جامعة الحسين التقنية
Al Hussein Technical University

Center of Professional Development and Community Outreach

Generative AI Training Program - Professional Development (Reskilling) Training Program

Generative AI - Prompt Engineering & Interaction

جامعة الحسين التقنية
Al Hussein Technical University

Module 3: Prompt Engineering & Interaction

Dr. Abeer Al-Hyari
Abeer.alhyariups@htu.edu.jo

Module-3

Module	Module Name and Focus	Key Topics to Cover	PLOs Addressed
Module 3	**Prompt Engineering & Interaction**	**1. Fundamentals of Effective Prompting** **2. Advanced Prompting Techniques** **3. Programmatic Prompting & Modalities**	Foundations (Primary), Model Development

Objectives

1. Understand what makes a prompt effective.
2. Learn how ambiguity affects model outputs.
3. Apply clarity, specificity, and context to improve results.
4. Learn how assigning roles affects model tone and reasoning.
5. Design prompts that align with specific professional personas.
6. Understand how LLMs learn “in context.”
7. Learn when and how to use examples in prompts.
8. Learn structured reasoning prompting for complex tasks.
9. Understand how CoT and ToT improve problem-solving.
10. Understand reusable, scalable prompting methods.
11. Learn to design prompts for text, code, and image generation.

Topics Covered

- **Fundamentals of Effective Prompting**
  - Clarity, Specificity, and Context: How to structure prompts to minimize ambiguity.
- **Role-Based and Persona Prompting**
  - Guiding the model by assigning it a role (e.g., "Act as a software engineer").
- **Advanced Prompting Techniques**
  - **Zero-shot** and **Few-shot** Learning: Adapting the model with zero or minimal examples directly in the prompt.
  - **Chain-of-Thought (CoT)** and **Tree-of-Thought (ToT)**: Techniques to force the model to demonstrate step-by-step reasoning for better results.
- **Programmatic Prompting & Modalities**
  - Prompt Templates: Creating reusable prompt structures.
  - Prompting for Modalities: Strategies for generating **Text**, **Code**, and **Images** (e.g., controlling style and negative prompts).

جامعة الحسين التقنية
Al Hussein Technical University

Fundamentals of Effective Prompting: Clarity, Specificity & Context

What is prompting?

- A prompt is a natural language text that requests the generative AI to perform a specific task. Generative AI is powered by very large machine learning models that use deep neural networks that have been pretrained on vast amounts of data.
- The large language models (LLMs) are very flexible and can perform various tasks. For example, they can summarize documents, complete sentences, answer questions, and translate languages. For specific user input, the models work by predicting the best output that they determine from past training.

What is prompting?

- However, because they're so open-ended, your users can interact with generative AI solutions through countless input data combinations. The AI language models are very powerful and don't require much to start creating content. Even a single word is sufficient for the system to create a detailed response.
- That being said, not every type of input generates helpful output. Generative AI systems require context and detailed information to produce accurate and relevant responses. When you systematically design prompts, you get more meaningful and usable creations. In prompt engineering, you continuously refine prompts until you get the desired outcomes from the AI system.

Why prompting matters in LLM

- Prompt engineering jobs have increased significantly since the launch of generative AI. Prompt engineers bridge the gap between your end users and the large language model. They identify scripts and templates that users can customize and complete to get the best result from the language models. These engineers experiment with different types of inputs to build a prompt library that application developers can reuse in different scenarios.
- Prompt engineering makes AI applications more efficient and effective. Application developers typically encapsulate open-ended user input inside a prompt before passing it to the AI model.

Why prompting matters in LLM

- For example, consider AI chatbots. A user may enter an incomplete problem statement like, *"Where to purchase a shirt."* Internally, the application's code uses an engineered prompt that says, *"You are a sales assistant for a clothing company. A user, based in Alabama, United States, is asking you where to purchase a shirt. Respond with the three nearest store locations that currently stock a shirt."* The chatbot then generates more relevant and accurate information.

