Building

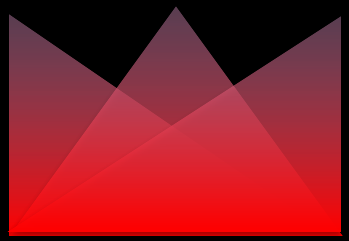
Interactive Web Application with Streamlit

Developing Application with Object-Oriented Design

Object-Oriented Programming 2/2567

@wichit2s

https://wichit2s.github.io/courses/oop/



Hourly Agenda

Hour 1: Introduction to Streamlit

Hour 2: Components & Widgets

Hour 3: Data Visualization

Hour 4: Mini-Project

Objectives

- What Streamlit is and why it's useful
- How to build interactive web apps with minimal code
- Using Streamlit components for UI and data visualization
- Deploying Streamlit apps

What is Streamlit?

https://docs.streamlit.io/

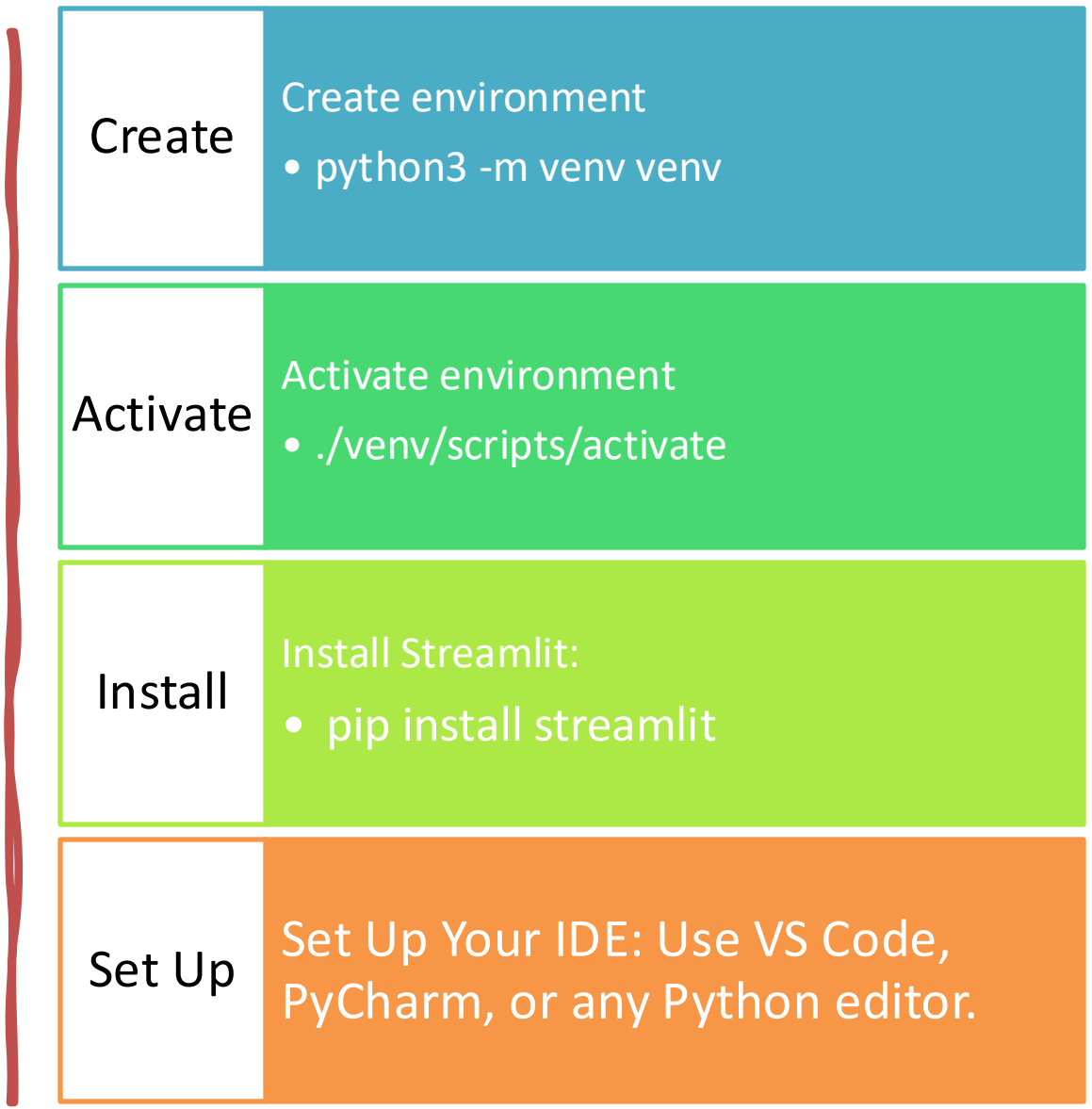
Definition:

