



Prompt Patterns Library

Prompt and eval library 01 | Twelve copy-paste templates for everyday LLM tasks (brief, few-shot classify, extract, summarise, rewrite, grounded answer, decompose, reason-then-answer, critique-revise, clarify-first, compare, meta-prompt), each with its variables and when to use it; theory in Ilm 09

#Prompting #Templates #Library

How to use this library

A law firm's form letters	Prompt patterns
Standard letter for each situation	One pattern per task shape
Blanks for names, dates, amounts	{{VARIABLES}} filled by code
Clauses that must never change	Fixed instructions and output format
A partner reviews before sending	An eval set checks before shipping

WHERE THE ANALOGY BREAKS A form letter gives the same result every time; a prompt gives a distribution of outputs, so each pattern still needs measuring on your own cases.

A pattern library works like form letters at a law firm

Before Using prompt patterns, can you explain...

	Not yet	Partly	Yes
MUST KNOW System vs user message Learn: Ilm 09	☐	☐	☐
MUST KNOW Temperature and sampling Learn: Ilm 09	☐	☐	☐
HELPFUL Tokens per character (≈ 4 chars/token in English) Learn: Ilm 01	☐	☐	☐
HELPFUL Reasoning models and effort settings Learn: Ilm 08	☐	☐	☐

Check before copying; each item is recalled in one line, not re-taught

Understand text	Produce text	Hard or vague tasks
Put text into fixed labels P2 few-shot classify	Any one-off job P1 task brief	Many steps P7 decompose / chain
Pull fields out of text P3 extract (schema: sheet 02)	Change tone, length, format P5 rewrite	Maths, logic, non-reasoning model P8 reason-then-answer
Shorten for a reader P4 summarise for audience	Weigh options P11 compare with criteria	Ambiguous request P10 clarify-first
Answer from documents P6 grounded answer	Improve quality of a draft P9 critique-revise	Write or fix a prompt P12 meta-prompt

Pattern finder: task shape → template on this sheet

P1 Task brief: the base of every other pattern

Template		≈ tokens		
role	You are a {{ROLE}} writing for {{AUDIENCE}}	44 ch 11		
context	Context: {{CONTEXT}} Why it matters: {{PURPOSE}}	48 ch 12		
instructions	Task: {{TASK}} Constraints: - Use only facts given in the context. - If something is missing, write MISSING: <what>.	116 ch 29		
input	<input> {{INPUT}} </input>	26 ch 7		
output	Output format: {{FORMAT}} No preamble.	39 ch 10		
11	12	29	7	10

Template ≈ 69 tokens (273 characters / 4, rounded up per block; a rule of thumb for English, real tokenizers differ)

variable	meaning	used in	example value	check
{{ROLE}}	job title with domain	role	senior HR adviser	ok
{{AUDIENCE}}	who reads the output	role	new team managers	ok
{{CONTEXT}}	facts the model needs	context	Company of 300 staff, hybrid work, 25 ...	ok
{{PURPOSE}}	what the output is used for	context	onboarding pack	ok
{{TASK}}	one-sentence job	instructions	Explain the leave policy	ok
{{INPUT}}	the text to work on	input	Leave requests go through the HR porta...	ok
{{FORMAT}}	exact shape of the reply	output	5 bullet points, max 20 words each	ok

Filled example

≈ 105 tokens

You are a senior HR adviser writing for new team managers.

Context: Company of 300 staff, hybrid work, 25 days leave

Why it matters: onboarding pack

Task: Explain the leave policy

Constraints:

- Use only facts given in the context.
- If something is missing, write MISSING: <what>.

<input>

Leave requests go through the HR portal 2 weeks ahead.

</input>

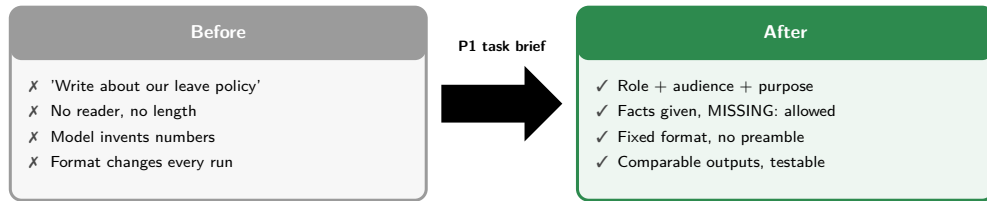
Output format: 5 bullet points, max 20 words each. No preamble.

