1. What is the difference between Predictive/Discriminative AI and Generative AI?

- **Answer:**
  - **Discriminative AI:** Models the decision boundary between classes. It learns the conditional probability $P(Y|X)$ (Given input X, what is the probability it belongs to class Y?).
    - *Examples:* Logistic Regression, SVM, CNNs for classification (Cat vs. Dog).
  - **Generative AI:** Models the distribution of the data itself. It learns the joint probability $P(X, Y)$ or just $P(X)$. It tries to understand *how* the data is created so it can generate *new* samples that look like the training data.
    - *Examples:* GANs, VAEs, and LLMs (Next Token Prediction).
- **Follow-up:** *Which requires more data to train and why?*
  - **A:** Generative models generally require significantly more data. Learning the entire distribution of data points (to generate new ones) is a much harder mathematical problem than simply finding a line/boundary that separates two groups.

---

2. What is LLM, and how are LLMs trained?

- **Answer:**
  - An LLM is a deep learning model (usually Transformer-based) trained on massive text corpora to predict the next token in a sequence.
  - **Training Pipeline:**
    1. **Pre-training:** Self-supervised learning on massive datasets (web, books). Objective: Next Token Prediction. Result: Base Model (learns grammar, facts, reasoning).
    2. **Supervised Fine-Tuning (SFT):** Training on high-quality (Instruction, Output) pairs. Result: Instruct Model (learns to follow commands).
    3. **RLHF/DPO (Alignment):** Reinforcement Learning from Human Feedback or Direct Preference Optimization. Result: Chat Model (aligned with human values like safety and helpfulness).
- **Follow-up:** *What is the difference between Pre-training and Fine-tuning regarding computational cost?*
  - **A:** Pre-training is astronomically expensive (months on thousands of GPUs). Fine-tuning is relatively cheap (hours/days on a few GPUs).

---

3. What is a token in the language model?

- **Answer:**
  - A token is the fundamental unit of text an LLM processes. It is **not** always a word. It can be a character, a sub-word, or a whole word.
  - Modern LLMs use **BPE (Byte Pair Encoding)**.

  - *Rule of Thumb:* 1,000 tokens $\approx$ 750 words.
  - *Example:* "apple" might be 1 token, but "friendship" might be split into "friend" + "ship" (2 tokens).
- **Follow-up:** *Why do LLMs struggle with arithmetic/math?*
  - **A:** Tokenization often splits numbers inconsistently. "100" might be one token, but "101" might be "10" and "1". This makes it hard for the model to learn digit-by-digit operations.

---

4. How to estimate the cost of running SaaS-based and Open Source LLM models?

- **Answer:**
  - **SaaS (OpenAI/Anthropic):** Cost = (Input Tokens $\times$ Price) + (Output Tokens $\times$ Price). *Note: Output tokens are usually 3x-10x more expensive.*
  - **Open Source (Self-Hosted):**
    - **GPU Cost:** Hourly rate of the GPU (e.g., A100/H100).
    - **VRAM Requirement:** ~2 bytes per parameter (FP16). A 7B model needs ~14GB just to load, plus extra for the KV Cache (context).
    - **Utilization:** Cost depends heavily on batch size. Low utilization = high cost per token.

---

5. Explain the Temperature parameter and how to set it.

- **Answer:**
  - Temperature scales the logits (raw scores) before the Softmax function during decoding.
  - **Low Temperature (<1.0):** Sharpens the probability distribution. The model picks the most likely token. Result: Deterministic, factual, focused. (Use for: Coding, Math).
  - **High Temperature (>1.0):** Flattens the distribution. Less likely tokens get a chance. Result: Creative, random, diverse. (Use for: Creative writing, Brainstorming).

---

6. What are different decoding strategies for picking output tokens?

- **Answer:**
  - **Greedy:** Always pick the highest probability token. (Fast, but repetitive).
  - **Beam Search:** Track top K sequences (beams) at each step. (Better quality, computationally expensive).
  - **Top-K Sampling:** Sample from the top K most likely tokens only.

    - **Top-P (Nucleus) Sampling:** Sample from the smallest set of tokens whose cumulative probability adds up to P (e.g., 0.9). This is **dynamic**—if the model is unsure, the pool is big; if sure, the pool is small.
- **Follow-up:** *Why is Top-P generally preferred over Top-K?*
    - **A:** Top-P adapts to the context. Top-K is rigid (always 50 tokens), which cuts off good options in "flat" distributions or includes bad options in "sharp" distributions.