an open-source Python library that makes it easy to build **interactive web applications** with just a few lines of code—without needing HTML, CSS, or JavaScript.

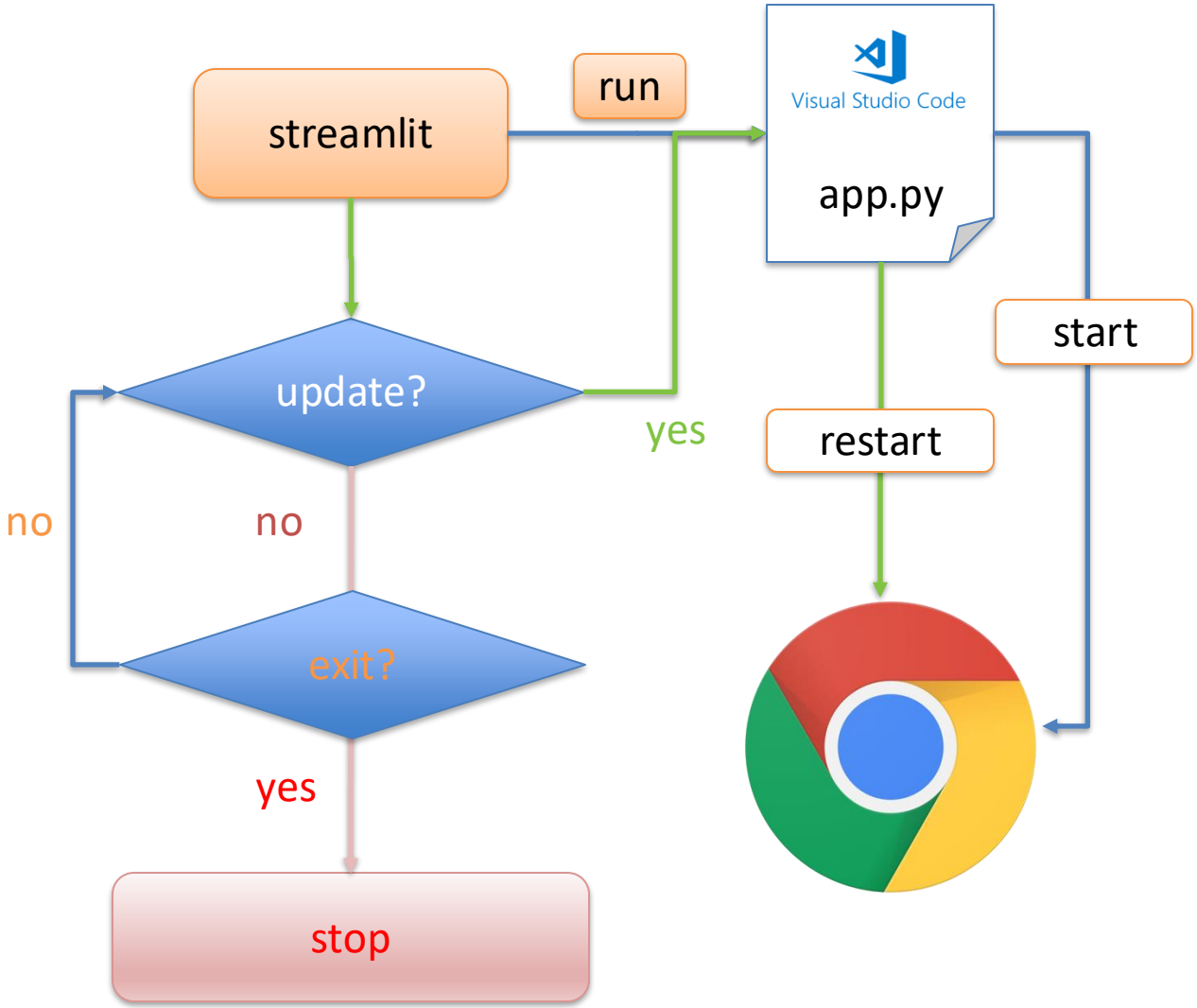
Features:

