

# Building Data Apps In Python With Streamlit

## Session 1: Setup & Basics

**Ari Lamstein**

**[AriLamstein.com](https://AriLamstein.com)**

# Exercise: What is a Data App?

- Introduce yourself to your neighbor.
- Ask them 2 questions:
  - *What do you already know about Data Apps, Python and Streamlit?*
  - *What do you want to build after completing this course?*

# About me

## Ari Lamstein

- Started out as a PhD student in Computer Science -> left to join industry
- Worked as a software engineer at large tech companies in San Francisco
- Authored several R packages that deal with analyzing public datasets. R Trainer.
- Recent years: Python.
- Created two data apps using Streamlit:
  - Covid Demographics Explorer
  - Immigration Enforcement Explorer



# Exercise: What is a Data App?

- My answer has two parts
  - "Data" apps vs. "Real" apps vs. Dashboards
  - Data apps vs. Jupyter Notebooks

Dashboard

Data App

"Real" App



More interaction / customization



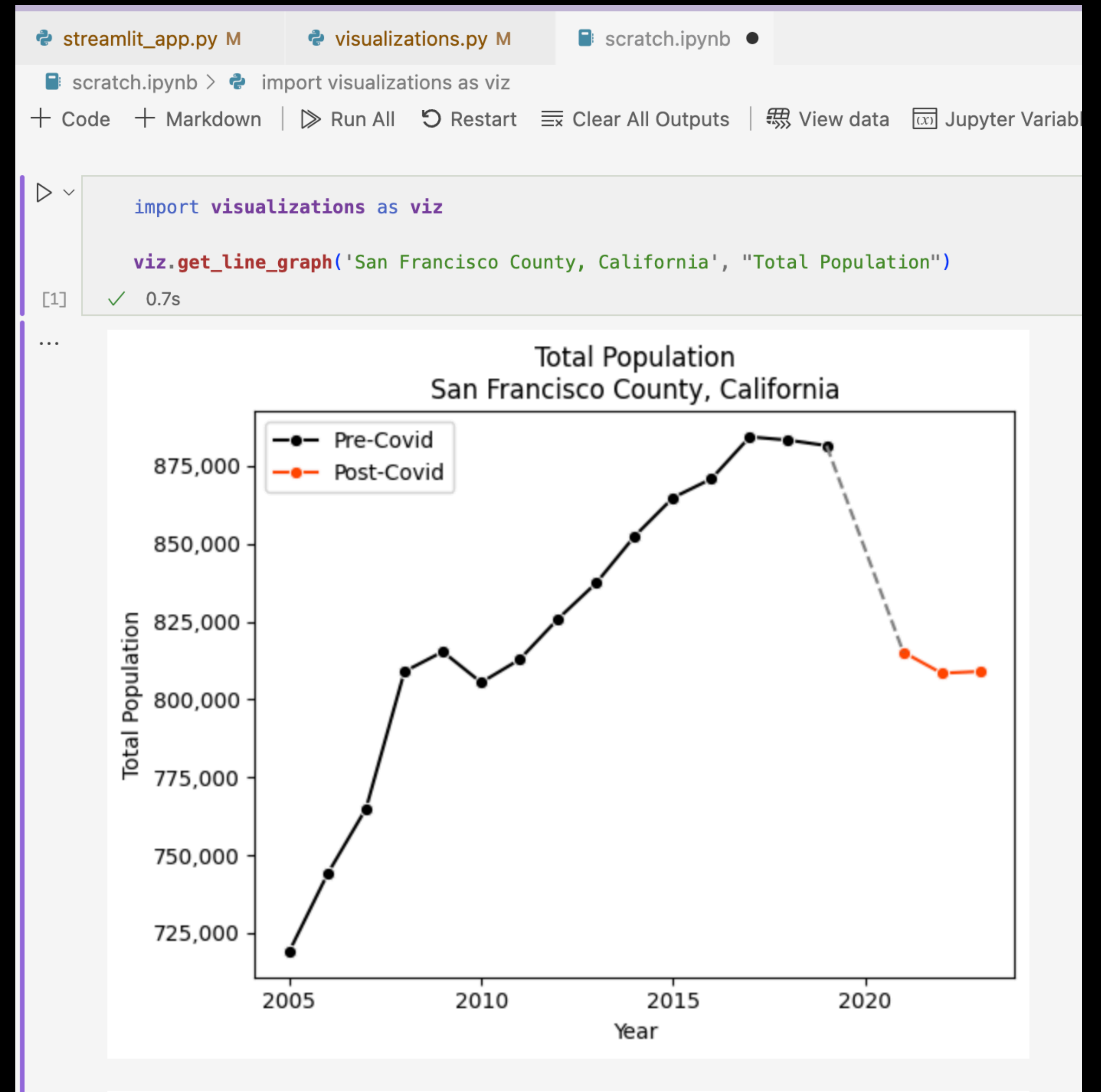
More engineering

# Data Apps vs. Jupyter Notebooks

- Jupyter Notebooks let me **create a single story about a dataset**. I can share the notebook with others for free via github.
- Streamlit lets me **create an app to explore a dataset**. I can share the app with others for free via Streamlit Cloud.
- I often start with a notebook, and end with a Streamlit app.

# Example Jupyter Notebook

- This notebook tells a story about **one city**.



# Example Streamlit App

- This Streamlit App lets users see how the population of **their** county changed
- <https://census-explorer.streamlit.app/>

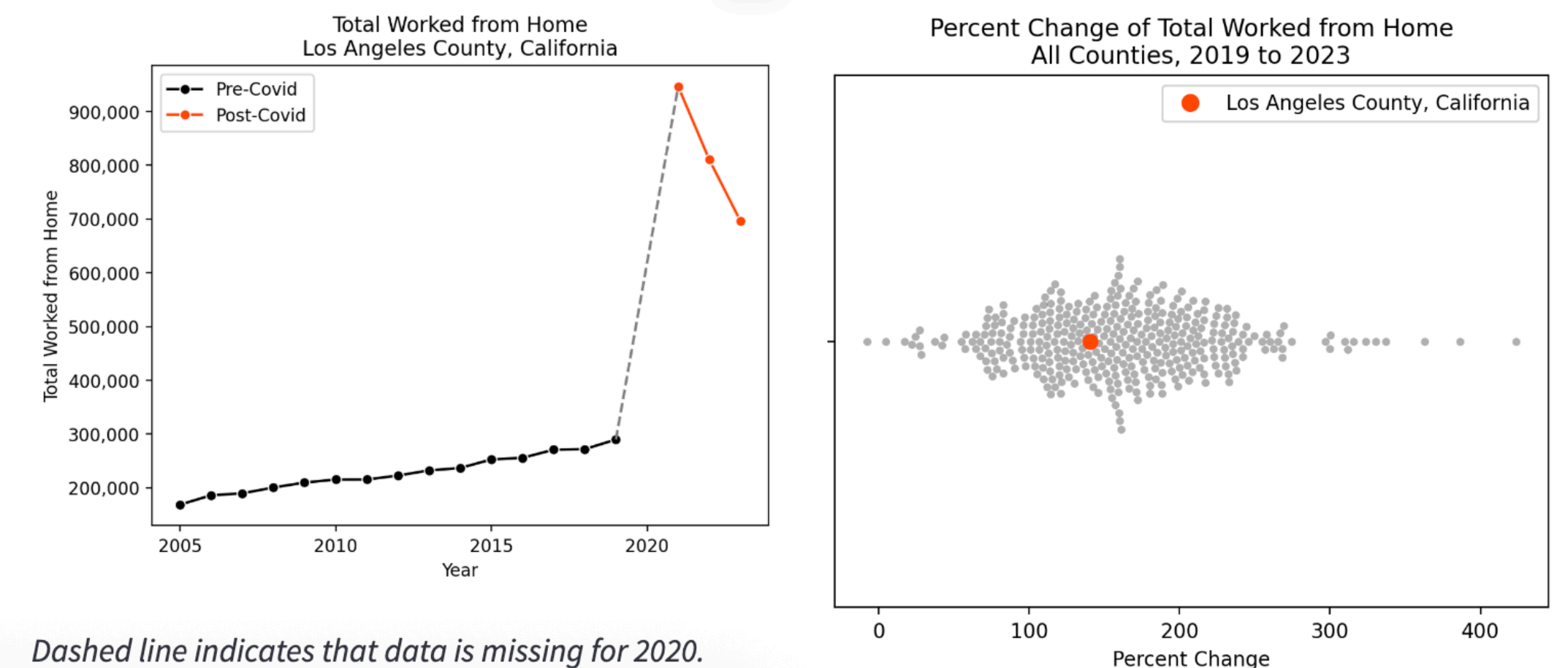
## How has America Changed Since Covid?

