INFO5002: Intro to Python for Info Sys

Streamlit

https://streamlit.io/

Slides created by: Zachary Doucet

Northeastern University

We have been terminal bound

- Terminal does not enable user interactivity easily
- GUI (Graphical User Interface) enable the user to have a rich display with easy interactivity
  - Imagine if everything you did on your computer was through a terminal—including Photoshop.

Getting started

- First install streamlit:

```
pip install streamlit
```

- Can then import it:

```
import streamlit as st
```

- Can then run the script:

```
streamlit run script.py
```

Whenever you update your code file the webpage will automatically rerun for fast programming

Streamlit

- Package that makes writing GUI code easily
- The screen is updated (repainted) when either
  - Code has been changed
  - User interacts with an element on screen
- The update is done by Streamlit rerunning the entire file again.

Start displaying with Magic

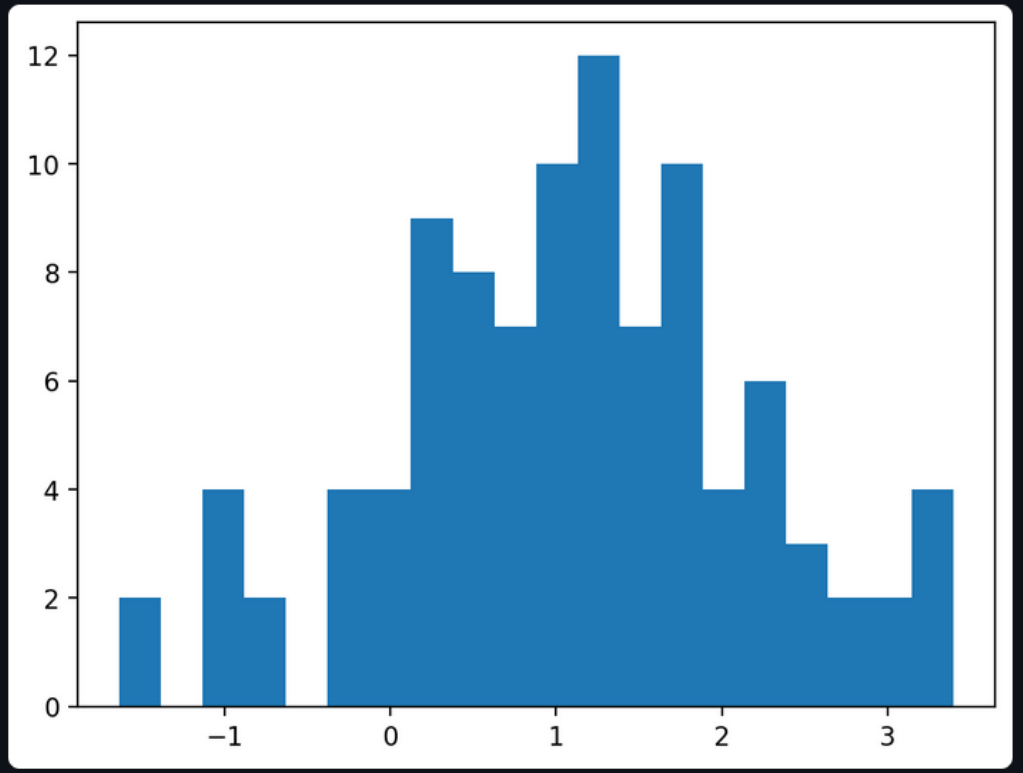
- Any **variable** or **literal value** you display on its own line will be displayed to the screen

```
import streamlit as st
import pandas as pd
df = pd.DataFrame(
   {'col1': [1,2,3]})
df

x = 10
'x', x
```

```
import matplotlib.pyplot as plt
import numpy as np
arr = np.random.normal(1, 1,
   size=100)
fig, ax = plt.subplots()
ax.hist(arr, bins=20)

fig
```

	col1
0	1
1	2
2	3

X 10



Magic calls st.write()

- The magic will behind the scenes pass the variables and literal values into st.write(). They do the same thing.
- You will not want to use magic if you want to customize the look. In that case use the more specific function like st.dataframe() and st.table().
  - Can see all here: https://docs.streamlit.io/develop/api-reference

```python
import streamlit as st
import pandas as pd
df = pd.DataFrame({'col1': [1,2,3],
   'col2': [4,5,6]})

"Magic"
df

st.write("st.write")
st.write(df)

"st.dataframe"
st.dataframe(df, width="content",
height="content", hide_index=True,
on_select="ignore", row_height=50)
```