{{variable}} slot

example value

no value given

Task brief anatomy, token estimate per block and the filled example



Same request, vague vs briefed

⌘ P2-P6: understanding and transforming text

P2 few-shot classify + P3 extract fields

Classify the ticket into exactly one label:
 {{LABELS}} (use OTHER if none fits)

```
<example><ticket>I was charged twice</ticket>
<label>billing</label></example>
<example><ticket>App crashes on login</ticket>
<label>bug</label></example>
{{MORE_EXAMPLES}}
```

```
<ticket>{{TICKET}}</ticket>
```

Reply with the label only.

```
# ---- P3 extract fields (schema: sheet 02)
Extract these fields from the document:
{{FIELD_LIST}} (name: meaning: type)
Rules: copy values verbatim; dates as YYYY-MM-DD;
null when absent, never guess.
<document>{{DOCUMENT}}</document>
Return JSON with exactly these keys.
```

P4 summarise + P5 rewrite + P6 grounded answer

Summarise the text for {{AUDIENCE}}, who need it
 to {{DECISION_OR_ACTION}}.
 Length: {{N}} bullets, each under {{W}} words.
 Keep numbers, names and dates exact. Put the single
 most important point first. No new facts.
 <text>{{TEXT}}</text>

```
# ---- P5 rewrite (tone, length, format)
Rewrite the text below.
Keep: meaning, facts, {{KEEP}}.
```

```
Change: tone to {{TONE}}, length to about {{LENGTH}},
format to {{FORMAT}}.
<text>{{TEXT}}</text>
Return only the rewritten text.
```

```
# ---- P6 grounded answer, citations, abstention
Answer the question using only the sources.
Cite each claim as [S<n>]. If the sources do not
contain the answer, reply exactly: NOT_IN_SOURCES.
<sources>
```

```
<source id="S1">{{CHUNK_1}}</source>
<source id="S2">{{CHUNK_2}}</source>
</sources>
<question>{{QUESTION}}</question>
```

💡 Few-shot hygiene (P2)

Use 3-8 examples that cover every label and at least one hard boundary case; vary their order across labels so the last example does not bias the answer; never copy test cases into the examples or your eval will overstate accuracy.

⌘ P7-P12: hard, vague and multi-step tasks

P7 decompose (chain step 1) + P8 reason-then-answer

Break the goal into at most {{K}} ordered steps.
 Each step: id, action, input it needs, done-when.
 Do not execute anything yet.
 <goal>{{GOAL}}</goal>
 Return a JSON list of steps.

```
# ---- P8 reason-then-answer (non-reasoning models)
{{PROBLEM}}
Think through the problem inside <reasoning> tags,
then give only the final result inside <answer> tags.
```

P9 critique-revise + P10 clarify-first

Review the draft against this checklist:
 {{CHECKLIST}}
 For each item: PASS or FAIL with a one-line reason.
 Then write the revised draft fixing every FAIL.
 <draft>{{DRAFT}}</draft>
 Output: <review>...</review><revised>...</revised>

```
# ---- P10 clarify-first
If the request is ambiguous in a way that changes
the answer, ask at most {{N}} short questions and
stop. Otherwise answer directly.
<request>{{REQUEST}}</request>
```

P11 compare with criteria + P12 meta-prompt

Compare {{OPTIONS}} for {{USE_CASE}}.
 Criteria and weights: {{CRITERIA}}
 Output: a table option x criterion with a 1-5 score
 and a reason per cell, then the weighted total and
 a recommendation with its main risk.

```
# ---- P12 meta-prompt
Improve the prompt below for {{GOAL}}.
Failures seen: {{FAILURE_EXAMPLES}}
Keep its variables {{VARIABLE_LIST}} unchanged.
Return the new prompt, then a list of changes and
the failure each change targets.
<prompt>{{PROMPT}}</prompt>
```

⚠ Reasoning models need less, not more

On reasoning models (llm 08) skip P8: they already reason before answering; state the goal, constraints and output format, and raise the provider's effort or reasoning setting instead. Some current APIs also reject assistant prefill and forced tool choice, so use structured outputs (sheet 02) to fix the format.

