Building Data Apps In Python With Streamlit

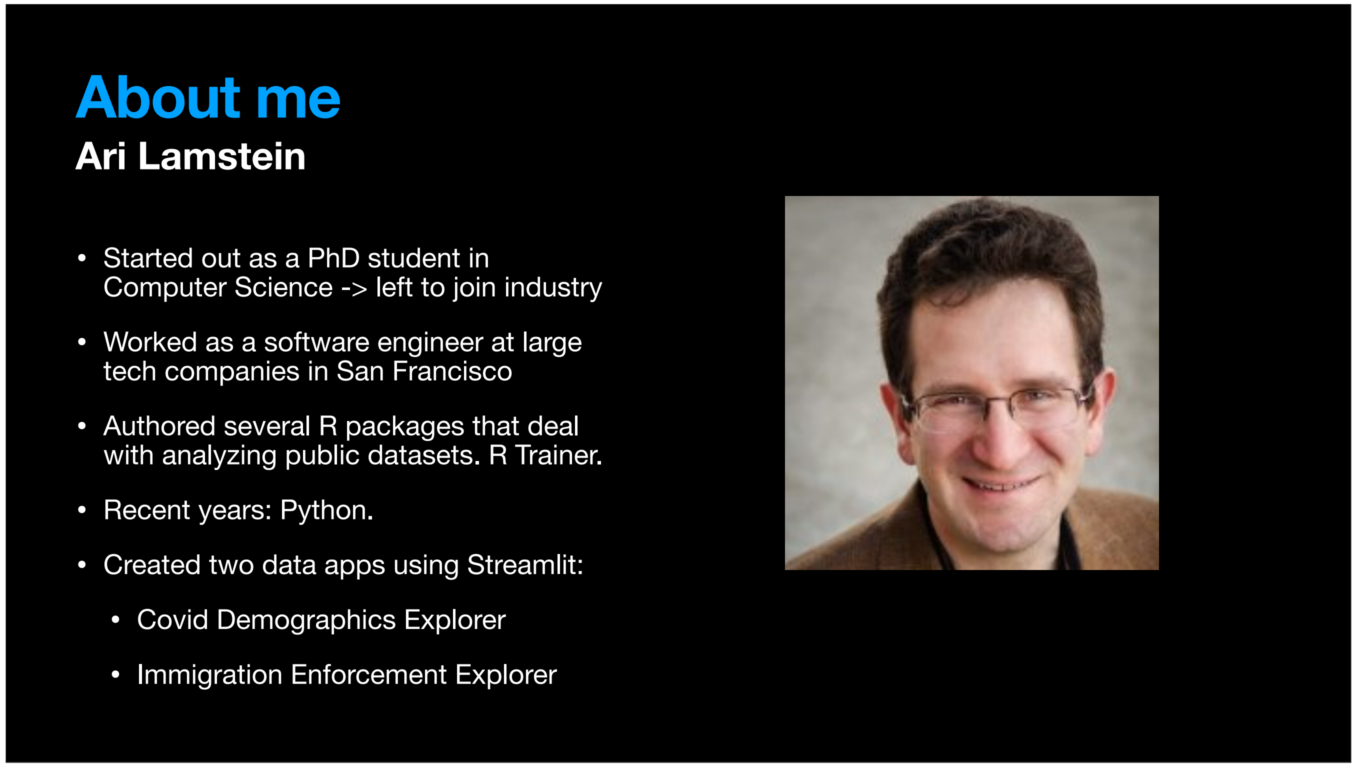
Session 1: Setup & Basics

Ari Lamstein
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?*

4 minute exercise



About me

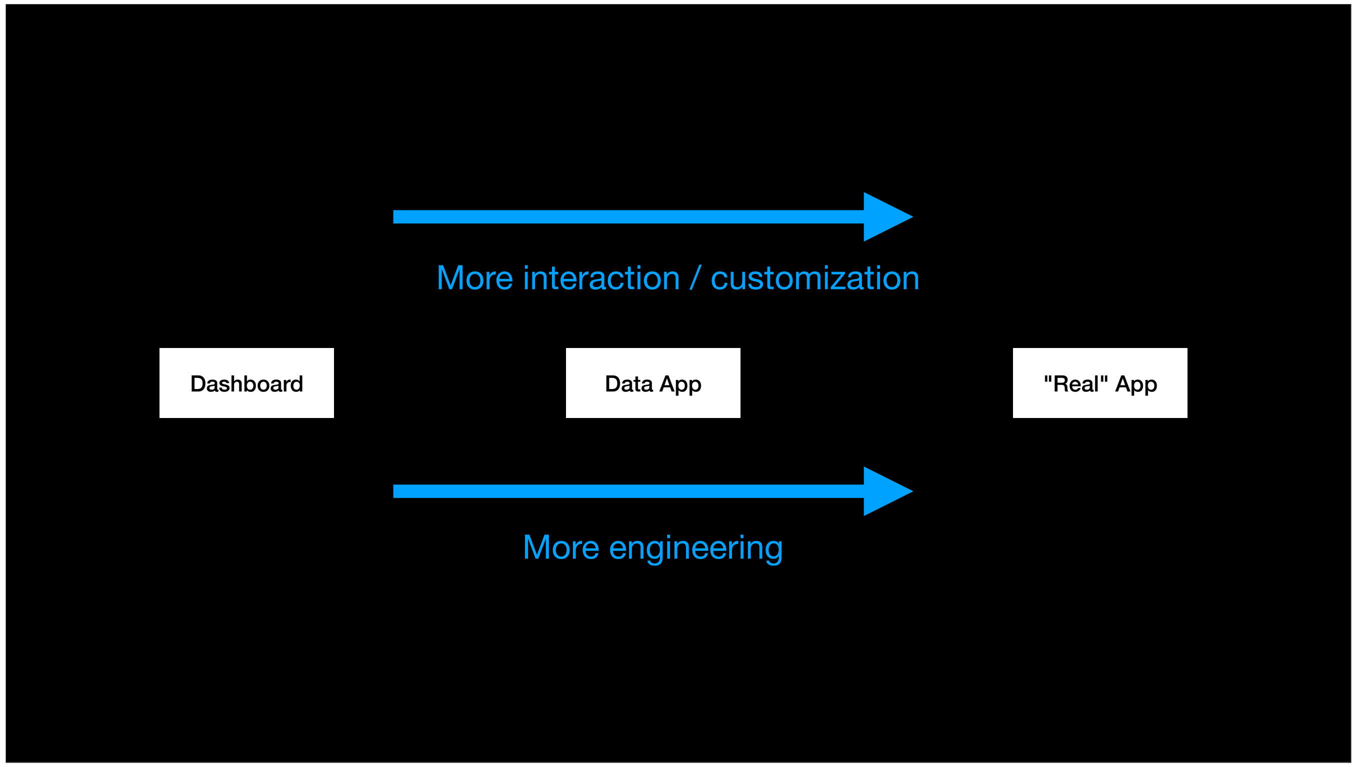
Ari Lamstein

- 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

Before I give my own answer about what a "data app" is, let me introduce myself

Exercise: What is a Data App?

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



More interaction / customization

Dashboard

Data App

"Real" App

More engineering

- A Dashboard has no interactivity whatsoever - think of the dashboard of car. And as a result, there are a lot of ways to create them with using little to no code.
- A "real" app, like Amazon, has a lot of interaction: creating an account, browsing items, purchase items and leave reviews. As a result, there is a lot of engineering involved.
- Data Apps are somewhere in the middle. Like a dashboard, they're designed to inform. But unlike a dashboard, you can interact with them. But unlike a "Real" app, that interaction is typically limited to exploring a dataset.
- Data apps require *some* engineering. In this course, we'll be writing small amounts of Python code. Streamlit will take care of all the front end code (like HTML, CSS and JavaScript). We won't be using a database / writing SQL.

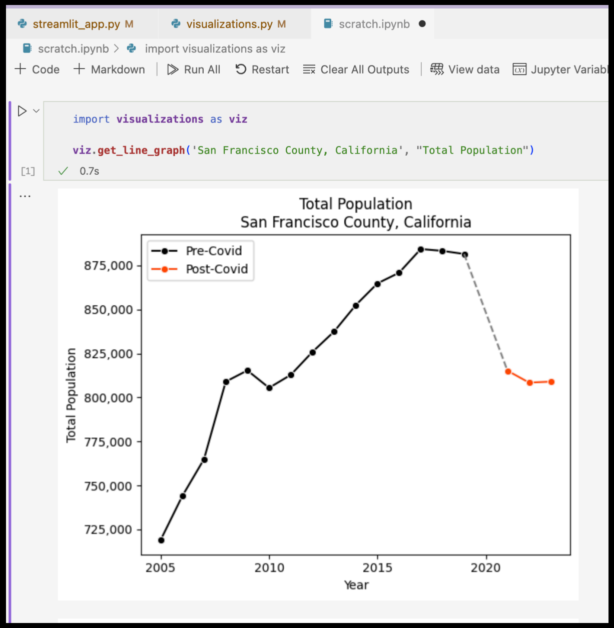
Data Apps vs. Jupyter Notebooks

- 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.

Ask: How many people here have used Jupyter notebooks before?
In my own work, I view Streamlit as a companion to Jupyter notebooks.

Example Jupyter Notebook

- This notebook tells a story about one city.

