

Building Data Apps In Python With Streamlit

Session 2: Inputs & Graphics

Ari Lamstein
AriLamstein.com

Exercise: What Inputs? What Graphics?

At the start of the first session I asked you to ask your neighbor this question:

- *What do you want to build after completing this course?*

Now ask your neighbor these followup questions:

- *What inputs do you want your app to have?*
- *What do you want your app to do with those inputs?*

4 minute exercise

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

This is the single most important session in the entire course. Why?

If you understand how to add input widgets to an existing streamlit app, and use those values to decide what graphics to display, then you can build useful apps with Streamlit. Full stop. The other material in this course is important, but not critical in the same way.

Please give this section your undivided attention.

Inputs & Graphics

- Inputs
- Graphics

Please Open


- `session2/input.ipynb`
- `session2/input_app.py`

"Select Box"

- Let the user "Select" which state they want to see data on
- Filter the dataframe to just see that data
- Print the 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



Demo Streamlit App

State:

California

	State	Year	Total Population	Median Household Income
72	California	2005	35278768	
73	California	2006	36457549	
74	California	2007	36553215	

I want the user to be able to see data on just one state. To do this, we'll add what's called a "select box". Think of it like this: "the user gets to 'select' which state to view data on."

In the first image, the table has data on all states. In the second image, the user selects a state, and the dataframe shows data on just that state.

3 Engineering Tasks

1. Create a list of all the state names
2. Input Widget: a select box which contains those names and the user can see
3. Output: dataframe filtered to just the state the user selected

Demo Streamlit App

State:

California

Alabama

Alaska

Arizona

Arkansas

California

Colorado

Connecticut

Delaware

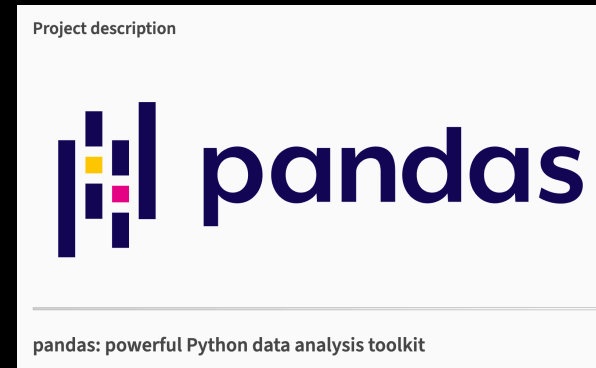
79	California	2012	38041430	
80	California	2013	38332521	
81	California	2014	38802500	

This feature (letting the user filter the dataframe to just one state) actually has 3 separate sub-tasks. When you're working on your own projects, it's helpful to break down tasks in this way as well. It normally takes some experience to get a knack for how to do this.

Let's now focus on the first task - getting a list of all the state names

pandas

- Already saw
 - `import pandas as pd`
 - `pd.read_csv`
- Classes and functions for working with tabular data (i.e. csv files)
- "DataFrame" library
- Downloaded 412M times last month



In order to get the list of the states in the data frame, we're going to use the pandas library. Pandas is the most popular library for working with tabular data in Python. I expect that most people in this course have some prior experience with Pandas. But I want the course to be accessible to people who are new to it as well. So I'll give you a short introduction

Open 2-input-graphics/input.ipynb

- Two ways to open:
 1. VS Code with Jupyter plugin
 2. Jupyter lab via command line:

```
uv run --with jupyter jupyter lab
```

Does anyone need an introduction to Jupyter Notebooks?

- selecting a kernel
- Adding markdown / code cells
- Running code cells

What is a DataFrame?

- A "Rectangle of Data"
- A "collection of columns"
- Each column
 - has a name
 - has values of the same type (text, number, boolean)
 - has the same length

```
type(df)
✓ 0.0s
pandas.core.frame.DataFrame
```

	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

It can be helpful to think of a dataframe as "a rectangle of data". More precise is "a collection of columns, each of the same length".

-Each column has a name.

