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

Feature	TensorFlow	PyTorch
Developed by	Google	Meta (Facebook)
Computation Graph	Static graph (optimized for production)	Dynamic graph (easier for debugging)
Usability	More complex but highly scalable	Pythonic & beginner-friendly
Deployment	TensorFlow Serving for production	Less robust deployment tools
Performance	Optimized for speed & scaling	Great for flexible research models
Debugging	Harder to debug (static graph issues)	Easier debugging via Python syntax
Community	Large deep-learning community	Strong presence in academia & research

Use **TensorFlow** for large-scale production environments, cloud deployments, and situations requiring optimized performance such as mobile apps. Use **PyTorch f**or research, experimentation and quick prototyping where ease of debugging and flexibility are crucial.

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

Interactive AI experimentation where AI researchers and developers use Jupyter notebooks to write, test and iterate models dynamically. The Jupyter code cells will allow running small experiments without executing entire programs.

In vizualization and debugging AI models where data scientists use integrated libraries to visualize training progress, dataset distributions and model predictions. The step-by-step debugging is easier due to its ability to execute cells separately.

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

It utilizes the tokenization feature hence recognizes complex words and contractions.

It supports Named Entity Recognition (NER) hence extracts entities like company names, locations and products.

Identifies nouns, verbs and adjectives.

Understands grammar relationships such as the subject and object.

Optimized for large-scale NLP applications.

In summary, by using spaCy, AI developers can build efficient NLP pipelines for applications such as chatbots and automated text classifications.

COMPARE SCIKIT-LEARN AND TENSORFLOW

Target Applications;

Scikit-learn is best for classical ML algorithms; TensorFlow is best for deep learning models.

Scikit-learn uses algorithms such as regression and decision trees; TensorFlow uses Neural Networks (CNN, RNN and Transformer).

Scikit-learn uses structured/ tabular data; TensorFlow uses unstructured data such as images, text and video.

Ease of use for beginners;

Scikit-learn is more beginner-friendly; TensorFlow is for more advanced users.

Scikit-learn uses simple python syntax; TensorFlow requires knowledge of tensors.

Scikit-learn is easy to debug since it uses NumPy-like operations;TensorFlow is harder due to complex computation graphs.

Community Support;

Scikit-learn has a community size that is large but ML focused; TensorFlow is huge with deep learning experts.

Scikit-learn is used in research and practical ML; TensorFlow is adopted in AI products.

Scikit-learn has extensive beginner-friendly docs for documentation; TensorFlow has detailed but complex for newcomers documentations.