# Part 1: Theoretical Understanding

## 1) Short answer questions

Q1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

**Answer:**

**The main difference lies in their computation graph approach:**

- ✓ **TensorFlow** traditionally uses static computation graphs, which are defined before running the model. This can make debugging harder but is more optimized for production.
- ✓ **PyTorch** uses dynamic computation graphs (define-by-run), which are built as the code runs, making it more intuitive and flexible for research and experimentation.

**When to use:**

- ➢ **Use TensorFlow** for deploying models in production (e.g., with TensorFlow Serving, TFLite).
- ➢ **Use PyTorch** for research, rapid prototyping, and academic environments due to its easier debugging and native Pythonic design.

Q2: Describe two use cases for Jupyter Notebooks in AI development.
Answer:

- ✓ **Prototyping Machine Learning Models:** Developers can write and test code in cells, visualize outputs, and iterate quickly without restarting full scripts.
- ✓ **Data Exploration and Visualization:** Jupyter Notebooks support rich visualizations (using tools like Matplotlib or Seaborn), making it easier to analyze datasets before building models.

Q3: How does spaCy enhance NLP tasks compared to basic Python string operations?

**Answer:**

**spaCy provides advanced NLP capabilities that go beyond basic string handling, such as:**

- ✓ **Tokenization**: Understands the grammatical structure, not just splitting on spaces.
- ✓ **Named Entity Recognition (NER):** Automatically identifies entities like names, places, and dates.
- ✓ **Part-of-speech tagging and dependency parsing**: Helps analyze the grammatical structure, which isn't possible with basic string methods.

## 2) Comparative Analysis : Scikit-learn vs TensorFlow

| Aspect | Scikit-learn | TensorFlow |
|---|---|---|
| Target Applications | Classical machine learning (e.g., SVM, decision trees, linear regression). | Deep learning (e.g., neural networks, CNNs, RNNs). |
| Ease of Use | Very beginner-friendly with simple, consistent APIs. | Steeper learning curve, especially for beginners. |
| Community Support | Strong community for traditional ML; excellent documentation. | Large global community; backed by Google; rich ecosystem (Keras, TFLite, etc.). |