Components of a prompt

- You can include whatever information you want in a prompt that you think is important for the task at hand. Generally, prompt content fall within one of the following components:
  - Task (required)
  - System instructions (optional)
  - Few-shot examples (optional)
  - Contextual information (optional)
  - https://docs.cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design

Dimensions of prompt quality

- Clarity: Avoiding vague instructions.
- Specificity: Adding constraints (format, tone, style).
- Context: Giving background or situational framing.

Common mistakes in prompting

1. Being Too Vague or Ambiguous
2. Lack of Role or Context
3. Overloading a Single Prompt
4. Not Providing Examples or Format Guidance
5. Ignoring Iteration

Common mistakes in prompting

6. Not Setting Constraints or Criteria
7. Ignoring the Model’s Limitations
8. Forgetting to Set the Output Format
9. Using Negative Prompts Poorly (for Image/Creative Tasks)
10. Skipping Chain-of-Thought or Step-by-Step Guidance

جامعة الحسين التقنية
Al Hussein Technical University

Role-Based and Persona Prompting

Defining roles

- Role prompting is a prompt engineering technique that instructs an artificial intelligence (AI) model to take on a specific role or persona when generating a response. This technique can be used to guide the model's tone, style and behavior, which can lead to more engaging outputs.

Defining roles

- Role prompting can be used to give a chatbot a persona to better interact with users or an AI agent to better interact with other agents. For example, many agentic frameworks use role-playing agents to complete tasks and collaborate effectively.
- ChatDev uses a role-prompting technique called a self-attention mechanism. This mechanism clearly defines the agent’s role that acts as a guideline for its generated outputs.

Role vs. context distinction

- **Role = Who the model should "be"**
  - **Definition:**
    The *role* tells the model **which identity, perspective, or expertise** to adopt when generating the response.
  - It shapes **tone, vocabulary, and priorities** — as if the model is performing a role in a scenario.
- **Context = What situation or information surrounds the task**
  - **Definition:**
    The *context* provides **background, goals, constraints, and audience** for the prompt.
    It helps the model understand *why* and *for whom* it is generating the response.

Combining roles and constraints

- combining roles and constraints is one of the most effective techniques in prompt engineering, because it lets you control how the model behaves and what boundaries it must respect at the same time.
- Combining both makes the prompt precise and reliable.

Element	Purpose	Example
Role	Defines *who or what* the model should act as — shaping its tone, perspective, and reasoning style.	"You are a university lecturer in data science."
Constraint	Defines *rules or limits* the output must follow — shaping format, scope, or style.	"Answer in bullet points under 200 words."

Prompt: A Good Structure

- [ROLE / CONTEXT SETTING]

  You are a [specific role] who [main goal or expertise area].

- [TASK DEFINITION]

  Your task is to [do X].

- [CONSTRAINTS / OUTPUT GUIDELINES]

  Follow these rules:

  - [constraint 1]
  - [constraint 2]
  - [constraint 3]

Personality and tone control

- **Persona/Personality:**
  - Who or what the model is acting as. Also called "role" or "vision.“
  - Example: You are a math tutor here to help students with their math homework.
- **Tone:**
  - The tone of the response. You can also influence the style and tone by specifying a persona. Also called "style," "voice," or "mood.“
  - Example: Respond in a casual and technical manner.