-The values in each column are of the same type

Each Column is a "Series"

- "Bracket notation"
- You can get "just" a column by typing `df[<column_name>]`
- Column name needs to be in quotes
- Each column is called a "Series"

```
type(df['State'])
```

✓ 0.0s

pandas.core.series.Series

```
df['State']
```

✓ 0.0s

```
0    Alabama
1    Alabama
2    Alabama
3    Alabama
4    Alabama
```

...

```
931   Wyoming
932   Wyoming
933   Wyoming
934   Wyoming
935   Wyoming
```

Name: State, Length: 936, dtype: object

Working with "just" a column of a dataframe is very common. Pandas lets you do this with "bracket notation" - put the column name, in brackets, after the dataframe. The column name needs to be in quotes. You get the data in just the column back. The resulting object is of type "Series".

Series Methods

- `.unique()`
- `.min()`
- `.max()`
- ...

```
df['State'].unique()
```

✓ 0.0s

```
array(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',  
      'Colorado', 'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho',  
      'Illinois', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine',  
      'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi',  
      'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire',  
      'New Jersey', 'New Mexico', 'New York', 'North Carolina',  
      'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania',  
      'Puerto Rico', 'Rhode Island', 'South Carolina', 'South Dakota',  
      'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia',  
      'Washington', 'West Virginia', 'Wisconsin', 'Wyoming'])
```

The Series class has a lot of useful methods. This is one reason why pandas is such a popular library - it makes common operations on dataframes easy. The method we want here is called `unique()`, and it returns the unique values in the series. It's perfect for letting the user select a state from our dataframe.

Exercise 2.1

- Groups of 2
- 10 minutes

Exercise 2.1: Series Methods

1. Calculate the unique values in the `Year` column. Does anything surprise you?
2. Use the `min()` and `max()` methods to calculate the min and max population in the dataset.
3. Ask an LLM for a suggestion of another method to experiment with, and then try it. Try a prompt like "Suggest a pandas Series method to experiment with." and max methods. Please suggest another method for me to try."

Working with Series is so important that I've created 3 exercises for you to work on.

3 Engineering Tasks

1. Create a list of all the state names
2. Input: a select box which has those names
3. Output: dataframe filtered to just the state the user selected

Demo Streamlit App

State:

California

Alabama

Alaska

Arizona

Arkansas

California

Colorado

Connecticut

Delaware

79	California	2012	38041430	
80	California	2013	38332521	
81	California	2014	38802500	

Now let's figure out how to get this select box into the app. This itself actually has two distinct subtasks. The first is to get any dummy selectbox into the app. The second is to update that dummy select box to have the right label (the text that says "State:") and the right values. There are two ways to do this: Google and an LLM.



- Search: "How do I create a select box in Streamlit?"
- Wind up on the official documentation for **st.selectbox**
- Useful to be able to read. But a lot of info we don't need for our first box!
- Count the number of parameters

st.selectboxVersion 1.49.0

Display a select widget.

Function signature[source]

```
st.selectbox(label, options, index=0, format_func=special_internal_function, key=None, help=None, on_change=None, args=None, kwargs=None, *, placeholder=None, disabled=False, label_visibility="visible", accept_new_options=False, width="stretch")
```

Parameters

label (<i>str</i>)	<p>A short label explaining to the user what this select widget is for. The label can optionally contain GitHub-flavored Markdown of the following types: Bold, Italics, Strikethroughs, Inline Code, Links, and Images. Images display like icons, with a max height equal to the font height.</p> <p>Unsupported Markdown elements are unwrapped so only their children (text contents) render. Display unsupported elements as literal characters by backslash-escaping them. E.g., <code>"1\.</code> Not an ordered list".</p> <p>See the <code>body</code> parameter of <code>st.markdown</code> for additional, supported Markdown directives.</p> <p>For accessibility reasons, you should never set an empty label, but you can hide it with <code>label_visibility</code> if needed. In the future, we may disallow empty labels by raising an exception.</p>
options (<i>Iterable</i>)	<p>Labels for the select options in an <code>Iterable</code>. This can be a <code>list</code>, <code>set</code>, or anything supported by</p>

Ask AI

If you google "How to create a select box in Streamlit" you'll wind up on the official documentation page for the function `st.selectbox`. This is the correct function. And learning how to navigate pages like this is a useful skill to gain. But it can be a bit of overkill for a beginner. It's most useful for understanding all the options.

