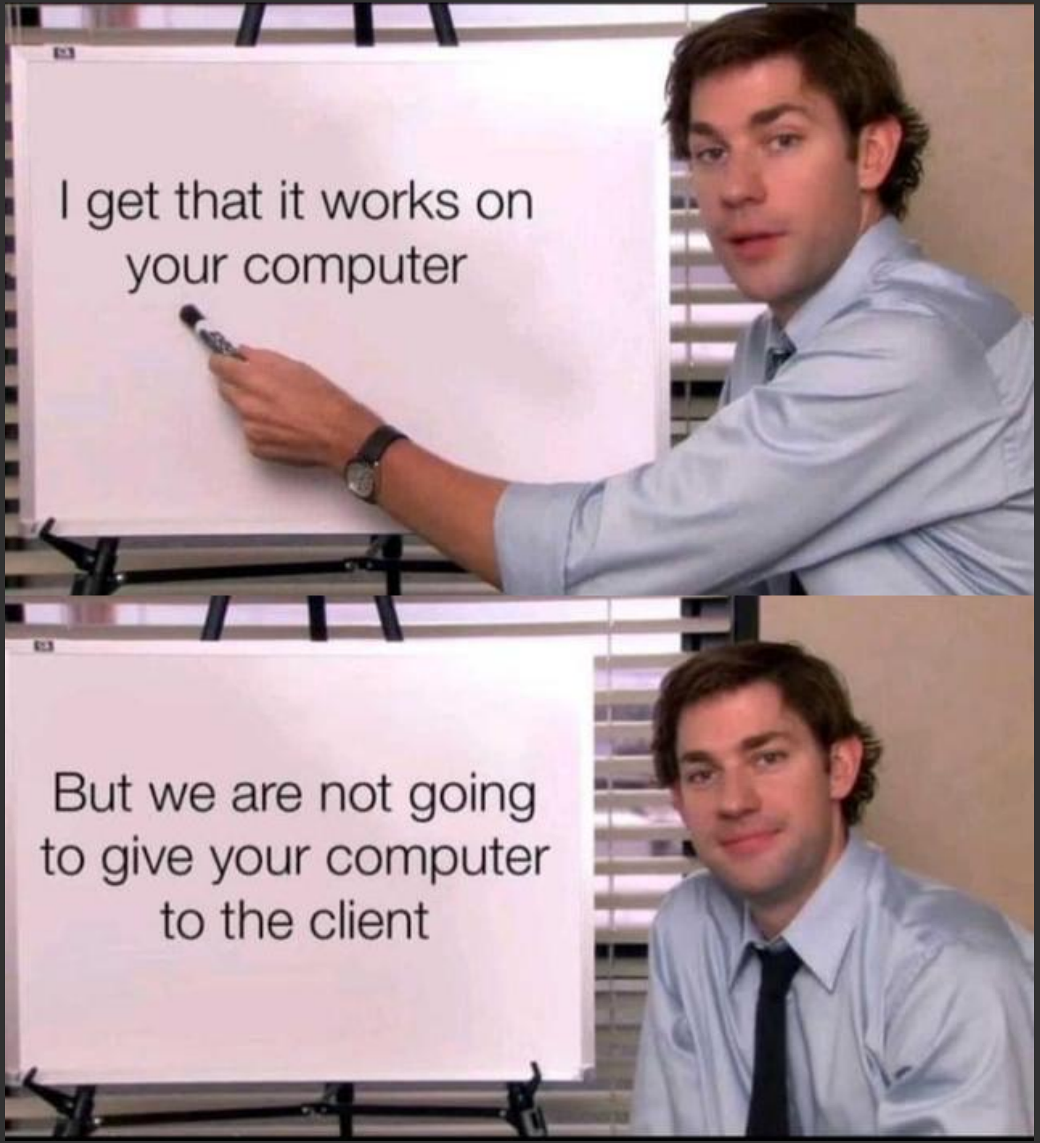
D. `git push feature-login`

What’s the difference between **staging** and **committing**?