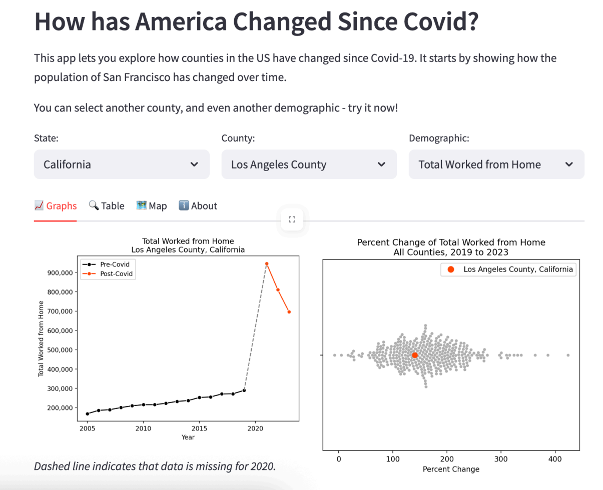


```
import visualizations as viz

viz.get_line_graph('San Francisco County, California', "Total Population")
```

At one point I was interested in understanding how San Francisco's population changed during Covid. My first step was to create a Jupyter Notebook that had graphs like this.

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

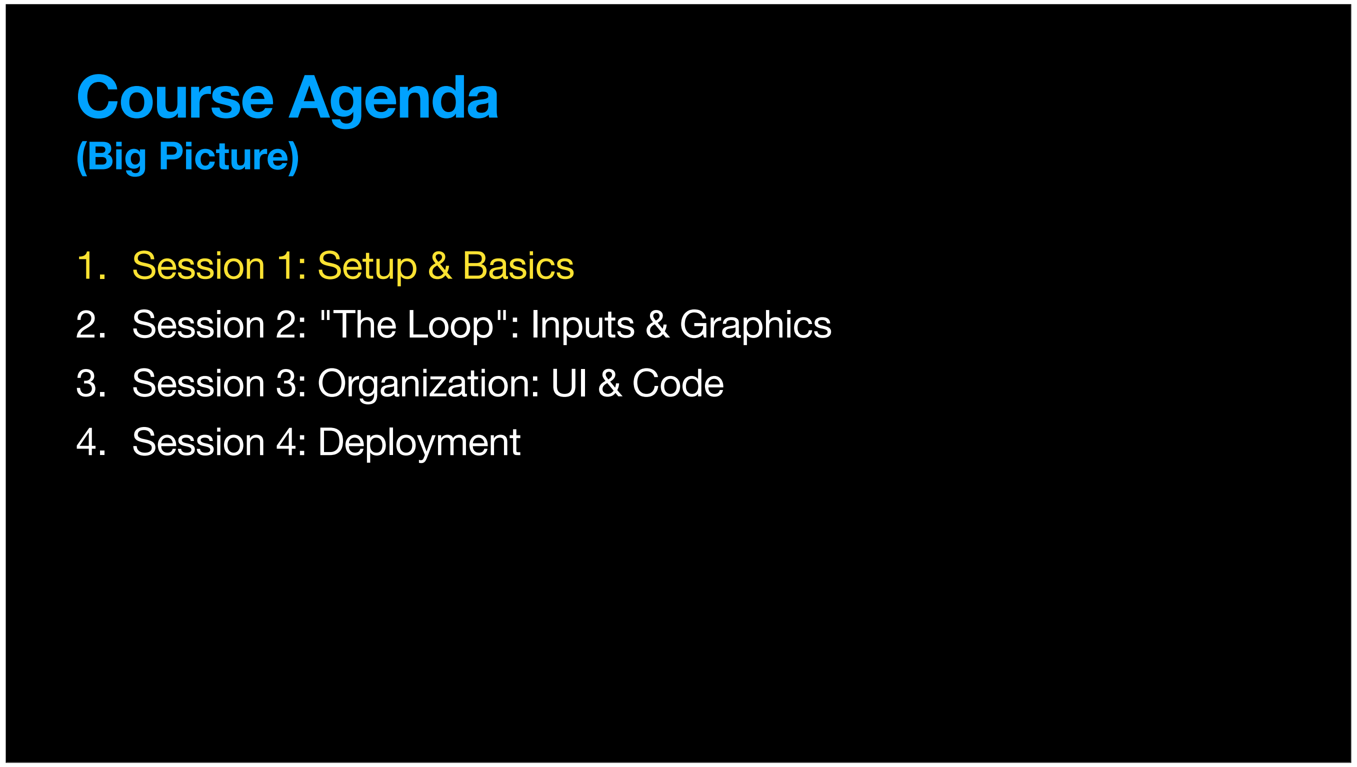
Demographic: Total Worked from Home

Dashed line indicates that data is missing for 2020.

Note the interactivity:

-select boxes -> graphs change.

-layout: tabs are clickable

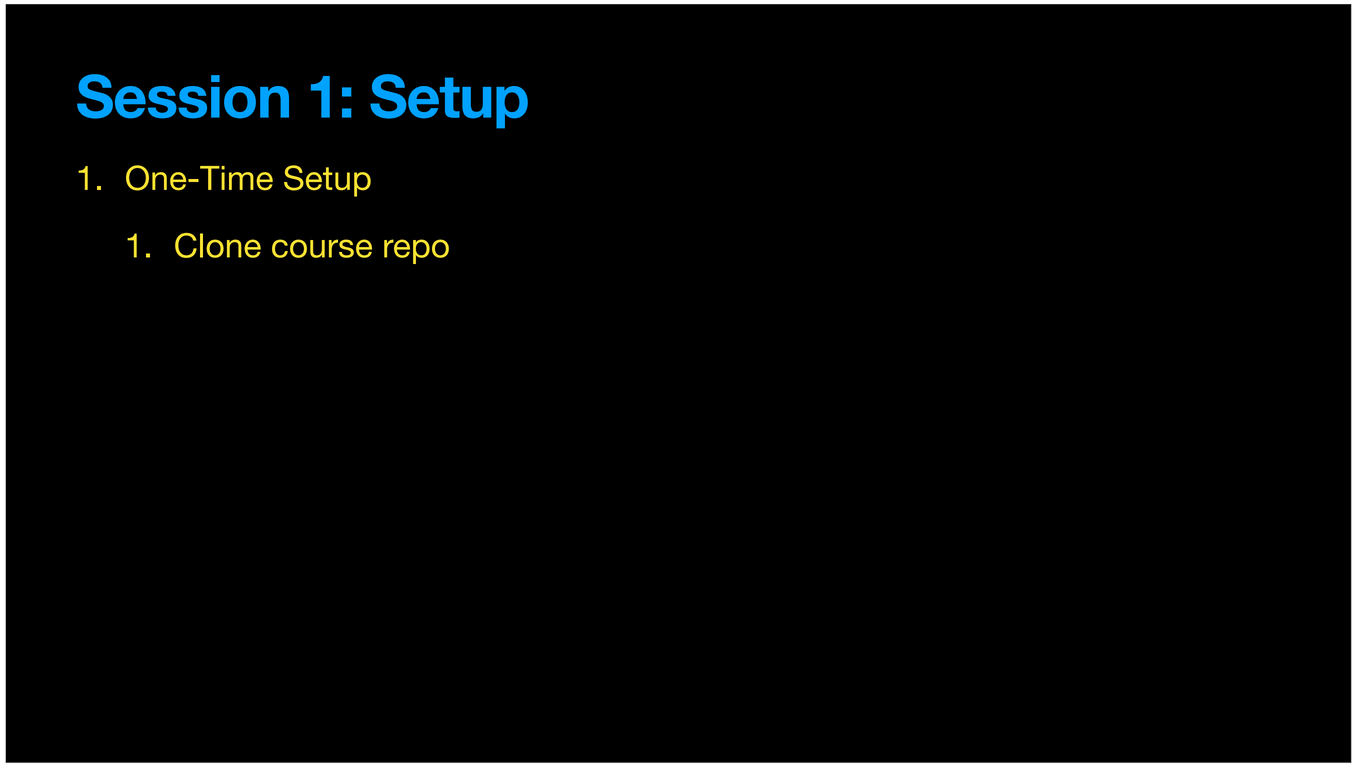


Course Agenda

(Big Picture)

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

I mentioned earlier that because there is some interaction and customization to Streamlit apps, they require some engineering. This engineering work normally has two components, and it's the same components for every project: setup and development. In this first session, we'll focus exclusively on setting up your environment.



Session 1: Setup

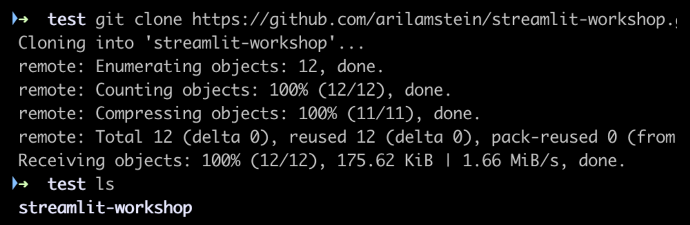
1. One-Time Setup
   1. Clone course repo

The first part of the setup is to get the course code onto your local machine. This is called cloning the course repository.