This app lets you explore how counties in the US have changed since Covid-19. It starts by showing how the population of San Francisco has changed over time.

You can select another county, and even another demographic - try it now!

State: California County: Los Angeles County Demographic: Total Worked from Home

[Graphs](#) [Table](#) [Map](#) [About](#)





# Course Agenda

## (Big Picture)

1. Session 1: Setup & Basics
2. Session 2: "The Loop": Inputs & Graphics
3. Session 3: Organization: UI & Code
4. Session 4: Deployment

# Session 1: Setup

## 1. One-Time Setup

### 1. Clone course repo

# Clone course repo

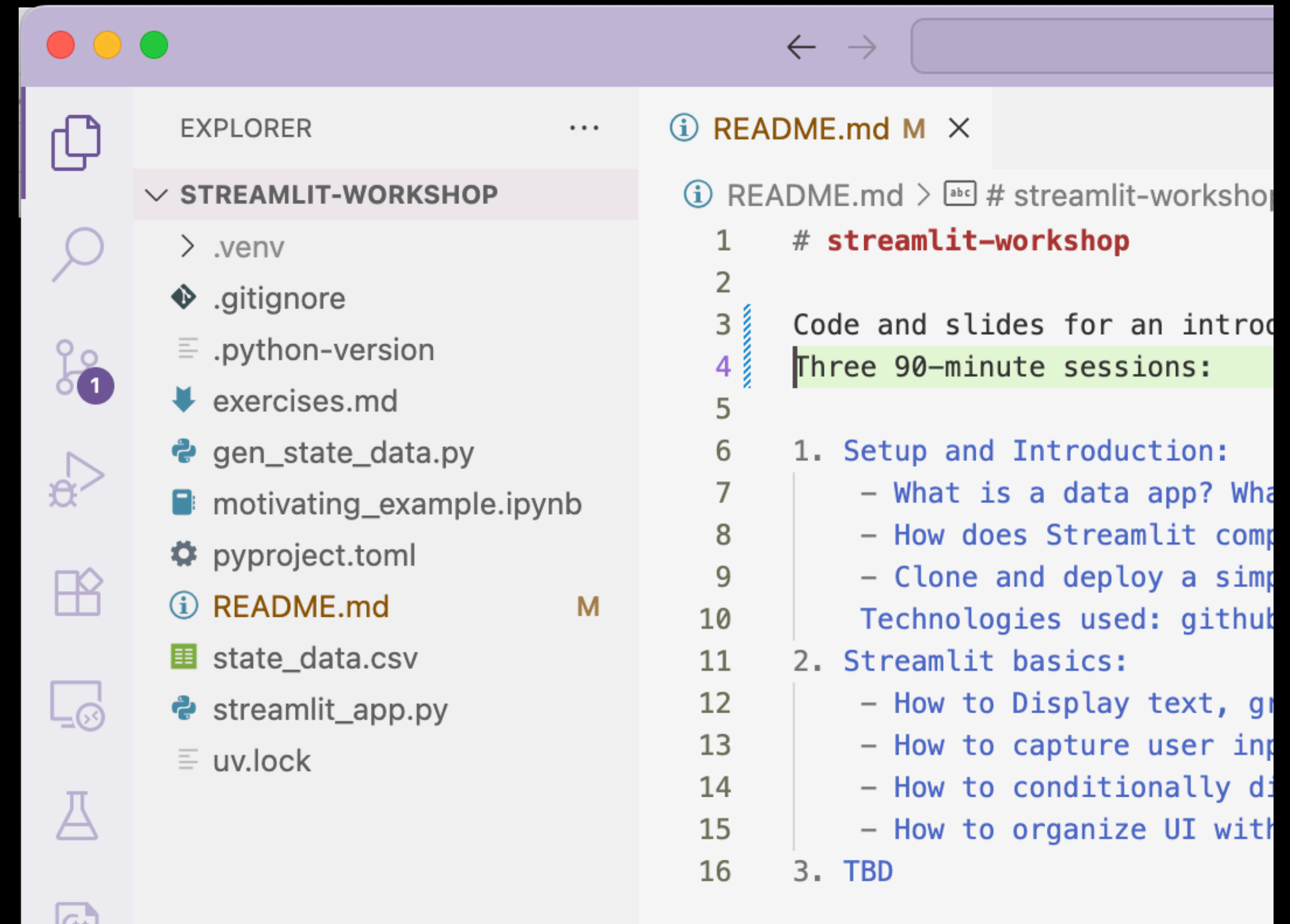
- The course repo is at URL is at <https://github.com/arilamstein/streamlit-workshop>
- To clone it, go to the terminal and type