Python Virtual Environment (venv)



¯_(ツ)_/¯
IT WORKS
on my machine

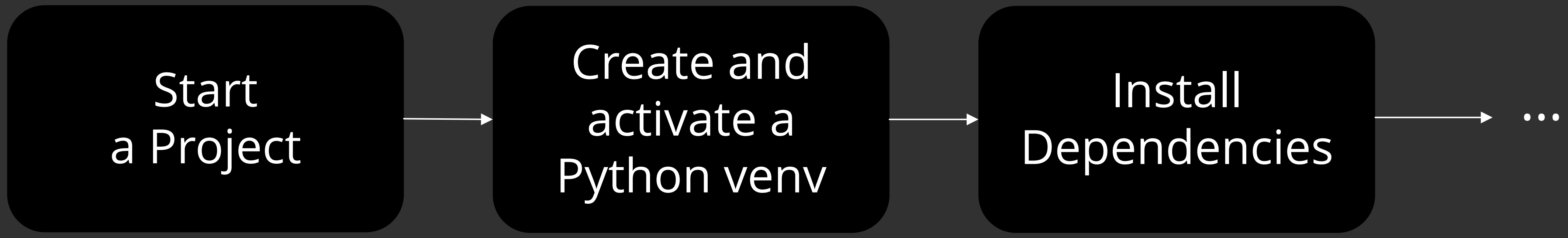


I get that it works on your computer
But we are not going to give your computer to the client

