



Prompt Engineering

Revision Notes for Students

Zero-Shot • Few-Shot • Chain of Thought • Role • ReAct • Self-Consistency • and more

1. What is a Prompt?

1.1 Definition

A prompt is the input text you send to a Large Language Model (LLM) to instruct it what to do. It is the primary — and often only — way humans communicate with an LLM. The quality of a prompt directly determines the quality of the model's response. A vague prompt produces a vague answer; a precise, well-structured prompt produces a precise, useful answer.

Prompt Engineering is the skill of designing, structuring, and refining prompts to reliably get the best possible output from an LLM. It does not require coding — it is a communication skill applied to AI.

1.2 Why Prompts Matter So Much

LLMs are extremely sensitive to how a request is worded. The same underlying question, phrased differently, can produce wildly different outputs — from a perfect answer to a completely wrong one. This is because LLMs are next-token predictors: they statistically continue whatever text you start. The prompt is the 'starting text' that shapes everything that follows.

Compare these two prompts asking the same thing:

WEAK PROMPT

Tell me about Python.

STRONG PROMPT

You are a Python tutor for beginners. In 3 bullet points, explain what Python is, why it is popular, and one simple use case. Keep each point under 2 sentences.


The second prompt specifies: persona (tutor), audience (beginners), format (3 bullets), scope (what, why, one use case), and length (under 2 sentences). The result is far more useful and predictable.

💡 Key Point: The golden rule of prompting: the more specific and structured your prompt, the more specific and structured the LLM's output will be.

1.3 Anatomy of a Well-Structured Prompt

A strong prompt typically contains one or more of these building blocks. Not all are required for every task — use the ones that match the complexity of your request:


Component	What It Does	Example
Role / Persona	Assigns an identity to the LLM	"You are a senior data scientist with 10 years of experience."
Context	Provides background information	"We are building a customer churn prediction model using telecom data."
Task / Instruction	Clearly states what to do	"Explain the difference between precision and recall in simple terms."
Format	Specifies how the output should look	"Respond in exactly 3 bullet points. Use plain English."
Examples	Shows the LLM the expected input/output pattern	"Input: 'great product!' → Output: Positive"
Constraints	Sets limits on length, tone, or content	"Keep your answer under 100 words. Avoid technical jargon."
Output indicator	Marks where the model should start its response	"Answer:" or "Classification:" at the end of the prompt

 **Remember:** You do not need all seven components in every prompt. Start simple — add role, format, or constraints only when the LLM's default output is not meeting your needs.

1.4 Prompt vs System Message vs User Message

When using LLMs via APIs (like LangChain), prompts are typically split into two parts — a system message and a user message. Understanding the difference helps you structure prompts correctly in code:

Message Type	Purpose	Who Writes It	Example
System Message	Sets the LLM's persistent persona, tone, and rules for the entire conversation	Developer / application	"You are a helpful Python tutor. Always explain with examples. Never use jargon."
User Message	The actual task or question for this specific turn	End user	"What is a list comprehension and when should I use it?"
Assistant Message	Previous LLM response — used to maintain conversation history	LLM (stored by app)	"A list comprehension is a concise way to create lists..."

 **Key Point:** System messages are your most powerful prompting tool. A well-written system message can transform a generic LLM into a specialised tutor, code reviewer, data analyst, or customer service agent — without any fine-tuning.

2. Zero-Shot Prompting

2.1 What is Zero-Shot Prompting?

Zero-shot prompting means giving the LLM a task with no examples — you just describe what you want and the model uses its pre-trained knowledge to respond. It is the most natural way to interact with an LLM, like asking a question to a knowledgeable friend. The 'zero' refers to zero examples provided in the prompt.

Modern LLMs like Gemini and GPT-4 are surprisingly capable with zero-shot prompts for a wide range of tasks because they have been trained on vast amounts of text covering almost every domain. For everyday tasks, zero-shot prompting is usually your starting point.

