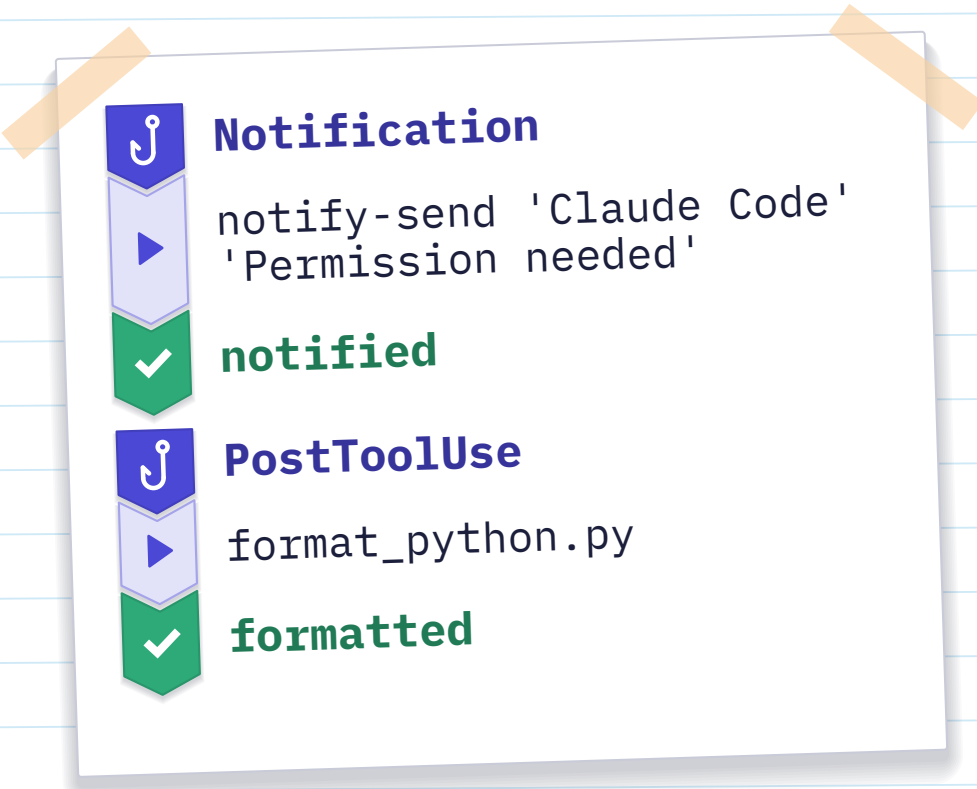
Prof. Aviv Zohar

Semester A, 2026–7

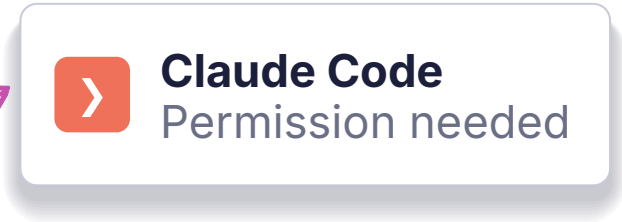
Your First Hook
Event to command

A notification, then a formatter



Start with a command that works

```shell
~/demo $ notify-send 'Claude Code' 'Permission needed'
```



By hand, in your shell: does it pop up?

Linux desktop `notify-send`, a standard command

WSL a small script from the agent: PowerShell shows a Windows notification

Mac a script from the agent, too

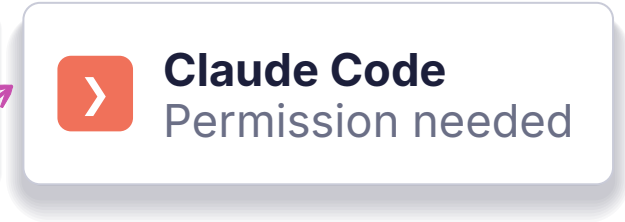
Check it on its own, outside the hook.

notification and prompt illustrative

Start with a command that works

shell

```
~/demo $ notify-send 'Claude Code' 'Permission needed'
```



on a permission notification, a hook starts it

It alerts you while another window is in front.