1. Avoid dependency conflicts.

2. Ensure easy reproducibility.

3. Cleanliness is next to Godliness ☺

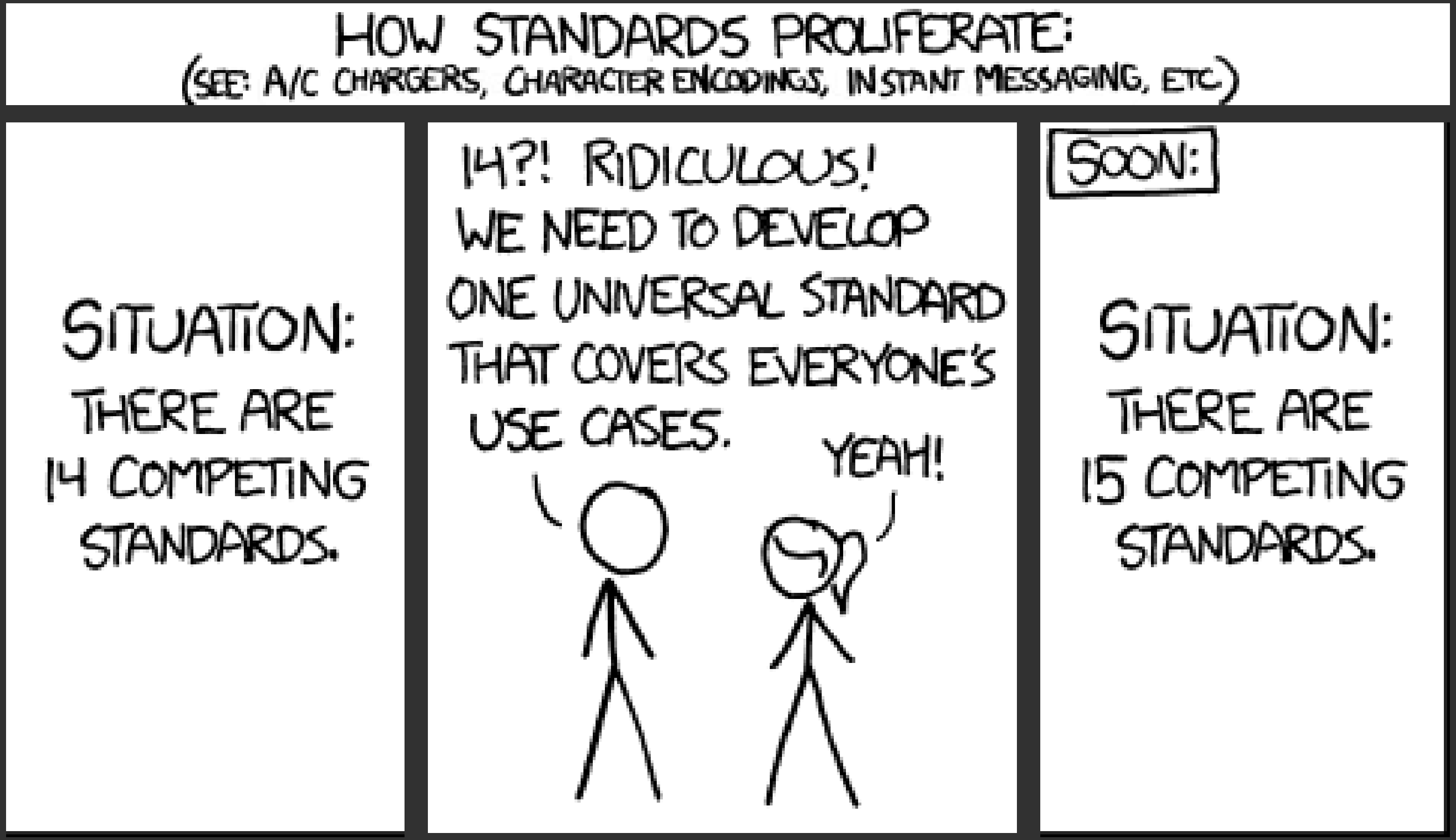


Start a Project

Create and activate a Python venv

Install Dependencies

…

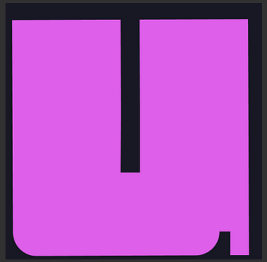

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Creating a Virtual Environment (venv)

pyenv

ANACONDA

virtualenv



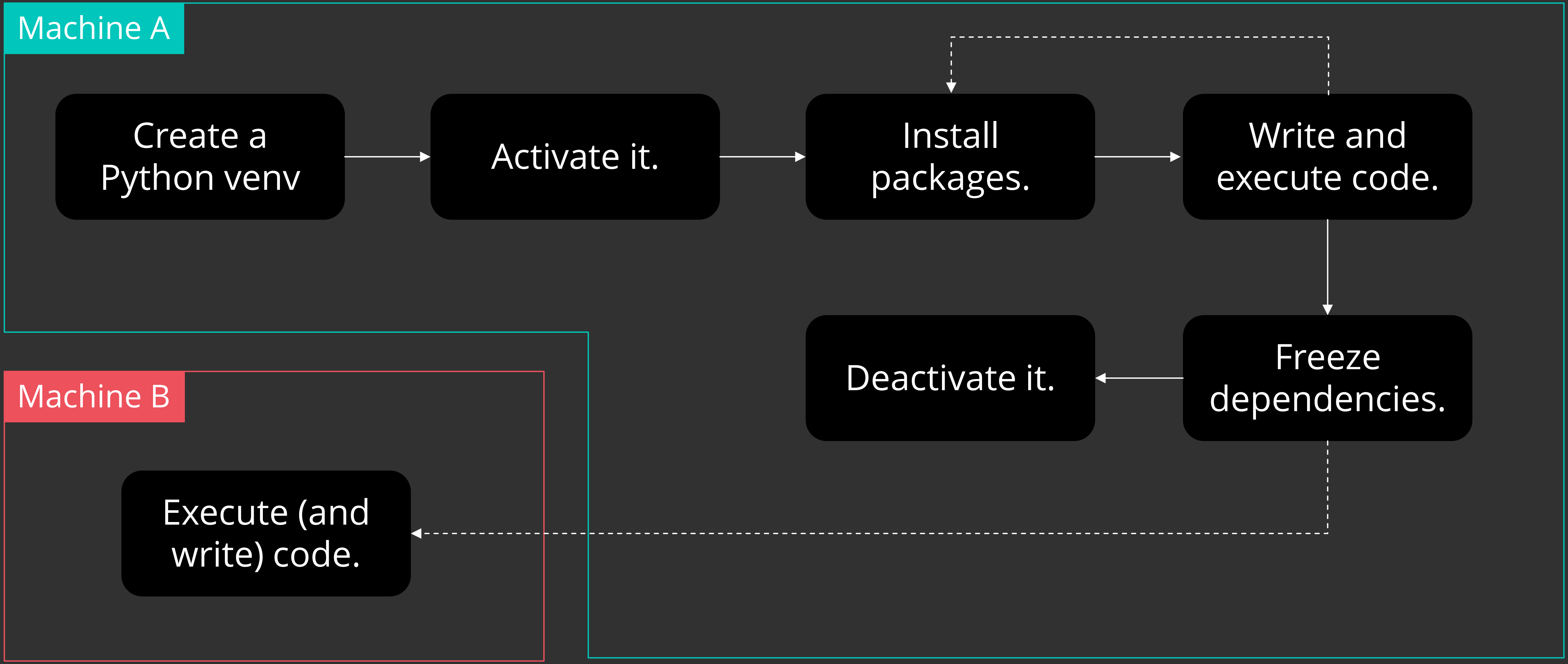
Poetry

```
python -m venv <venv-name>
```