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.
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
```

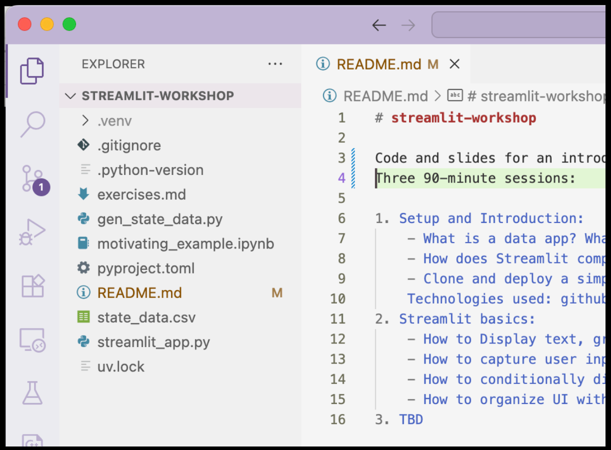
In order to get the course repo on your local machine, you will need to clone it.

Open the terminal to the directory where you want the repository and type "git clone https://github.com/arilamstein/streamlit-workshop.git".

If it worked correctly, you should see a new subdirectory called "streamlit-workshop"

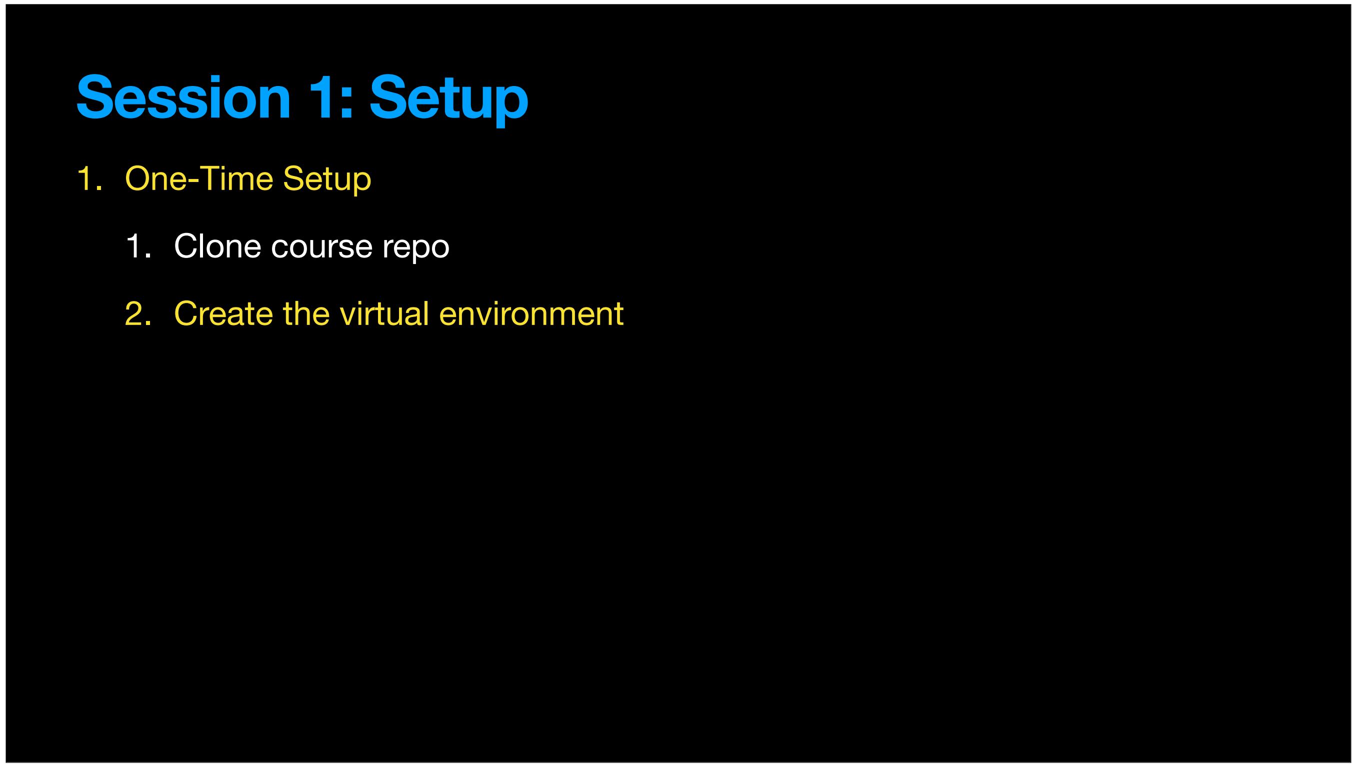
Open Course Repo

- Open the course repo using your favorite IDE



```
# streamlit-workshop

Code and slides for an introd
Three 90-minute sessions:

1. Setup and Introduction:
    - What is a data app? Wha
    - How does Streamlit comp
    - Clone and deploy a simp
    Technologies used: github
2. Streamlit basics:
    - How to Display text, gr
    - How to capture user inp
    - How to conditionally di
    - How to organize UI with
3. TBD
```

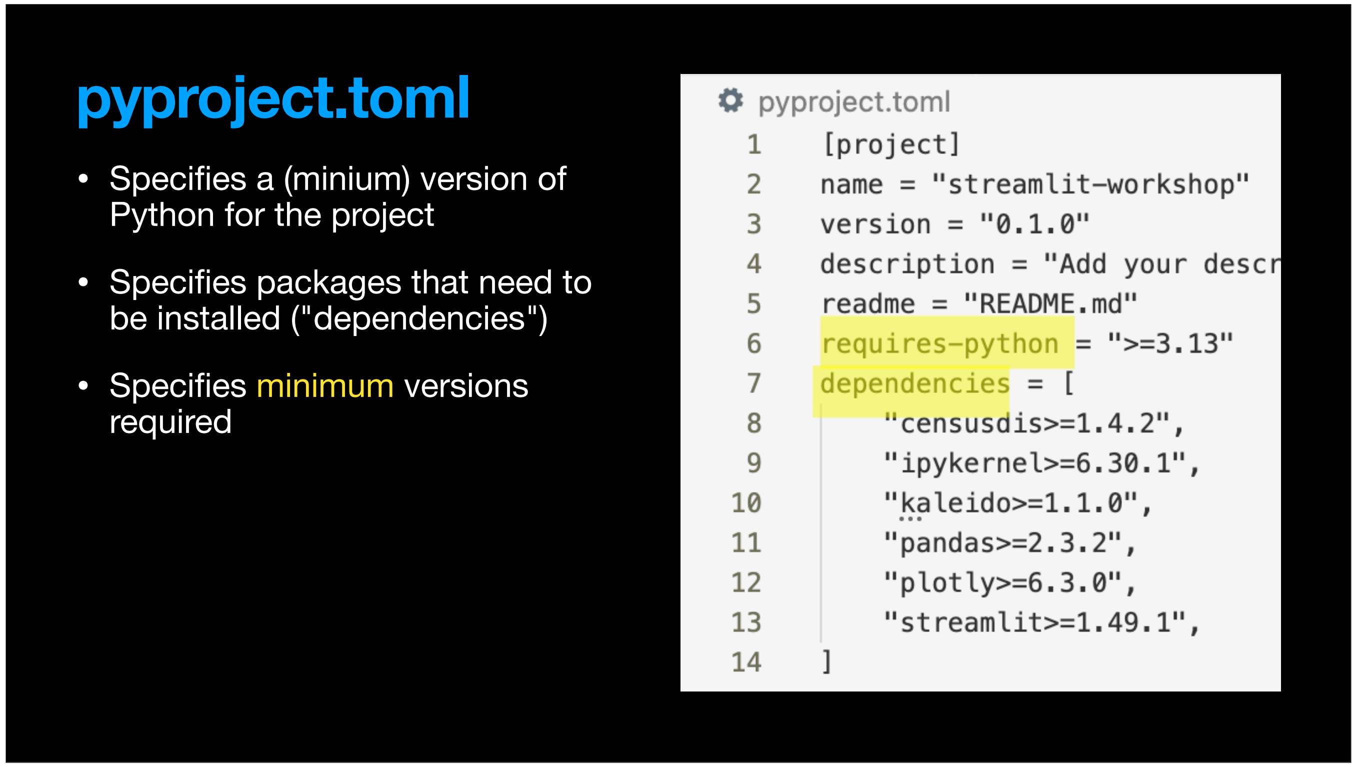
Open up the "streamlit-workshop" directory in your favorite IDE. Here I'm using VS Code.



Session 1: Setup

1. One-Time Setup
    1. Clone course repo
    2. Create the virtual environment

Now that you have the code installed, it's time to create a virtual environment for the course. A virtual environment is basically a version of Python that is tied to a specific project.



pyproject.toml

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

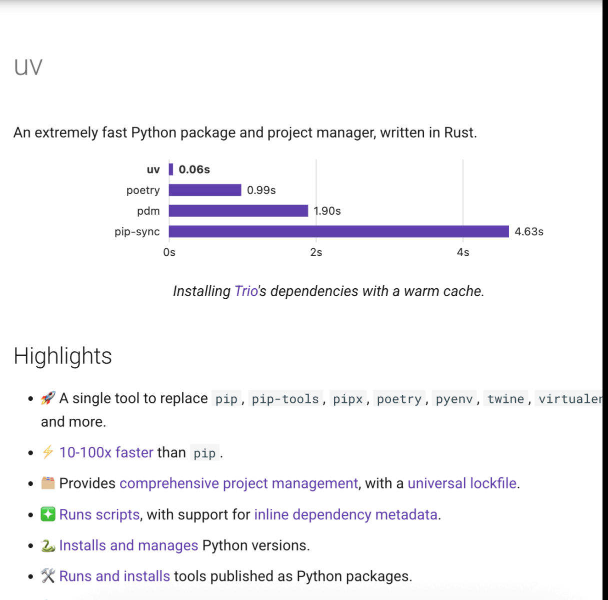
pyproject.toml