- **Simple & Fast** – Write Python scripts, and Streamlit converts them into web apps instantly.
- **No Front-End Required** – No need to learn HTML, CSS, or JavaScript.
- **Interactive UI** – Supports widgets, forms, charts, and media with minimal effort.
- **Data Science Friendly** – Easily integrates with Pandas, Matplotlib, Plotly, and machine learning libraries.
- **Easy Deployment** – Deploy apps with **Streamlit Community Cloud** or platforms like Heroku and AWS.

Installation and Setup



Streamlit Application Life Cycle Basics



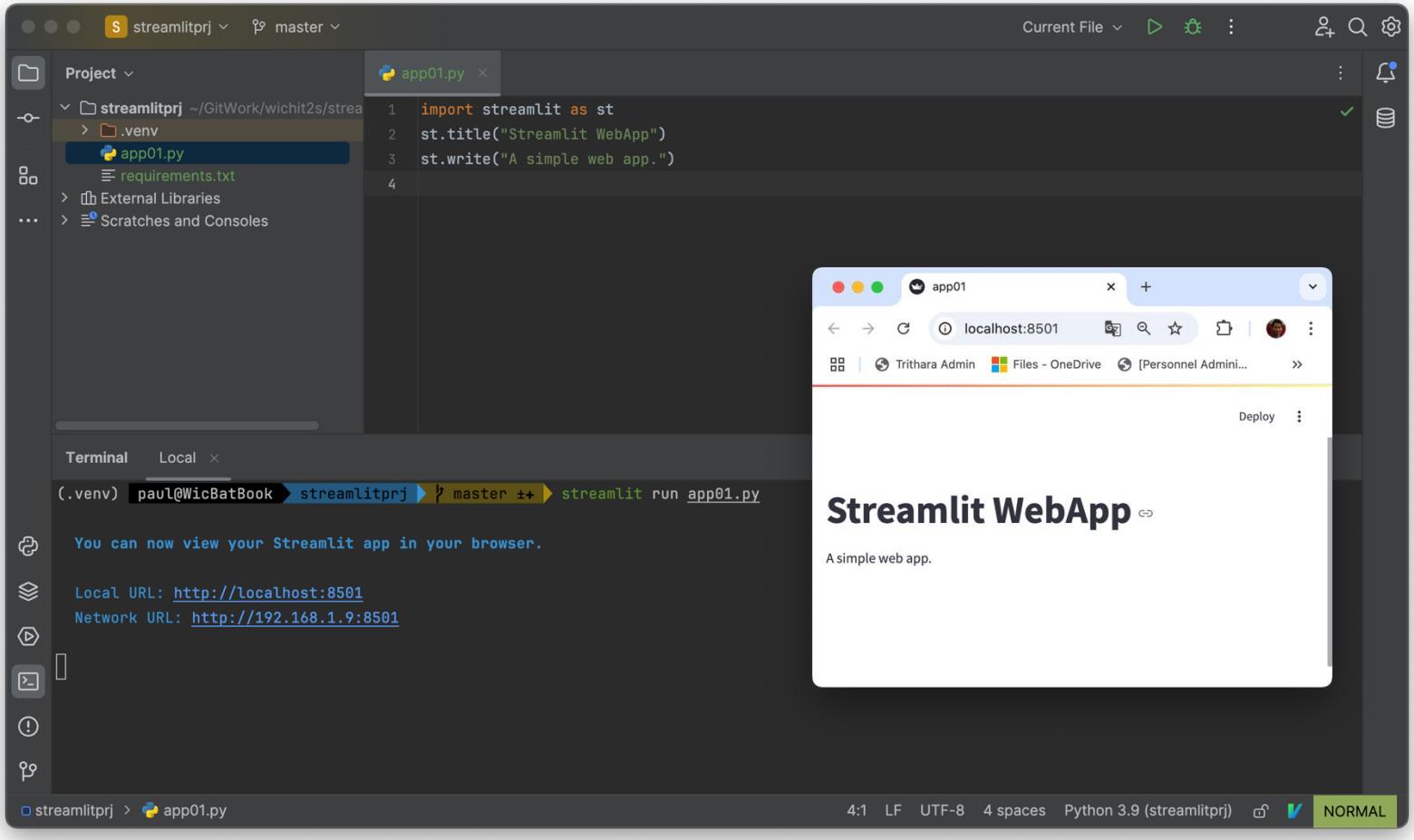
@wichit2s
https://wichit2s.github.io/courses/oop/