How It Works: $PATH

- When you run the `python` command, or execute code in Jupyter Notebook, the Python interpreter looks for a python executable file in the system environment variable called $PATH…
- … Unless you are inside a virtual environment.
- When you create a venv, a new (hidden) directory is created. It contains:
  - A directory named `site_packages` that will store all installed packages.
  - A file named `pyvenv.cfg` which includes a path to the Python executable.
  - The Python interpreter will now use this Python executable to execute Python scripts.
  - And all Python packages will be installed within the “local” `site_packages` directory.

The Process



Machine A
Create a Python venv
Activate it.
Install packages.
Write and execute code.
Freeze dependencies.
Deactivate it.
Machine B
Execute (and write) code.

Python Virtual Environment

What’s the main reason to use a virtual environment?

A. Make Python run faster
B. Keep project dependencies isolated
C. Share secrets with teammates
D. Compile Python to C

What’s the command to create a virtual environment named `vcu` using the standard library?

True or False: `deactivate` deletes the virtual environment folder.

Put these in the usual order for a fresh project:

A. `pip install -r requirements.txt`

B. `python -m venv .venv`

C. `pip freeze > requirements.txt`

D. `activate the venv`

You forgot to activate and installed packages globally by accident. Describe a way to undo or mitigate that.

Unit Testing

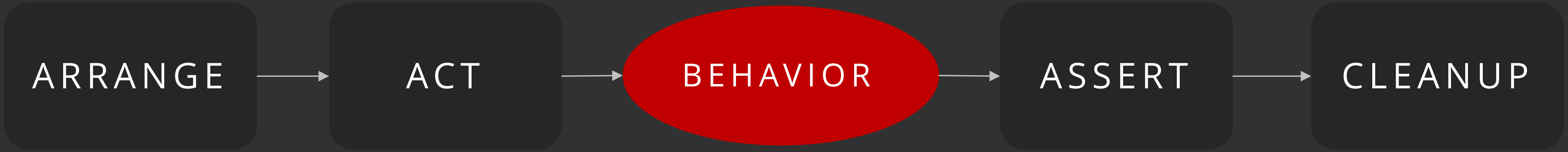
1. Is the code working as expected?

2. Any edge cases where the code fails?

pytest

Anatomy of a Test

A test checks the outcome of a specific behavior and ensures it aligns with the expectations.



ARRANGE → ACT → BEHAVIOR → ASSERT → CLEANUP

ARRANGE: Prepare everything for the test, e.g., enter records into a database.

ACT: A singular, state-changing action that kicks off the behavior under evaluation.

BEHAVIOR: How the system acts in response to the given situation or stimulus.