---

7. What are different ways you can define stopping criteria in large language model?

- **Answer:**
    1. **Max Tokens:** Hard limit (e.g., stop after 512 tokens).
    2. **Stop Sequences:** Specific strings (e.g., "\n", "User:") that trigger a halt.
    3. **EOS Token:** The model predicts its own <End_Of_Sequence> token.
    4. **Time Limit:** Stop after X seconds (latency constraints).

---

8. How to use stop sequences in LLMs?

- **Answer:**
    - Stop sequences prevent the model from rambling or hallucinating the next part of a conversation.
    - **Example:** In a chatbot, if the prompt format is `User: Hi \n AI: Hello`, you must set `User:` as a stop sequence. Otherwise, the AI might generate `User: How are you?` immediately after its own answer, effectively talking to itself.

---

9. Explain the basic structure prompt engineering.

- Answer:
  A robust prompt has four parts:
    1. **Persona/Role:** "Act as a Senior Python Developer."
    2. **Instruction:** "Write a script to scrape a website."
    3. **Context/Constraint:** "Use the BeautifulSoup library. Handle 404 errors."
    4. **Format:** "Output the code in Markdown."

---

10. Explain in-context learning.

- **Answer:**
    - The ability of an LLM to learn a new task at inference time by seeing examples in the prompt, **without any weight updates**.

    - If you show it English: Dog -> French: Chien, it learns the pattern "Translation" and applies it to the next input.
- **Follow-up:** *Does In-Context Learning update the model's weights?*
    - **A:** No. It relies on the model's existing internal representations and attention mechanism to copy the pattern.

---

11. Explain types of prompt engineering.

- **Answer:**
    - **Zero-Shot:** No examples ("Translate this").
    - **Few-Shot:** Providing examples (k-shot) before the query.
    - **Chain of Thought (CoT):** Asking the model to "think step by step".
    - **RAG:** Injecting retrieved documents into the context.

---

12. What are some of the aspects to keep in mind while using few-shot prompting?

- **Answer:**
    1. **Class Balance:** If you give 3 positive examples and 1 negative, the model will be biased toward "Positive" (Majority Label Bias).
    2. **Recency Bias:** LLMs tend to repeat the label of the *last* example provided.
    3. **Diversity:** Examples should cover edge cases, not just simple ones.

---

13. What are certain strategies to write good prompts?

- **Answer:**
    1. **Delimiters:** Use """, ---, or XML tags <data> to separate instructions from content.
    2. **Structured Output:** Explicitly ask for JSON or HTML to make parsing easier.
    3. **Positive Constraints:** Tell the model *what to do* ("Use short sentences") rather than *what not to do* ("Don't use long sentences").

---

14. What is hallucination, and how can it be controlled using prompt engineering?

- **Answer:**
    - **Hallucination:** Confidently stating false information.
    - **Controls:**
        1. **Grounding:** "Answer ONLY using the text below. If the answer is not there, say 'I don't know'."
        2. **Chain of Thought:** Asking for reasoning steps reduces logic errors.

3. **Citations:** Asking the model to quote the source text.

---

15. How to improve the reasoning ability of LLM through prompt engineering?

- **Answer:**
  - **Chain of Thought (CoT):** Force the model to generate intermediate steps.
  - **Zero-Shot CoT:** Just adding "Let's think step by step" improves math/logic performance significantly.
  - **Few-Shot CoT:** Providing examples that include the reasoning trace (e.g., `Q: ... A: First, we calculate X, then Y...`).

---

16. How to improve LLM reasoning if your COT prompt fails?

- **Answer:**
  1. **Self-Consistency:** Generate 5 different CoT paths and pick the most common answer (Majority Voting).
  2. **Reflexion:** Ask the model: "Review your previous answer. Did you miss anything? Correct it."
  3. **Least-to-Most:** Break the problem into sub-questions. Ask the model to solve Q1, then use that to solve Q2.
  4. **Tree of Thoughts (ToT):** Explore multiple reasoning branches and "backtrack" if a branch looks unpromising (like a search algorithm).