app01.py

```
import streamlit as st
st.title("Streamlit WebApp")
st.write(”A simple web app.")
```

```
streamlit run app01.py
```

Hands-on Activity



FUNCTION	DESCRIPTION	EXAMPLE
st.title()	Large title text	st.title("Welcome to My App!")
st.header()	Section header	st.header("Section Title")
st.subheader()	Subsection header	st.subheader("Subsection")
st.text()	Plain text	st.text("This is simple text.")
st.markdown()	Supports Markdown formatting	st.markdown("**Bold** and *Italic*")
st.write()	Displays text, objects, and more	st.write("Hello, Streamlit!")
st.code()	Displays code blocks	st.code("print('Hello, World!')", language="python")
st.latex()	Displays latex equation	st.latex(”\sqrt{e^{xy}}")

Text Elements

https://docs.streamlit.io/develop/api-reference/text

Widget	Function	Example Code
Button	Creates a clickable button	st.button("Click Me")
Text Input	Accepts user text input	st.text_input("Enter your name")
Chat Input	Accepts user chat input	st.chat_input(”Say something…”)
Text Area	Multi-line text input	st.text_area("Enter a description")
Number Input	Accepts numeric input	st.number_input(”Age", min_value=0, max_value=100)
Checkbox	A simple toggle option	st.checkbox("I agree")
Radio Buttons	Select one option from a list	st.radio("Choose an option", ["A", "B", "C"])
Select Box	Dropdown menu for selection	st.selectbox("Pick a color", ["Red", "Blue", "Green"])
Slider	Select a value from a range	st.slider("Choose a number", 0, 100, 50)
File Uploader	Uploads files	st.file_uploader("Upload a file")
Stream Writer	Render stream data in sequence	st.write_stream(stream_data_function)

Input Widgets

https://docs.streamlit.io/develop/api-reference/widgets

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

st.title("File Upload Example")

uploader = st.file_uploader("Choose a file", type=["csv"])

if uploader is not None:
  df = pd.read_csv(uploaded_file)
  st.write(df)
```

```python
import streamlit as st
from PIL import Image

st.title("Image Upload Example")

uploader = st.file_uploader("Choose a file", type=["jpg", "png"])

if uploader is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image", use_column_width=True)
```

File Upload Sample

https://docs.streamlit.io/develop/api-reference/widgets

Handling User Input

st.session_state

a **persistent state storage** in Streamlit that allows variables to retain values across user interactions, preventing resets when the app reruns.

- Store user inputs, counters, or app settings.
- Keep UI elements in sync across reruns.
- Maintain state between interactions (e.g., form submissions, button clicks).

Session counter

