Building interactive data apps with Python

A gentle introduction to Streamlit

Lorenzo

StatBytes
10/30/2024

Why data apps?

- If either:
  - You have a **theoretical** result (simulations, proofs of concept)
  - You want to disseminate your new **method** (tutorials, simulations)
  - You have an **applied** scientific finding (showcasing)
  - You are **collecting** data (surveys)
  - You are making data available to others (**database** navigation)
  - You want to organize a **conference**, or just a tennis tournament (social events)
  - You want to keep track of how much **coffee** people drink in FMS (our working example today!)
- Then data apps are right for you!

My 2 cents

- Interactive web platforms should not be exceptional tools in research
- They should instead be the norm

Why Streamlit?

- It is easy to use
- It embraces scripting
- It weaves in interactions
- It allows instantaneous deployment
- It is compatible with a lot of libraries and data storage platforms

Streamlit

Installation and validation

- `pip install streamlit`
- `streamlit hello`

Locally running an app

- `streamlit run your_app.py`
- The two fundamental flows:
  - Development flow: connect to local server, write your script, save it, refresh
  - (Default) Data flow: Streamlit reruns your script top-to-bottom every time there is a state change

Data flow

- (Default) Data flow: Streamlit reruns your script top-to-bottom every time there is a state change
- What is a state change?
  - Whenever you modify source code
  - Whenever an user interacts with the app, through widgets

Basic elements

- `import streamlit as st`
- Display and style data
- Draw charts and maps
- Widgets
- Let’s see how these work: https://weightedclimatedata.streamlit.app/

Dealing with app refresh

- How to store information across reruns? Session state!
  - `st.session_state[“my_key”]`
- Widgets do this on their own: no need to worry

Dealing with massive data flows

- `@st.cache_data`: a decorator that is very useful to avoid rerunning chunks of code if the data they manage are already stored in cache
  - Typical example: reading big datasets
- `on_change` (or `on_click`): a parameter through which you can pass a function (a callback) to widgets that is going to be run before the script

Connections

- Many times `st.connection("my_database")` does the job
- Username and password go into the secrets, into `.streamlit/secrets.toml`
  - You can access your secrets using `st.secrets["my_key"]`

Pages

- Create a `pages` folder in the same folder as your home page script, `home.py`
- Add scripts to `pages` folder, one per page
- `streamlit run home.py`

Deployment

- `requirements.txt` (or other dependency files)
  - Only for third party libraries
  - I particularly like Poetry
- `.gitignore` your secrets!
- `.config.toml` your style
- Streamlit Community Cloud share.streamlit.io

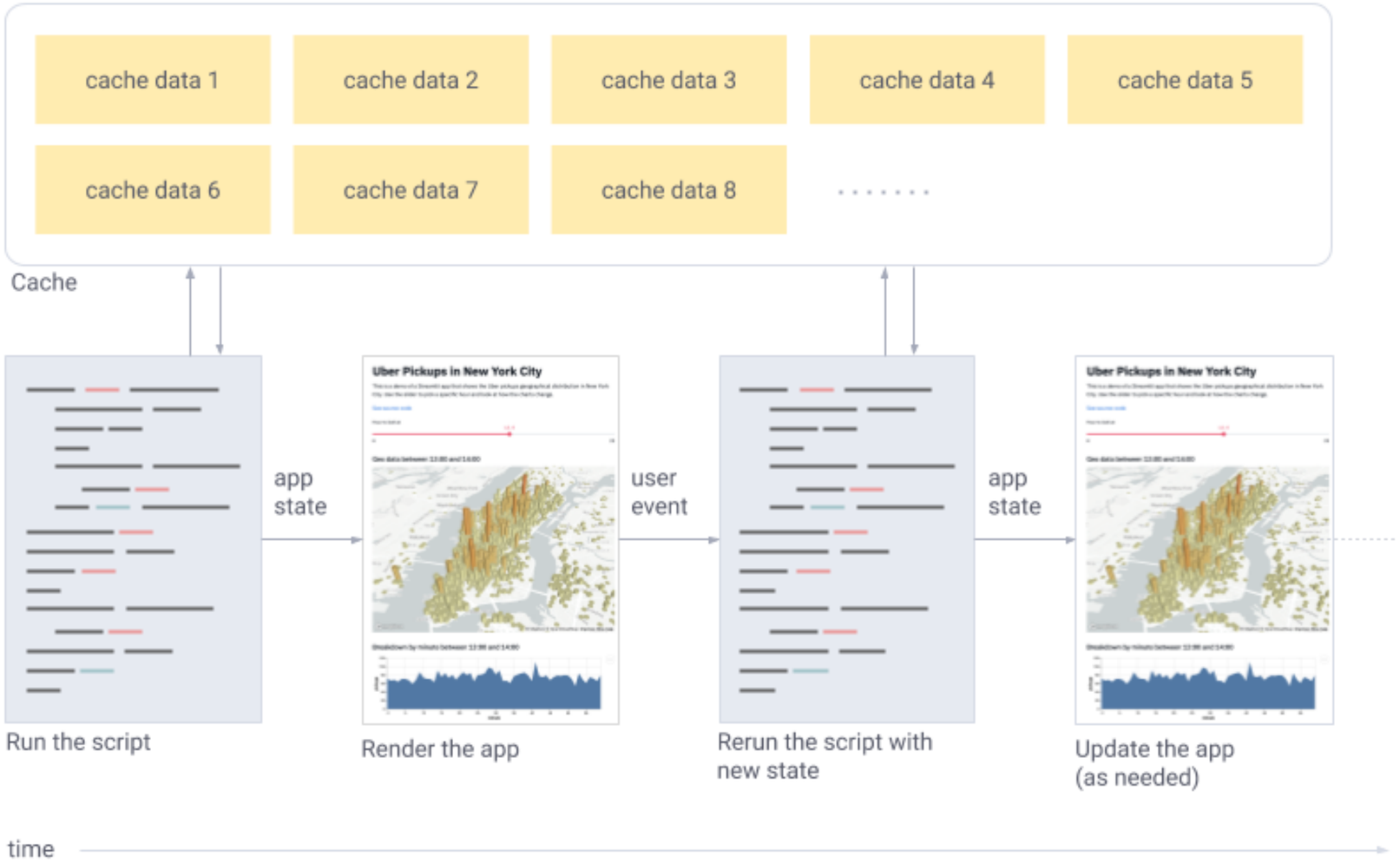
Let’s get started…

- The “lady tasting coffee” problem:
  - Assume you observe ~60 human beings pursuing their PhD in the so-called Facility Management Services (FMS) building. Assume you are a member of the Caffeine Committee, which has just brought a new coffee machine to FMS
  - You want to collect, analyze, and share data on coffee consumption
- Solution: Build a Streamlit data app

The structure of our data app

- 3 pages:
  - Intro: talk about the project
  - Registration: allow users to increase their coffee consumption counts
  - Visualization: show collected data

Summary


cache data 1
cache data 2
cache data 3
cache data 4
cache data 5
cache data 6
cache data 7
cache data 8
Cache
app state
user event
app state
Uber Pickups in New York City
Uber Pickups in New York City
Run the script
Render the app
Rerun the script with new state
Update the app (as needed)
time

Questions?

- Streamlit documentation is superb: https://docs.streamlit.io/
- Community is also great: https://discuss.streamlit.io/
- Feel free to reach out to me at ltesta@andrew.cmu.edu

Thank you!