```
[project]
name = "streamlit-workshop"
version = "0.1.0"
description = "Add your descr
readme = "README.md"
requires-python = ">=3.13"
dependencies = [
    "censusdis>=1.4.2",
    "ipykernel>=6.30.1",
    "kaleido>=1.1.0",
    "pandas>=2.3.2",
    "plotly>=6.3.0",
    "streamlit>=1.49.1",
]
```

pyproject.toml is the most important file when creating a virtual environment. It says what minimum version of Python to use (here >=3.13). It also says what packages the project needs to have installed ('dependencies'), and what minimum version they need as well. You can open this file now.

uv

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



UV

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

Installing Trio's dependencies with a warm cache.

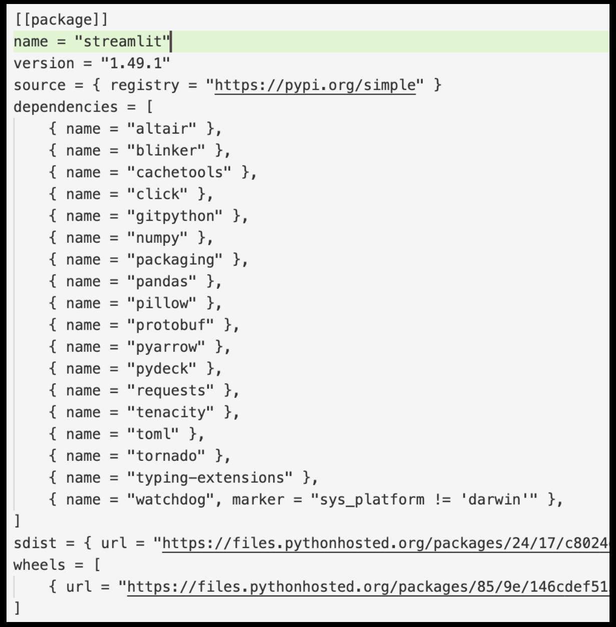
Highlights

- 🚀 A single tool to replace `pip`, `pip-tools`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualen` 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.

We'll be using a program called 'uv' to create the virtual environment. It's a new Python package manager that is taking the Python world by storm. It will install everything faster than older systems, and with less steps.

uv.lock

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

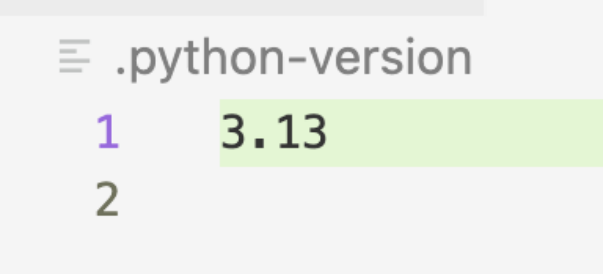


```
[[package]]
name = "streamlit"
version = "1.49.1"
source = { registry = "https://pypi.org/simple" }
dependencies = [
    { name = "altair" },
    { name = "blinker" },
    { name = "cachetools" },
    { name = "click" },
    { name = "gitpython" },
    { name = "numpy" },
    { name = "packaging" },
    { 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
]
```

In addition to the pyproject.toml file, uv also lets you specify the exact version of each package to install. This information is in the file uv.lock. This helps to reduce a common source of errors.

.python-version

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

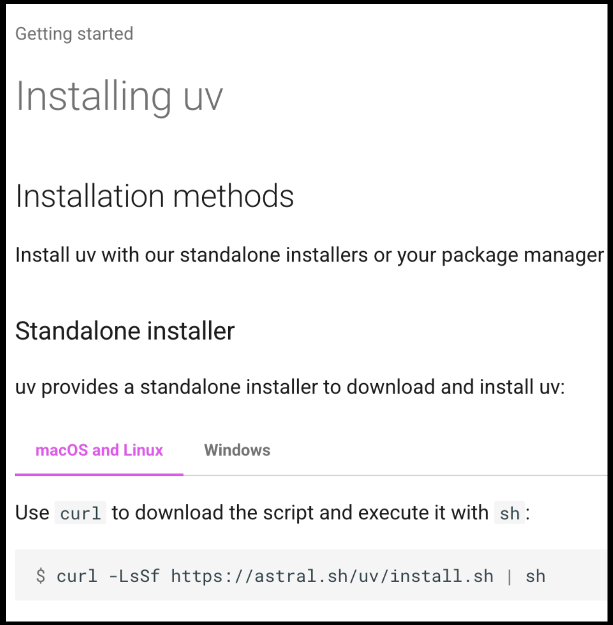


```
3.13

```

uv also lets you "pin" a version of Python to use. Again, this mimimizes the chance of something working on my computer but not on yours.

Install uv

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



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:

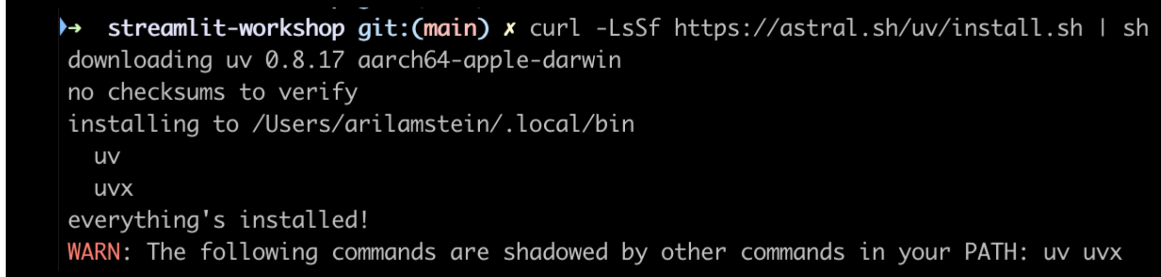
Use `curl` to download the script and execute it with `sh`:

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

Copy the url into the chat

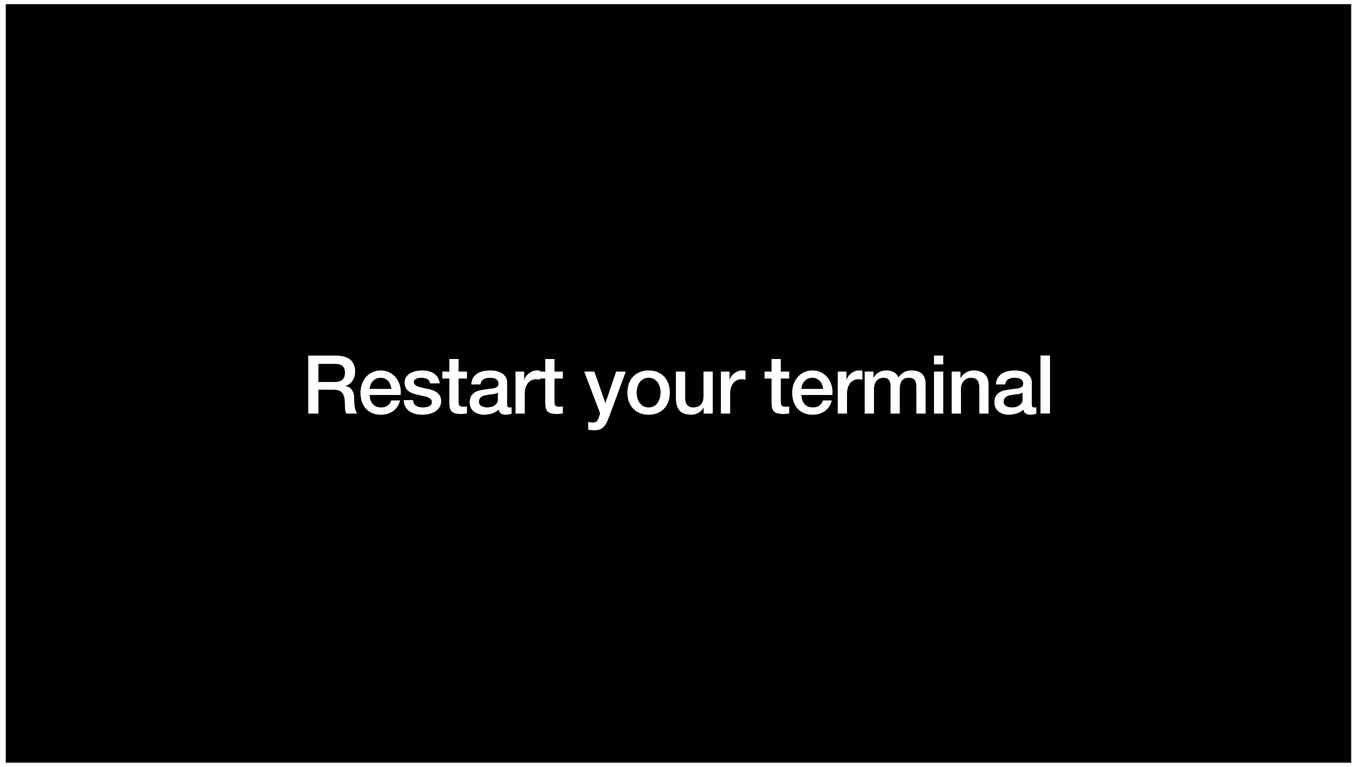
Installing uv

- What I see after installation



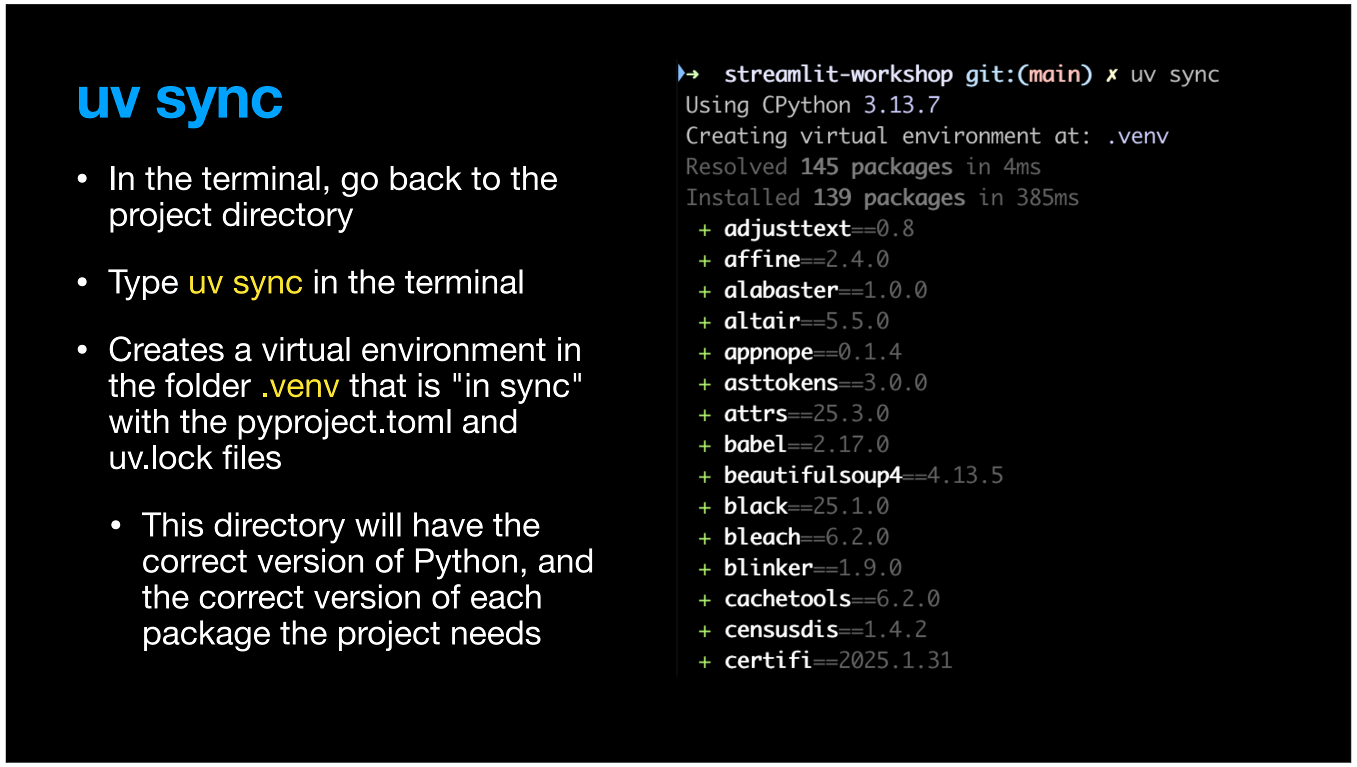
```
➜  streamlit-workshop git:(main) ✗ curl -LsSf https://astral.sh/uv/install.sh | sh
downloading uv 0.8.17 aarch64-apple-darwin
no checksums to verify
installing to /Users/arilamstein/.local/bin
  uv
  uvx
everything's installed!
WARN: The following commands are shadowed by other commands in your PATH: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