```
import streamlit as st

if "counter" not in st.session_state:
   st.session_state.counter = 0

st.write(f"Counter: {st.session_state.counter}")

if st.button("Increase"):
    st.session_state.counter += 1

if st.button("Reset"):
   st.session_state.counter = 0
```

Remember inputs

```
name = st.text_input("Name: ", key="user_name")
age = st.number_input("Age: ", min_value=0, max_value=100, key="user_age")

st.write(f"Hello, {st.session_state.user_name}!")

st.write(f"You are {st.session_state.user_age} years old.")
```

Data Visualization

Why Use Visualizations in Streamlit?

Streamlit supports various charting libraries to create interactive data visualizations effortlessly. These include **Matplotlib, Seaborn, Plotly, Altair, and Streamlit's built-in charts**.

Function	Description	Example
st.line_chart()	Line chart for time-series data	st.line_chart(df)
st.bar_chart()	Bar chart for categorical data	st.bar_chart(df)
st.area_chart()	Area chart for trends	st.area_chart(df)
st.pyplot()	Supports Matplotlib figures	st.pyplot(fig)
st.plotly_chart()	Displays interactive Plotly charts	st.plotly_chart(fig)
st.altair_chart()	Uses Altair for declarative charting	st.altair_chart(chart)
st.map()	Scatter plots on map	st.map(df)

https://docs.streamlit.io/develop/api-reference/charts

Line Chart

```
import streamlit as st
import pandas as pd
import numpy as np

st.title("📊 Interactive Data Visualization")
data = np.random.randn(20, 3)
columns = columns=["A", "B", "C"]
df = pd.DataFrame(data, columns)

st.line_chart(df)
```

Plotly Chart

```
import plotly.express as px

df = px.data.iris()
x = "sepal_width”
y = "sepal_length"
fig = px.scatter(df, x=x, y=y, color="species")

st.plotly_chart(fig, use_container_width=True)
```

Map

```
import streamlit as st
import pandas as

pd data = pd.DataFrame({
  'lat': [13.7563, 14.5995, 35.6895],
  'lon': [100.5018, 120.9842, 139.6917]
})

st.title("📍 Simple Map Example")
st.map(data)
```

Folium Map

```
import streamlit as st
import folium
from streamlit_folium import st_folium

m = folium.Map(location=[13.7563, 100.5018], zoom_start=5)

th = folium.Marker([13.7563, 100.5018], tooltip="Bangkok", popup="Thailand")
th.add_to(m)
ph = folium.Marker([14.5995, 120.9842], tooltip="Manila", popup="Philippines")
ph.add_to(m)

