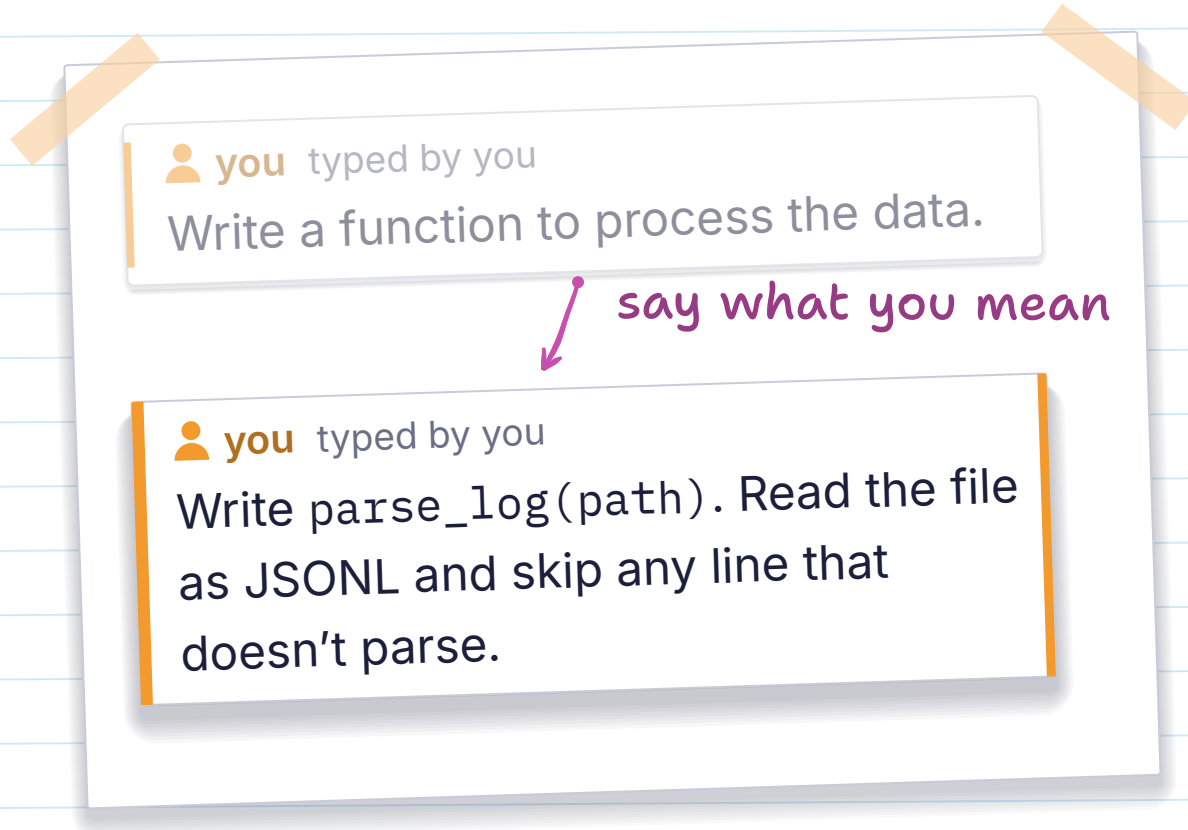
Prof. Aviv Zohar

Semester A, 2026–7

Writing a Good Prompt

Being specific, the goal and the check, examples, and a few habits



A vague request leaves the model guessing

you typed by you

Write a function to process the data.

A task, but the behavior is left open.

A vague request leaves the model guessing

you typed by you

Write a function to process the data.

A task, but the behavior is left open.

What format is the data in?

the model guesses: CSV?

Which fields actually matter?

the model guesses: all of them?

What if the input is malformed?

the model guesses: crash?

Each guess is a place where the result can differ from what you meant.

Or let the model interview you

agent session `~/logs`

you typed by you

Ask me questions until the requirements are clear, then write the function. Before you start, summarize what we agreed.

A short conversation, instead of writing every detail yourself.

Or let the model interview you

agent session `~/logs`

you typed by you

Ask me questions until the requirements are clear, then write the function. Before you start, summarize what we agreed.

agent written by the model

1. What format is the data in?
2. Which fields matter?
3. What should happen with a malformed line?

you typed by you

JSONL, one event per line. Keep the time, level and message. Skip a malformed line.

Also: call it `parse_log` and return `list[Event]`.

The same three gaps, now asked instead of guessed.

You decide the behavior. Add what it did not ask.

Answered, the request is specific

you typed by you

Write `parse_log(path: str) -> list[Event]`.
Read the file as JSONL, one event per line,
keeping the time, level and message.
Skip any line that doesn't parse.