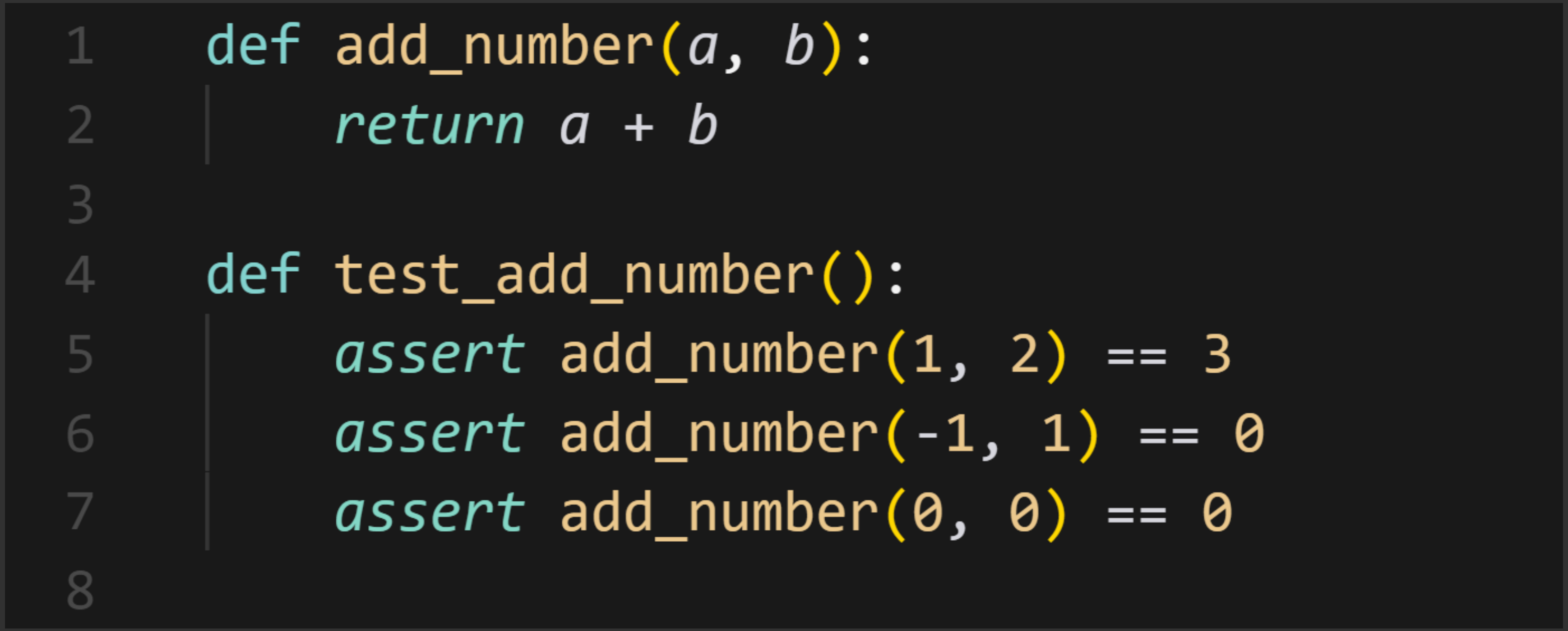
ASSERT: Check the resulting state to ensure it aligns with what was expected.

CLEANUP: The test picks up after itself, so it doesn’t unintentionally affect other tests.

The focus is less on *how* or *why* it happened, and more on *what* happened.

https://docs.pytest.org/en/stable/explanation/anatomy.html

Example



```python
def add_number(a, b):
    return a + b

def test_add_number():
    assert add_number(1, 2) == 3
    assert add_number(-1, 1) == 0
    assert add_number(0, 0) == 0
```

Pytest

```
pip install pytest
```

Pytest Exercise

1. Create a new Python virtual environment for this exercise or reuse the one that we created in the previous section.
2. Install `pandas`.
3. Write a simple function to load `heart.csv` file into a Pandas dataframe.
4. Write a test to verify that the dataframe contains twelve columns.
5. Check if you already have `pytest` in the virtual environment. Install it if needed.
6. Run the test and check the results. The test should **pass**.
7. Write another test to verify that there are no null values in any columns.
8. Run the test and check the results. The test should **pass**.
9. Write another test to verify that the column `heart_disease` is present.
10. Run the test and check the results. Inspect the results and figure out why the test failed.

Project Code Structure

Noble's Principles

1. Someone unfamiliar with your project should be able to look at your files and understand in detail what you did and why.

2. Everything you do, you will probably have to do over again.

A Quick Guide to Organizing Computational Biology Projects:
https://dx.plos.org/10.1371/journal.pcbi.1000424

“Let us change our traditional attitude to
the construction of programs.

Instead of imagining that our main task is
to instruct a computer what to do,
let us concentrate rather on explaining to human beings
what we want a computer to do.”

– Donald E. Knuth

Elements of a Data Science Project

README.md

Why this project exists, how things are organized, conventions used in the project, etc.

Data

Raw, Interim, Processed

Models

Model artifacts

Notebooks

Jupyter Notebooks

Reports

Plots, Excel files,...