Hopefully you don't get the same warnings I do :)



Restart your terminal

This is something that tripped up people in the last workshop I ran. After installing uv, you need to restart your terminal before using it.

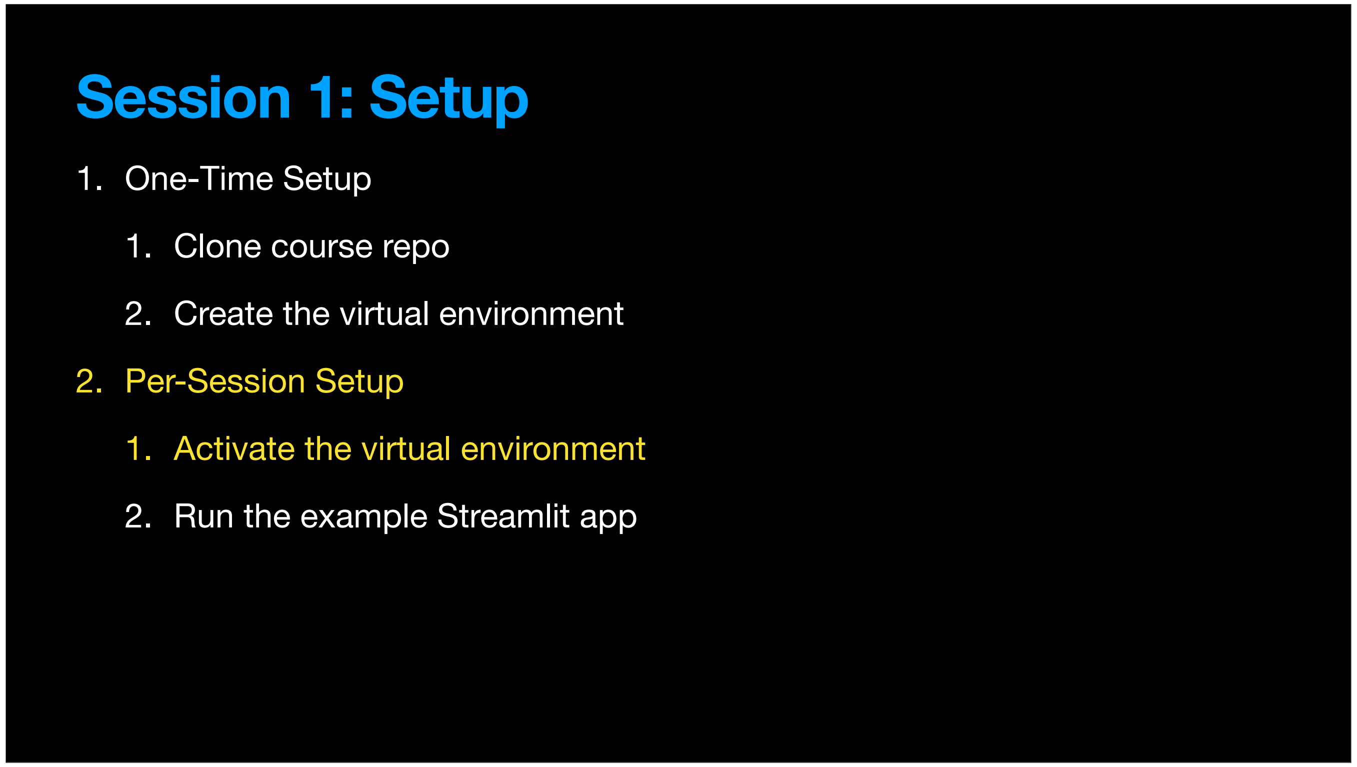


uv sync

- 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) ✗ 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
```

You can see the versions of the packages being installed in the output from uv sync.



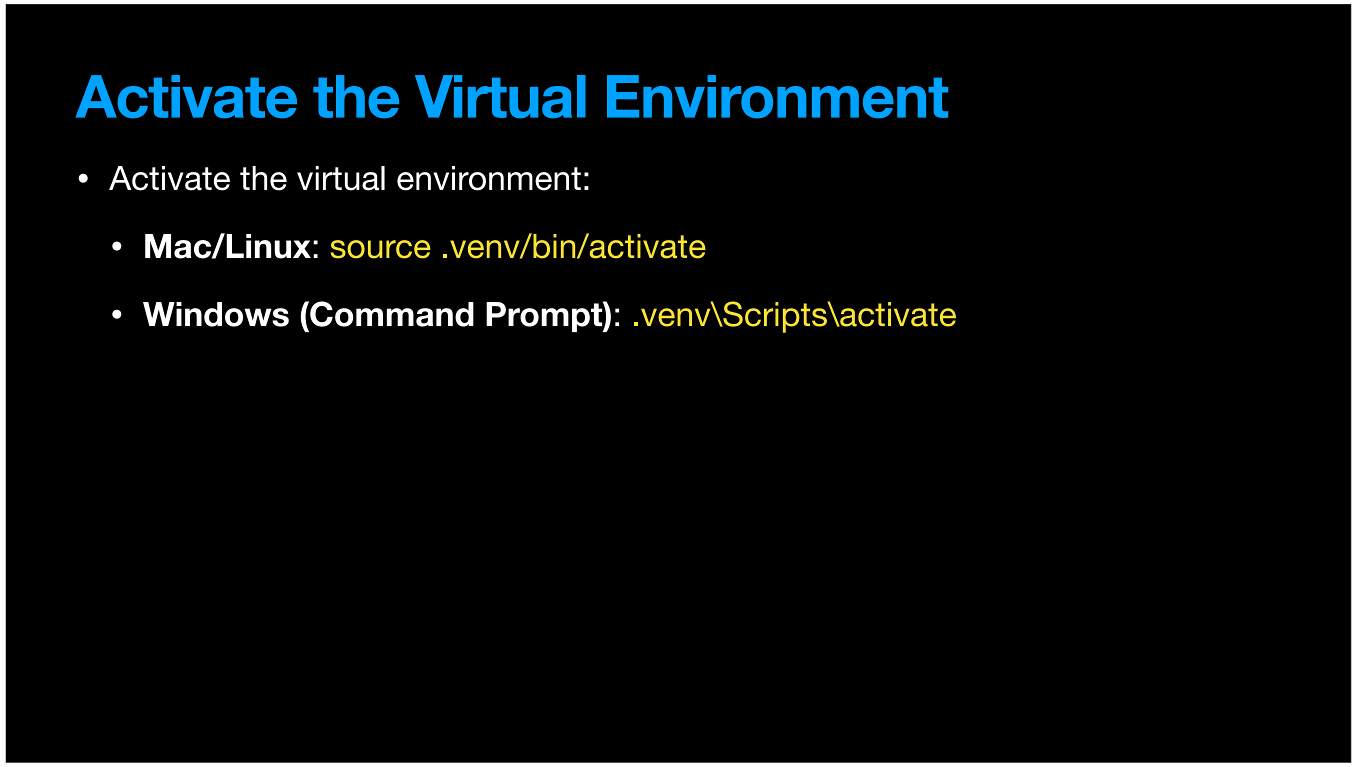
Session 1: Setup

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

Cloning a repo only needs to be done once. And creating the virtual environment also only needs to be done once.

But each time you open the terminal, you'll need to activate the virtual environment. "Activating" the virtual environment just means "When I type 'python' in the terminal, it uses the version of python in the project's .venv directory."

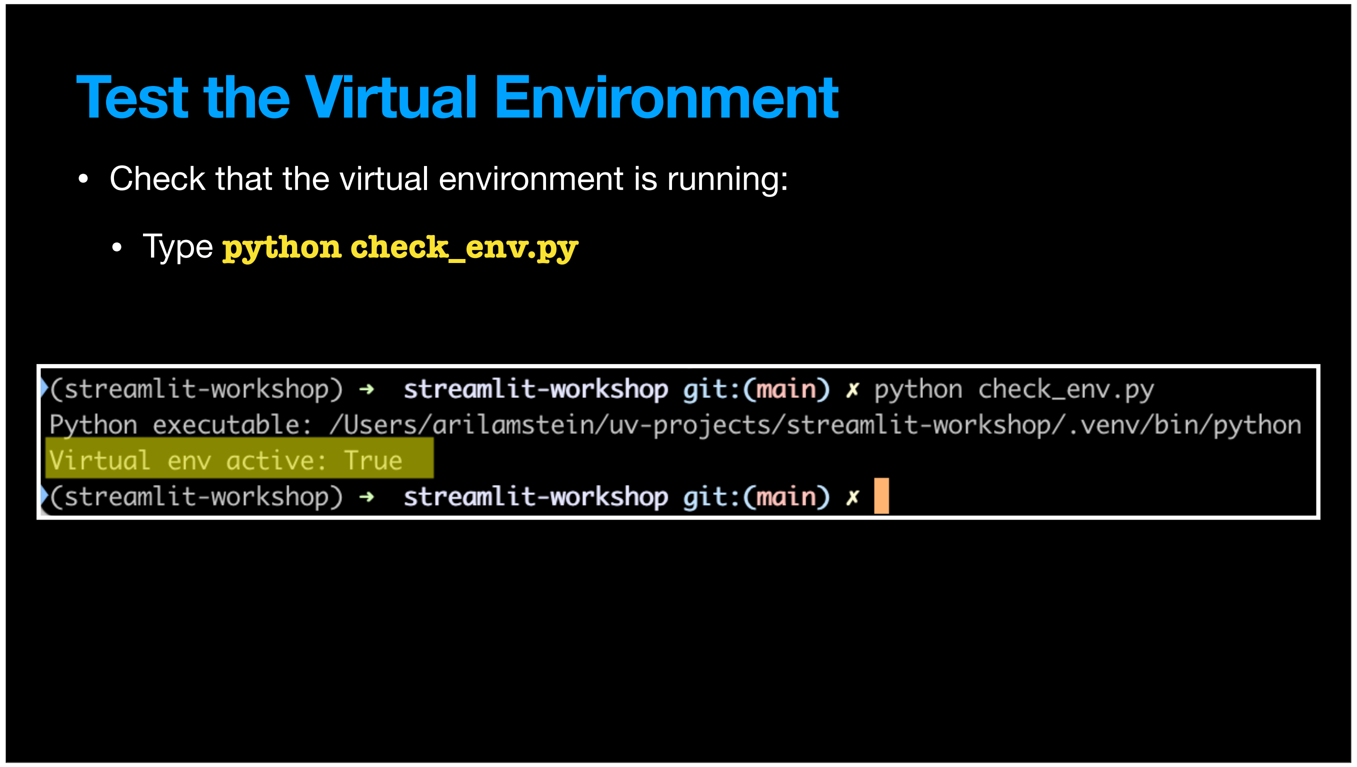
Also, each time you want to run the app, you'll need to type a specific command.



Activate the Virtual Environment

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