Magic way as before

st.write explicit

Specific function which gives more customization

Magic

	col1	col2
0	1	4
1	2	5
2	3	6

st.write

	col1	col2
0	1	4
1	2	5
2	3	6

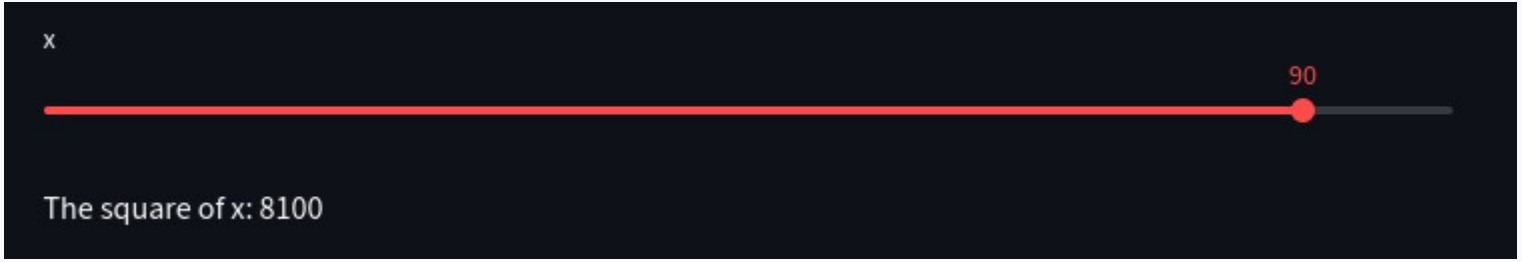
st.dataframe

col1	col2
1	4
2	5
3	6

Functionality with widgets

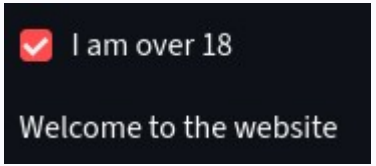
- Widgets should be treated like variables and will return data based on the interactions of the user

```
import streamlit as st
x = st.slider("x")
f"The square of x: {x*x}"
```



```
first_name = st.text_input("First name")
f"Welcome {first_name}"
```


```
over_18 = st.checkbox("I am over 18")
if over_18:
    "Welcome to the website"
else:
    "Sorry we cannot let you in"
```



```
option = st.selectbox("Favourite fruit?",
   ["apple", "pear", "banana", "orange"])
f"I love {option} too"
```

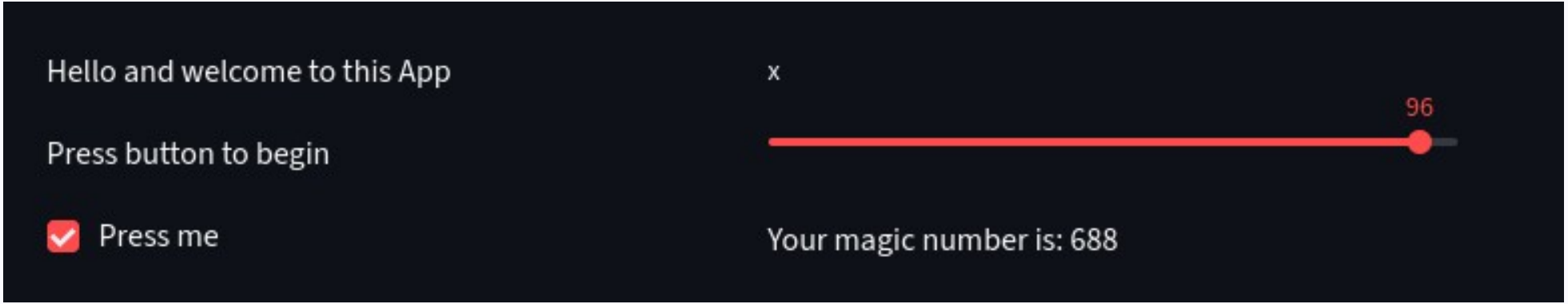
And so much more: https://docs.streamlit.io/develop/api-reference/widgets