LLM

- Prompt: *"How to add a select box in streamlit"*
- Response: copy-paste sample code
 - Only has the two required parameters

how to add a select box in streamlit

To add a **select box** in Streamlit, use the `st.selectbox()` function. It lets users choose one option from a dropdown menu—perfect for filtering or setting parameters in your app.

Basic Example

```
Python ^ Copy
import streamlit as st

option = st.selectbox(
    'Choose your favorite fruit:',
    ['Apple', 'Banana', 'Cherry']
)

st.write('You selected:', option)
```

Choose your favorite fruit:

Apple

You selected: Apple

Exercise: input_app.py

- The file `session2/input_app.py` has the Streamlit example code for select boxes added to the app.
- Update it to:
 - Let the user select from the unique states
 - Have a label that says something related to the options

how to add a select box in streamlit

To add a **select box** in Streamlit, use the `st.selectbox()` function. It lets users choose one option from a dropdown menu—perfect for filtering or setting parameters in your app.

Basic Example

```
Python ^ Copy
import streamlit as st

option = st.selectbox(
    'Choose your favorite fruit:',
    ['Apple', 'Banana', 'Cherry']
)

st.write('You selected:', option)
```

Choose your favorite fruit:

Apple

You selected: Apple

3 Engineering Tasks

1. Create a list of all the state names
2. Input: a select box which has those names
3. **Output: dataframe filtered to just the state the user selected**

Demo Streamlit App

Select a State:

Wyoming

You selected: Wyoming

	State	Year	Total Population	Median Household In
0	Alabama	2005	4442558	
1	Alabama	2006	4599030	
2	Alabama	2007	4627851	

Right now we have a perfectly functioning select box. But we're still displaying the full dataframe. For example, I selected "Wyoming" but the dataframe still has rows for Alabama. To fix this, we'll have to use what Pandas calls "Boolean Indexing".

Boolean Indexing

- Filter rows based on a condition
- First create a boolean Series
- Then feed the series to the dataframe using `[]` notation
- Open [session2/input.ipynb](#).
 - Section "Filtering a DataFrame (Boolean Indexing)"
- **Example:** Show me rows where the population is $< 1M$

	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

Boolean indexing lets you filter rows in a DataFrame based on a condition. You create a Series of True and False values—called a Boolean mask—and use it to select only the rows where the condition is True.

I'll walk you through an example of selecting rows that have a population less than 1 million. Then I'll give you an exercise where you select rows with a particular state.

Note that Alabama has a population $> 1M$, and Wyoming has a population $< 1M$

Column / Series

- We care about the **Total Population** column / series
- We want to create a new Series where each element is
 - **True** if the element is $< 1\text{M}$
 - **False** if the element is not $< 1\text{M}$

```
df['Total Population']  
✓ 0.0s  
0      4442558  
1      4599030  
2      4627851  
3      4661900  
4      4708708  
...  
931     577737  
932     578759  
933     578803  
934     581381  
935     584057  
Name: Total Population, Length: 936, dtype: int64
```

Vectorization

- Pandas lets you **vectorize** a test over a Series
- This applies the test to each element of the Series
- The result is a boolean Series
- The length of this series is the same as the length of the original series
 - Which is the same as the length of the original dataframe

```
df['Total Population'] < 1000000
✓ 0.0s
```

0	False
1	False
2	False
3	False
4	False
...	...
931	True
932	True
933	True
934	True
935	True

Name: Total Population, Length: 936, dtype: bool

Boolean Indexing

- Put a Boolean Series inside `[]` after a dataframe
- First element of the series corresponds to the first row of the dataframe.
- Rows that have `True` are returned

```
df[df['Total Population'] < 1000000]
```

✓ 0.0s

	State	Year	Total Population	Me
18	Alaska	2005	641724	
19	Alaska	2006	670053	
20	Alaska	2007	683478	
21	Alaska	2008	686293	
22	Alaska	2009	698473	
...	
931	Wyoming	2018	577737	
932	Wyoming	2019	578759	

You can read it as "df where the total population column is less than 1M"