Example — Sentiment Classification

PROMPT

Classify the sentiment of the following review as Positive, Negative, or Neutral.

Review: "The delivery was fast but the packaging was damaged."

Sentiment:

LLM RESPONSE

Negative

Example — Code Explanation

PROMPT

Explain what the following Python code does in one sentence:

```
squares = [x**2 for x in range(10)]
```

LLM RESPONSE

This creates a list of the squares of numbers from 0 to 9 using a list comprehension.

☒ **When to use:** Simple tasks the LLM clearly understands, quick questions, tasks where you just need a fast answer, initial testing before adding complexity.

☐ **Avoid when:** Complex multi-step reasoning, specialised domain tasks with precise output formats, or when you need very consistent structured outputs.

3. Few-Shot Prompting

3.1 What is Few-Shot Prompting?

Few-shot prompting provides the LLM with a small number of worked examples (typically 2–5) before the actual task. These examples demonstrate the exact pattern you want — the input format, the reasoning style, and the expected output format. The model then follows the established pattern for the new input.

Think of it like showing a new employee how the job is done with 3 worked examples before asking them to do it themselves. The examples act as an implicit instruction manual — far more effective than trying to describe the pattern in words alone.

3.2 Why Few-Shot Works

LLMs learn from patterns. When you provide consistent input-output pairs, the model identifies the pattern and continues it for the new input. Few-shot prompting is especially powerful when:

- **The output format is complex or unusual** — examples show exactly what structure is expected.
- **The task has domain-specific conventions** — examples teach the model the right vocabulary and style.
- **Zero-shot gives inconsistent results** — examples anchor the model's output style.

3.3 Few-Shot Example — Sentiment Analysis with Format Control

Here we want the model to always respond with a single word in a specific format. Examples enforce this:

PROMPT

```
Classify each review as Positive, Negative, or Neutral.  
Respond with only the label – no explanation.
```

```
Review: "Absolutely loved it! Will buy again."  
Sentiment: Positive
```

```
Review: "Completely broken on arrival. Terrible."  
Sentiment: Negative
```

```
Review: "It arrived on time, nothing special."  
Sentiment: Neutral
```

```
Review: "Works okay but the battery drains fast."  
Sentiment:
```

LLM RESPONSE

```
Negative
```

3.4 Few-Shot Example — Data Extraction

Extracting structured data from unstructured text — examples teach the exact JSON format:

PROMPT

```
Extract the name and city from each sentence.  
Respond in this exact format: Name: <name>, City: <city>
```

```
Input: "Sarah lives in New York and works in finance."  
Output: Name: Sarah, City: New York
```

```
Input: "My colleague Ahmed just moved to Berlin last month."  
Output: Name: Ahmed, City: Berlin
```

```
Input: "The conference was attended by Priya from Singapore."  
Output:
```


LLM RESPONSE


```
Name: Priya, City: Singapore
```

3.5 Few-Shot Best Practices

- **Keep examples consistent** — all examples must follow the exact same input/output format. Mixed formats confuse the model.
- **Use 2 to 5 examples** — 2 is often enough for simple patterns; 5 gives stronger anchoring for complex formats. More than 6 rarely helps.
- **Include edge cases** — if some inputs have unusual properties, show an example of how to handle them.
- **Put the new input last** — always place the actual task at the very end, after all examples, to continue the pattern naturally.
- **Label clearly** — use consistent labels like 'Input:' / 'Output:' or 'Q:' / 'A:' to separate examples cleanly.

☒ **When to use:** Tasks requiring strict output format (JSON, CSV, labels), domain-specific classification, data extraction, zero-shot gives inconsistent or wrong format results, building reliable pipelines.

 **Avoid when:** Conversational chat (examples feel unnatural), when the task is already working well with zero-shot, when you only have poor-quality examples.

 **Remember:** Quality over quantity: 2 excellent, representative examples beat 8 mediocre ones. Bad examples actively mislead the model.