Can place widgets into sidebar

```
import streamlit as st
x = st.sidebar.slider("x")
f"Value of x: {x}"
```

Organize with columns

```
import streamlit as st
left_col, right_col = st.columns(2)

left_col.write("Hello and welcome to this App")
left_col.write("Press button to begin")
clicked = left_col.checkbox("Press me")

if clicked:
    x = right_col.slider("x")
    right_col.write(f"Your magic number is:
      {(x**3+52)%1263}")
```


Hello and welcome to this App
Press button to begin
Press me
x
96
Your magic number is: 688

Data processing can take time

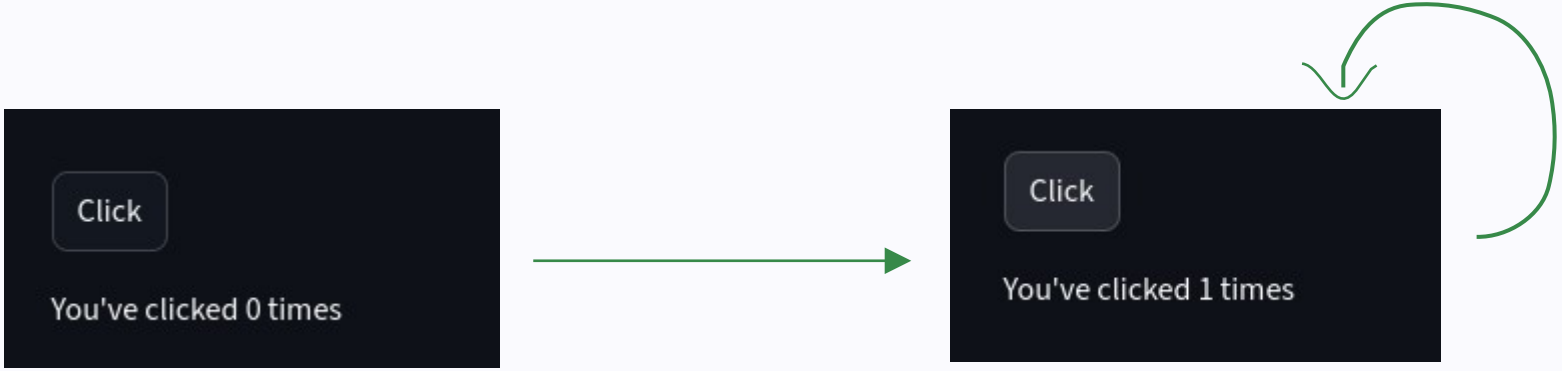
```
import streamlit as st
import time

"Starting a long computation..."
# Add a placeholder
latest_iteration = st.empty()
bar = st.progress(0)

for i in range(100):
  # Update the progress bar with each iteration.
  latest_iteration.text(f"Iteration {i+1}")
  bar.progress(i + 1)
  time.sleep(0.1)
"...and now we're done!"
```


Starting a long computation...
Iteration 19

What if we want to track data across renders?

- Whenever user interacts with a widget the whole script runs from scratch. Let's imagine creating a click counter.

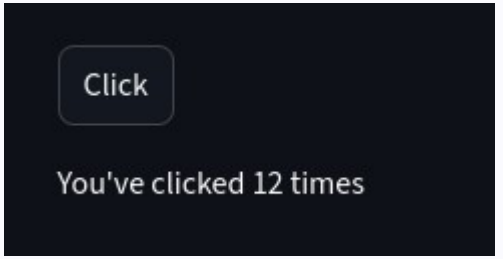
```
import streamlit as st

cnt = 0
clicked = st.button("Click")
if clicked:
    cnt += 1
f"You've clicked {cnt} times"
```


Click
You've clicked 0 times
Click
You've clicked 1 times

Use session state

- Each tab will have its own session state until the tab is **closed** or **refreshed**.

```
import streamlit as st

if "cnt" not in st.session_state:
    st.session_state["cnt"] = 0
clicked = st.button("Click")
if clicked:
    st.session_state["cnt"] += 1
f"You've clicked
  {st.session_state['cnt']} times"
```



Let’s practice

- Create a streamlit app where you will be using the StudentPerformanceFactors.csv file.
  - Have a dropdown where you can select either Female or Male
  - Have a button that says “Generate histogram” that will then generate a histogram of either the females’ or males’ grades
  - Add other features as you see fit