claude

```
❯ create notes.txt
● Bash(touch notes.txt)
Do you want to proceed?
❯ 1. Yes   2. No
```

notification and prompt illustrative

One entry: the event, a matcher, a handler

.claude/settings.json

```
{
  "hooks": {
    "Notification": [{
      "matcher": "permission_prompt",
      "hooks": [{
        "type": "command",
        "command": "notify-send 'Claude Code' 'Permission needed'"
      }]
    }]
  }
}
```

hooks holds all our hook configuration

event

when

to consider running: a notification

One entry: the event, a matcher, a handler

.claude/settings.json

```
{
  "hooks": {
    "Notification": [{
      "matcher": "permission_prompt",
      "hooks": [{
        "type": "command",
        "command": "notify-send 'Claude Code' 'Permission needed'"
      }]
    }]
  }
}
```

event
matcher

when
to consider running: a notification

which
occurrences match: permission prompts only

One entry: the event, a matcher, a handler

.claude/settings.json

```
{
  "hooks": {
    "Notification": [{
      "matcher": "permission_prompt",
      "hooks": [{
        "type": "command",
        "command": "notify-send 'Claude Code' 'Permission needed'"
      }]
    }]
  }
}
```

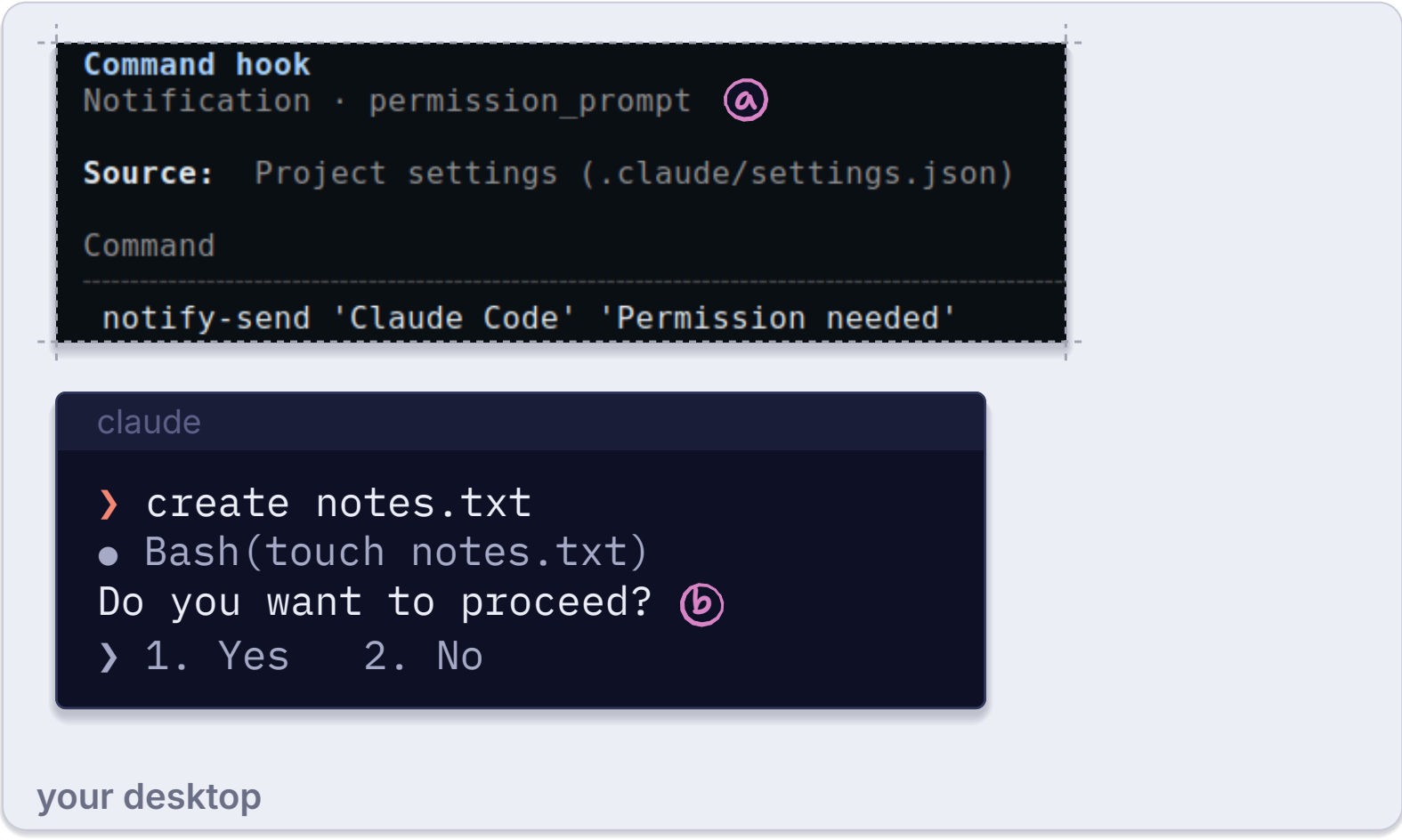
event
matcher
handler

when
to consider running: a notification

which
occurrences match: permission prompts only

what
to run: the command we tested by hand

Check that it is loaded, then trigger it

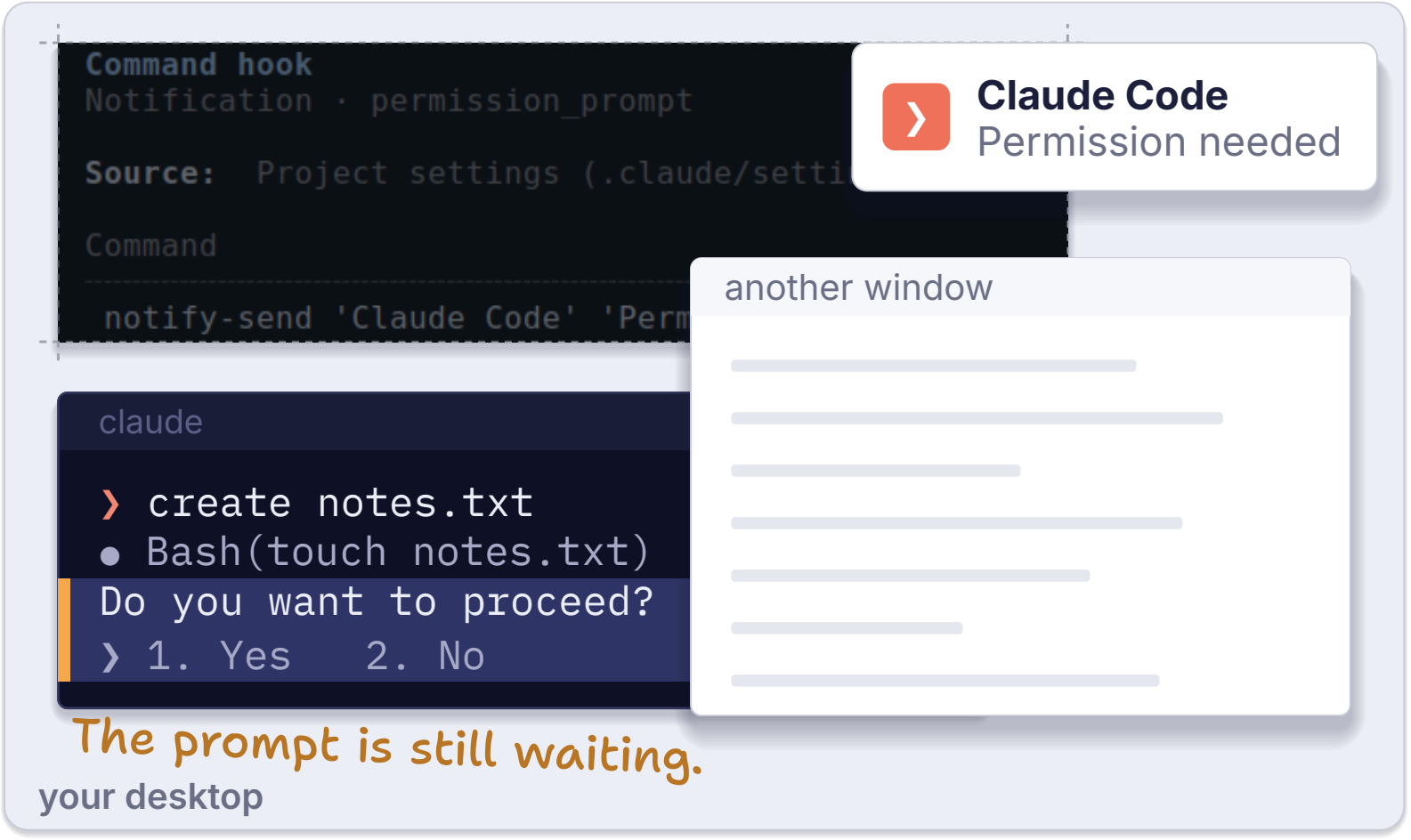


ⓐ /hooks: the hook is loaded

ⓑ a harmless action that needs permission

the /hooks screen captured, Claude Code 2.1.302 · the rest illustrative

Check that it is loaded, then trigger it



Notification

`notify-send …`

notified

An event matched, the harness started the command, and we saw its effect.

the /hooks screen captured, Claude Code 2.1.302 · the rest illustrative

The harness tells the hook what happened



stdin a PostToolUse event · selected fields

```
{
  "hook_event_name": "PostToolUse",
  "cwd": "/home/you/demo",
  "tool_name": "Edit",
  "tool_input": {
    "file_path": "/home/you/demo/src/counts.py"
  }
}
```

the event
the working directory
the tool
the tool's input

The harness tells the hook what happened