Zero-Shot and Few-Shot Prompting

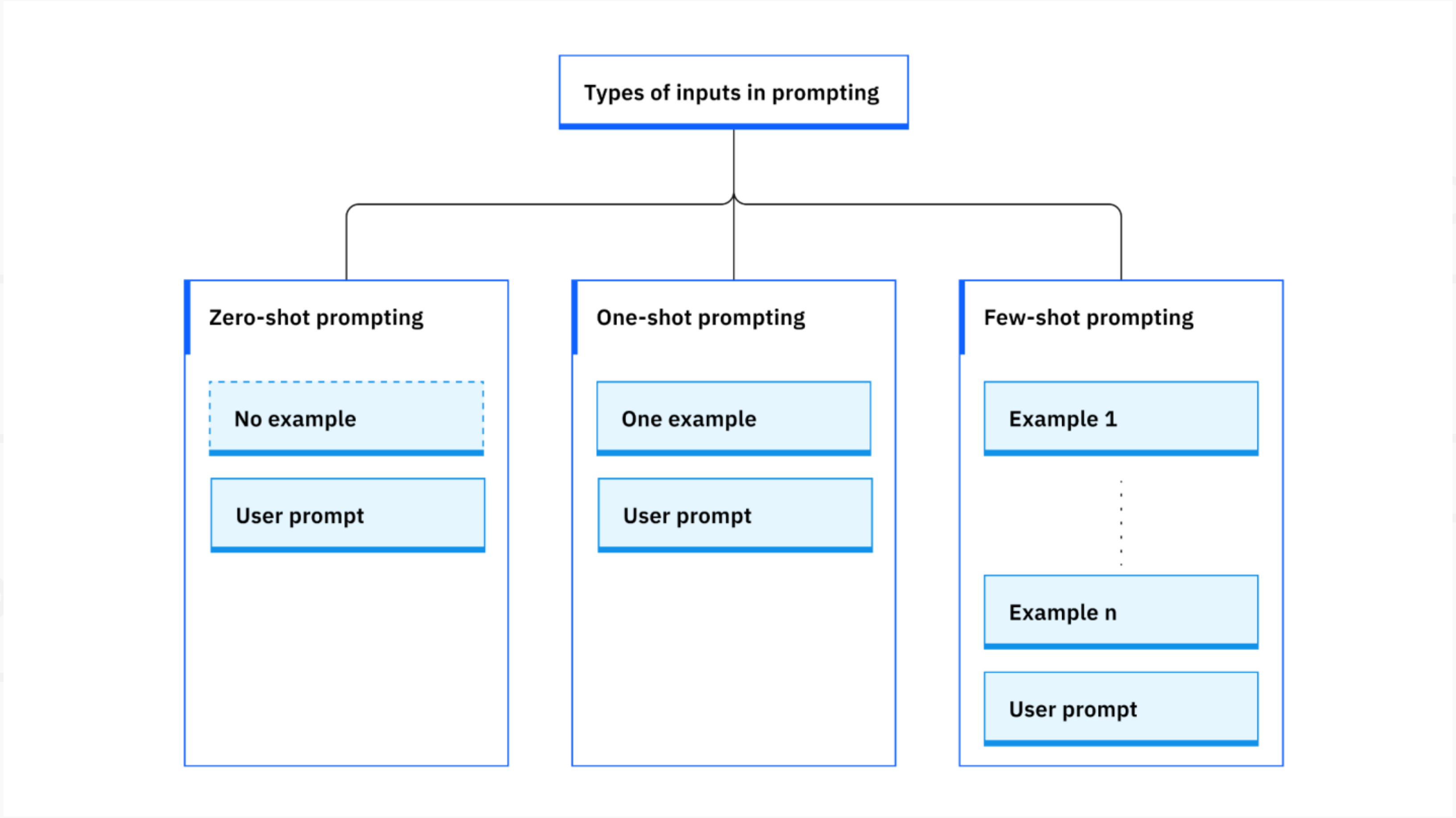
Zero-shot prompting

- Zero-shot prompting means that the prompt used to interact with the model won't contain examples or demonstrations. The zero-shot prompt directly instructs the model to perform a task without any additional examples to steer it.
- https://www.datacamp.com/tutorial/zero-shot-prompting
- Advantages:
    - Flexibility
    - Can handle new situations or tasks that weren't considered before
    - Efficiency

Few-shot prompting

- **Few-shot prompting** provides two or more examples, which helps the model recognize patterns and handle more complex tasks. With more examples, the model gains a better understanding of the task, leading to improved accuracy and consistency.
- Examples: https://learnprompting.org/docs/basics/few_shot?srsltid=AfmBOoraDi_oSfj2MHb4fzArS-jSGkkYJxws5Sd4ewvPMBuAFtEHMOXV