`git clone https://github.com/arilamstein/streamlit-workshop.git`

```
➤ test git clone https://github.com/arilamstein/streamlit-workshop.git
Cloning into 'streamlit-workshop'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 0), reused 12 (delta 0), pack-reused 0 (from
Receiving objects: 100% (12/12), 175.62 KiB | 1.66 MiB/s, done.
➤ test ls
streamlit-workshop
```

# Open Course Repo

- Open the course repo using your favorite IDE



# Session 1: Setup

## 1. One-Time Setup

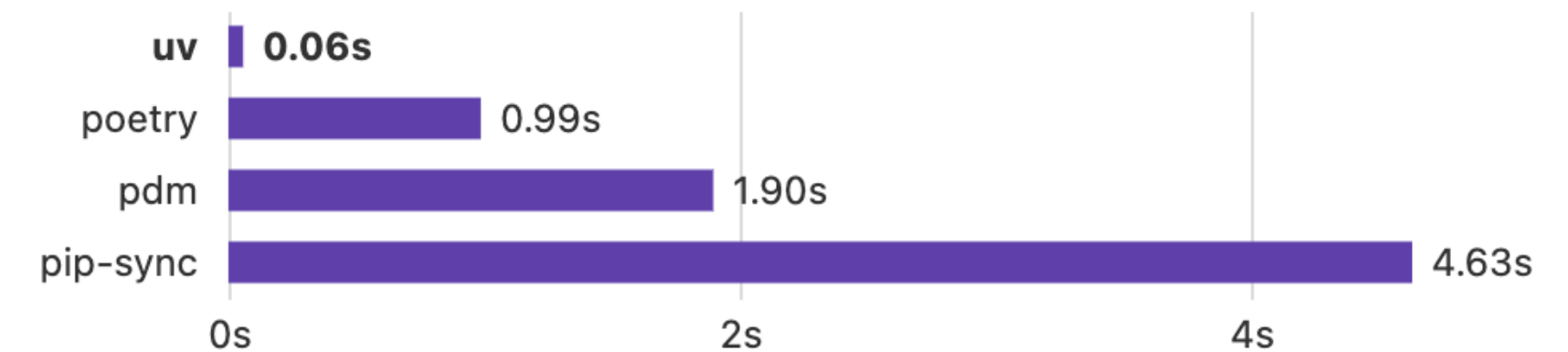
1. Clone course repo
2. Create the virtual environment



- "An extremely fast Python package and project manager, written in Rust"
- Uses 3 files:
  - pyproject.toml
  - uv.lock
  - .python-version

## uv

An extremely fast Python package and project manager, written in Rust.



*Installing [Trio's](#) dependencies with a warm cache.*

## Highlights

- 🚀 A single tool to replace `pip`, `pip-tools`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualenv` and more.
- ⚡ 10-100x faster than `pip`.
- 📦 Provides [comprehensive project management](#), with a [universal lockfile](#).
- ✨ [Runs scripts](#), with support for [inline dependency metadata](#).
- 🐍 [Installs and manages](#) Python versions.
- 🛠️ [Runs and installs](#) tools published as Python packages.
- 🔑 Includes a [pip-compatible interface](#) for a performance boost with a familiar CLI.