stdin a PostToolUse event · selected fields

```
{
  "hook_event_name": "PostToolUse",
  "cwd": "/home/you/demo",
  "tool_name": "Edit",
  "tool_input": {
    "file_path": "/home/you/demo/src/counts.py"
  }
}
```

the edited file: what our handler needs

Other events carry their own fields: see the docs, or ask the agent.

Register the formatter the same way

.claude/settings.json

```
{
  "hooks": {
    "PostToolUse": [{
      "matcher": "Edit|Write",
      "hooks": [{
        "type": "command",
        "command": "\"$CLAUDE_PROJECT_DIR/.venv/bin/
      }]
    }]
  }
}
```

`PostToolUse`: after a tool has succeeded

`Edit|Write`: the Edit and Write tools

The same structure as the notification hook, with other values.

Register the formatter the same way

.claude/settings.json

```
{
  "hooks": {
    "PostToolUse": [{
      "matcher": "Edit|Write",
      "hooks": [{
        "type": "command",
        "command": "\"$CLAUDE_PROJECT_DIR/.venv/bin/
      }]
    }]
  }
}
```

the command, as the shell receives it

```
"$CLAUDE_PROJECT_DIR/.venv/bin/python"
"$CLAUDE_PROJECT_DIR/.claude/hooks/format_python.py"
```

