Q1: TensorFlow vs. PyTorch

- **TensorFlow** (by Google) is known for deployment-ready capabilities, scalable production pipelines, and extensive tools like TensorBoard and TFX.
- **PyTorch** (by Meta) is loved for its **dynamic computation graph**, which feels more "Pythonic" and intuitive—especially during experimentation or research.

When to choose:

- Use **PyTorch** for **prototyping, academic research**, and when you want fast debugging.
- Use **TensorFlow** for **production-scale apps**, mobile/embedded deployment (via TensorFlow Lite), and model serving in enterprise environments.

Q2: Use Cases for Jupyter Notebooks in AI

1. **Data Exploration & Visualization**: Jupyter lets you interactively explore data, run code snippets, and visualize outputs inline using libraries like Matplotlib or Seaborn.
2. **Model Prototyping**: You can iteratively build, train, and tweak models while documenting the process with Markdown, which makes it ideal for showcasing workflows or teaching.

Q3: spaCy vs. Basic Python String Ops

- While Python's split(), find(), and regex can manipulate text, **spaCy offers pre-trained NLP pipelines** with tokenization, part-of-speech tagging, named entity recognition (NER), dependency parsing, and more.
- It's **faster and more accurate** for real-world language understanding, especially across large corpora. For example, identifying “Apple” as a company vs. fruit—Python string ops alone can’t do that without contextual models like spaCy’s.

2. Comparative Analysis: Scikit-learn vs. TensorFlow

Aspect	**Scikit-learn**	**TensorFlow**
Target Applications	Classical ML (e.g. SVM, Decision Trees, k-NN)	Deep Learning (CNNs, RNNs, GANs, Transformers)
Ease of Use	Very beginner-friendly; intuitive APIs	Steeper learning curve, especially for custom models
Community Support	Strong in academia & traditional ML communities	Massive global community, production-focused tools