4. Chain of Thought (CoT) Prompting

4.1 What is Chain of Thought?

Chain of Thought (CoT) prompting encourages the LLM to think through a problem step by step — showing its reasoning before giving the final answer. Instead of jumping straight to an answer, the model explains its logic in intermediate steps, just like how a student works through a maths problem by writing out each step.

This technique, introduced by Google researchers in 2022, dramatically improved LLM performance on reasoning-intensive tasks — especially maths, logic puzzles, multi-step problems, and common sense reasoning. The key insight is: forcing the model to 'think out loud' reduces errors because each step provides context and a check for the next step.

4.2 Zero-Shot CoT — The Magic Phrase

The simplest form of CoT requires no examples at all. Simply appending the phrase 'Let's think step by step.' to almost any problem triggers step-by-step reasoning in modern LLMs. This is called Zero-Shot Chain of Thought.

Without CoT — LLM answers directly (often wrong for hard problems)

PROMPT

A store sells apples for \$0.50 each. Maria buys 12 apples and pays with a \$10 note. How much change does she receive?

LLM RESPONSE

\$4.00

With Zero-Shot CoT — LLM reasons step by step (more reliable)

PROMPT

A store sells apples for \$0.50 each. Maria buys 12 apples and pays with a \$10 note. How much change does she receive?

Let's think step by step.

LLM RESPONSE

Step 1: Cost of 12 apples = $12 \times \$0.50 = \6.00

Step 2: Change = $\$10.00 - \$6.00 = \$4.00$

Maria receives \$4.00 in change.

4.3 Few-Shot CoT — Teaching the Reasoning Style

For more complex or domain-specific tasks, you can combine few-shot prompting with Chain of Thought — providing examples that show both the step-by-step reasoning AND the final answer. This teaches the model your preferred reasoning style.

PROMPT

Solve each problem by showing your working first, then give the final answer on a new line.

Q: A train travels 120km in 2 hours. What is its average speed?
A: Speed = Distance / Time
Speed = 120km / 2h = 60 km/h
Answer: 60 km/h

Q: A rectangle has length 8m and width 5m. What is its area?
A: Area = Length x Width
Area = 8 x 5 = 40
Answer: 40 square metres

Q: A class has 30 students. 40% passed the exam.
How many students passed?
A:

LLM RESPONSE

Number passed = 40% of 30
= 0.40 x 30 = 12
Answer: 12 students passed.

4.4 CoT Trigger Phrases

These phrases reliably activate step-by-step reasoning in modern LLMs:

Phrase	Best For
Let's think step by step.	General maths, logic, and multi-step reasoning
Think through this carefully before answering.	Ambiguous problems or questions needing careful judgement
Show your working, then give the final answer.	Maths and calculation problems
Break this down into steps.	Complex instructions or planning tasks
First, identify what is being asked. Then...	Questions with hidden sub-tasks or ambiguous scope
Reason through each option before choosing.	Multiple choice or decision tasks
Explain your reasoning, then conclude.	Analytical or argumentative tasks

✅ **When to use:** Maths and arithmetic problems, logical reasoning and puzzles, multi-step instructions, decisions requiring justification, debugging code, any task where accuracy is more important than speed.

⚠️ **Avoid when:** Simple factual lookup (CoT adds unnecessary length), real-time speed-critical applications, creative tasks like writing poems or stories where linear reasoning is not needed.

💡 **Key Point:** CoT increases the token usage (and API cost) because responses are longer. Use it selectively for complex tasks — not for simple factual questions.

5. Role / Persona Prompting

5.1 What is Role Prompting?

Role prompting assigns a specific identity, expertise level, or persona to the LLM before giving it a task. By telling the model who it is, you prime it to draw on a particular style, vocabulary, depth of knowledge, and communication approach — without any fine-tuning or extra training required.

It works because LLM training data includes text written by experts, teachers, doctors, engineers, and virtually every other profession. When you say 'You are a senior Python engineer', the model statistically draws on text patterns associated with that role.

5.2 Examples

Without role — generic, shallow response

PROMPT

Explain recursion.