the project's Python

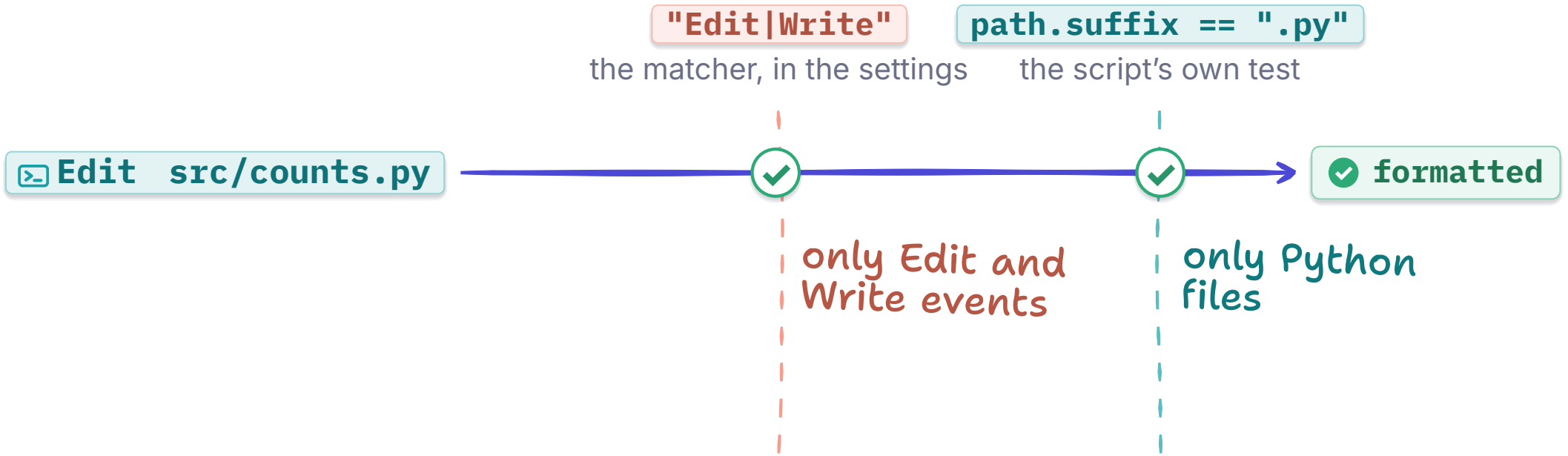
our handler script

`$CLAUDE_PROJECT_DIR`: the project directory

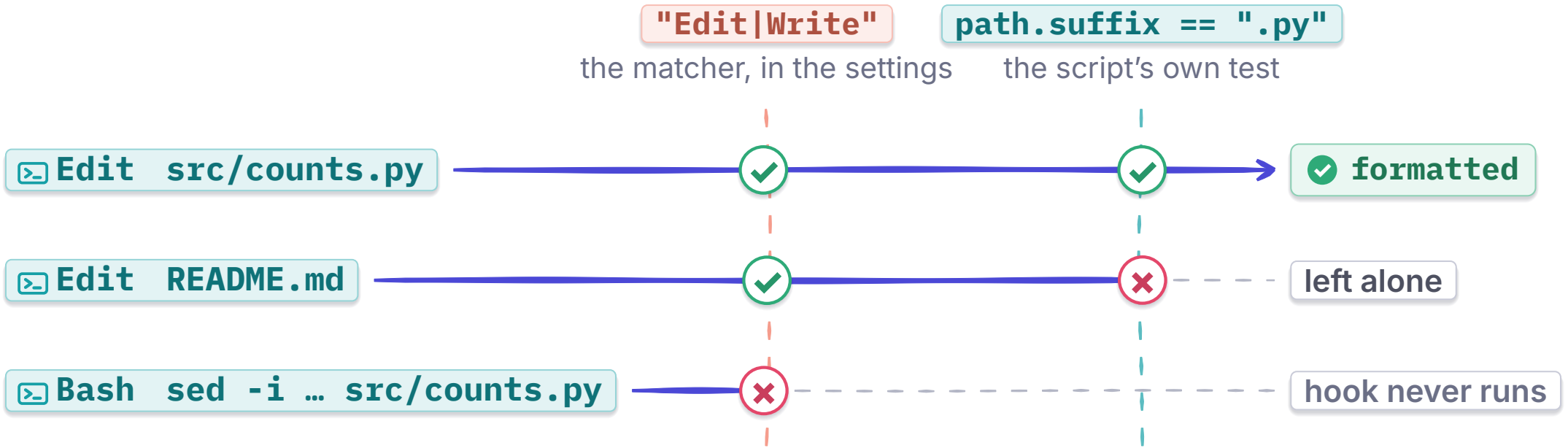
The " keep a path whole, even with a space.

In the settings file they are escaped: \"

