Prepared by Gone Phishing

Streamlit/Docker

How to Host ML on Streamlit Hosted on Docker

1 April 2025 | DS460 Section 12:45

Project Objectives

Streamlit

- Set up Streamlit
- Deploy Streamlit
- Have basic template for future use

ML Development

- Create ML model
- Host on Streamlit

Docker

- Understand how to set up a Docker file in your Streamlit App
- Understand the basics of what the background is doing

Why You Should Listen

Super Fast Prototyping

Perfect for quick MVPs, data demos, or visualizing models without needing to learn front-end dev

Easy to Deploy

Takes second to deploy fully functionig app

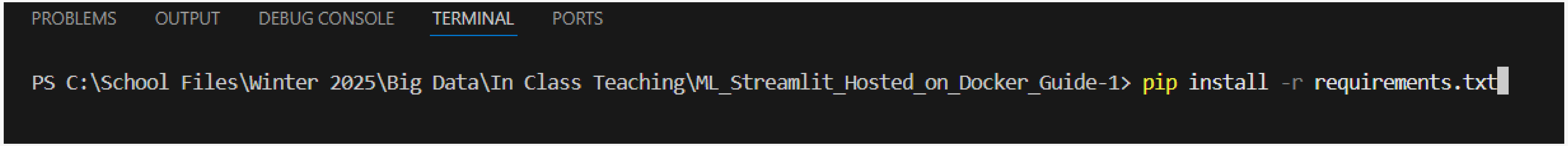
We Will Be Doing This For Class

We are using this for our final project and is used often in industry

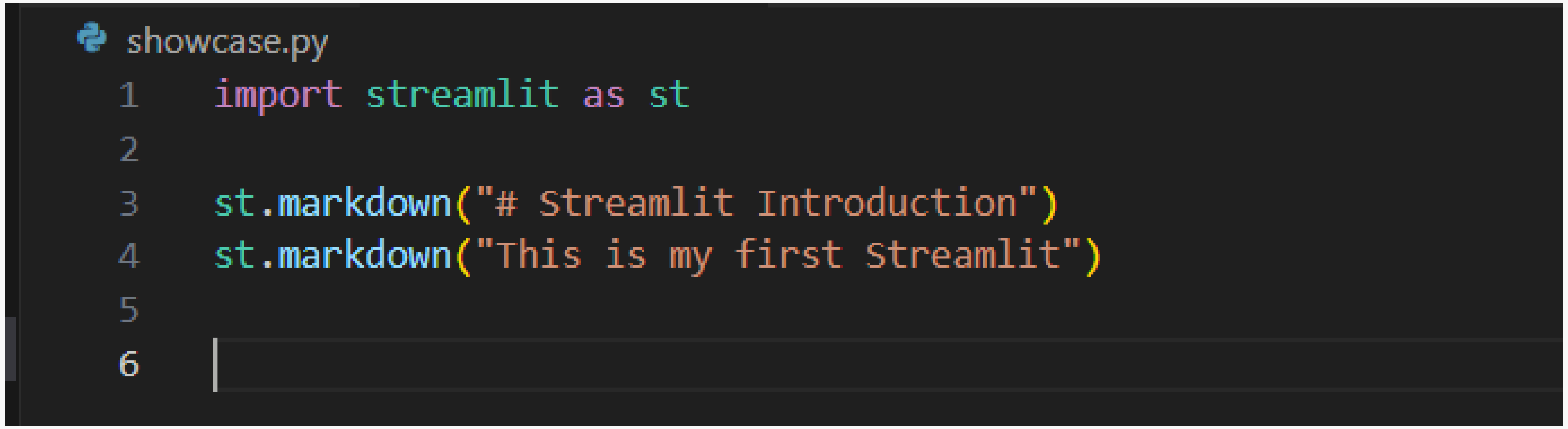
Importance of Docker

It streamlines application development, deployment, and management, enabling consistent environments, efficient resource usage, and faster software delivery