Scripts

`.py` files

Tests

Unit tests

Configs

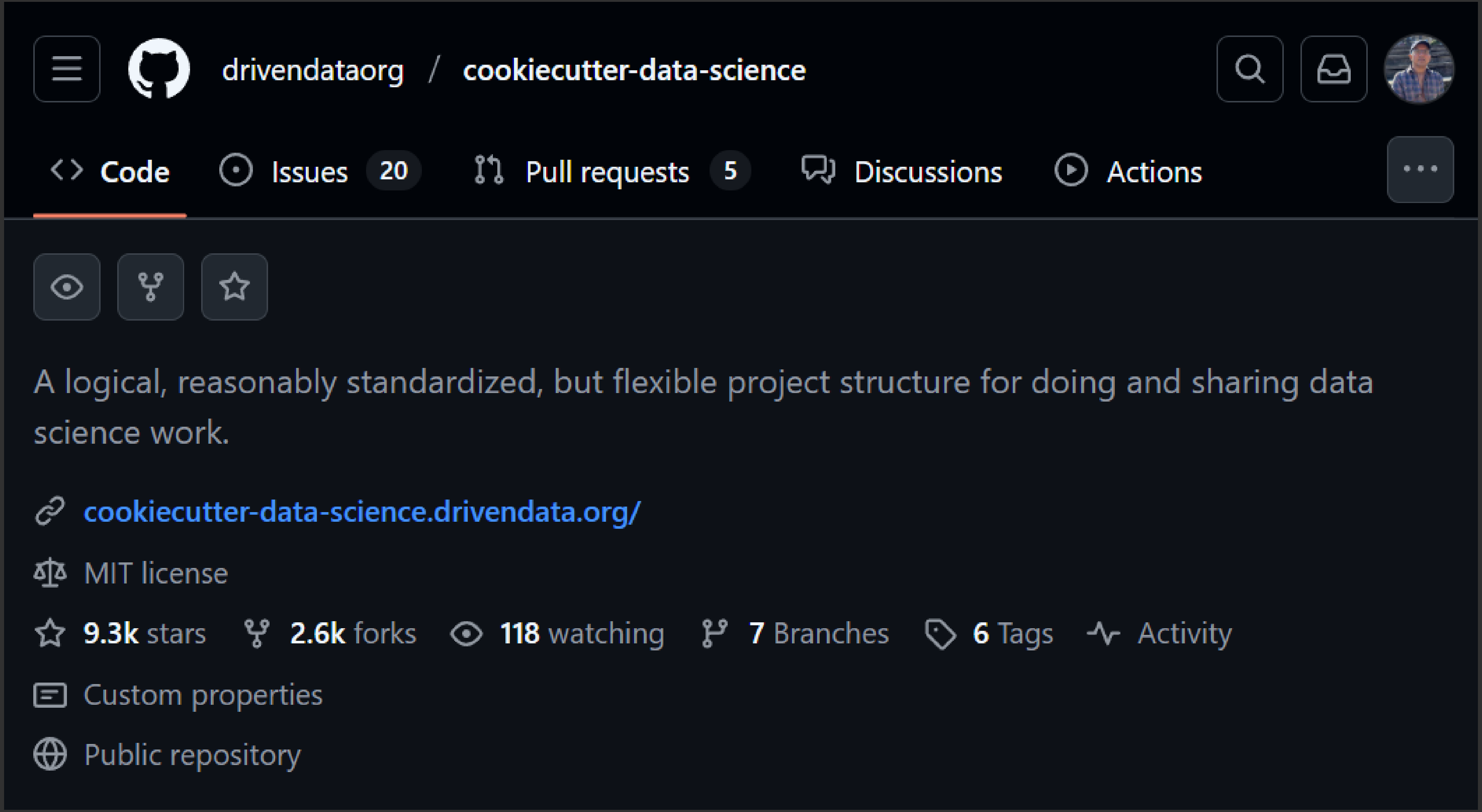
Paths, options, ...

Secrets

API keys, ...

Requirements

Python dependencies



drivendataorg / cookiecutter-data-science

A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.

cookiecutter-data-science.drivendata.org/

MIT license

9.3k stars 2.6k forks 118 watching 7 Branches 6 Tags Activity

Custom properties

Public repository

Cookiecutter Data Science: Opinions

1 Data analysis is a directed acyclic graph.

The best way to ensure reproducibility is to treat your data analysis pipeline as a directed acyclic graph (DAG).

This means each step of your analysis is a node in a directed graph with no loops. You can run through the graph forwards to recreate any analysis output, or you can trace backwards from an output to examine the combination of code and data that created it.

