STREAMLIT- SUBJECTIVE TEST

Q 1) What is Streamlit and what are its main features?

Streamlit: A Python library for creating interactive web apps for data science and machine learning projects.

Main Features:

1. Simple syntax similar to writing Python scripts.
2. Auto-updating app as you code.
3. Built-in support for widgets (sliders, buttons, text inputs).
4. Easy data visualization with popular libraries (Matplotlib, Plotly).
5. Integration with machine learning models.

Q2) How does Streamlit differ from other web application frameworks like Flask or Django?

Streamlit	Flask/Django
Specifically designed for data science and machine learning apps.	General-purpose web frameworks.
Easy to use with minimal code.	Requires more reusable code.
Automatically updates the app in real-time.	More control over the structure and components of the app.

Q3) What are some typical use cases for Streamlit?

Typical Use Cases:

1. Data visualization dashboards.
2. Prototyping machine learning models.
3. Interactive data exploration tools.
4. Sharing data insights with non-technical stakeholders.

Q4) How do you create a simple Streamlit app?

Steps to Create a Simple Streamlit App:

1. Install Streamlit: pip install streamlit.
2. Create a Python script (e.g., app.py).
3. Write code using Streamlit commands (e.g., st.write, st.slider).
4. Run the app: streamlit run app.py.

Q5) Can you explain the basic structure of a Streamlit script?

Basic Structure:

1. import streamlit as st
2. st.title('My Streamlit App')
3. st.write('Hello, Streamlit!')
4. slider_value = st.slider('Select a value', 0, 100)
5. st.write('Slider value:', slider_value)

Q6) How do you add widgets like sliders, buttons, and text inputs to a Streamlit app?
Adding Widgets:

1. Slider: st.slider('Label', min_value, max_value)
2. Button: st.button('Button Label')
3. Text Input: st.text_input('Label')

Q7) How does Streamlit handle user interaction and state management?
User Interaction and State Management:

1. Widgets automatically capture user input.
2. Streamlit reruns the script from top to bottom with each interaction.
3. Use st.session_state to manage persistent state between interactions.

Q8) What are some best practices for organizing and structuring a Streamlit project?
Best Practices:

1. Organize code into functions.
2. Use a clear and consistent naming convention.
3. Separate data loading, processing, and visualization code.
4. Keep the app script clean and modular.

Q9) How would you deploy a Streamlit app locally?
Deploying Locally:

1. Ensure Streamlit is installed.
2. Navigate to the directory containing the app script.
3. Run: streamlit run app.py.

Q10) Can you describe the steps to deploy a Streamlit app?
Steps to Deploy a Streamlit App:

1. Develop the app locally and test it.
2. Create a requirements.txt file listing all dependencies.

3. Choose a hosting service (e.g., Streamlit Sharing, Heroku).
4. Follow the hosting service's deployment instructions (usually involves pushing code to a repository and connecting it to the hosting service).

Q11) What is the purpose of the requirements.txt file in the context of Streamlit deployment?

1. Purpose of requirements.txt:
2. Lists all the Python libraries and versions needed to run the app.
3. Ensures that the deployment environment has all necessary dependencies installed.
4. Facilitates easy replication of the app environment on different machines.