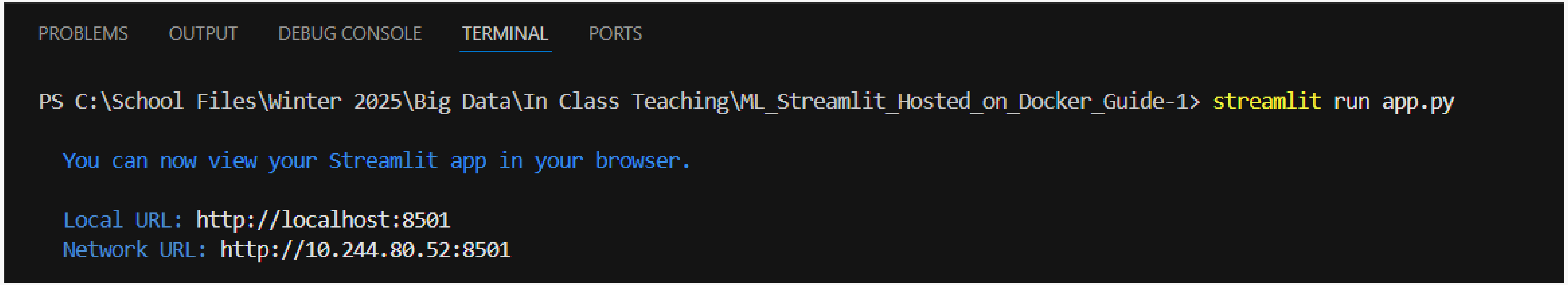
Streamlit Set Up



```
PS C:\School Files\Winter 2025\Big Data\In Class Teaching\ML_Streamlit_Hosted_on_Docker_Guide-1> pip install -r requirements.txt
```

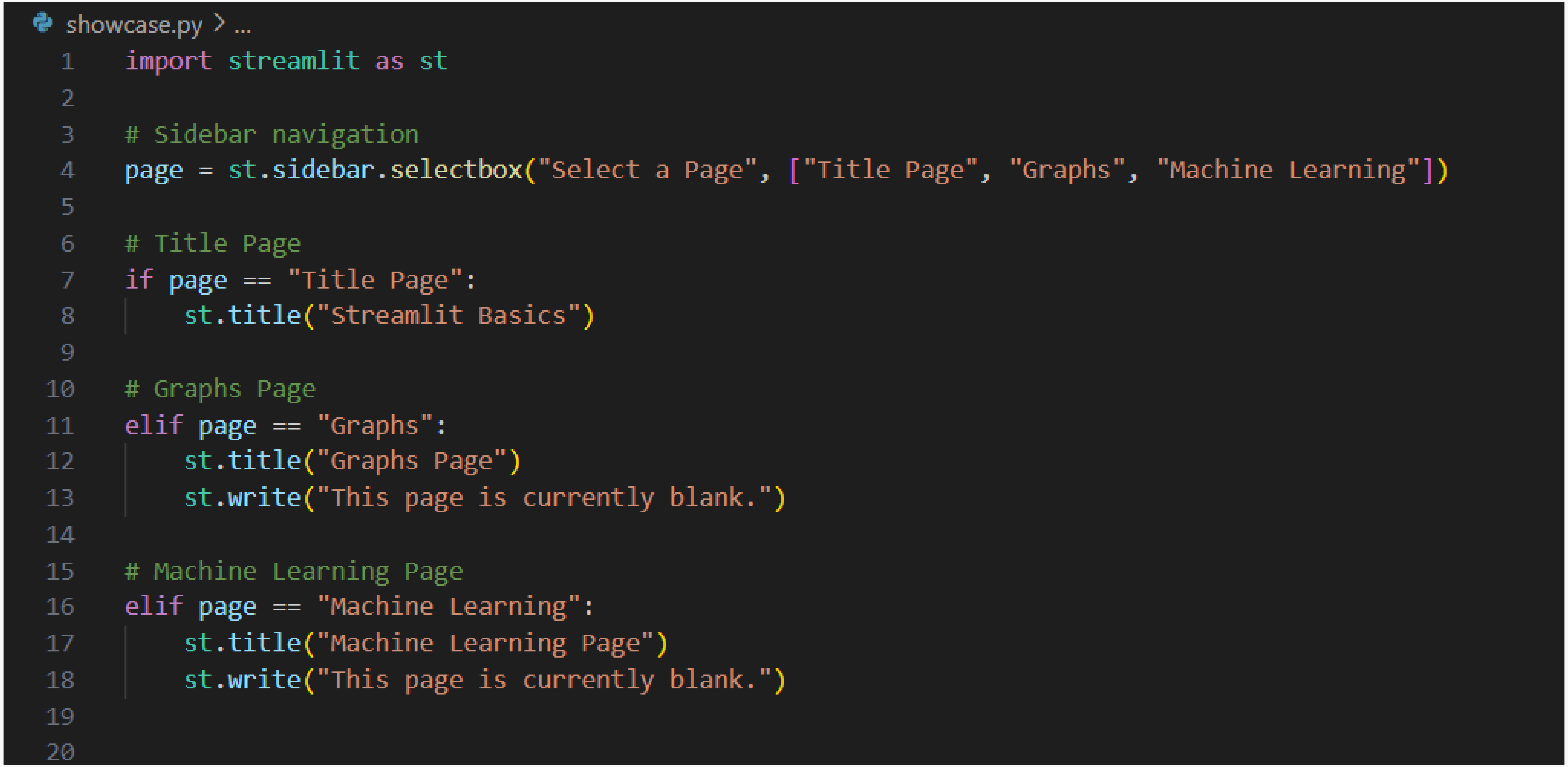


```python
import streamlit as st

st.markdown("# Streamlit Introduction")
st.markdown("This is my first Streamlit")