The model is no smarter. It just has fewer guesses to make.

decided now, and checkable:

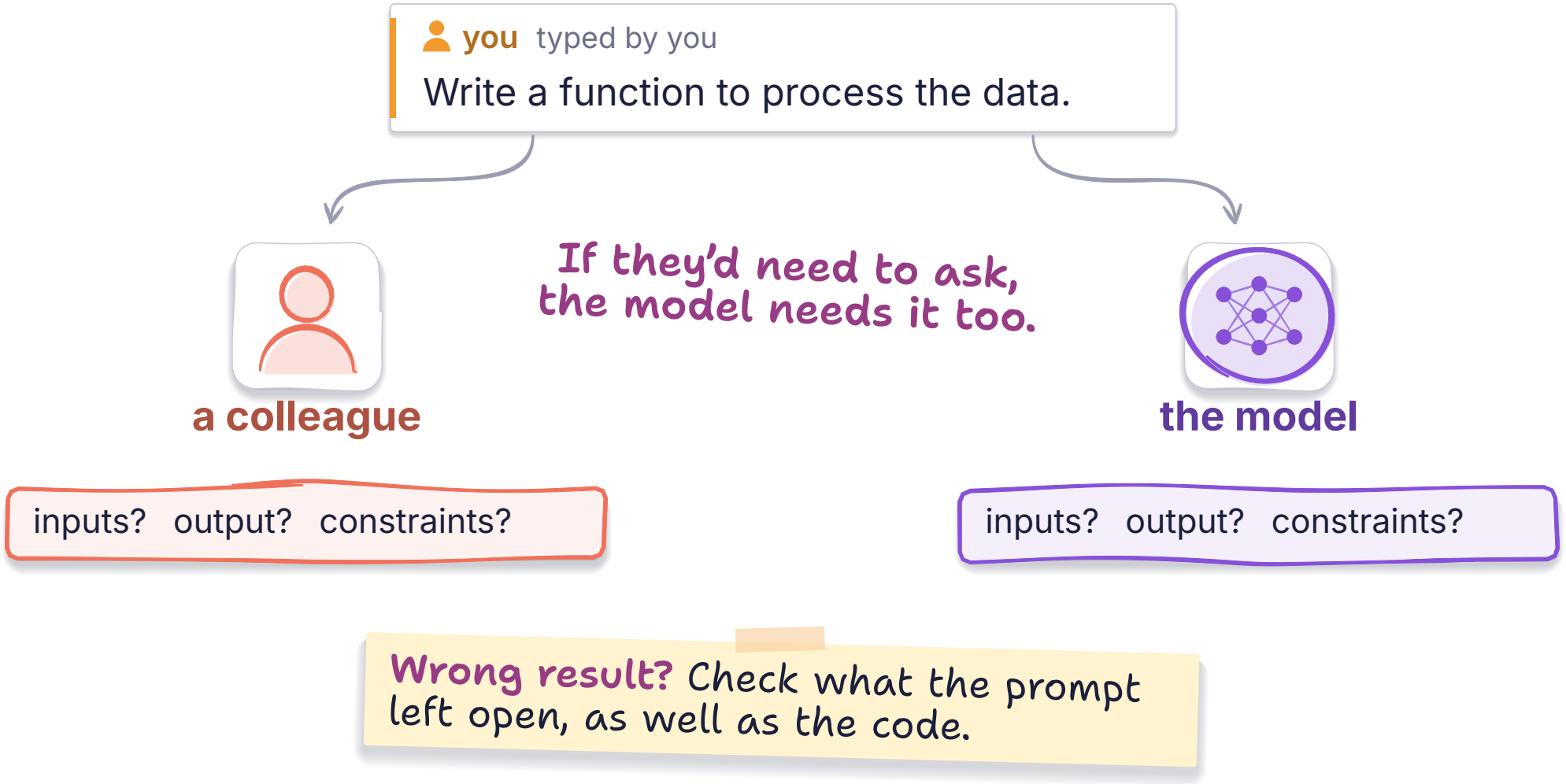
the function's type

its scope

its behavior

Check the code against these decisions.

Would a colleague need to ask?



Name the problem and a way to check it

you typed by you

Improve the code.

faster?

shorter?

more comments?

Better how? And when is it done?

Name the problem and a way to check it

you typed by you

Improve the code.

faster? *shorter?* *more comments?*

Better how? And when is it done?

you typed by you

Login fails when the email has capital letters. Add a test that reproduces the bug, and then fix it.

the problem and a concrete check

test fails before the fix

↓

test passes after the fix

Also check that the test covers **the behavior you intended**.