# pyproject.toml

- Specifies a (minium) version of Python for the project
- Specifies packages that need to be installed ("dependencies")
- Specifies **minimum** versions required

```
⚙ pyproject.toml
1  [project]
2  name = "streamlit-workshop"
3  version = "0.1.0"
4  description = "Add your descr
5  readme = "README.md"
6  requires-python = ">=3.13"
7  dependencies = [
8      "censusdis>=1.4.2",
9      "ipykernel>=6.30.1",
10     "kaleido>=1.1.0",
11     "pandas>=2.3.2",
12     "plotly>=6.3.0",
13     "streamlit>=1.49.1",
14 ]
```



# uv.lock

- Says the **exact version** of each package to install
- Ensures reproducible environments
- Eliminates a common source of errors

```
[[package]]
name = "streamlit"
version = "1.49.1"
source = { registry = "https://pypi.org/simple" }
dependencies = [
    { name = "altair" },
    { name = "blinker" },
    { name = "cachetools" },
    { name = "click" },
    { name = "gitpython" },
    { name = "numpy" },
    { name = "packaging" },
    { name = "pandas" },
    { name = "pillow" },
    { name = "protobuf" },
    { name = "pyarrow" },
    { name = "pydeck" },
    { name = "requests" },
    { name = "tenacity" },
    { name = "toml" },
    { name = "tornado" },
    { name = "typing-extensions" },
    { name = "watchdog", marker = "sys_platform != 'darwin'" },
]
sdist = { url = "https://files.pythonhosted.org/packages/24/17/c8024"
wheels = [
    { url = "https://files.pythonhosted.org/packages/85/9e/146cdef51"
]
```



# .python-version

- Says which version of Python to install
  - Major and minor version are specified
  - uv will always install the latest patch version
- This eliminates a common source of errors

A screenshot of a code editor showing the content of a file named .python-version. The file name is at the top with a hamburger menu icon to its left. Below it, the text '1 3.13' is displayed, with the '1' in purple and '3.13' in black. The line containing '3.13' is highlighted with a light green background. A '2' is visible below the '1', likely indicating the current line number in the editor.

```
≡ .python-version
```

1

3.13

2

# Install uv

- Google "install uv".
- The first hit should be this page.
- Copy the "curl" command into your terminal.

Getting started

## Installing uv

### Installation methods

Install uv with our standalone installers or your package manager

### Standalone installer

uv provides a standalone installer to download and install uv:

**macOS and Linux**

Windows

Use `curl` to download the script and execute it with `sh`:

```
$ curl -LsSf https://astral.sh/uv/install.sh | sh
```

# Installing uv

- What I see after installation

```
➔ streamlit-workshop git:(main) x curl -LsSf https://astral.sh/uv/install.sh | sh
downloading uv 0.8.17 aarch64-apple-darwin
no checksums to verify
installing to /Users/arilamstein/.local/bin
  uv
  uvx
everything's installed!
WARN: The following commands are shadowed by other commands in your PATH: uv uvx
```

Restart your terminal

# uv sync

- In the terminal, go back to the project directory
- Type `uv sync` in the terminal
- Creates a virtual environment in the folder `.venv` that is "in sync" with the `pyproject.toml` and `uv.lock` files
  - This directory will have the correct version of Python, and the correct version of each package the project needs

```
➔ streamlit-workshop git:(main) x uv sync
Using CPython 3.13.7
Creating virtual environment at: .venv
Resolved 145 packages in 4ms
Installed 139 packages in 385ms
+ adjusttext==0.8
+ affine==2.4.0
+ alabaster==1.0.0
+ altair==5.5.0
+ appnope==0.1.4
+ asttokens==3.0.0
+ attrs==25.3.0
+ babel==2.17.0
+ beautifulsoup4==4.13.5
+ black==25.1.0
+ bleach==6.2.0
+ blinker==1.9.0
+ cachetools==6.2.0
+ censusdis==1.4.2
+ certifi==2025.1.31
```

# Session 1: Setup

## 1. One-Time Setup

1. Clone course repo
2. Create the virtual environment

## 2. Per-Session Setup

1. Activate the virtual environment
2. Run the example Streamlit app

# Activate the Virtual Environment

- Activate the virtual environment:
  - **Mac/Linux:** `source .venv/bin/activate`
  - **Windows (Command Prompt):** `.venv\Scripts\activate`