```



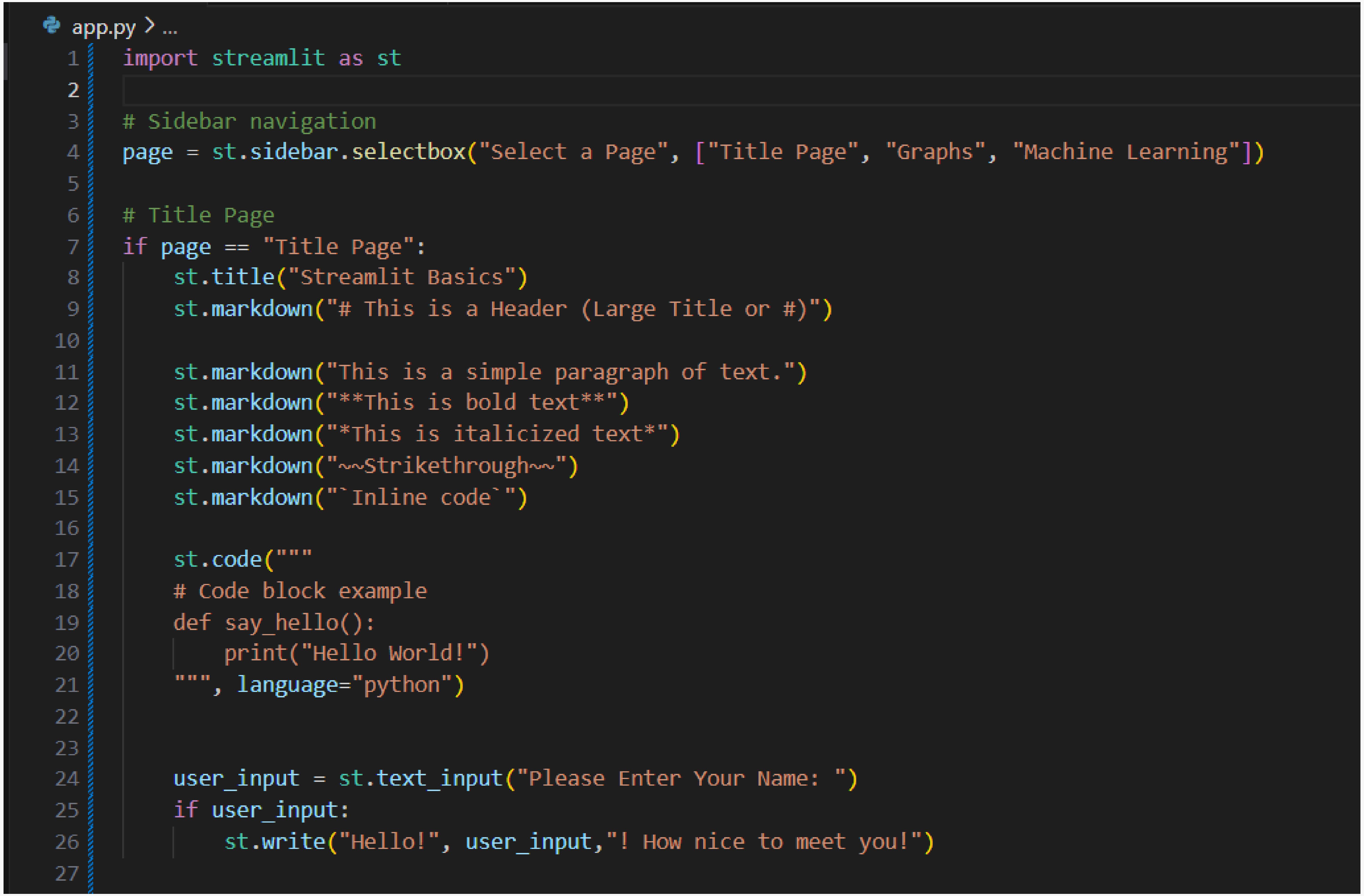
```
PS C:\School Files\Winter 2025\Big Data\In Class Teaching\ML_Streamlit_Hosted_on_Docker_Guide-1> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.244.80.52:8501
```