LLM RESPONSE

Recursion is when a function calls itself.

With role — tailored depth and style

PROMPT

You are a computer science professor teaching first-year students. Explain recursion using a simple real-world analogy, then show a 5-line Python example. Avoid jargon.

LLM RESPONSE

Think of recursion like Russian dolls: each doll contains a smaller version of itself until you reach the smallest doll. A function calls itself with a smaller input until it hits the base case (the smallest doll) and stops.

```
def countdown(n):
    if n == 0:          # base case - stop here
        print('Go!')
    else:
        print(n)
        countdown(n - 1) # call itself with smaller n
```

5.3 Useful Role Templates

Role Prompt	Effect on Output
You are a senior Python engineer with 10 years of experience.	Production-quality code with edge case handling and best practices
You are a data science tutor explaining to a complete beginner.	Simple language, analogies, avoids jargon, builds up gradually
You are a strict code reviewer. Find all bugs and issues.	Critical, detailed feedback — won't be polite about problems

You are a concise technical writer. Be precise and brief.	Dense, accurate, no filler words — documentation style
You are a Socratic teacher. Guide with questions, don't just answer.	Responds with questions to develop student thinking
You are a sceptical scientist. Challenge assumptions.	Pushes back on unsupported claims, asks for evidence
You are a friendly customer support agent for AlBees platform.	Warm, helpful, brand-consistent tone

☒ **When to use:** Any time you want the LLM to adopt a specific communication style, when targeting a particular audience (beginners vs experts), building specialised assistants or chatbots.

☐ **Avoid when:** Casual one-off questions where the extra framing adds no value.

6. Instruction & Format Control Prompting

6.1 What is Format Control?

Format control prompting explicitly specifies the structure, length, and layout of the expected output in the prompt itself. LLMs default to a flowing prose style — format control overrides this to produce JSON, tables, bullet lists, numbered steps, or any other structure you need for downstream processing or display.

6.2 Examples

Output as structured JSON

PROMPT

Extract the following details from the job posting and return ONLY a valid JSON object. No explanation, no markdown, just JSON.

Job posting: 'Senior ML Engineer at TechCorp in Berlin.
Salary: 90,000 EUR. Required: Python, TensorFlow, 5 years exp.'

Required fields: role, company, location, salary, skills (list)

LLM RESPONSE

```
{
  "role": "Senior ML Engineer",
  "company": "TechCorp",
  "location": "Berlin",
  "salary": "90,000 EUR",
  "skills": ["Python", "TensorFlow"]
}
```

Controlling length and structure


PROMPT

Summarise the key differences between SQL and NoSQL databases.
Format: exactly 3 bullet points, each max 20 words.
Start each bullet with a bold keyword.

LLM RESPONSE

```
- Schema: SQL uses fixed schemas; NoSQL is schema-less and flexible.
- Scalability: SQL scales vertically; NoSQL scales horizontally across servers.
- Use Case: SQL suits structured data; NoSQL suits unstructured or real-time data.
```

☒ **When to use:** Building pipelines that parse LLM output programmatically (JSON extraction), dashboards, reports, any production application where consistent output structure is critical.

 **Key Point:** Always say 'ONLY return JSON — no explanation, no markdown code fences' when you need parseable JSON. LLMs tend to wrap JSON in markdown backticks unless told not to.

7. Self-Consistency Prompting

7.1 What is Self-Consistency?

Self-consistency is a technique where you send the same prompt to the LLM multiple times (with temperature > 0 so responses vary), collect all the different reasoning paths and answers, and then take the most frequent answer as the final output — like majority voting. This reduces the impact of occasional reasoning errors in any single response.

It was introduced alongside Chain of Thought and is particularly powerful for problems where the model's reasoning can vary but the correct answer is deterministic (like maths or logic). Instead of hoping the model gets it right on the first try, you sample many attempts and trust the consensus.