st.title("🗺️ Interactive Map with Folium")
st_folium(m, width=700, height=500)
```

Sidebar

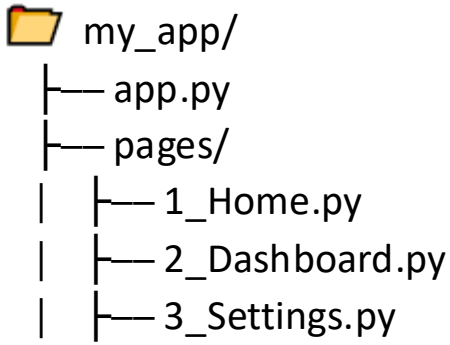
The sidebar (st.sidebar) helps organize UI elements, allowing users to navigate and interact with your app efficiently.

- Keeps the main interface clean
- Useful for filters, settings, and navigation
- Works with all Streamlit widgets

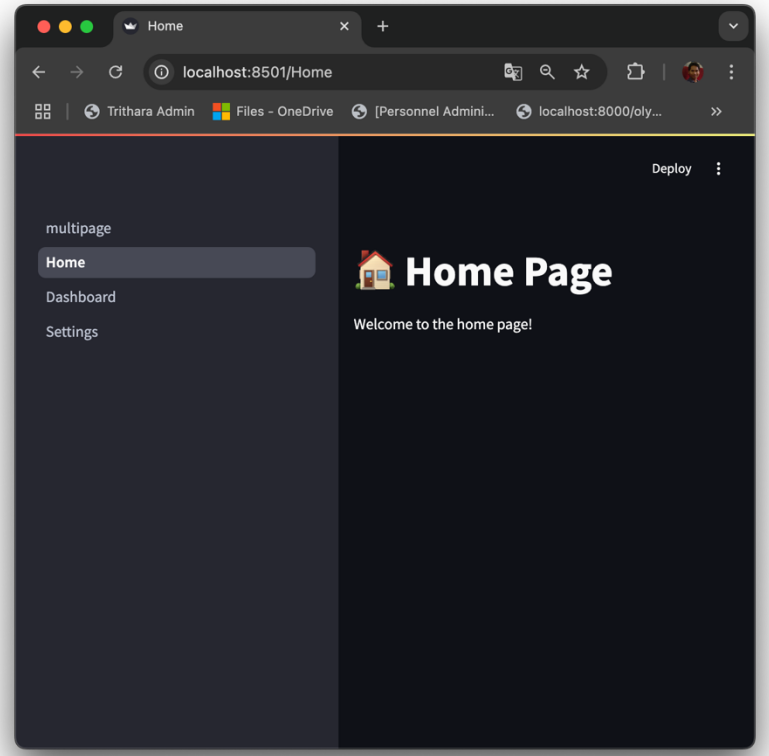
```
st.sidebar.title("Navigation")
pages = ["Home", "Data", "About"]
page = st.sidebar.radio("Go to", pages)

if page == "Home":
  st.write("Welcome to Home Page! 🎉 ")
elif page == "Data":
  st.write("📊 View Data Here")
elif page == "About":
  st.write("ℹ️ About This App")
```

Multipages



```
my_app/
├── app.py
├── pages/
│   ├── 1_Home.py
│   ├── 2_Dashboard.py
│   └── 3_Settings.py
```



Layouts

Layout Feature	Function	Description
Main Layout	st.write(), st.title(), etc.	Default area for content rendering.
Sidebar	st.sidebar.*	Moves elements to a collapsible sidebar.
Columns	st.columns(n)	Creates n equal-width columns for content layout.
Expander	st.expander("Label")	Collapsible section for hiding/showing content.
Tabs	st.tabs(["Tab 1", "Tab 2"])	Creates a tabbed layout for organizing content.
Container	st.container()	A flexible container for grouping elements dynamically.
Empty Space	st.empty()	Placeholder for dynamic content updates.
Divider	st.divider()	Adds a horizontal line to separate sections.
Dialog	@st.dialog()	Define function as an independent dialog

https://docs.streamlit.io/develop/api-reference/layout

st.cache_data

@st.cache_data decorator **stores function outputs** so they don't need to be recalculated every time.

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

@st.cache_data
def load_data():
  df = pd.read_csv("large.csv")
  return df

st.title("⚡ Cached Data Loading")
df = load_data()
st.dataframe(df)
```

```python
import streamlit as st
import requests

@st.cache_data
def fetch_data():
  url = "https://api.publicapis.org/entries"
  response = requests.get(url)
  return response.json()

st.title("🌎 Cached API Example")
data = fetch_data()
st.write(data)
```

https://docs.streamlit.io/develop/api-reference/caching-and-state

OOP Chat

Goal: AI chatbot with ollama

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression
scoop install ollama
```

```
wget https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh
sh ./install.sh
brew install ollama
```

