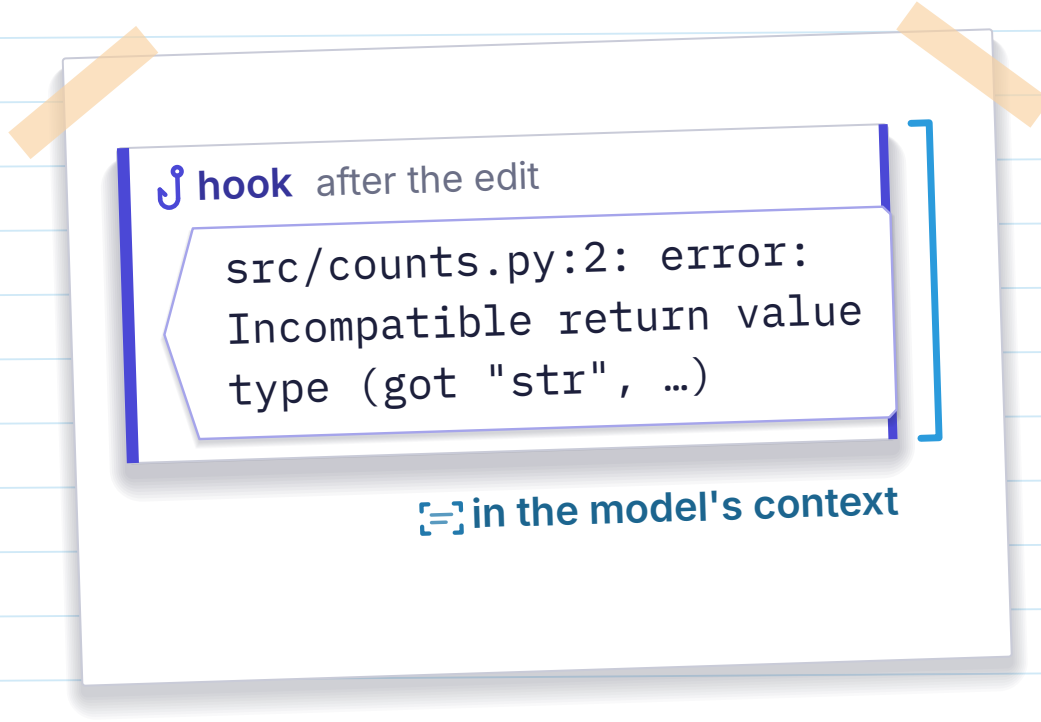
Prof. Aviv Zohar

Semester A, 2026–7

Returning feedback to the agent

Send a checker's error back to the model after an edit



The error has to reach the model

src/counts.py

```
1 def total_items(counts: list[int]) -> int:
2     return str(sum(counts))
```

Promises an `int`, returns `str`.