7.2 How it Works

PROMPT (sent 5 times with temperature=0.7)

A farmer has 17 sheep. All but 9 die. How many are left?
Let's think step by step.

RESPONSES FROM 5 RUNS

```
Run 1: 'All but 9 means 9 survive. Answer: 9' ✓
Run 2: 'All but 9 die, so 17 - 9 = 8 remain. Answer: 8' ✗
Run 3: '9 sheep survive. Answer: 9' ✓
Run 4: '17 - 9 = 8. Answer: 8' ✗
Run 5: 'All but 9 die means 9 are left. Answer: 9' ✓

Majority vote: 9 appears 3 times → Final Answer: 9 ✓
```

☒ **When to use:** High-stakes reasoning tasks where accuracy is critical, maths and logic problems, when a single CoT response is occasionally wrong and you need higher confidence.

☐ **Avoid when:** Real-time applications (too slow), creative tasks where diversity is good, when cost is a concern — multiple API calls multiply the cost linearly.

8. ReAct Prompting (Reason + Act)

8.1 What is ReAct?

ReAct (Reasoning + Acting) is a prompting technique that interleaves reasoning steps with action steps — allowing the LLM to think about what to do, take an action (like calling a tool, searching the web, or querying a database), observe the result, and then reason further based on that observation.

It combines Chain of Thought (reasoning) with tool use (acting) in an alternating Thought → Action → Observation loop. This is the foundation of modern LLM Agents — AI systems that can autonomously use tools to complete complex multi-step tasks.

8.2 The Thought → Action → Observation Loop

PROMPT

Answer the following question by reasoning step by step.
You have access to a calculator tool.

Q: What is 15% of 847, rounded to the nearest whole number?

LLM RESPONSE

Thought: I need to calculate 15% of 847.

15% = 15/100 = 0.15

0.15 x 847 = ?

Action: calculator(0.15 * 847)

Observation: 127.05

Thought: 127.05 rounded to nearest whole number is 127.

Final Answer: 127

☑ **When to use:** Building LLM agents that use external tools (search, calculator, database, APIs), complex tasks requiring real-time information retrieval, multi-step automated workflows.

⚠ **Avoid when:** Simple single-step tasks, when no tools are available, when you need a purely generative response without tool calls.

💡 **Key Point:** ReAct is the backbone of LangChain Agents. When you see 'agents' in LangChain documentation, the underlying technique is usually a variation of ReAct prompting.

9. Contextual Prompting (RAG Pattern)

9.1 What is Contextual Prompting?

Contextual prompting injects relevant background information directly into the prompt before asking the question. The LLM uses this provided context to answer rather than relying solely on its training data. This is the core idea behind Retrieval-Augmented Generation (RAG) — one of the most important patterns in production LLM applications.

LLMs have a knowledge cutoff date and no access to your private documents. By retrieving relevant text from your own data source (documents, databases, PDFs) and inserting it into the prompt as context, you give the LLM everything it needs to answer accurately — without any fine-tuning.

PROMPT

```
Use ONLY the information in the context below to answer the question. If the answer is not in the context, say 'I don't know based on the provided information.'
```

CONTEXT:

```
AIBees is an online education platform focused on AI and data science. The Python Fundamentals course covers core Python, NumPy, and Pandas. It is 6 hours long and costs Rs 2,499. Students get lifetime access after purchase.
```


```
QUESTION: How long is the Python Fundamentals course and does it include NumPy?
```

```
ANSWER:
```

LLM RESPONSE

```
The Python Fundamentals course is 6 hours long and yes, it does include NumPy as part of its curriculum alongside core Python and Pandas.
```

☒ **When to use:** Q&A over private documents (PDFs, wikis, knowledge bases), customer support bots, domain-specific assistants, any time the LLM needs access to information it wasn't trained on.

 **Key Point:** RAG = Retrieve relevant document chunks → inject as context → ask the question. This simple pattern powers most production AI assistants used by companies today.