Confusing Notation

- `[]` is being used twice
- Inner `[]`:
 - `df['Total Population']`
 - Given a string
 - Returns a Series
 - Conceptually: Column extraction
- Outer `[]`:
 - `df[<Boolean Series>]`
 - Given a Boolean Series
 - Returns a dataframe
 - Conceptually: Row filtering

```
df[df['Total Population'] < 1000000]
```

✓ 0.0s

	State	Year	Total Population	Me
18	Alaska	2005	641724	
19	Alaska	2006	670053	
20	Alaska	2007	683478	
21	Alaska	2008	686293	
22	Alaska	2009	698473	
...	
931	Wyoming	2018	577737	
932	Wyoming	2019	578759	

Sometimes people find Boolean Indexing to be confusing. I think this is because on a single line, the bracket notation is being used twice. And each time, it's being used in a completely different way.

2-line Option

- Put the creation of the Boolean Series on its own line
- Convention: call the Series **mask**.

```
mask = df['Total Population'] <= 1000000  
df[mask]
```

✓ 0.0s

	State	Year	Total Population	Median Household Income
18	Alaska	2005	641724	7500
19	Alaska	2006	670053	7500
20	Alaska	2007	683478	7500
21	Alaska	2008	686293	7500
22	Alaska	2009	698473	7500
...
931	Wyoming	2018	577737	7500
932	Wyoming	2019	578759	7500
933	Wyoming	2021	578803	7500
934	Wyoming	2022	581381	7500
935	Wyoming	2023	584057	7500

130 rows x 4 columns

Exercise 2.2: Boolean Indexing

- Groups of 2
- 10 minutes

Exercise 2.2: Boolean Indexing

1. Create a variable called `mask` that says which rows in `df` are in California. Use it to subset the dataframe.
2. Update your app to filter the dataframe to show rows in the state the user selected.

Use masking in a Jupyter Notebook to subset which rows in the dataframe are in California.
Then update the app to subset the dataframe based based on the state the user selected

```
session2 > input_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  # Exercise: Change this code to:
9  # 1. Ask the user to select a state
10 # 2. Have it populate with all the list of
11 option = st.selectbox(
12     'Select a State:',
13     df['State'].unique()
14 )
15
16 mask = df['State'] == option
17 df = df[mask]
18
19 st.dataframe(df)
20
```

Demo Streamlit App

Select a State:

California

	State	Year	Total Population
72	California	2005	
73	California	2006	
74	California	2007	

My solution

Inputs & Graphics

- Inputs
- Graphics

Please Open

- [2-input-graphics/graphs.ipynb](#)
- [2-input-graphics/graph_app.py](#)

Exercise: What Graphics?

At the start of the first session I asked you to ask your neighbor this question:

- *What do you want to build after completing this course?*

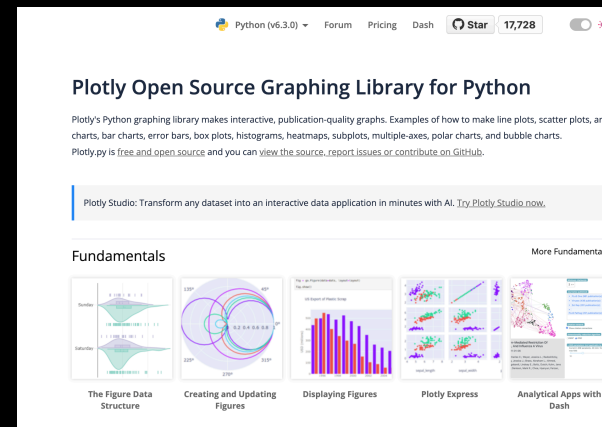
Now ask your neighbor this followup question:

- *What graphs do you want your app to have?*

4 minute exercise



- Popular library for interactive graphics
- "Plotly Express"
 - `import plotly.express as px`
 - `px.line`
 - `px.bar`
 - `px.choropleth`
- Downloaded 31M times last month



Plotly is a popular Python library for making interactive graphics. This lets users hover their mouse over a point and see more information about it. It's common to use interactive graphics when making web apps.