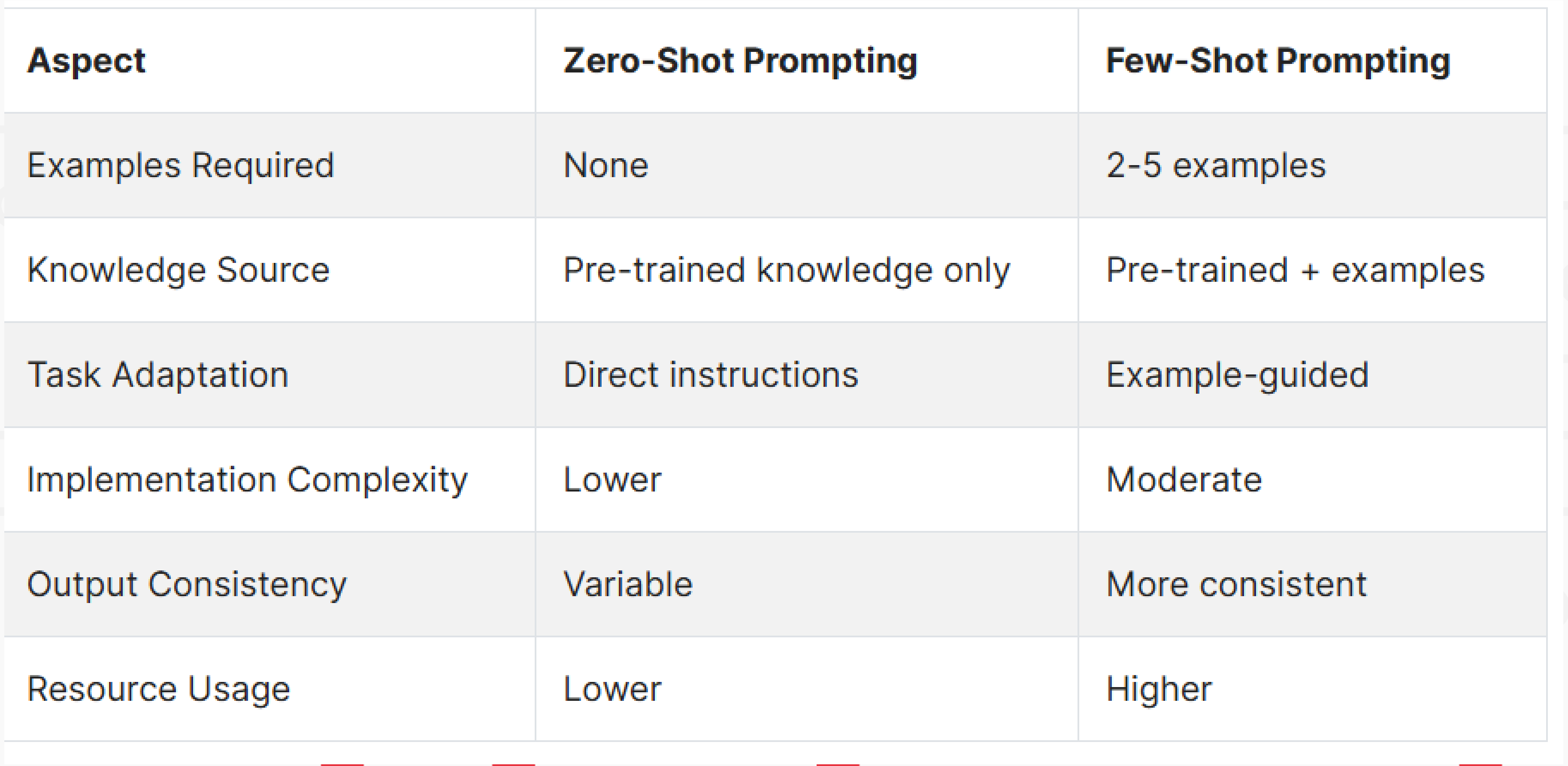
Types of Inputs in Prompting



https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/creative-assets/s-migr/ul/g/00/13/types-of-prompting.component.m.ts=1752514858252.png/content/adobe-cms/us/en/think/topics/zero-shot-prompting/jcr:content/root/table_of_contents/body-article-8/image_414581189

Prompt calibration: choosing examples carefully

جامعة الحسين التقنية
Al Hussein Technical University



Aspect	Zero-Shot Prompting	Few-Shot Prompting
Examples Required	None	2-5 examples
Knowledge Source	Pre-trained knowledge only	Pre-trained + examples
Task Adaptation	Direct instructions	Example-guided
Implementation Complexity	Lower	Moderate
Output Consistency	Variable	More consistent
Resource Usage	Lower	Higher

Prompt calibration: choosing examples carefully

- **Key Takeaways**
  - Use **zero-shot** for simple, general tasks like categorization or quick answers.
  - Use **few-shot** for complex, nuanced tasks where precision and context matter.
  - Combining both methods can balance efficiency and accuracy for varied tasks.