📊 Worked example: budget a few-shot prompt

PROBLEM	
P2 classifier: is a 6-example prompt affordable at 50 000 tickets per day?	
1. Fixed part ⇒ $480 + 6 \times 240 = 1\,920$ chars ≈ 480 tokens why: instructions and examples repeat on every call	GIVEN - Instructions + labels: 480 characters - Each example: 240 characters - Average ticket: 1 200 characters - Rule of thumb: tokens $\approx$ characters / 4 (English) ANSWER ≈ 780 input tokens per call, 39 M per day; 62 % is cacheable, so keep the examples first and the ticket last. Check: Real tokenizers differ: count tokens with the provider's tokenizer before committing a budget.
2. Variable part ⇒ $1\,200 / 4 = 300$ tokens per ticket	
3. Input per call ⇒ $480 + 300 = 780$ tokens	
4. Input per day ⇒ $780 \times 50\,000 = 39\text{ M}$ tokens why: multiply by your provider's input price per million tokens	
5. Cache the fixed part ⇒ $480 / 780 \approx 62\%$ of input is a stable prefix why: prompt caching (llm 10) bills it at a fraction of the price	

Token estimate before choosing how many examples to keep

temperature	0-0.3 for classify/extract; 0.7-1 for creative variants (where the API still exposes it)
max output tokens	label: about 10; summary: $2 \times$ expected length; never so low that JSON gets cut
stop sequences	e.g. <code></answer></code> to end parsing early
model and version	pinned id, recorded with the prompt version (sheet 08)

🚩 Misconceptions and recap

MYTH	FACT
✗ "You are a world-class expert" boosts accuracy	✓ Specific context beats flattering personas Audience, purpose and constraints change outputs far more than superlatives.
✗ More examples are always better	✓ Few-shot gains flatten after a handful Cover labels and edge cases; extra examples cost tokens and can anchor style.
✗ Capital letters and 'MUST' make rules stick	✓ Clear, reasoned rules work better on current models Explain why a rule exists; shouting can make models over-apply it.
✗ The model will say when it does not know	✓ Only if abstention is allowed and named Give an exact token such as NOT_IN_SOURCES and test it (P6).

Pattern myths

Start from P1 role, context, task, constraints, format	Data inside tags instructions outside	Name the abstain token MISSING, OTHER, NOT_IN_SOURCES
Fixed prefix first variable input last, for caching	Reasoning model? goal + format, raise effort, skip CoT scaffolds	Measure a pattern is a hypothesis until sheet 05 runs it

Remember

Q1 Five blocks of the task brief (P1)?

A Role, context (with purpose), task with constraints, delimited input, output format.

Q2 Why add OTHER to a classifier's labels?

A Without it the model forces out-of-scope inputs into a wrong label; OTHER makes those errors visible.

Q3 How does P6 stop the model inventing answers?

A It restricts answers to the sources, requires `[S<n>]` citations and names an exact abstention string.

Q4 When do you use P8 reason-then-answer?

A On non-reasoning models for maths or logic; the `<answer>` tag lets code parse the result. Skip it on reasoning models.

Q5 What does P7 produce?

A An ordered list of steps (id, action, input, done-when) that the next prompts of the chain execute one by one.

Q6 Why put examples before the variable input?

A The fixed part forms a stable prefix that prompt caching can reuse; the varying ticket goes last.

Q7 Tokens for 2 000 English characters, rule of thumb?

A About 500 (characters / 4); verify with the real tokenizer.

Q8 What must a meta-prompt (P12) keep unchanged?

A The template variables, so the improved prompt still plugs into the same code.

Q9 Typical temperature for extraction?

A 0 to 0.3, for stable, repeatable output (when the API exposes temperature).