Plotly has a module called "Plotly Express" that lets you easily make common graphics. Each function in that module has a descriptive name. We'll be using Plotly Express in this course.

px.line()

Graph Population over Time

	State	Year	Total Population	Median Household Income
72	California	2005	35278768	53629
73	California	2006	36457549	56645
74	California	2007	36553215	59948
75	California	2008	36756666	61021
76	California	2009	36961664	58931
77	California	2010	37349363	57708
78	California	2011	37691912	57287

```
import plotly.express as px

px.line(
    df_ca,
    x="Year",
    y="Total Population",
    title="Population of California over Time"
)
```

How would we graph population over time? Let's start by reviewing the structure of our data.

We have a dataframe with columns like Year and Total Population. Plotly uses those column names when making the x- and y-axis.

We start by importing plotly express as px. Then we call px.line. The first argument is our dataframe. Then we specify the column we want to be the x-axis. Then we specify the column we want to be the y-axis. Then we specify the title.

Exercise: Plotly Graphs in Jupyter Notebooks

Exercise: Plotly Graphs in Jupyter Notebooks


1. Use `px.bar` to create a graph of California's population over time. What are the pros and cons of each visualization?
2. The line graph would be clearer if it also had a point for each data point. Copy the `px.line` code above into Copilot and ask it to modify it so that it also had points for each datapoint.

Hint: `px.bar` takes the same parameters as `px.line`.

modify this code so it has a marker for each point:

```
px.line(  
    df_ca, x="Year", y="Total Population", title="Population of  
California over Time"  
)
```

To add a marker for each point in your Plotly Express line chart, you can use the `markers=True` argument. Here's the modified code:

```
Python ^  Copy  
  
import plotly.express as px  
  
px.line(  
    df_ca,  
    x="Year",  
    y="Total Population",  
    title="Population of California over Time",  
    markers=True # This adds a marker at each data point  
)
```

Plotly Graphs in Streamlit

st.plotly_chart

Version 1.49.0 ▾

Display an interactive Plotly chart.

Plotly is a charting library for Python. The arguments to this function closely follow the ones for Plotly's `plot()` function.

To show Plotly charts in Streamlit, call `st.plotly_chart` wherever you would call Plotly's `py.plot` or `py.iplot`.

Calling `px.line()` in a Streamlit app won't render the plot. You need to call `st.plotly_chart`.

Plotly Graphs in Streamlit

```
fig = px.line(...)  
st.plotly_chart(fig)
```

This is the most common pattern for rendering plotly charts in Streamlit. You call a plotly express function like normal. But you store the result in a variable call fig. And then you pass fig to st.plotly_chart. This is what I'll ask you to do in the next exercise.

f-Strings

f-Strings

For the next exercise, it will help to know about "f-strings".

```
state = "New York"  
title = f"Graph for {state}"  
title
```

✓ 0.0s

'Graph for New York'

In the next exercise, I'm going to ask you to make a graph that updates based on the state that the user selects. You'll probably want the title of the graph to update with the name of the state as well. To do that, it's helpful to know about f-strings.

The "f" stands for "formatted". To make an f-string, put the letter 'f' in front of the quotes. Then put a variable inside curly braces. The value inside the variable will become part of the string.

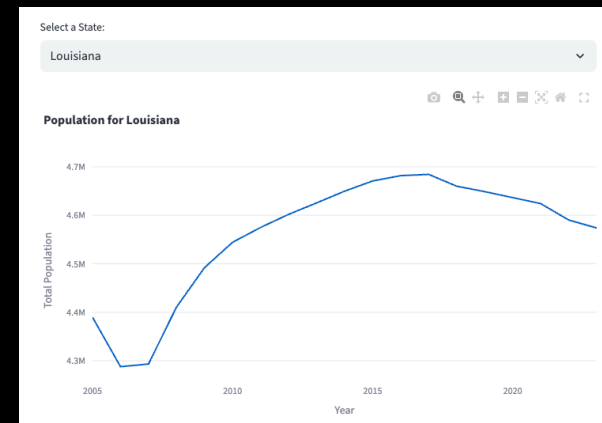
Exercise: Plotly in Streamlit

Exercise: Plotly Graphs in Streamlit

Update `graph_app.py` to create a line graph of the population of the selected State.