after the edit

hook
runs mypy





Claude Code



model

saves its output



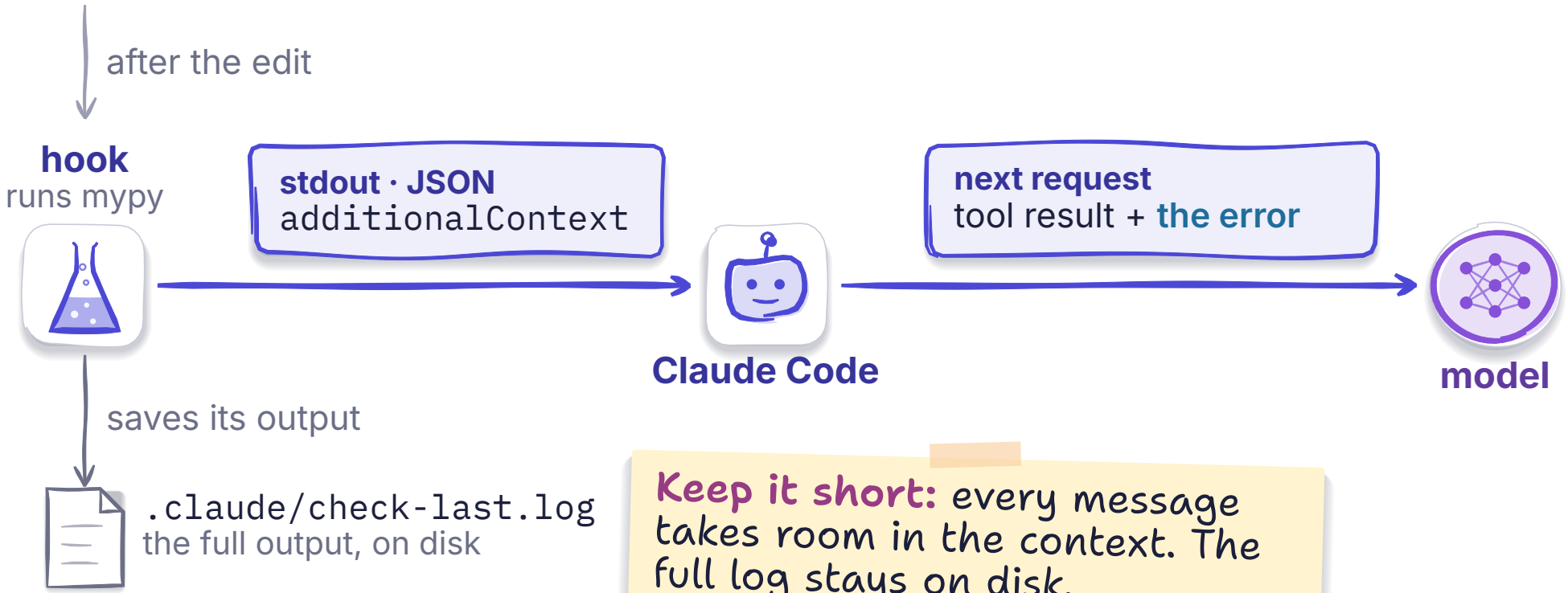
`.claude/check-last.log`
the full output, on disk

On disk only. Nothing here puts the error in the model's context.

The error has to reach the model

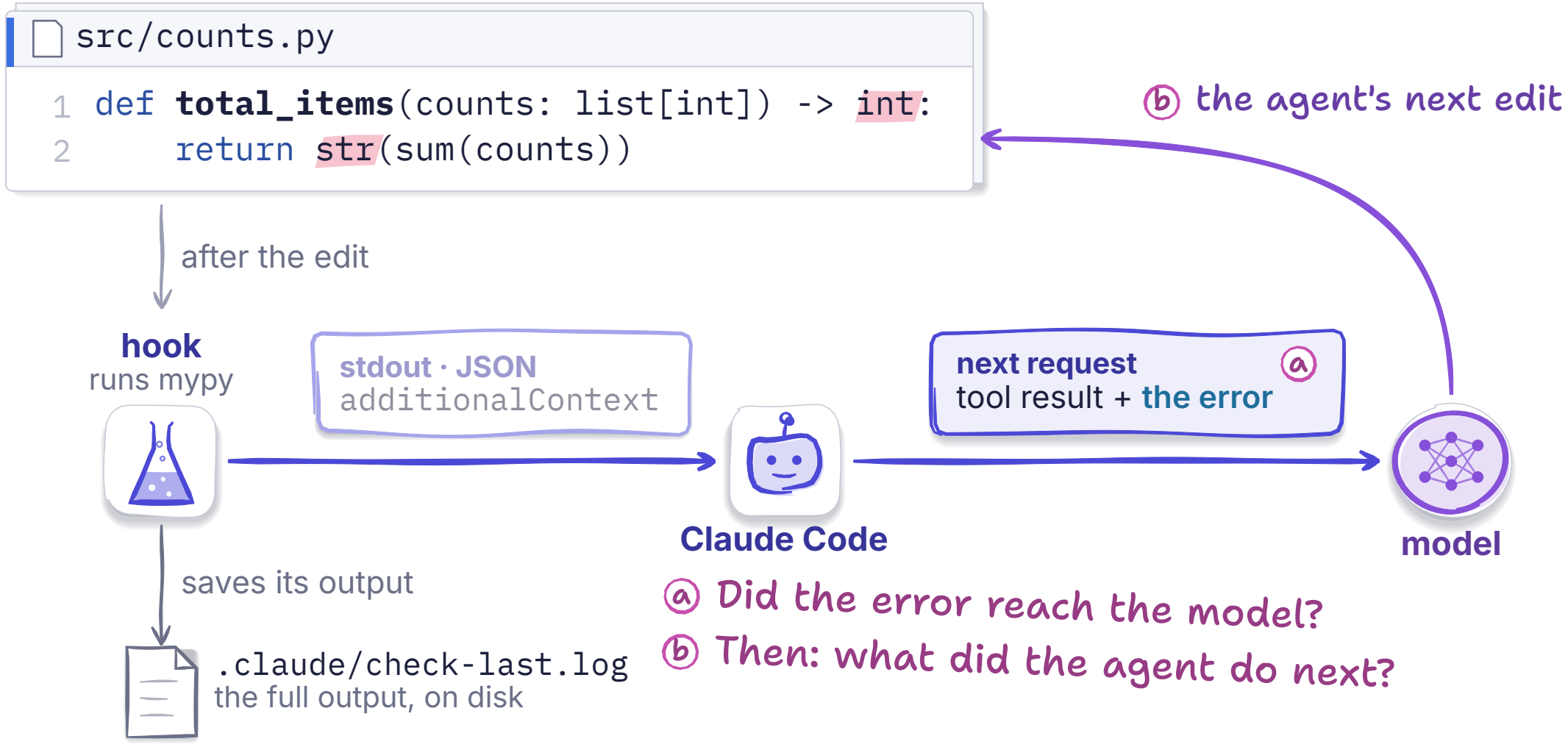
src/counts.py

```
1 def total_items(counts: list[int]) -> int:
2     return str(sum(counts))
```