# Test the Virtual Environment

- Check that the virtual environment is running:
  - Type **python check\_env.py**

```
▶(streamlit-workshop) → streamlit-workshop git:(main) x python check_env.py
Python executable: /Users/arilamstein/uv-projects/streamlit-workshop/.venv/bin/python
Virtual env active: True
▶(streamlit-workshop) → streamlit-workshop git:(main) x █
```



# Session 1: Setup

## 1. One-Time Setup

1. Clone course repo
2. Create the virtual environment

## 2. Per-Session Setup

1. Activate the virtual environment
2. Run the example Streamlit app

# Run the Demo App

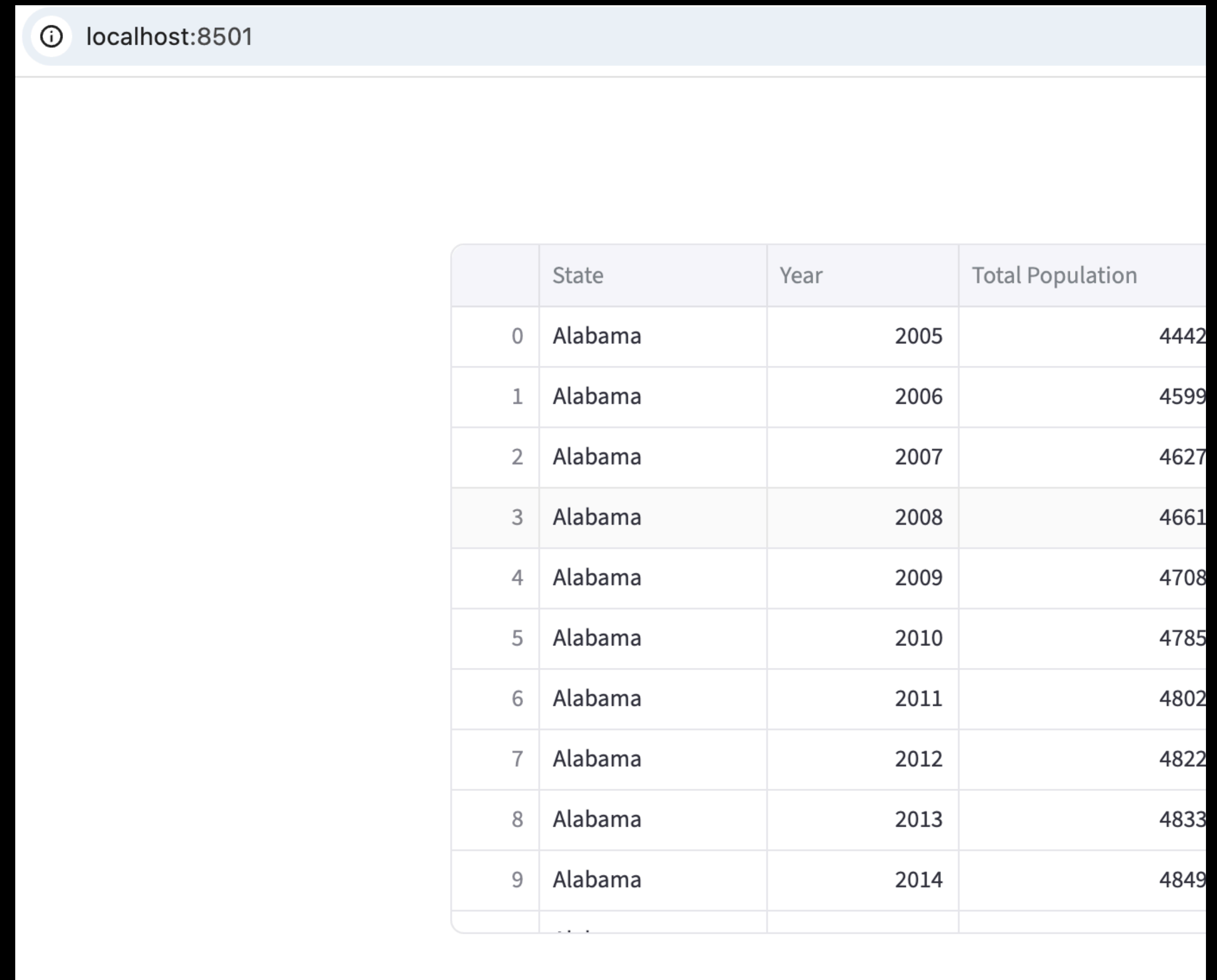
- Each session has its own directory
- Type `cd 1-intro-setup`
  - This moves to the directory for this session
- Type `streamlit run streamlit_app.py`
  - This runs the demo app for this section

```
▶(streamlit-workshop) → streamlit-workshop git:(main) x cd 1-intro-setup  
▶(streamlit-workshop) → 1-intro-setup git:(main) x streamlit run streamlit_app.py
```