Solution

```
fig = px.line(df_state,  
              x="Year",  
              y="Total Population",  
              title=f"Population for {state}")  
st.plotly_chart(fig)
```

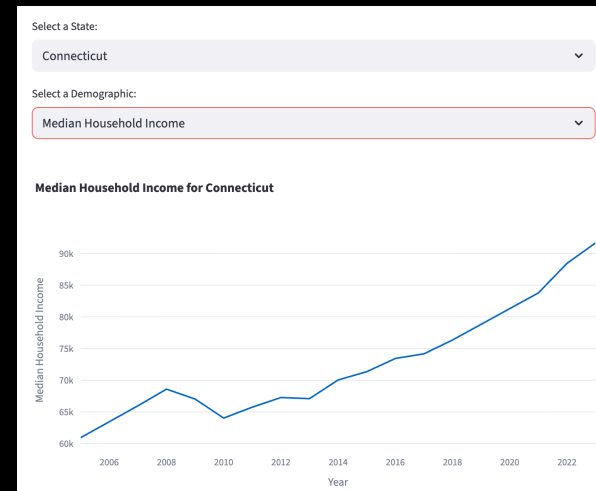


And here's the result.

Final Exam: Q1

UI & Graphs

- Add a new select box to `graph_app.py`.
 - Options: **Total Population** and **Median Household Income**.
- Store the value returned from that selectbox in a variable called **demographic**.
- Use the value of **demographic** as the value for **y** in your line graph.



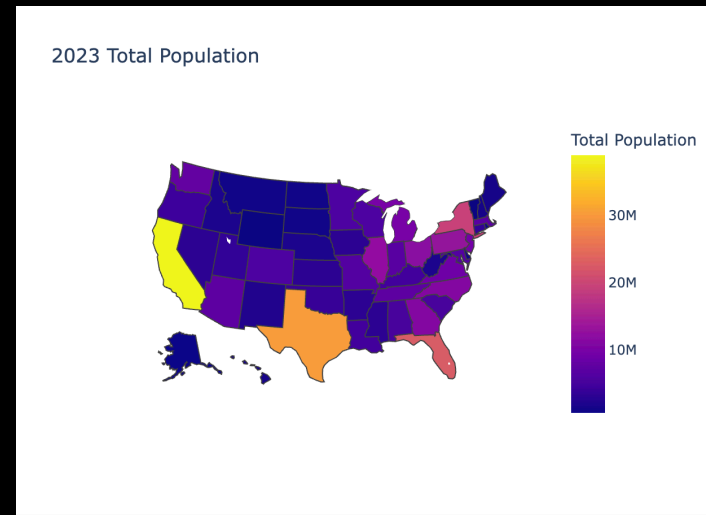
In some sense this is the final exam, because if you can do this, you can do anything with Streamlit. But after this I'm going to teach you how to make maps, and then we'll do a fun exercise with an LLM.

Solution

```
1  import streamlit as st
2  import pandas as pd
3  import plotly.express as px
4
5  st.title("Demo Streamlit App")
6
7  df = pd.read_csv("state_data.csv")
8
9  state = st.selectbox("Select a State:", df["State"].unique())
10 demographic = st.selectbox(
11     "Select a Demographic:", ["Total Population", "Median Household Income"]
12 )
13
14 mask = df["State"] == state
15 df_state = df[mask]
16
17 fig = px.line(df_state, x="Year", y=demographic, title=f"{demographic} for {state}")
18 st.plotly_chart(fig)
19
20 st.dataframe(df_state)
```

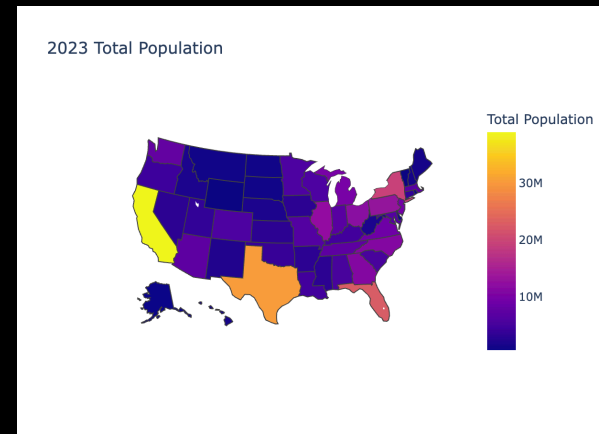

Choropleth Maps

- Shows **locations** such as states
- Uses **color** to express values
- Different than a heat map