The command to activate the virtual environment depends on your operating system. Type the right command now

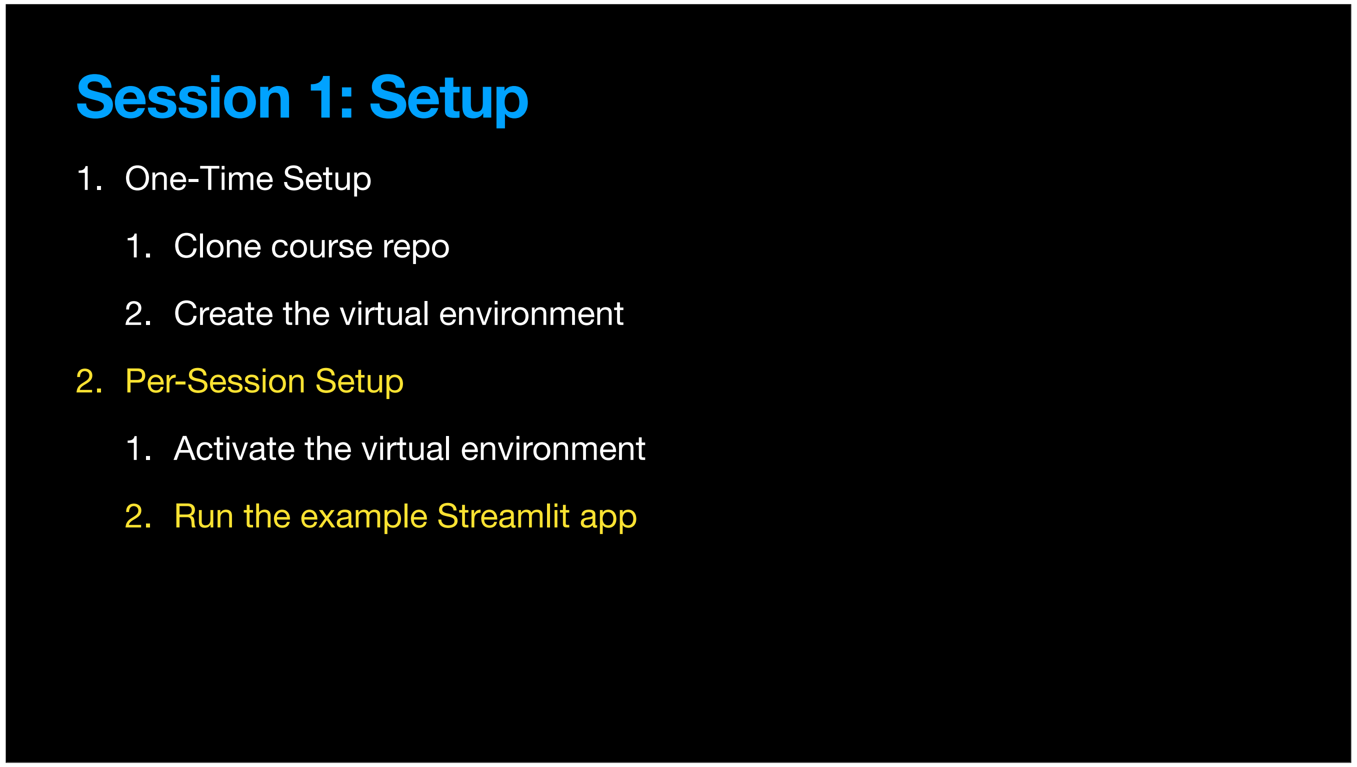


Test the Virtual Environment

- Check that the virtual environment is running:
  - Type **`python check_env.py`**

```
(streamlit-workshop) ➜  streamlit-workshop git:(main) ✗ 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) ✗ 
```

Before we move on, please verify that your virtual environment is running. To do this, type "python check_env.py". This will run a test that should say "Virtual env active: True".



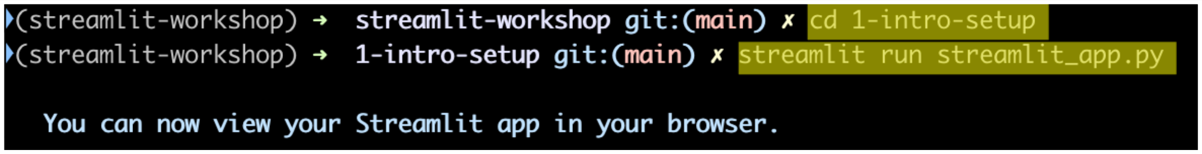
Session 1: Setup

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

Each time you open the terminal, you'll need to activate the virtual environment. And each time you want to run a streamlit app, you'll need to type a specific command.

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

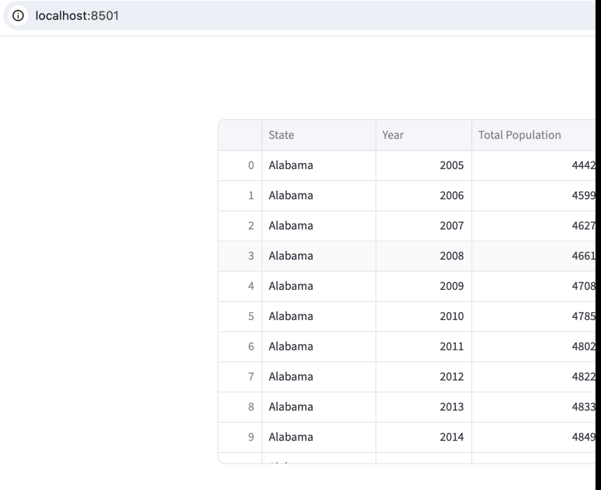


```
(streamlit-workshop) ➜  streamlit-workshop git:(main) ✗ cd 1-intro-setup
(streamlit-workshop) ➜  1-intro-setup git:(main) ✗ streamlit run streamlit_app.py