You can now view your Streamlit app in your browser.

# Running the App

- Type `streamlit run streamlit_app.py`
- It should open a browser like this
- Click the table headers to sort (e.g. State, Year, Total Population)
- **Quiz:**
  - What state had the lowest population?
  - What state had the highest income?



	State	Year	Total Population
0	Alabama	2005	4442
1	Alabama	2006	4599
2	Alabama	2007	4627
3	Alabama	2008	4661
4	Alabama	2009	4708
5	Alabama	2010	4785
6	Alabama	2011	4802
7	Alabama	2012	4822
8	Alabama	2013	4833
9	Alabama	2014	4849

# Recap: Setup

## 1. One-Time Setup

1. Clone course repo
2. Create the virtual environment

## 2. Per-Session Setup

1. Activate the virtual environment
2. Run the example Streamlit app

# Course Agenda

## (Big Picture)

1. Session 1: Setup & Basics


# Rerun

- Streamlit is designed for fast iteration
- Each time you modify and save the script the word **Rerun** will appear in the top right of the app
- Click it to rerun the app with your changes
- Do not stop and start the app from the command line



# Exercise: Rerun

- In groups of 2
- Open the file `streamlit_app.py` in the directory `1-intro-setup`
- Type `streamlit run streamlit_app.py` to run the app
- Uncomment the `st.title` line
- Save the file
- Click "rerun" and verify that the title appears

```
session1 >  streamlit_app.py > ...  
1  import streamlit as st  
2  import pandas as pd  
3    
4  #st.title("Demo Streamlit App")  
5    
6  df = pd.read_csv("state_data.csv")  
7    
8  st.dataframe(df)
```

# Exercise: Rerun

- Result

## Demo Streamlit App



	State	Year	Total Population	Median Household Income
0	Alabama	2005	4442558	36879
1	Alabama	2006	4599030	38783
2	Alabama	2007	4627851	40554
3	Alabama	2008	4661900	42666
4	Alabama	2009	4708708	40489
5	Alabama	2010	4785298	40474
6	Alabama	2011	4802740	41415
7	Alabama	2012	4822023	41574
8	Alabama	2013	4833722	42849
9	Alabama	2014	4849377	42830



# Demo App Walkthrough

# The data

- `state_data.csv`
- Basic demographic data of US States
- Each state, 2005-2023
- Comes from the American Community Survey (ACS)

```
session1 >  state_data.csv >  data
```

	State	Year	Total Population	Median Household Income
1	Alabama	2005	4442558	36879
2	Alabama	2006	4599030	38783
3	Alabama	2007	4627851	40554
4	Alabama	2008	4661900	42666
5	Alabama	2009	4708708	40489
6	Alabama	2010	4785298	40474
7	Alabama	2011	4802740	41415
8	Alabama	2012	4822023	41574
9	Alabama	2013	4833722	42849
10	Alabama	2014	4849377	42830
11	Alabama	2015	4858979	44765
12	Alabama	2016	4863300	46257

# Pandas

- Pandas is the most common Python library for working with tabular data
- Open **scratch.ipynb**. Two ways:

1. VS Code with Jupyter plugin
2. Jupyter lab via command line:

```
uv run --with jupyter jupyter lab  
scratch.ipynb
```

- It's always imported as **pd**
- **pd.read\_csv** returns a **dataframe**

```
import pandas as pd

df = pd.read_csv('state_data.csv')
df
```


