Prompt Engineering

Table of contents

Tactic 1 — use delimiters 1
Tactic 2 — ask for structured output 1
Tactic 3 — ask it to check whether conditions are satisfied 2
Tactic 4 — specify the steps required 2
Tactic 5 — iterate 2
Tactic 6 — be as specific as possible 2
Tactic 7 — build code incrementally 3
LLMs as tools for thought 3

Seven tactics, aimed at programming tasks. See large-language-models[1] for why they work.

Tactic 1 — use delimiters

Mark where the input starts and stops, and mark the shape of the output:

```
Here is some code in Python: $$$ … $$$

… format the output as follows: """On [DATE], [NAME] … """
```

Tactic 2 — ask for structured output

```
… Put these into an HTML table. The table should have three columns.

… format it as a JSON object: """On [DATE], [NAME] … """
```

Example:

[1] courses/comp1100/large-language-models.pdf

> Write a python program that can generate 5 sample emails, including email header. Each email should contain a body that has a greeting, then the main part, and then a sign-off. The main part should contain an order for a part, of the format:
>
> `""" I would like to order [NUMBER] [PARTS]. """`

Tactic 3 — ask it to check whether conditions are satisfied

Tell it what to do when the input doesn't match the expected shape, rather than leaving it to improvise:

> Below is an email from someone. Extract the following into a structured format of: `"""On [DATE], [PERSON] ([SENDER EMAIL ADDRESS]) from [COUNTRY] sent an email with a message about [SUMMARY]"""`. If the email contains a phone number, determine the country number, and insert the country name. **Otherwise, write "unknown country" in place of [COUNTRY].**

Tactic 4 — specify the steps required

Tactic 5 — iterate

> The following is an email from a customer outlining their opinion on a project. Create an email that responds to them, thanks them for their review, and addresses any items in the email.

Tactic 6 — be as specific as possible

A before-and-after:

This:

> Create a Python program that downloads emails from the email address "uqt-miller@gmail.com" and gets an image from them.

Versus this:

> Create a Python program that: 1. Downloads any email in the account "uqt-miller@gmail.com" 2. Saves any image attachments (JPG or PNG) in the directory called "images" 3. Deletes the email from the email inbox.

A longer example, showing how far the specificity goes — including the error case and the runtime environment:

> Create a simple html page that: 1. Use javascript to select a random image from the directory 'images/' 2. When the image is selected, display a message that says "a new image has been received" 3. When the user clicks the message, the randomly selected image is displayed. 4. If no image is available, print "There is no image in the directory <DIRECTORY>", where DIRECTORY is the complete path. 5. A message "An email has been sent to your_family@fake.com" should be displayed. 6. Make sure this runs locally.

Tactic 7 — build code incrementally

Don't ask for the whole system at once. Ask for the page, then ask for the specific piece:

> **Prompt 1:** Create a simple html page: 1. Use javascript to select a random image from the directory 'images/' …
>
> **Prompt 2:** Create javascript code that selects the file with the most recent timestamp from a list called 'images'.

LLMs as tools for thought

The most interesting use is not asking for an answer, but asking to be interrogated:

> I am doing a project using the Lean Startup methodology. I have a hypothesis about my solution that I want to test. I would like to construct a test card for it.
>
> Can you please interact with me to help me construct this?
>
> **Do not give me answers, just ask a series of questions until I have the test card.**
>
> Provide feedback on my answers.

This inverts the usual pattern: the model runs the process and you supply the content, so the test card[2] ends up containing your team's actual hypotheses rather than a plausible-sounding generated one. It stays on the right side of the course's AI policy for exactly that reason — see comp1100[3].

[2] courses/comp1100/test-cards-and-learning-cards.pdf
[3] courses/comp1100/comp1100.pdf