Streamlit Pages



```python
import streamlit as st

# Sidebar navigation
page = st.sidebar.selectbox("Select a Page", ["Title Page", "Graphs", "Machine Learning"])

# Title Page
if page == "Title Page":
    st.title("Streamlit Basics")

# Graphs Page
elif page == "Graphs":
    st.title("Graphs Page")
    st.write("This page is currently blank.")

# Machine Learning Page
elif page == "Machine Learning":
    st.title("Machine Learning Page")
    st.write("This page is currently blank.")


```

Text Features



```python
import streamlit as st

# Sidebar navigation
page = st.sidebar.selectbox("Select a Page", ["Title Page", "Graphs", "Machine Learning"])

# Title Page
if page == "Title Page":
    st.title("Streamlit Basics")
    st.markdown("# This is a Header (Large Title or #)")

    st.markdown("This is a simple paragraph of text.")
    st.markdown("**This is bold text**")
    st.markdown("*This is italicized text*")
    st.markdown("~~Strikethrough~~")
    st.markdown("`Inline code`")

    st.code("""
    # Code block example
    def say_hello():
        print("Hello World!")
    """, language="python")


    user_input = st.text_input("Please Enter Your Name: ")
    if user_input:
        st.write("Hello!", user_input,"! How nice to meet you!")

```

Graph Setup

For Streamlit:

- Using pandas to read a .parquet file
- Caching data with @st.cache_data
- Displaying the first few rows using st.dataframe(df.head())

```
@st.cache_data
  def load_data():
    file_path = os.path.join(DEPEND_DIR, "idaho_target_2023.parquet")
    return pd.read_parquet(file_path)
  df = load_data()
```

Data Preprocessing

Filtering and Grouping Data:

- Converting DAY_OF_PREDICTION to a datetime format
- Filtering the dataset for a specific date range
- Grouping the data by date and summing TOTAL_QUANTIT