Example Choropleth Code

```
px.choropleth(  
    df_2023,  
    locations='State Abbrev', # Column for region  
    locationmode='USA-states',  
    color='Total Population', # Column for color  
    scope='usa',  
    title='2023 Total Population'  
)
```



Different parameters than line and bar chart. First parameter is still a dataframe.

Still have title. But no x or y.

`locations` is the column plotly uses to decide where to draw.

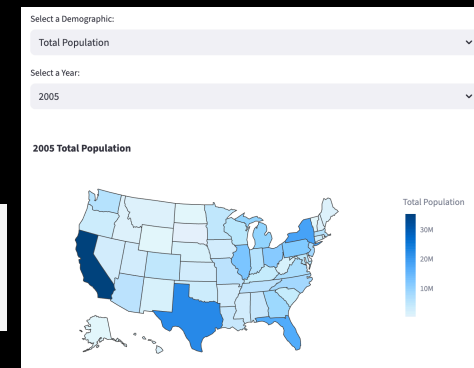
`color` is the column plotly uses to decide what to draw.

scope tells plotly to zoom in on the us. Locationmode tells plotly that we're using postal abbreviations to identify the states. This is the only mode for mapping states.

Final Exam: Q2

UI & Graphs

1. Copy the above code, verbatim, to the app. Verify that it works.
2. Connect the map to the **demographic** selectbox. Do the results surprise you?
3. Create a new selectbox that lets the user select which **Year** of data to map.
4. Connect the year selectbox to the map.



The second question of you final exam.

At the start, the map will not be connected to any UI widgets. Gradually connect it: first demographic then year.

Solution

```
demographic = st.selectbox(
    "Select a Demographic:", ["Total Population", "Median Household Income"]
)
year = st.selectbox("Select a Year:", df["Year"].unique())

mask = df["Year"] == year
df_choro = df[mask]

fig = px.choropleth(
    df_choro,
    locations="State Abbrev", # Column for region
    locationmode="USA-states",
    color=demographic, # Column for color
    scope="usa",
    title=f"{year} {demographic}",
)
st.plotly_chart(fig)
```

The year selectbox works just like the state selectbox - use .unique() because 2020 is missing.

Up Next: Layouts

- Organize the User Interface (UI) a bit better
- Move select boxes into columns
- Move graphs into tabs

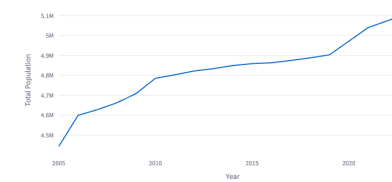
Demo Streamlit App

Select a State:
Alabama

Select a Demographic:
Total Population

Select a Year:
2005

(demographic) of (state) over Time



2005 Total Population



	State	Year	Total Population	Median Household Income	State Abbrev
0	Alabama	2005	4442558	36879	AL
1	Alabama	2006	4599510	37883	AL

The app is interesting but a bit too crowded. In the next section we'll learn how to organize the graphs and selectboxes a bit better.

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

Reiterate course outline, and that this section is the most important skill when working with Streamlit.

Next Steps?

- Visit any LLM (ex. copilot.microsoft.com)
- Prompt: "Here's an app I wrote. Any ideas how it could be improved?"
- Copy-paste your app into the box.
- With a partner: evaluate the suggestions.

Here's a streamlit app I wrote. Any ideas on how to make it better?

```
import streamlit as st
import pandas as pd
import plotly.express as px

st.title("Demo Streamlit App")

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

state = st.selectbox("Select a State:", df["State"].unique())
demographic = st.selectbox("Select a Demographic:", ["Total Population", "Median Household Income"])

mask = df["State"] == state
```

Perhaps you have some thoughts on how the app could be improved.
One idea is to ask an LLM for suggestions.