10. Constrained & Negative Prompting

10.1 What is Constrained Prompting?

Constrained prompting explicitly tells the LLM what NOT to do, what to avoid, and what hard limits apply to its response. LLMs tend to be helpful to a fault — they add qualifiers, disclaimers, introductions, and verbose padding unless explicitly told not to. Constraints trim this noise.

Without constraints — verbose with unwanted padding

PROMPT

```
Translate this to French: 'Machine learning is fascinating.'
```

LLM RESPONSE

```
Sure! Here is the translation of your sentence into French:  
"L'apprentissage automatique est fascinant."  
Let me know if you need any other translations!
```

With constraints — clean, direct output only

PROMPT

```
Translate to French. Return ONLY the translated sentence.  
No explanation, no greeting, no extra text.
```

```
'Machine learning is fascinating.'
```

LLM RESPONSE

```
L'apprentissage automatique est fascinant.
```

10.2 Useful Constraint Phrases

Constraint Type	Prompt Language to Use
No preamble	"Do not start with 'Sure', 'Certainly', or any greeting. Begin directly."
No explanation	"Return ONLY the answer. No explanation, no extra text."
Length limit	"Answer in exactly 3 sentences." or "Keep under 50 words."
No hallucination	"If you are not certain, say 'I don't know.' Do not guess."
No markdown	"Return plain text only. Do not use bullet points, headers, or bold."
Strict format	"Return ONLY valid JSON. No code fences, no extra text."
Stay on topic	"Answer only the question asked. Do not add unsolicited advice."
Audience level	"Explain as if to a 10-year-old. No technical terms."

☒ **When to use:** Production pipelines where the output is parsed by code, when building chatbots with strict brand voice, any time previous prompts produce verbose or incorrectly formatted responses.

11. Prompting Techniques — Quick Reference


Use this table as a revision cheat sheet. Match the technique to your task type:

Technique	Core Idea	Best For	Avoid When
Zero-Shot	Task only — no examples	Simple, well-known tasks	Complex formats or reasoning
Few-Shot	2–5 input/output examples before task	Strict format, classification, extraction	No good examples available
Chain of Thought	Add 'Let's think step by step'	Maths, logic, multi-step reasoning	Simple lookups, creative tasks
Few-Shot CoT	Examples include reasoning steps too	Domain-specific complex reasoning	Token budget is tight
Self-Consistency	Same prompt N times → majority vote	High-accuracy critical reasoning	Real-time, cost-sensitive apps
Role / Persona	Assign identity before task	Tone, audience targeting, specialised assistants	One-off casual queries
Format Control	Specify exact output structure	JSON, bullets, tables, pipelines	Open-ended creative tasks
ReAct	Reason → Act (use tool) → Observe → Repeat	Agents, tool use, live data retrieval	No tools available
Contextual / RAG	Inject retrieved context into prompt	Private docs, Q&A bots, knowledge bases	Public general knowledge only
Constrained	State explicit do-nots and hard limits	Parsing output in code, strict format	Exploratory / creative tasks

11.1 Prompting Decision Tree

When choosing a technique, follow this simple decision path:

Question	If Yes → Use
Does the LLM already know this type of task well?	Zero-Shot — start here, simplest option
Is the output format critical (JSON, labels, tables)?	Few-Shot — show examples of the exact format
Does the task require multi-step reasoning or maths?	Chain of Thought — add 'Let's think step by step'
Do you need very high accuracy on a reasoning task?	Self-Consistency — run N times, take majority
Do you need a specific tone, style, or expertise level?	Role Prompting — assign a persona first
Does the LLM need your private documents or live data?	Contextual / RAG — inject retrieved text as context
Does the LLM need to call external tools or APIs?	ReAct — use an agent with tool access
Is the LLM being too verbose or adding unwanted text?	Constrained Prompting — add explicit do-nots

 **Remember:** These techniques are not mutually exclusive. A production prompt often combines several: Role + Few-Shot + Format Control + Constraints in a single system message. Build up complexity gradually — start with zero-shot and add layers only when needed.