Evaluating model adaptation quality

- **Model adaptation quality** measures **how effectively a language model adjusts its behavior** to follow prompts, examples, roles, or context.
  It reflects how well the model:
  - Understands instructions,
  - Maintains context consistency,
  - Produces outputs aligned with the user’s intent or domain.
- In prompt engineering, this evaluation ensures that **the model is not just generating text**, but is **adapting intelligently** to the guidance provided.

جامعة الحسين التقنية
Al Hussein Technical University

Chain-of-Thought (CoT) and Tree-of-Thought (ToT) Reasoning

Why reasoning matters in LLMs

- **Standard LLMs remain incredibly powerful**. They are great at handling tasks like summarizing text, generating content, or answering day-to-day questions.
- Yet they can stumble when confronted with complex or multi-step challenges, mainly due to the fact that they are trained to make a "best guess" at what the answer should be.
- As a result, they are prone to logical errors – or what we call "hallucinations". Their ability to quickly guess the answer is great for speed, but it doesn't leave much room for explaining **how** they arrive at the output. For users, it can feel like a "black box."

Why reasoning matters in LLMs

- **Reasoning LLMs promise something more.** They aim to tackle questions not just by guessing the next likely word, but by systematically working through the underlying logic.
- A reasoning model is an LLM that has been fine-tuned to break complex problems into smaller steps, often called "reasoning traces," prior to generating a final output. Increasingly sophisticated means of training models to employ chain-of-thought reasoning and other multi-step decision-making strategies have yielded state-of-the-art performance, particularly on benchmarks for logic-driven tasks like math and coding.
- Example: https://www.linkedin.com/pulse/why-reasoning-matters-ai-katerina-zanos-382mf/

Chain-of-Thought (CoT)

- Chain-of-thought prompting is a technique that breaks down a complex question into smaller, logical parts that mimic a train of thought. This helps the model solve problems in a series of intermediate steps rather than directly answering the question. This enhances its reasoning ability.
- You can perform several chain-of-though rollouts for complex tasks and choose the most commonly reached conclusion. If the rollouts disagree significantly, a person can be consulted to correct the chain of thought.
- For example, if the question is *"What is the capital of France?"* the model might perform several rollouts leading to answers like *"Paris," "The capital of France is Paris,"* and *"Paris is the capital of France."* Since all rollouts lead to the same conclusion, *"Paris"* would be selected as the final answer.

Chain-of-Thought (CoT)- Example

Problem:
If a shirt costs $30 and is discounted by 20%, what is the final price?

Solution:

The discount is **20% of $30**.
20%×30=0.2×30=6

The discount amount is **$6**.

Subtract the discount from the original price:
30−6=24

The final price is **$24**.

Explanation:
This reasoning follows a **linear chain** — each step logically depends on the previous one. It's like a single train of thought moving forward without branching.

Tree-of-Thought (ToT)

- The tree-of-thought technique generalizes chain-of-thought prompting. It prompts the model to generate one or more possible next steps. Then it runs the model on each possible next step using a tree search method.
- For example, if the question is "What are the effects of climate change?" the model might first generate possible next steps like "List the environmental effects" and "List the social effects.“ It would then elaborate on each of these in subsequent steps.

Tree-of-Thought (ToT) - Example

Problem:
A farmer has 17 sheep. All but 9 run away. How many sheep are left?

Path A – Arithmetic interpretation:

"All but 9 run away" might mean:
17−9=8 sheep ran away, so 9 are left.
→**Possible answer: 9**

Path B – Language misinterpretation:

Maybe "All but 9 run away" means 9 ran away, and the rest stayed.
17−9=8 sheep are left.
→**Possible answer: 8**

Tree-of-Thought (ToT) - Example

Now evaluate:
The phrase “All but 9 run away” literally means **all except 9 ran away** — therefore **9 remained**.