The matcher picks the tool, the script picks the file



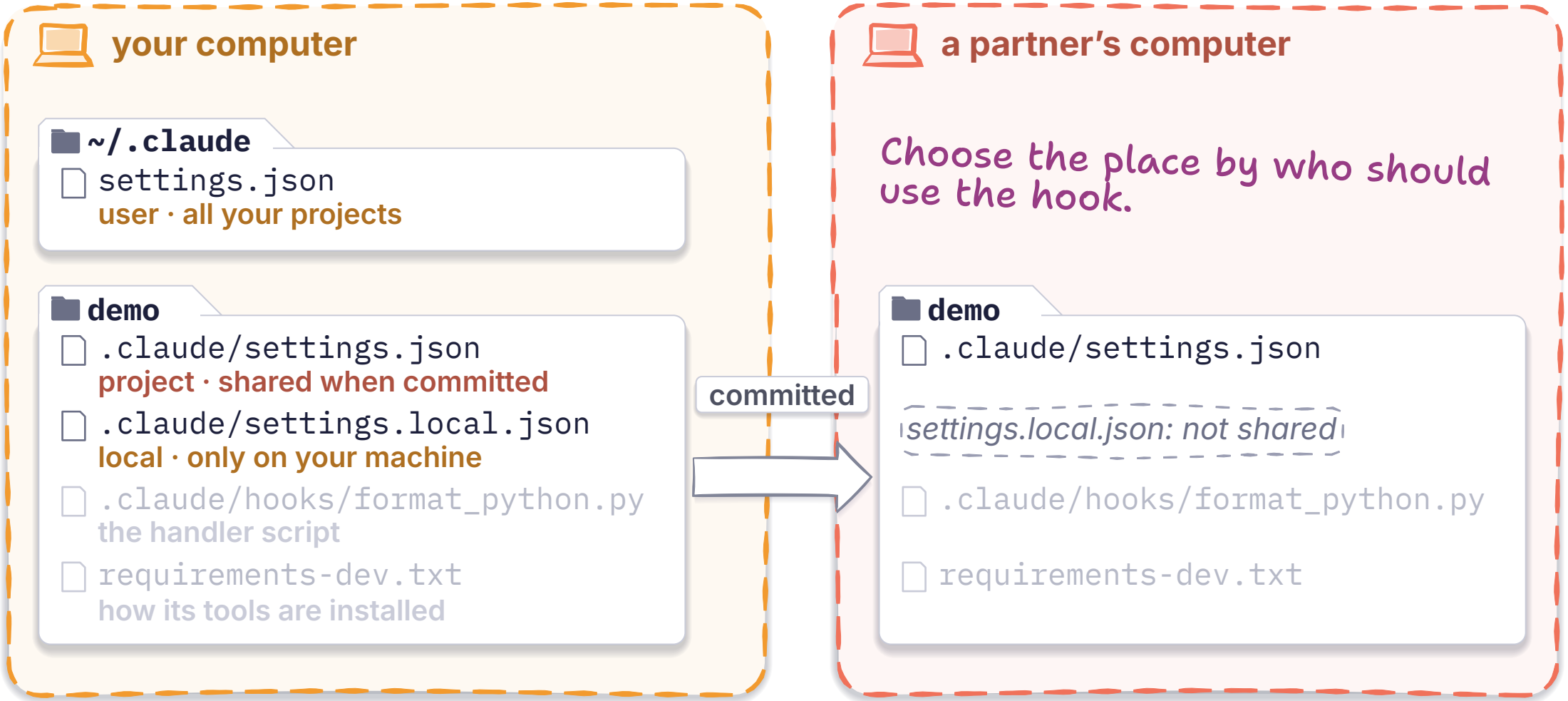
The matcher picks the tool, the script picks the file