[1] ✓ 0.4s Open 'df' in Data Wrangler

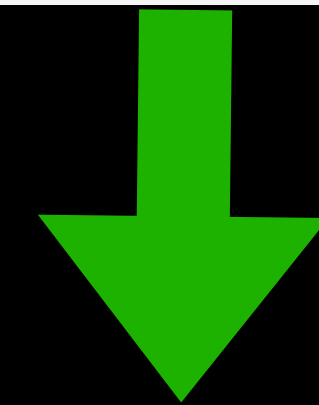
	State	Year	Total Population	Median Household Income
0	Alabama	2005	4442558	36879
1	Alabama	2006	4599030	38783
2	Alabama	2007	4627851	40554
3	Alabama	2008	4661900	42666
4	Alabama	2009	4708708	40489
...	...	...	...	...
931	Wyoming	2018	577737	61584
932	Wyoming	2019	578759	65003
933	Wyoming	2021	578803	65204
934	Wyoming	2022	581381	70042
935	Wyoming	2023	584057	72415

936 rows x 4 columns

# Streamlit

- Lets the script run as a webapp
- Cannot run in a Jupyter notebook - must be a regular .py file.
- Always imported as **st**
- Two output functions:
  - **st.title**
  - **st.dataframe**
- Processed top to bottom.

```
session1 >  streamlit_app.py > ...  
1  import streamlit as st  
2  import pandas as pd  
3  
4  st.title("Demo Streamlit App")  
5  
6  df = pd.read_csv("state_data.csv")  
7  
8  st.dataframe(df)  
9
```



## Demo Streamlit App

	State	Year	Total Population	Median Household Income
0	Alabama	2005	4442558	36879
1	Alabama	2006	4599030	38783
2	Alabama	2007	4627851	40554
3	Alabama	2008	4661000	42666

# Exercise: LLM

- In groups of 2
- Tell an LLM
  - What you've done with Streamlit so far
  - What you should try next
- Experiment with one of the functions it recommends.
  - Use "rerun" to see the changes
- I use [copilot.microsoft.com](https://copilot.microsoft.com), but I think any LLM is fine.
- Stuck? Ask it for help!

I just started using Streamlit. So far I've used two functions: `st.title` and `st.dataframe`. Please recommend a few other simple, common functions I can use on a demo app.

Nice start, Ari! Since you've already used `st.title()` and `st.dataframe()`, here are a few other beginner-friendly Streamlit functions that will make your demo app more interactive and polished:



## Presentation & Layout


These help structure your app and guide the viewer:

- `st.header("Section Title")` — Adds a bold section header
- `st.markdown("**Bold text**, _italic_, [link](https://example.com))"` — Rich formatting
- `st.write("Any object")` — Smart display for text, data, plots, etc.
- `st.image("path_or_url")` — Show an image (e.g. logo or chart)





# st.markdown

- Official docs
- Google "st.markdown"
- It can print emojis?!



Search



**st.markdown**

Version 1.49.0

Display string formatted as Markdown.

Function signature

[source]

```
st.markdown(body, unsafe_allow_html=False, *, help=None, width="stretch")
```

Parameters

**body** (*any*)

The text to display as GitHub-flavored Markdown. Syntax information can be found at: <https://github.github.com/gfm>. If anything other than a string is passed, it will be converted into a string behind the scenes using `str(body)`.

This also supports:

- Emoji shortcodes, such as `:+1:` and `:sunglasses:`. For a list of all supported codes, see <https://share.streamlit.io/streamlit/emoji-shortcodes>.
- Streamlit logo shortcode. Use `:streamlit:` to add a little Streamlit flair to your text.
- A limited set of typographical symbols. `"<- -> <-> -- >= <="` becomes "`<` `>` `<=>` `—` `≥` `≤` `≈`" when parsed as Markdown.

```
st.markdown("**I** :heart: *Streamlit*!")
```

# Demo Streamlit App

I ❤️ *Streamlit*!

	State	Year	Total Population
0	Alabama	2005	4

# Section 1 Review

- What is a data app?
- 1-time setup:
  - Clone course repo
  - Create virtual environment
- Per-session setup:
  - Activate virtual environment
  - Run app
- Rerun app without restarting Streamlit
- Review technical basics: Pandas, Streamlit, Jupyter
- Ask LLM for a suggestion and implemented it