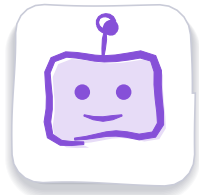
Give the goal, and let the agent work out the details

you give

the goal
Login accepts emails with capital letters.

a check
A test that fails before the fix, passes after.





the agent

the agent works out

- which file to change
- which lines
- how to fix it

~~Edit line 42 of auth.py ...~~ not a line-by-line edit

Give the goal, and let the agent work out the details

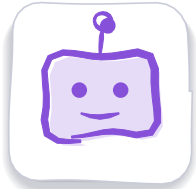
you give

the goal
Login accepts emails with capital letters.

a check
A test that fails before the fix, passes after.

constraints
Don’t change the public login API.

the agent works out



the agent

- which file to change
- which lines
- how to fix it

A required approach?
Say so, and say why.

~~Edit line 42 of auth.py …~~

not a line-by-line edit

Examples show the format you want

zero-shot

you typed by you

Label the sentiment of each restaurant review.

no worked examples

“Lovely view, but the fish was dry.” →

Examples show the format you want

few-shot

you typed by you

Label the sentiment of each restaurant review.

“Best pasta in town!”	→ **positive**
“Cold soup, rude waiter.”	→ **negative**
“Great food, but a long wait.”	→ **mixed**

“Lovely view, but the fish was dry.” →

Not what sentiment is: the labels and format you expect back.

Examples show the format you want

few-shot

you typed by you

Label the sentiment of each restaurant review.

“Best pasta in town!”	→ **positive**
“Cold soup, rude waiter.”	→ **negative**
“Great food, but a long wait.”	→ **mixed**

“Lovely view, but the fish was dry.” →

Not what sentiment is: the labels and format you expect back.

agent reply without examples

The review is fairly positive overall, though the customer was let down by the fish.

a sentence, not one of the labels

agent reply with examples

mixed

follows the pattern

Examples show the format you want

few-shot

you typed by you

Label the sentiment of each restaurant review.

“Best pasta in town!”	→ **positive**
“Cold soup, rude waiter.”	→ **negative**
“Great food, but a long wait.”	→ **mixed**

“Lovely view, but the fish was dry.” →

Not what sentiment is: the labels and format you expect back.

Examples can also clarify the task, or show an unusual case.

agent reply without examples

The review is fairly positive overall, though the customer was let down by the fish.

a sentence, not one of the labels

agent reply with examples

mixed

follows the pattern

Start with 3 to 5 representative examples, then test whether more help.

Six habits that make a prompt clear

1. **Say what to produce**
2. Give the reason
3. Make the action clear
4. Ask for structure
5. Ask for checks
6. Say it, don't type it

you typed by you

Don't use markdown.

then what?
plain text, HTML,
a numbered list?

you typed by you

Write prose.

says what
to produce

Six habits that make a prompt clear

1. Say what to produce
2. **Give the reason**
3. Make the action clear
4. Ask for structure
5. Ask for checks
6. Say it, don't type it

you typed by you

Write prose.

The text will be read aloud by text-to-speech.

The reason covers related cases you didn't spell out:

no tables *no emoji* *no code blocks*

Six habits that make a prompt clear

1. Say what to produce
2. Give the reason
3. **Make the action clear**
4. Ask for structure
5. Ask for checks
6. Say it, don't type it

you typed by you

Change the name `load` to `read_config`.

edits the code

you typed by you

Suggest a change to the name `load`.

gives advice

Politeness is fine. Say whether the agent should act, explain, or wait for your decision.

Six habits that make a prompt clear

1. Say what to produce
2. Give the reason
3. Make the action clear
4. **Ask for structure**
5. Ask for checks
6. Say it, don't type it

you typed by you

List the failing tests as JSON, with the keys `name` and `reason`.

agent written by the model

```
[
  {"name": "test_login_caps",
   "reason": "email not lowercased"}
]
```

Easy for your program to read, and easy for you to check.

Six habits that make a prompt clear

1. Say what to produce
2. Give the reason
3. Make the action clear
4. Ask for structure
5. **Ask for checks**
6. Say it, don't type it

you typed by you

Fix the bug. Run the tests and the type-checker before you report that you're done.

agent written by the model

Done. I ran the checks:

- ✓ `pytest`: 14 passed
- ✓ `mypy`: no issues

not run integration tests: no database here

Read the results, including what it could not run.

illustrative values

Six habits that make a prompt clear

1. Say what to produce
2. Give the reason
3. Make the action clear
4. Ask for structure
5. Ask for checks
6. **Say it, don't type it**

you typed by you

```
/voice
```

Claude Code's voice dictation.
In Codex: the desktop app.

space

hold it while you talk



Speaking takes less effort than typing, so your prompts get longer and more detailed.

Six habits that make a prompt clear

1. Say what to produce
2. Give the reason
3. Make the action clear
4. Ask for structure
5. Ask for checks
6. Say it, don't type it

you typed by you

Write `summarize(events) -> str`. ← task
Plain prose: it will be read aloud. ← constraint
Match this: *"3 errors since 09:00."* ← format
Run the tests before you report. ← check

Keep the parts that help on your project.

See you in the next video.