```
curl -fsSL https://ollama.com/install.sh | sh
```

requirements.txt

```
streamlit
ollama
```

Chat Messages

```python
import streamlit as st

with st.chat_message('user'):
  st.write("Hello from user!")

with st.chat_message('assistant'):
  st.write("Hello from assistant!")

with st.chat_message('robot'):
  st.write("Hello from robot!")

with st.chat_message('human'):
  st.write("Hello from human!")
```

```python
import streamlit as st

messages = [
  {'role': 'user', 'content': 'Hello from user!'},
  {'role': 'assistant', 'content': 'Hello from assistant!'},
  {'role': 'robot', 'content': 'Hello from robot!'},
  {'role': 'human', 'content': 'Hello from human!'},
]

for message in messages:
  with st.chat_message(message['role']):
    st.write(message['content'])
```

https://docs.streamlit.io/develop/api-reference/chat/st.chat_message

Chat Input

```
import streamlit as st

messages = [
 {'role': 'user', 'content': 'Hello from user!'},
 {'role': 'assistant', 'content': 'Hello from assistant!'},
 {'role': 'robot', 'content': 'Hello from robot!'},
 {'role': 'human', 'content': 'Hello from human!'},
]

for message in messages:
  with st.chat_message(message['role']):
    st.write(message['content'])

user_input = st.chat_input('message')
if user_input is not None:
  with st.chat_message('user'):
    st.write(user_input)
```

https://docs.streamlit.io/develop/api-reference/chat/st.chat_input

Session State

```python
import streamlit as st

if 'messages' not in st.session_state:
  st.session_state['messages'] = []

for message in st.session_state.messages:
  with st.chat_message(message['role']):
    st.write(message['content’])

user_input = st.chat_input('message', key='user’)

if user_input is not None:
  st.session_state['messages'].append({ 'role': 'user', 'content': user_input })
  with st.chat_message('user’):
    st.write(user_input)
```

https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state

Session State – add echo

```
import streamlit as st

if 'messages' not in st.session_state:
  st.session_state['messages'] = []

for message in st.session_state.messages:
  with st.chat_message(message['role']):
    st.write(message['content’])

user_input = st.chat_input('message', key='user’)