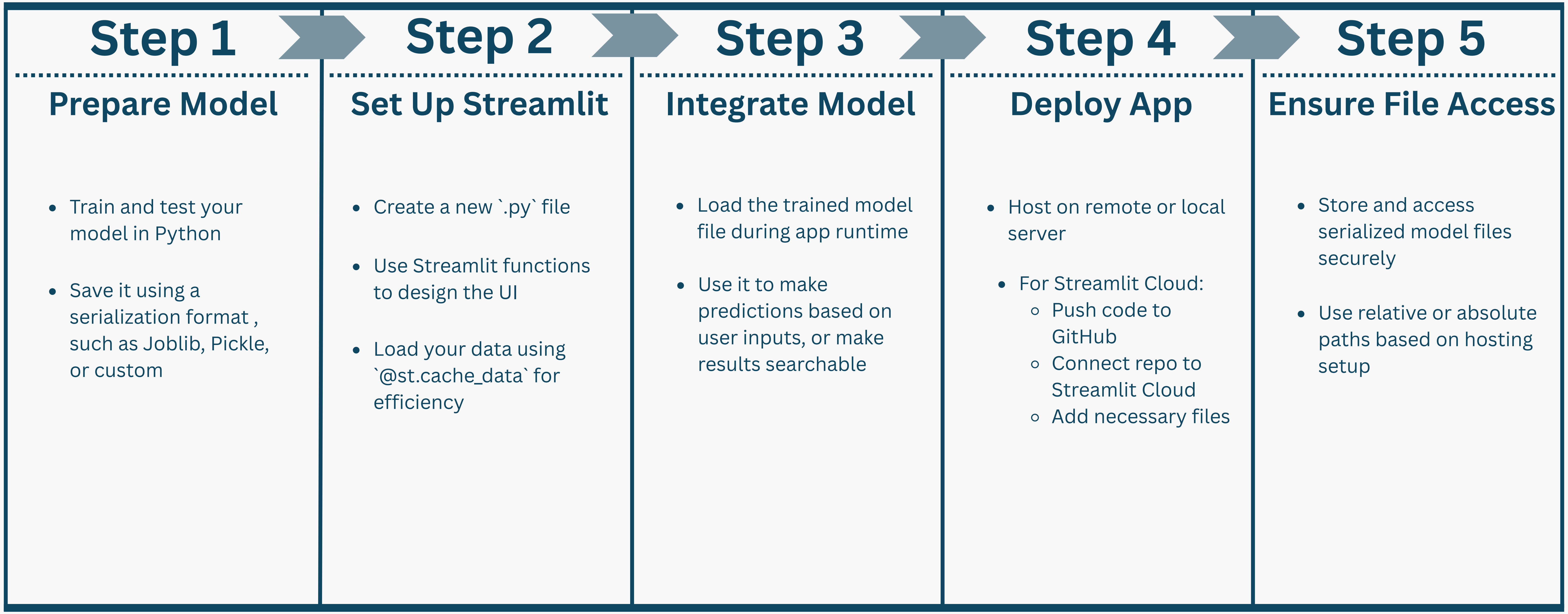
```
df['DAY_OF_PREDICTION'] =
pd.to_datetime(df['DAY_OF_PREDICTION'])

df_filtered = df[(df['DAY_OF_PREDICTION'] >= "2023-01-01")
& (df['DAY_OF_PREDICTION'] <= "2023-02-28")]

df_grouped = df_filtered.groupby('DAY_OF_PREDICTION',
as_index=False)['TOTAL_QUANTITY'].sum()
```

Building Interactive Charts with Plotly

- Using Plotly Express for line and bar charts
- Adding a st.selectbox() to let users choose between different chart types
- Displaying the chart with st.plotly_chart()

```
chart_type = st.selectbox("Select Chart Type", ["Line Chart",
"Bar Chart"])

if chart_type == "Line Chart":
    fig = px.line(df_grouped, x='DAY_OF_PREDICTION',
y='TOTAL_QUANTITY', title="TOTAL_QUANTITY Over Time")
    st.plotly_chart(fig)
elif chart_type == "Bar Chart":
    fig = px.bar(df_grouped, x='DAY_OF_PREDICTION',
y='TOTAL_QUANTITY', title="TOTAL_QUANTITY Distribution")
    st.plotly_chart(fig)
```

Hosting ML Models on Streamlit



Step 1

Prepare Model

- Train and test your model in Python
- Save it using a serialization format , such as Joblib, Pickle, or custom