Cookiecutter Data Science: Opinions

2 Raw data is immutable.

That proper data analysis is a DAG means that raw data must be treated as immutable—it's okay to read and copy raw data to manipulate it into new outputs, but never okay to change it in place.

Cookiecutter Data Science: Opinions

3 Data should (mostly) not be kept in source control.

Another consequence of treating data as immutable is that data doesn't need source control in the same way that code does.

Therefore, by default, the `data/` folder is included in the `.gitignore` file.

If you have a small amount of data that rarely changes, you may want to include the data in the repository.

Cookiecutter Data Science: Opinions

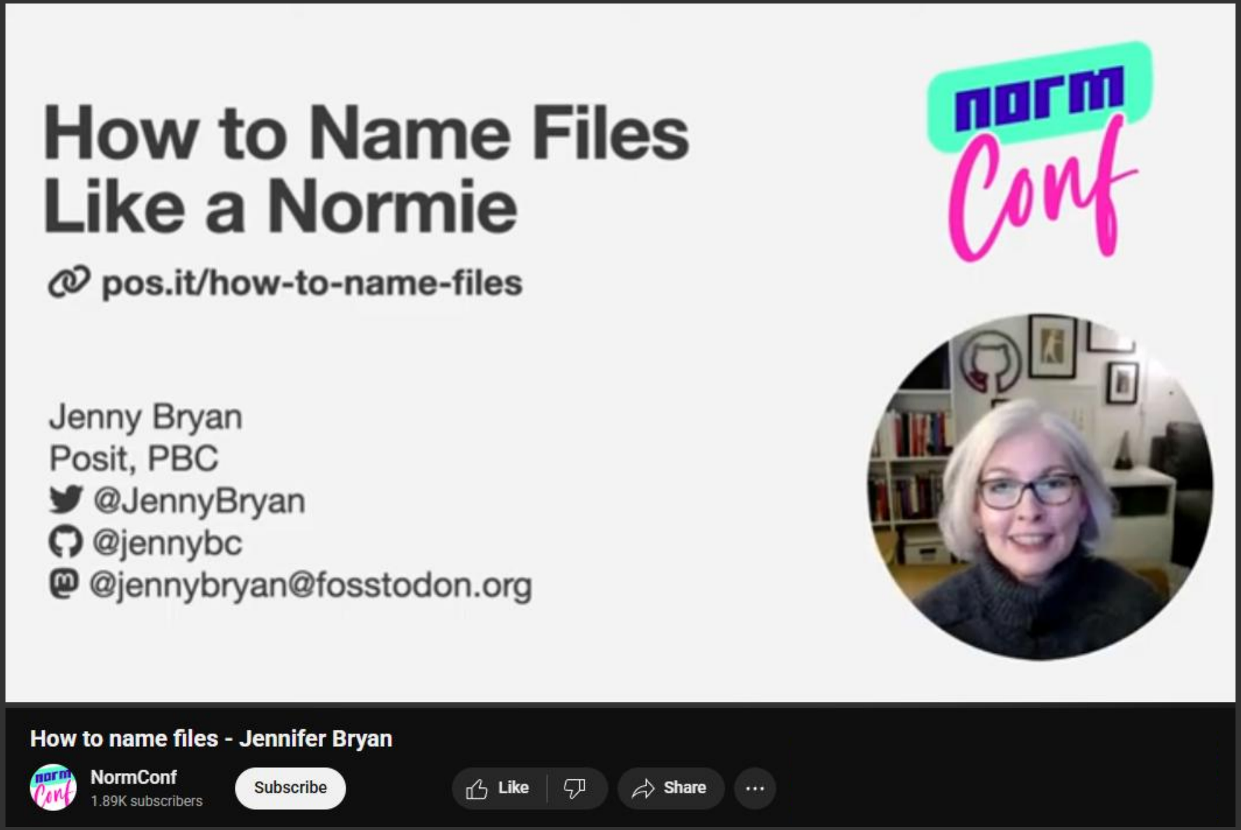
4 **Notebooks are for exploration and communication, source files are for repetition.**

Source code is superior for replicability because it is more portable, can be tested more easily, and is easier to code review.