if user_input is not None:
  st.session_state['messages'].append({ 'role': 'user', 'content': user_input })
  with st.chat_message('user’):
    st.write(user_input)
  st.session_state['messages'].append({ 'role': ‘robot’, ??? })
  with st.chat_message(‘robot’, avatar=‘old_robot.jpg’):
    st.write(f’you said {user_input} ’)
```

https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state

Response from Ollama

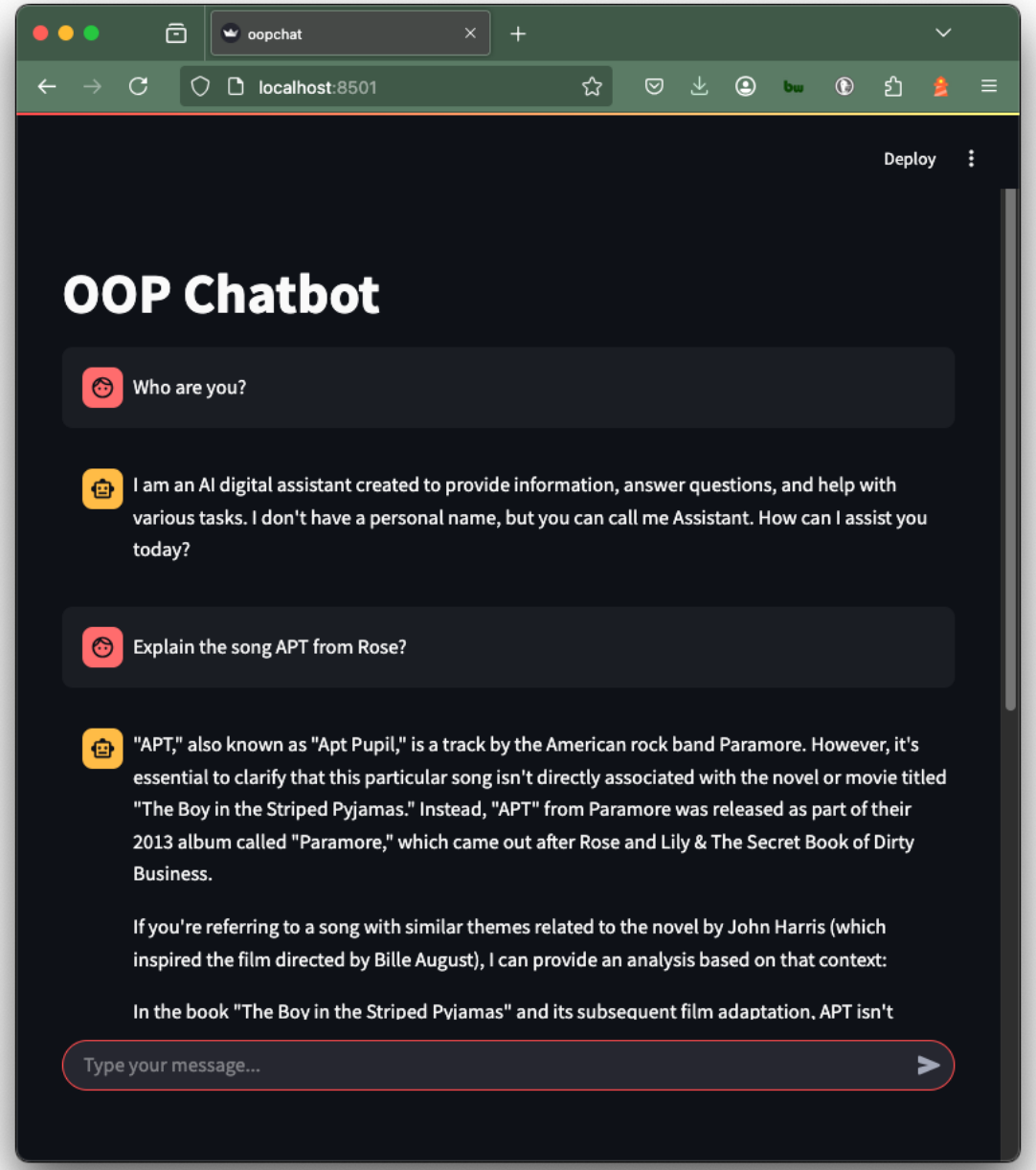
https://github.com/ollama/ollama

```
with st.chat_message("assistant"):
        response_container = st.empty()

  full_response = ""
  for chunk in ollama.chat(model="phi3", messages=[user_data], stream=True):
    full_response += chunk["message"]["content"]
    response_container.markdown(full_response + "▌")

  response_container.markdown(full_response)
st.session_state.messages.append({"role": "assistant", "content": full_response})
```

https://github.com/ollama/ollama-python


oopchat
localhost:8501
Deploy
OOP Chatbot
Who are you?
I am an AI digital assistant created to provide information, answer questions, and help with various tasks. I don't have a personal name, but you can call me Assistant. How can I assist you today?
Explain the song APT from Rose?
"APT," also known as "Apt Pupil," is a track by the American rock band Paramore. However, it's essential to clarify that this particular song isn't directly associated with the novel or movie titled "The Boy in the Striped Pyjamas." Instead, "APT" from Paramore was released as part of their 2013 album called "Paramore," which came out after Rose and Lily & The Secret Book of Dirty Business.
If you're referring to a song with similar themes related to the novel by John Harris (which inspired the film directed by Bille August), I can provide an analysis based on that context:
In the book "The Boy in the Striped Pyjamas" and its subsequent film adaptation, APT isn't
Type your message...