Step 2

Set Up Streamlit

- Create a new `.py` file
- Use Streamlit functions to design the UI
- Load your data using `@st.cache_data` for efficiency

Step 3

Integrate Model

- Load the trained model file during app runtime
- Use it to make predictions based on user inputs, or make results searchable

Step 4

Deploy App

- Host on remote or local server
- For Streamlit Cloud:
  - Push code to GitHub
  - Connect repo to Streamlit Cloud
  - Add necessary files

Step 5

Ensure File Access

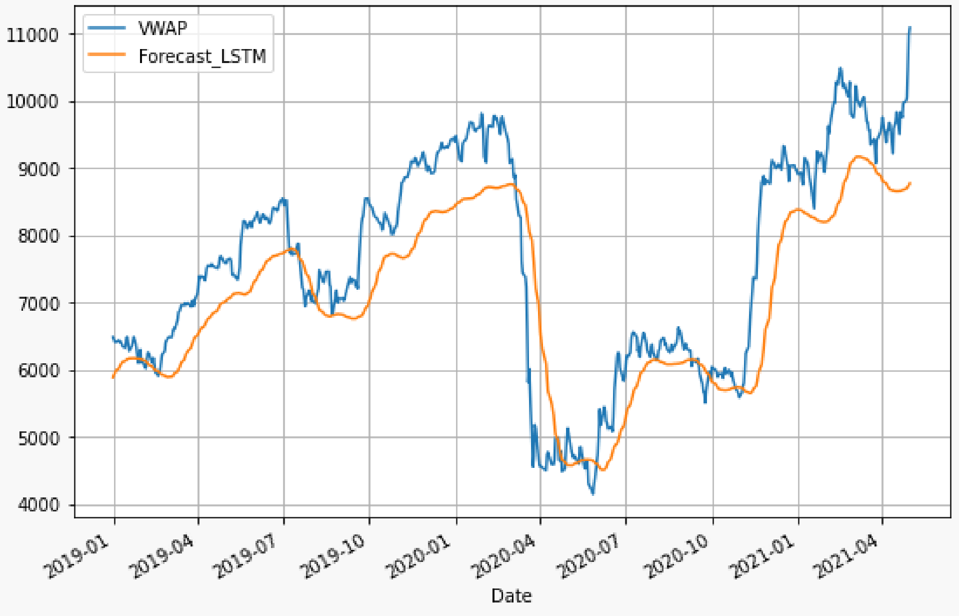
- Store and access serialized model files securely
- Use relative or absolute paths based on hosting setup

Demonstration Model: Meta Prophet

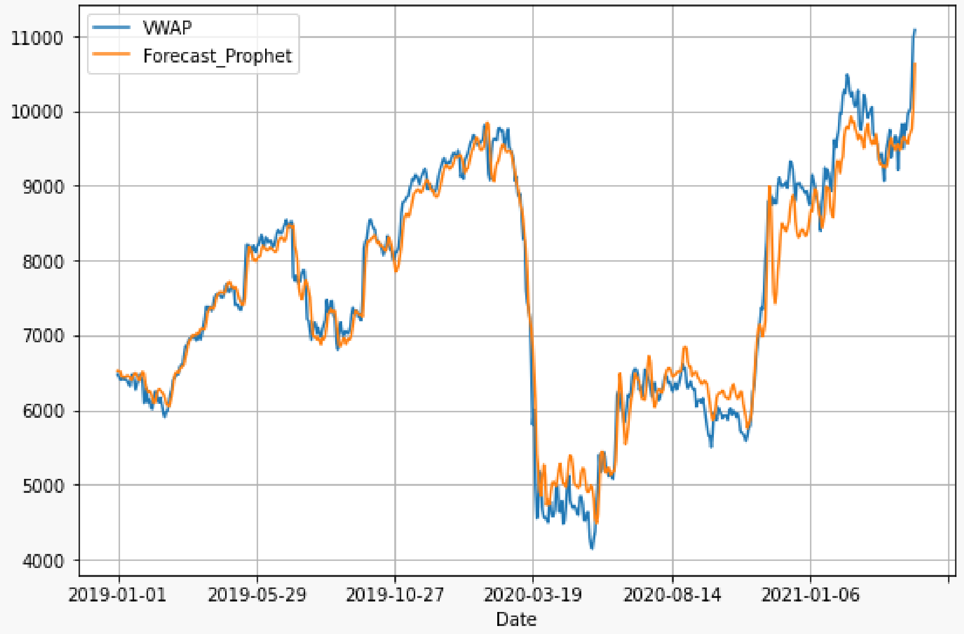
Prophet Overview:

- Built for time series forecasting
- Additive model: handles seasonality, holidays, trend shifts, etc.
- Ideal for business data with irregularities or gaps
- Automatically selects changepoints and handles missing data
- Performs better than LSTM (Long Short-Term Memory) models, which are specifically designed to process sequential data, like time series

LSTM Model Prediction



Prophet Model Prediction



(**VWAP**, or **volume-weighted average price**, is a technical analysis tool that shows the ratio of an asset's price to its total trade volume. Used in stock and other trading markets.)

Source: https://neptune.ai/blog/arima-vs-prophet-vs-lstm

Same Data, New Purpose

Context

- This model will be trained on the same data shown by the graphs you made previously
- Focus will now be on weekly GTIN-level sales forecasting

Training Strategy

- Models trained per GTIN and globally (all GTINs together)
- Hyperparameter tuning over:
  - `changepoint_prior_scale`
  - `seasonality_prior_scale`
- Parallelized with `ThreadPoolExecutor`

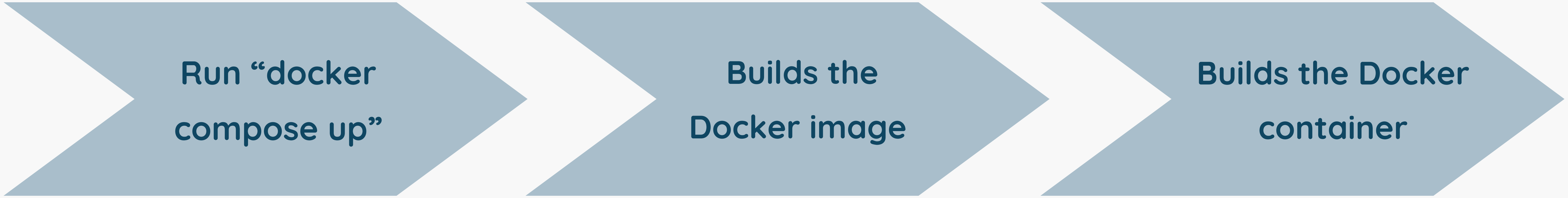
Prediction Goal

- Forecast total weekly sales for each GTIN based on historical data

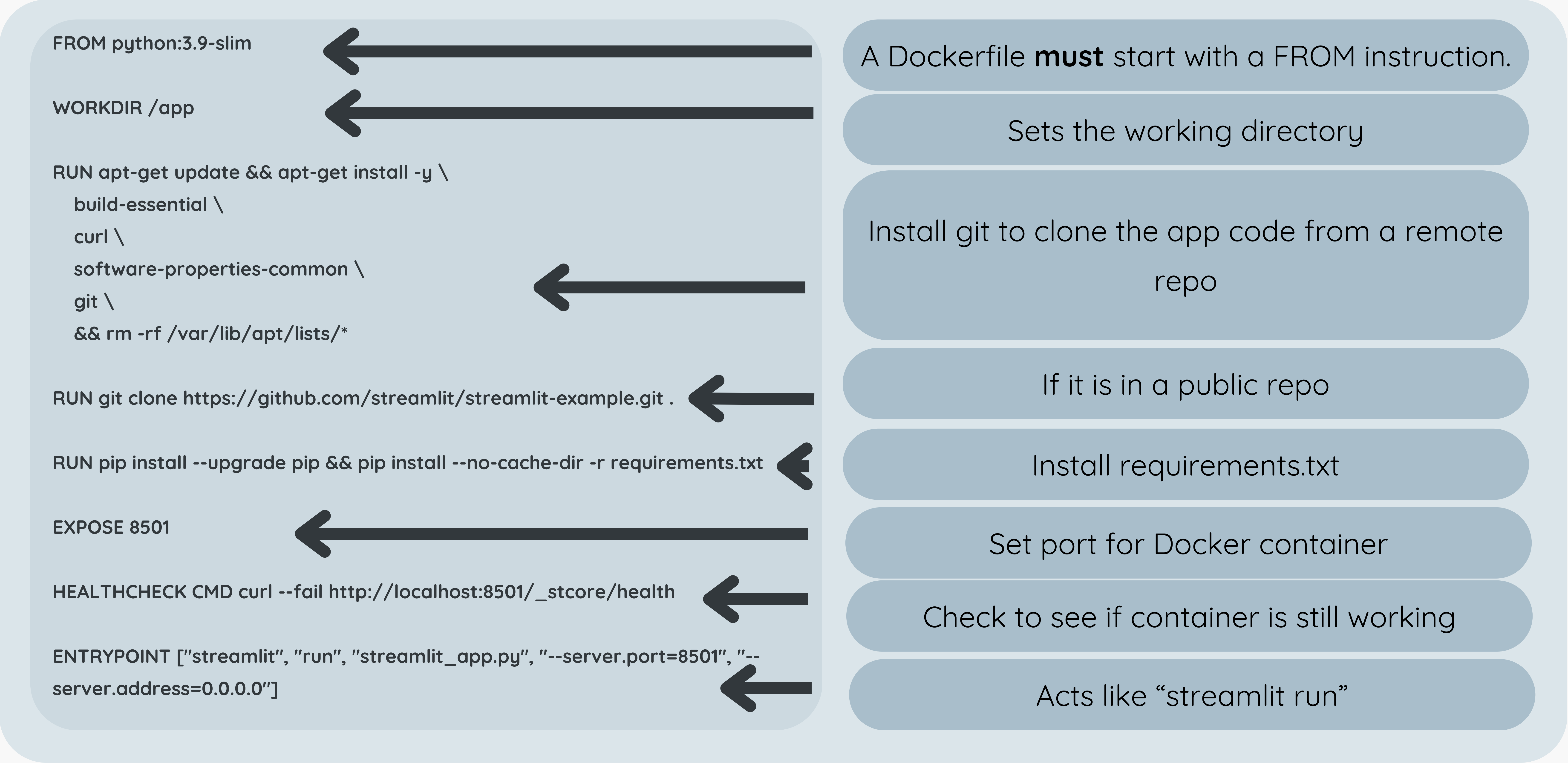
Docker Set Up

Step 1	Step 2	Step 3
Create a Dockerfile	Create docker-compose.yaml	run: docker compose up

Flow Architecture in the Background



DockerFile Example



```
FROM python:3.9-slim

WORKDIR /app

RUN apt-get update && apt-get install -y \
    build-essential \
    curl \
    software-properties-common \
    git \
    && rm -rf /var/lib/apt/lists/*

RUN git clone https://github.com/streamlit/streamlit-example.git .

RUN pip install --upgrade pip && pip install --no-cache-dir -r requirements.txt

EXPOSE 8501

HEALTHCHECK CMD curl --fail http://localhost:8501/_stcore/health

ENTRYPOINT ["streamlit", "run", "streamlit_app.py", "--server.port=8501", "--server.address=0.0.0.0"]
```

A Dockerfile **must** start with a FROM instruction.

Sets the working directory

Install git to clone the app code from a remote repo

If it is in a public repo

Install requirements.txt

Set port for Docker container

Check to see if container is still working

Acts like “streamlit run”

References

For Streamlit:

- Gather sales data, market research, and consumer feedback through surveys and analysis tools.
- Utilize both primary and secondary research methods to gather comprehensive insights.

For Docker:

- Hosting Streamlit on Docker
- Docker Tutorial

GitHub Repository for future references:

- Advanced Streamlit Hosted on Docker Example

Thank you