Keep it short: every message takes room in the context. The full log stays on disk.

The error has to reach the model



The handler answers in JSON on standard output

stdout of feedback_check.py, after the edit

```
{
  "hookSpecificOutput": {
    "hookEventName": "PostToolUse",
    "additionalContext": "python -m mypy  ▸ 3 lines
  }
}
```

Printed after the edit succeeded. The event name tells Claude Code which format this is.

real output · Claude Code 2.1.282, 2026-09-24

The handler answers in JSON on standard output

stdout of feedback_check.py, after the edit

```
{
  "hookSpecificOutput": {
    "hookEventName": "PostToolUse",
    "additionalContext": "python -m mypy  ▸ 3 lines
  }
}
```

Text for the model: what ran, where, what is wrong, and where the full log is.

what the model reads, beside the tool result

```
python -m mypy --strict src failed:
src/counts.py:2: error: Incompatible return value type
  (got "str", expected "int")  [return-value]
Full output: .claude/check-last.log
```

For this event, plain printed text does not reach the model. This field does.

The handler answers in JSON on standard output

stdout of feedback_check.py, after the edit


```
{
  "hookSpecificOutput": {
    "hookEventName": "PostToolUse",
    "additionalContext": "python -m mypy   ▸ 3 lines
  }
}
```


the checker

mypy

exit 1 found a problem

the handler

feedback_check.py

exit 0 reported it

Its stdout holds only the JSON. Debug messages go to `stderr`.

Say what ran, where, and what it reported

hook what our handler returns

```
python -m mypy --strict src failed:
src/counts.py:2: error: Incompatible return value type
  (got "str", expected "int")  [return-value]
Full output: .claude/check-last.log
```

what ran
where the problem is
what it reported

hook a message with less in it

```
Check failed.
```

Only this, and the agent has to rebuild what we already had.

The model gets the errors; the log keeps the rest

.claude/hooks/feedback_check.py

```
44 LOG.write_text(result.stdout + result.stderr) ⓐ
45 if result.returncode == 0:
46     sys.exit(0) # passing: add nothing to the conversation ⓒ
   ⋮
52     errors = [line for line in result.stdout.splitlines() if ": error:" in line]
   ⋮
55         + "\n".join(errors[:10]) ⓑ
56         + f"\nFull output: {Path(LOG).relative_to(project_root())}"
```

ⓐ The complete output goes to the log on disk.

ⓑ Up to ten error lines and the log's path go to the model.

ⓒ A passing check adds nothing.