Final answer: 9 sheep are left.

- **Explanation:**
  Here, we considered **two reasoning branches** (two interpretations), evaluated both, and then **selected the logically consistent path**. This is the essence of **Tree-of-Thought** reasoning — exploring, comparing, and selecting among multiple thought paths.

Integrating reasoning with role-based and few-shot techniques

- When prompting advanced language models (LLMs), combining **reasoning-based prompting** (like *Chain-of-Thought*) with **role-based** and **few-shot** techniques can greatly improve both **accuracy** and **consistency**.
- This integration helps the model:
  - Think through problems step-by-step (**reasoning**)
  - Respond with the expertise or tone of a professional (**role-based**)
  - Learn from examples in the same prompt (**few-shot learning**)

Group Task

Programmatic Prompting & Modalities

Prompt Templates

- A prompt template is a prompt that includes replaceable variables. Prompt templates enable you to test how different prompt formats perform with different prompt data, without requiring you to write multiple individual prompts.
- **Why Use Prompt Templates?**
  - Ensure **clarity and consistency** across tasks.
  - Make it easy to **adapt** prompts for new contexts.
  - Useful for **automation** and **programmatic prompting** (e.g., in apps or batch tasks).
  - Facilitate **collaboration**—everyone uses the same format.

Example: https://docs.cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/prompt-templates

Variable-based and reusable prompts

A **variable-based prompt** is a **template with placeholders (variables)** that can be filled in dynamically.

- It’s like a function with parameters.
- The text of the prompt remains the same, but the **inputs** (variables) change per use.
- This allows you to **reuse** the same prompt structure across multiple tasks.

Variable-based and reusable prompts

```
Act as a {role}.
Your task is to {task_description}.
Input: {input_data}
Output format: {output_format}
```

```
Act as a data analyst.
Your task is to summarize the trends in customer feedback.
Input: 500 reviews from 2024.
Output format: bullet points with sentiment labels.
```

The **variables** here are:

•{role}
•{task_description}
•{input_data}
•{output_format}

Programmatic prompting

- Programmatic prompting is using LLMs to generate prompts to use with LLMs. The idea is also known as "prompt auto-optimization" and LLMs are surprisingly good at it. Hence, it is an extra step before the main LLM inference to answer a query, whereby an LLM (possibly a smaller one) is used to tweak the words in the prompt and hopefully improve them.
- Programmatic Prompt Engineering is a process that involves the strategic creation and optimisation of prompts to effectively guide the behavior of AI models, particularly language models LLMs.

Modalities

- **Definition:**
  In prompt engineering, **modalities** refer to the **types or formats of data** that AI models can process and generate. Each modality requires slightly different prompting strategies because the model interprets and outputs information in distinct ways.

Modalities

1-Text Modality

- **Description:** Natural language input and output (e.g., essays, summaries, chat, code).
- **Prompting Focus:** Clarity, tone, role specification, and reasoning steps.
- **Example:**

“Act as a career coach. Write a 200-word cover letter for a data analyst applying to a sustainability firm.”

2- Code Modality

- **Description:** Prompts that generate, explain, or debug programming code.
- **Prompting Focus:** Specify the **language**, **constraints**, and **functionality** required.
- **Example:**

“Write a Python function that removes duplicate items from a list while preserving order.”

Modalities

3- Image Modality

- **Description:** Text-to-image generation or image editing.
- **Prompting Focus:** Detailed **visual description**, **style**, **composition**, and **mood**.
- **Example:**

“Generate a realistic portrait of a woman sitting in a modern office, soft lighting, neutral tones.”

4- Audio Modality

- **Description:** Involves speech-to-text, text-to-speech, or sound generation.
- **Prompting Focus:** Control **tone**, **emotion**, **voice style**, or **accent**.
- **Example:**

“Convert this text into a calm, professional female voice with a British accent.”

Modalities

5- Video Modality

- **Description:** Combining visuals, sound, and text for dynamic content.
- **Prompting Focus:** Define **scene sequence**, **camera movement**, **narration style**, and **duration**.
- **Example:**
  "Create a 30-second explainer video about renewable energy using infographic-style visuals."