  You can now view your Streamlit app in your browser.
```

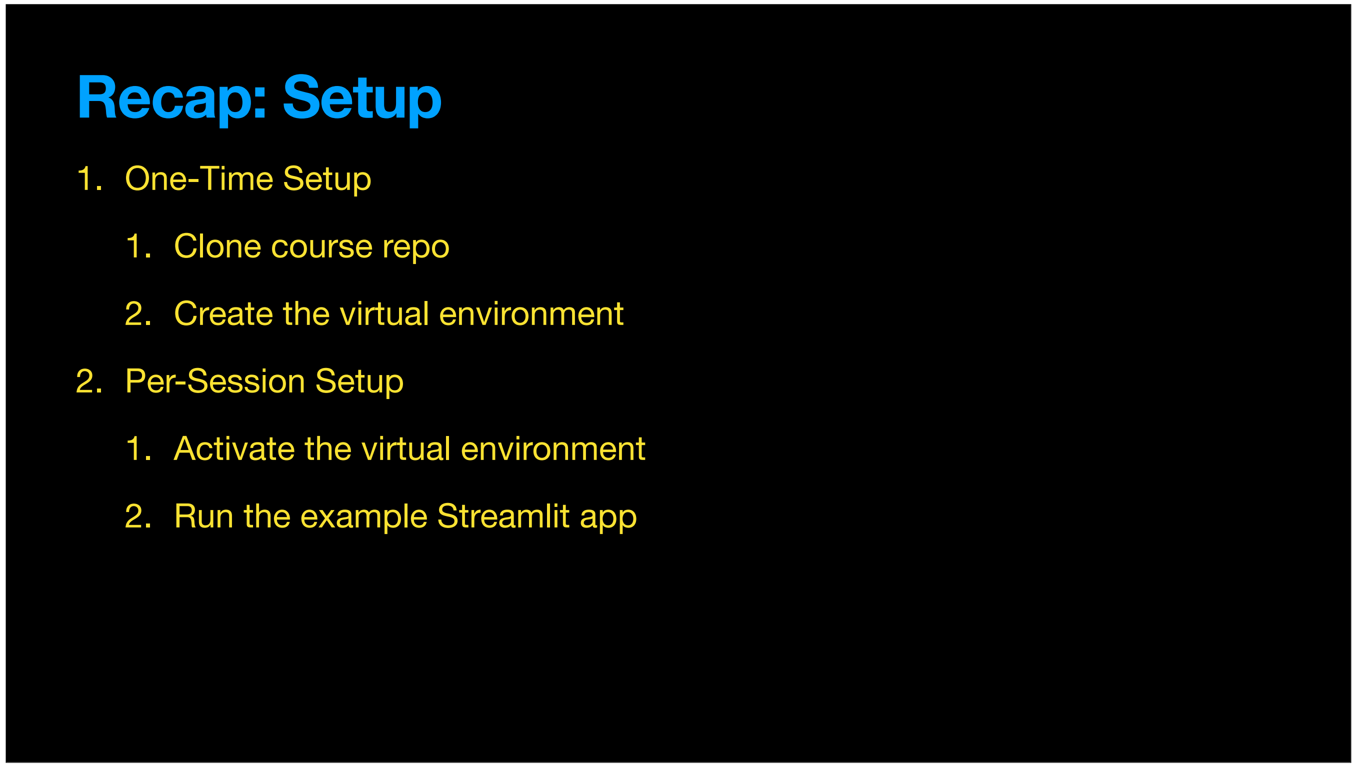
Move to the "session1" directory and type "streamlit run streamlit_app.py"

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

Cloning the repo and creating the virtual environment were one-time actions.
Each time you create a new terminal, you'll need to activate the virtual environment.
And each time you want to run the app, you'll need to type "streamlit run <file>"

Course Agenda

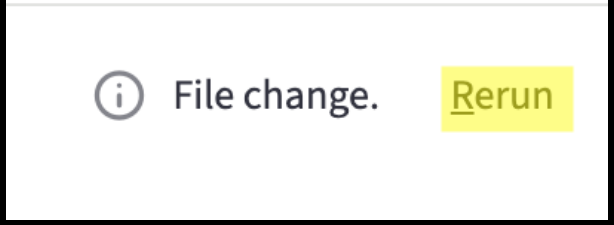
(Big Picture)

1. Session 1: Setup & Basics

Now that you have a simple Streamlit app running, let's run through some basic things that will help you throughout the course.

Rerun

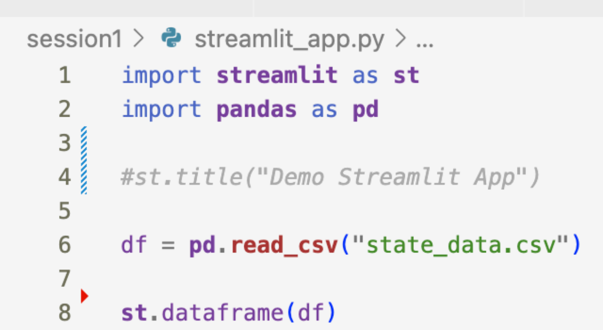
- Streamlit is designed for fast iteration
- Each time you modify and save the script he word Rerun will appear in the top right of the app
- Click it to rerun the app with your changes
- Do not stop and start the app from the command line



File change. Rerun

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

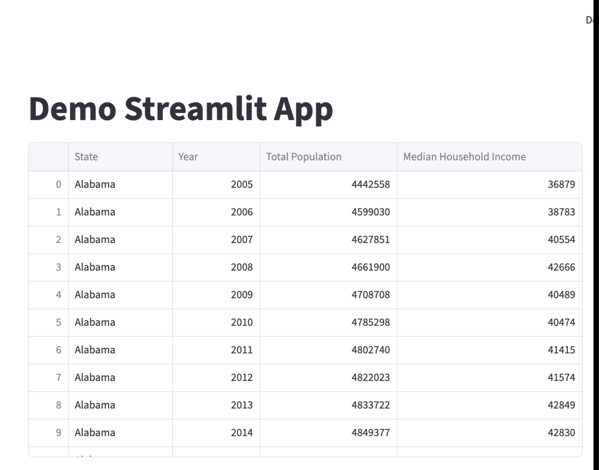


```python
import streamlit as st
import pandas as pd

#st.title("Demo Streamlit App")

df = pd.read_csv("state_data.csv")

st.dataframe(df)
```

This is an exercise to make sure you're comfortable changing the app, and clicking "rerun" to see your changes.

Exercise: Rerun

- Result



Demo Streamlit App

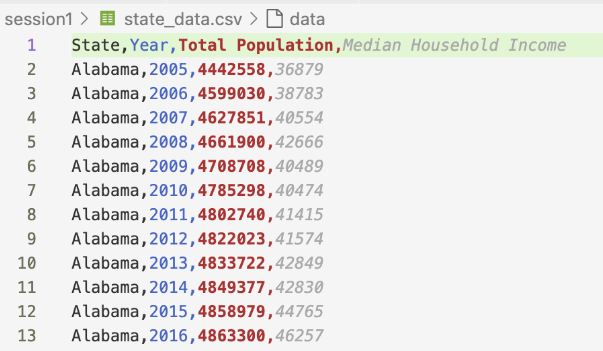
	State	Year	Total Population	Median Household Income
0	Alabama	2005	4442558	36879
1	Alabama	2006	4599030	38783
2	Alabama	2007	4627851	40554
3	Alabama	2008	4661900	42666
4	Alabama	2009	4708708	40489
5	Alabama	2010	4785298	40474
6	Alabama	2011	4802740	41415
7	Alabama	2012	4822023	41574
8	Alabama	2013	4833722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

Demo App Walkthrough

Before we start digging into real streamlit, let's cover what this basic app does. We'll be building on this app during the next session, so it pays to make sure we're all on the same page.

The data

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

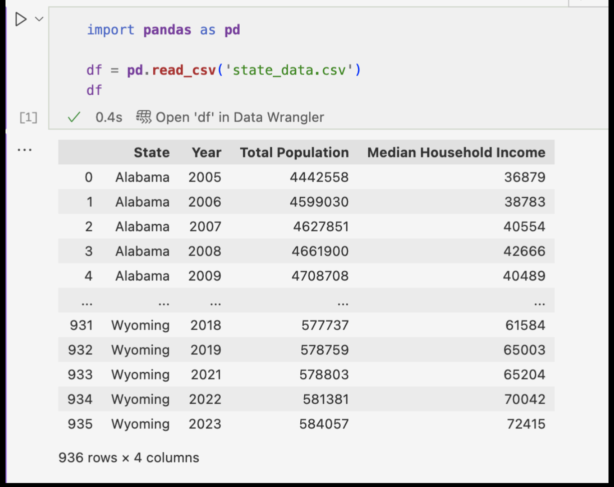


```
State,Year,Total Population,Median Household Income
Alabama,2005,4442558,36879
Alabama,2006,4599030,38783
Alabama,2007,4627851,40554
Alabama,2008,4661900,42666
Alabama,2009,4708708,40489
Alabama,2010,4785298,40474
Alabama,2011,4802740,41415
Alabama,2012,4822023,41574
Alabama,2013,4833722,42849
Alabama,2014,4849377,42830
Alabama,2015,4858979,44765
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:

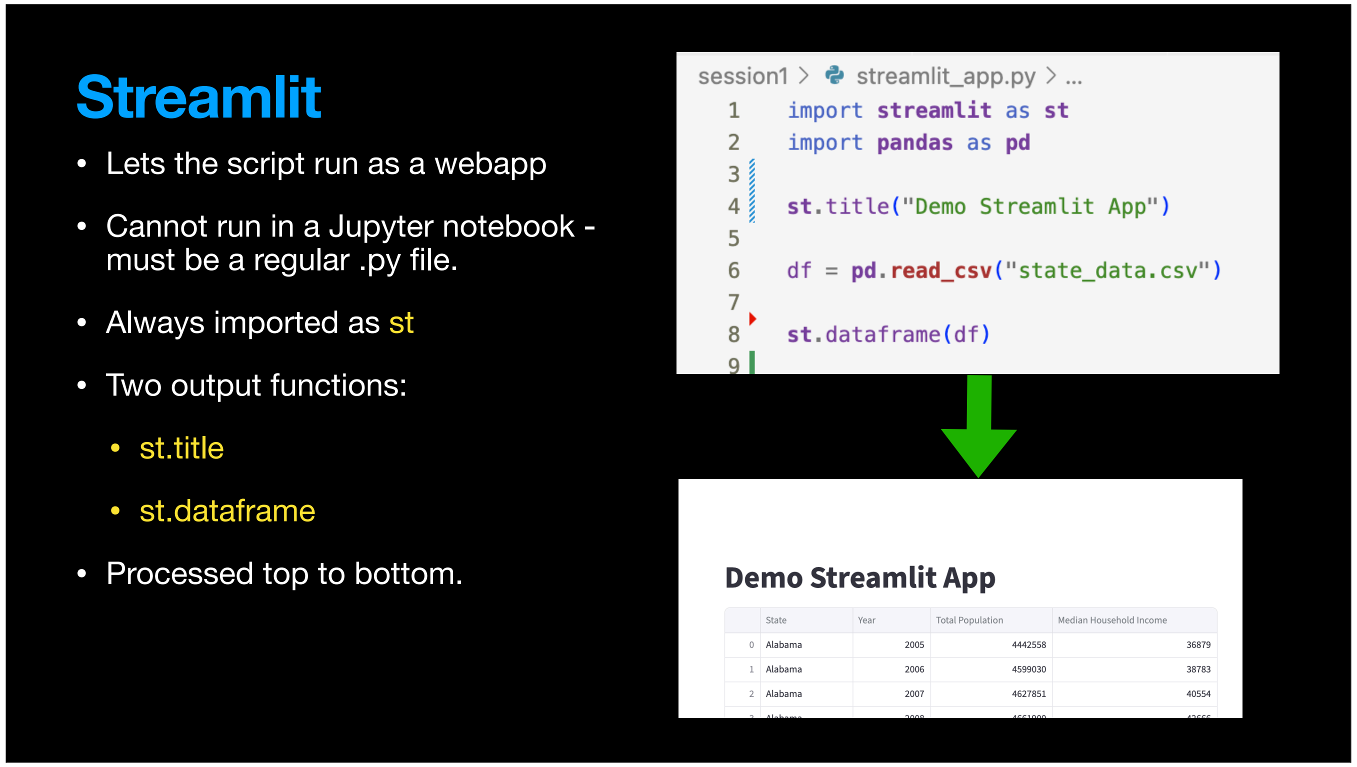
     `uv run --with jupyter jupyter lab`
- It's always imported as pd
- pd.read_csv returns a dataframe



```python
import pandas as pd

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

```
       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 × 4 columns
```

Our demo app has the line "import pandas as pd".
How many people have experience with Jupyter notebooks? With Pandas?
We will be using Notebooks throughout this course: select a kernel, add a cell both markdown and code). Should I demo?



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.

```python
import streamlit as st
import pandas as pd

st.title("Demo Streamlit App")

df = pd.read_csv("state_data.csv")

st.dataframe(df)
```

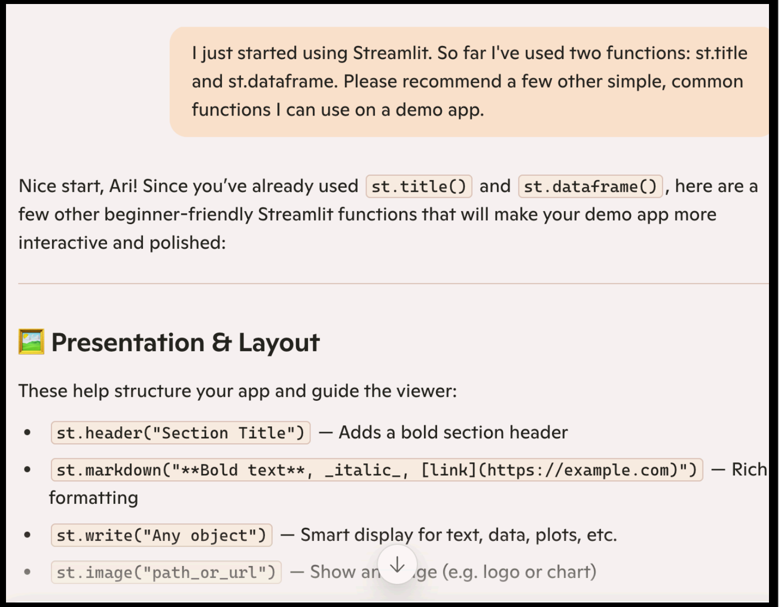
Demo Streamlit App

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

Our demo app also has the line "import streamlit as st".

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, 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.

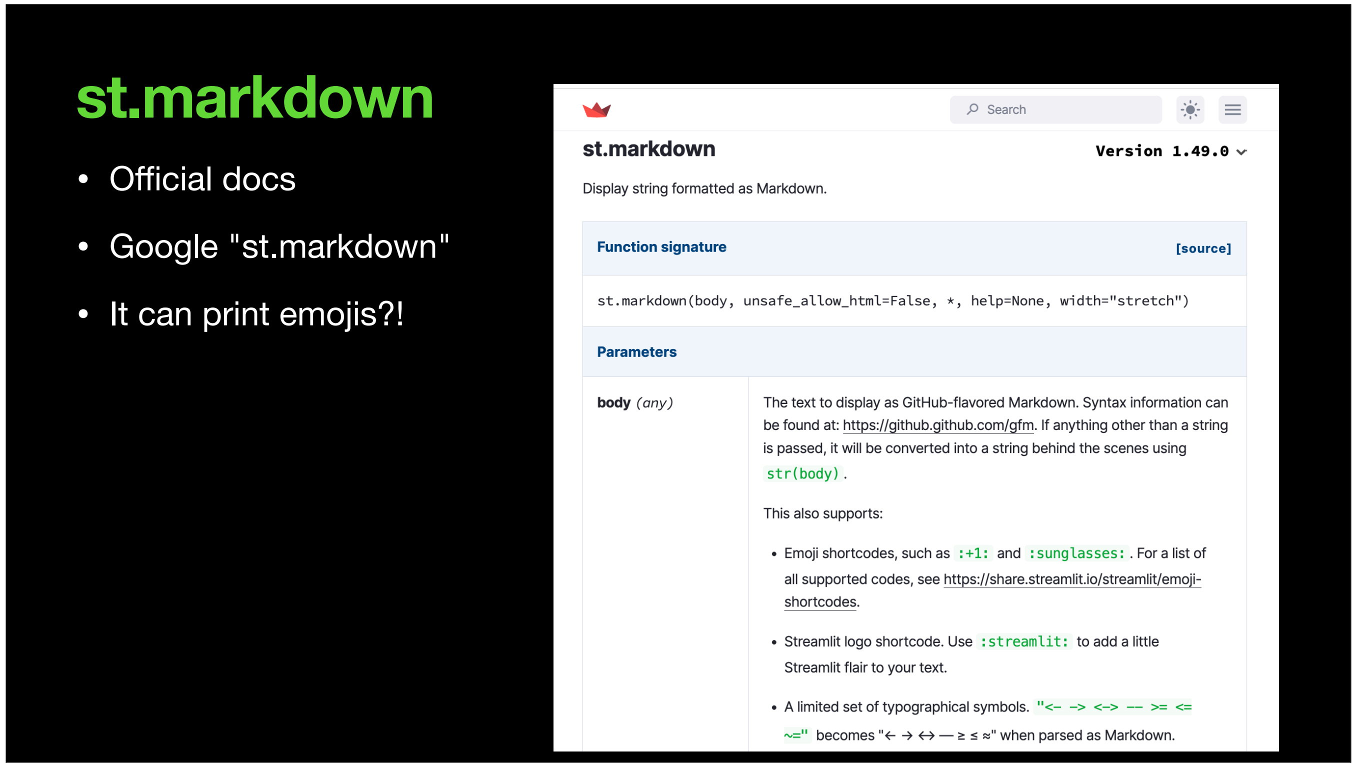
Nice start, Ari! Since you've already used `st.title()` and `st.dataframe()`, here are a few other beginner-friendly Streamlit functions that will make your demo app more interactive and polished:

🖼️ 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 ge (e.g. logo or chart)

I was asked to include advice on coding with LLMs in this course. Here I told an LLM about the demo app, and asked it for recommendations on how I could improve it.
<read prompt>



st.markdown

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

st.markdown

Display string formatted as Markdown.

Function signature

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

Parameters

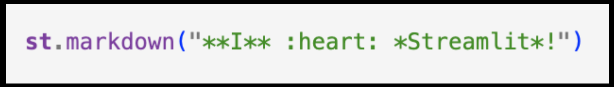
body *(any)*

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)`.

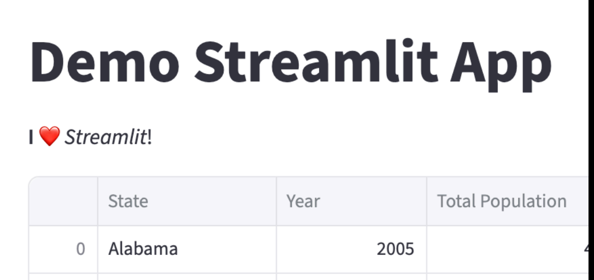
This also supports:

- 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.

I chose to use st.markdown for this exercise. I then googled "st.markdown" to get to the official docs, and was surprised to see that it also accepts emoji shortcodes! Sometimes it's helpful to read the official docs.



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



Demo Streamlit App

I ❤️ *Streamlit*!

	State	Year	Total Population
0	Alabama	2005	

I always get confused whether ** is bold (it is) or italics. You can always ask an LLM for a cheatsheet / introduction on Markdown

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