https://cookiecutter-data-science.drivendata.org/opinions/

Cookiecutter

How to Name Files



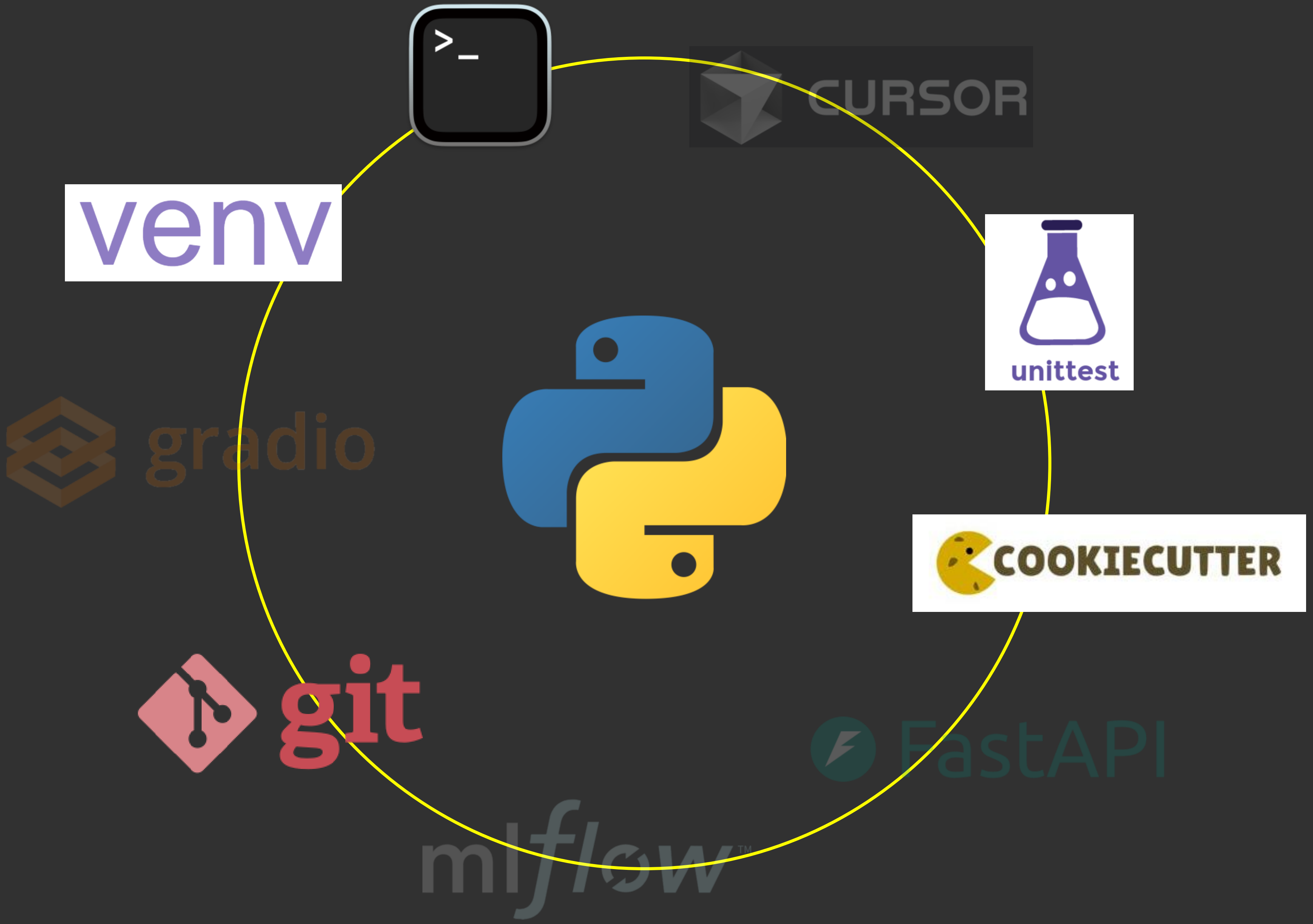
How to Name Files
Like a Normie

pos.it/how-to-name-files

Jenny Bryan
Posit, PBC
@JennyBryan
@jennybc
@jennybryan@fosstodon.org

How to name files - Jennifer Bryan

1. ProjectBrief_FINAL (2).pptx
2. Sales Report 12-04-09.csv
3. 2012-04-09_sales-report.csv
4. 2025-09-24_project-brief_v03.pptx
5. 20250924_lab_notes.md


>_
CURSOR
venv
unittest
gradio
COOKIECUTTER
git
FastAPI
mlflow