1. Short Answer Questions

Q1

Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

<u>Differences</u>

- **Eager Execution-** PyTorch uses dynamic computation graphs (eager execution by default), making debugging easier. TensorFlow originally used static graphs (via tf.Session), but TensorFlow 2.x now supports eager execution.
- **API Design-** PyTorch is more Pythonic and intuitive, while TensorFlow has a steeper learning curve but offers more deployment tools (e.g., TensorFlow Lite, TensorFlow Serving).
- **Deployment-** TensorFlow is better for production due to robust deployment options, whereas PyTorch is favored in research for its flexibility.
- **Community & Ecosystem-** TensorFlow has strong industry adoption, while PyTorch is dominant in academia.

When to Choose

- **TensorFlow-** For production deployments, mobile/edge devices, or when using TensorFlow Extended (TFX).
- **PyTorch-** For rapid prototyping, research, or dynamic neural networks (e.g., transformers).

Q2

Describe two use cases for Jupyter Notebooks in AI development.

1. **Exploratory Data Analysis (EDA)**
    - ✓ Jupyter Notebooks allow interactive visualization (e.g., Matplotlib, Seaborn) and real-time data manipulation (e.g., Pandas), making them ideal for understanding datasets before model training.
2. **Model Prototyping & Experimentation**

- ✓ Data scientists can iteratively train and evaluate models (e.g., Scikit-learn, TensorFlow) while documenting code, results, and visualizations in a single shareable notebook.

Q3

How does spaCy enhance NLP tasks compared to basic Python string operations?

- **Pre-trained Models-** spaCy provides optimized models for tasks like named entity recognition (NER), part-of-speech (POS) tagging, and dependency parsing, which would require complex regex or manual rules in basic string operations.
- **Efficiency-** Built-in tokenization and linguistic features (e.g., lemmatization) are faster and more accurate than manual string splitting/stemming.
- **Pipeline Approach-** spaCy's modular pipeline (e.g., nlp(text)) streamlines preprocessing, whereas string operations require custom, error-prone code for each task.

2. Comparative Analysis

Scikit-learn vs. TensorFlow

Aspect	Scikit-learn	TensorFlow
Target Applications	Classical ML (e.g., regression, SVM)	Deep Learning (e.g., CNNs, RNNs)
Ease of Use	Beginner-friendly (`fit()` / `predict()`)	Steeper (tensors, GPU, layers)
Community Support	Strong for traditional ML	Larger DL ecosystem (Google-backed)

When to Use

- **Scikit-learn:** Small/structured datasets, interpretability (e.g., feature importance), or non-neural models.
- **TensorFlow:** Large-scale data (e.g., images, text), neural networks, or production deployment.