A hook can point the agent at an image

agent session `~/figure-demo`

hook after the edit to plot.py

```
Rendered output/figure.png.
Use Read to inspect the image
before continuing.
```

tool call asked by the model

Read `output/figure.png`

tool result run by the harness

`[image content]`

agent written by the model

The title text overflows past the figure edges and overlaps the "Items counted" y-axis label.

text only: a path and an instruction

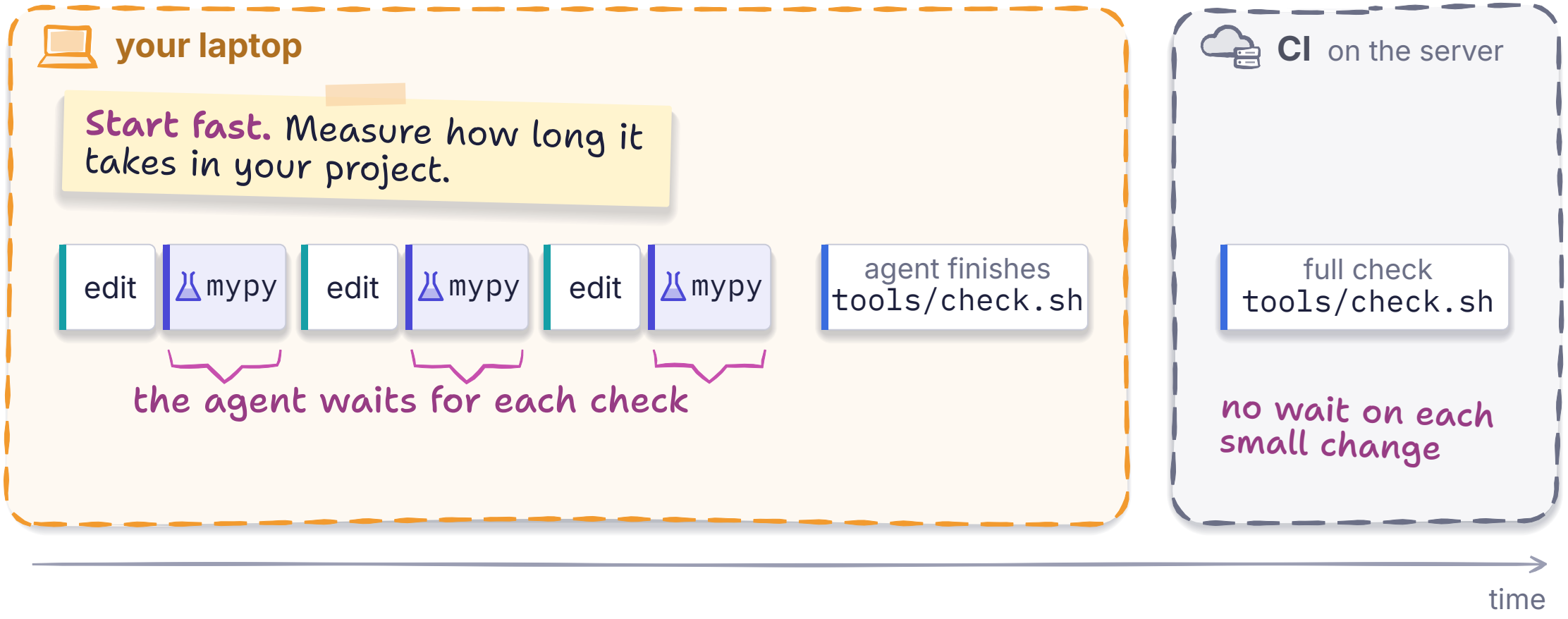
Text does not make a Read happen. Check that the agent reads each new render.



render · the render the Read call returned

real run, condensed · Claude Code 2.1.302, 2026-10-17

A fast check after each edit, the full suite later



A failed check is not always a code defect

a defect in the code

hook the check ran

```
src/counts.py:2: error:
Incompatible return value type
(got "str", expected "int")
```

the code was not checked

hook the check could not run

```
SETUP FAILURE: mypy is not
installed in the project
environment. The code was not
checked.
```

test the handler itself	for example	the handler
a file it should ignore	`notes.md`	**adds nothing**
a path with spaces	`src/my counts.py`	**checks it as usual**
broken source code	`def total(`	**reports a defect**
the checker missing	`no mypy installed`	**reports a setup failure**

See you in the next video.