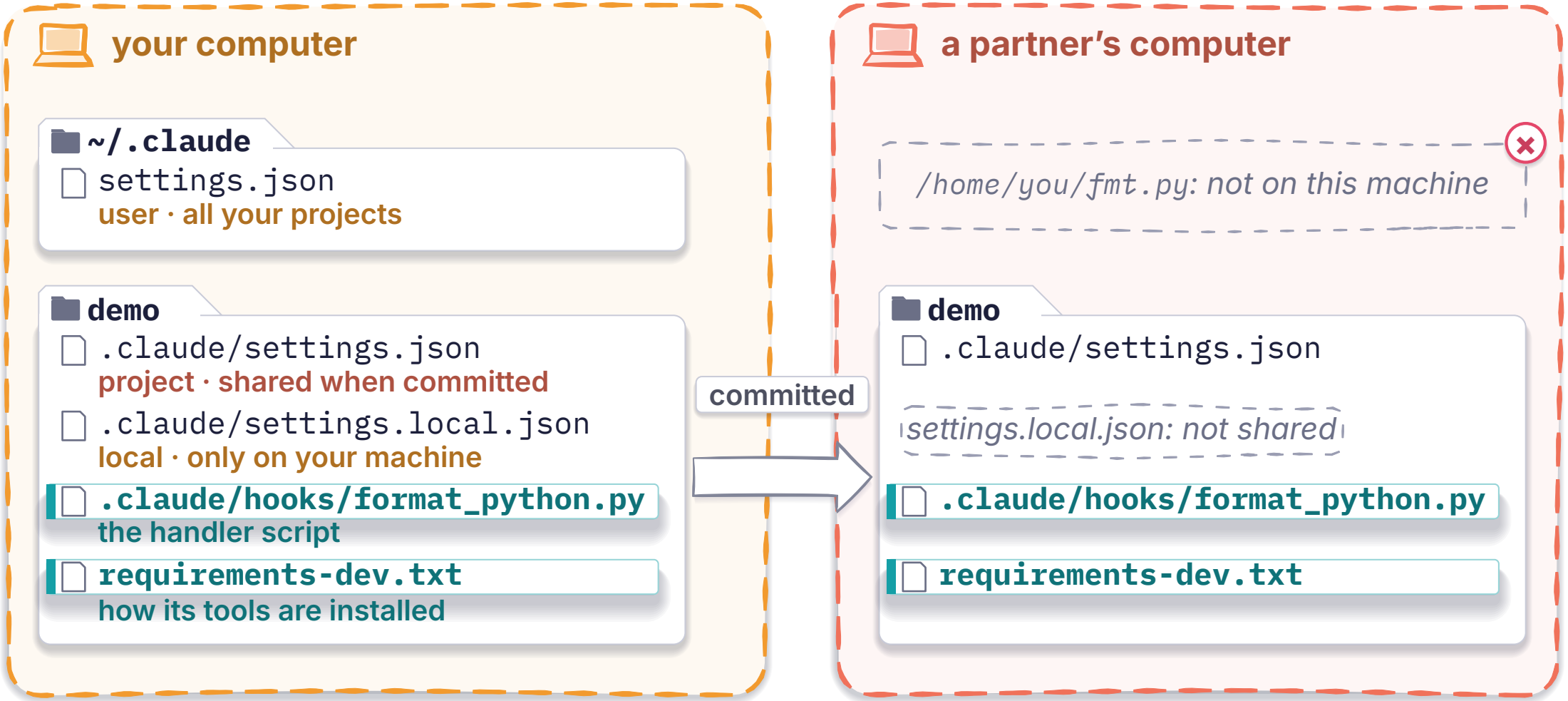
Agents also edit with Bash (sed, a small script): no Edit or Write event.

To cover it: also match Bash, and check for changed Python files.

Put the hook where its users are



Put the hook where its users are



Read the handler, then test it

.claude/hooks/format_python.py

```
 8 event = read_event()
 9 path = validated_project_path(event)
10 if path.suffix == ".py":
11     subprocess.run(
12         [sys.executable, "-m", "ruff", "format", str(path)],
13         check=True,
14     )
```

the event is parsed as JSON: data, never run

the path is one list item: no shell to split it

Test it with

`src/counts.py`	✅ formatted
`src/sales report.py`	✅ formatted: a space in the path
`README.md`	✅ left alone

It runs with your account's access. Read it before you enable it.

A command hook is a small program with a trigger



Start with a command that works

`notify-send …`, `format_python.py`: each run by hand first



Connect it to the intended event

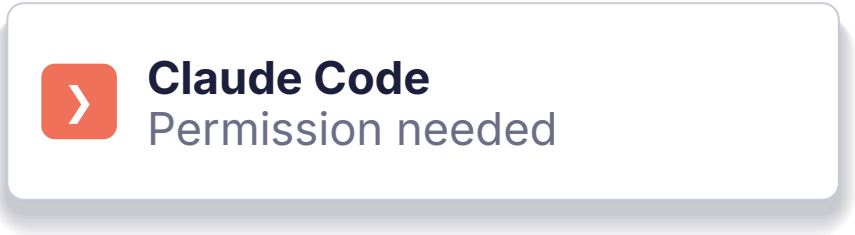
Notification with `permission_prompt`, PostToolUse with `Edit|Write`



Test the whole path

event, command, effect: an alert seen, a file formatted

Automation that runs at the right moment, without you starting it.



Go make a coffee.
The hook will call you back.

See you in the next video.