6- Multimodal Prompting

- **Description:** Uses multiple input types together (e.g., text + image).
- **Prompting Focus:** Clarify how the inputs relate and what output form you want.
- **Example:**
  "Describe the mood of the person in this image and write a caption that fits a motivational post."

Text Generation

- **Definition:**
Text modality refers to the use of **written natural language** as both input (prompt) and output (response) for AI models such as **Large Language Models (LLMs)**.
- **Goal:**
To guide the model to generate coherent, contextually relevant, and goal-aligned **text-based outputs**(e.g., summaries, essays, answers, instructions, or dialogues).
- **Examples of text-based tasks:**
  - Question answering
  - Summarization
  - Translation
  - Content writing (emails, reports, stories)
  - Dialogue or conversational agents

Code Prompting

- Designing prompts that make an LLM **write, explain, refactor, test, or review code** with high reliability. It combines precise specs, I/O contracts, constraints, and examples so the model behaves like a disciplined engineer.

Core patterns (use these a lot)

- Language + version
  - “Write **Python 3.11** code using **typing** and **pathlib** only.”
- I/O contract
  - “Implement `def slugify(text: str) -> str:`. No I/O. Pure function.”
- Constraints
  - “No external deps. O(n) time, O(1) extra space. Handle Unicode.”

Code Prompting

- **Spec + edge cases**
  - “Trim, lowercase, replace spaces with -, collapse dashes, strip punctuation. Examples: ‘A b’→‘a-b’, ‘Café’→‘cafe’.”
- **Tests-first (TDD)**
  - “Generate pytest tests first; then provide implementation that passes them.”
- **Few-shot formatting**
  - Show 1–2 miniature examples of desired docstring, type hints, and error handling.
- **Tooling-aware**
  - “Return only a unified diff patch.” or “Output JSON with keys: summary, warnings, patch.”
- **Guardrails**
  - “If unsure, ask for assumptions at top as # ASSUMPTIONS: then proceed.”

Image Prompting

- **Image prompting** is the process of crafting **text instructions (prompts)** that guide an **AI image generation model** (like DALL·E, Midjourney, or Stable Diffusion) to produce visuals that match the desired **content, style, and mood**.
- It's about **translating visual imagination into precise textual descriptions** that the model can interpret effectively.
- **Purpose**
  - To generate **new images** from text (text-to-image).
  - To **edit or modify** existing images (image-to-image or inpainting).
  - To control **style, realism, composition, and mood**.

Image Prompting - Structure

Element	Description	Example
Subject	What the image should show	“A cat with two different eye colors”
Details	Attributes, environment, objects	“sitting on a gaming chair, holding a joystick”
Style	Artistic or visual style	“realistic 8K photography, soft lighting”
Mood / Tone	Emotion or atmosphere	“playful and bright mood”

Ethical prompting

- Ethical prompting refers to the responsible design and use of prompts to ensure that interactions with AI models are safe, fair, transparent, and respectful of human values. It focuses on preventing the generation or amplification of harmful, biased, misleading, or privacy-violating content through prompt design and model use.

Ethical prompting – Core Principle

Principle	Description	Example
Fairness	Avoid prompts that reinforce stereotypes or discrimination.	✗ “Describe a typical nurse (only mentioning one gender).” ✓ “Describe the daily duties of a nurse in a hospital setting.”
Transparency	Clearly indicate when content is AI-generated.	Add: “This report was drafted with AI assistance.”
Privacy & Consent	Don’t prompt models to produce or reveal private or personal data.	Avoid: “Generate private info about [person].”
Non-harm / Safety	Avoid content that promotes hate, self-harm, or violence.	Never prompt for illegal or harmful instructions.
Accountability	The human user remains responsible for the outcomes of AI-generated text, code, or media.	Always review and verify before sharing or publishing.

Reading Sources

- Example prompt bank (good vs. bad prompts).
- LLM playground (ChatGPT / Claude / Gemini) for experiments.
- Suggested readings:
  - OpenAI “Prompt Engineering Guide
  - ”Anthropic “Prompt Crafting Techniques”
  - Papers: “Language Models are Few-Shot Learners” (Brown et al., 2020)

Module 3 Summary & Transition