LESSON 01 · CLAUDE SKILLS

# Build a Claude scraping skill

No external APIs. Just Claude, Python and one file of instructions.

↓ The skill and all materials are linked below

WHAT WE'RE BUILDING

# Two pieces

THE SKILL · THE METHOD

```
# ~/.claude/skills/
scrape-public-pages/
└── SKILL.md
```

How to scrape any public page. The same for every site.

PER SITE · WHAT IT LEARNED

```
# your project
targets/<site>/
├── scrape.py
└── recipe.md
```

What worked on this site, so it is solved once.

STEP 1 · SAY WHAT YOU WANT

# Before any code, five answers

01
### URL
The page to scrape

02
### Fields
What one record holds

03
### Format
CSV or JSON

04
### Count
How many records you expect

05
### One value
Something you know is on the page

STEP 2 · THE LADDER

# Start simple. Climb only if you have to.

1

**`requests`**

Plain HTTP. Fast, no browser.

→

2

**`curl_cffi`**

Looks like a browser on the wire, without launching one.

→

3

**`Playwright`**

Only when JavaScript builds the data.

→

!

**Stop**

Login or CAPTCHA: ask the user.

STEP 3 · CHECK BEFORE "DONE"

# Check the data, not the status code

- ✓ The fields are found
- ✓ Enough records
- ✓ Nothing empty
- ✓ A few records look right
- ✓ The file is saved

BEFORE / AFTER · QUOTES.TOSCRAPE.COM/JS

# Same page. Different answer.

✗ A QUICK SCRIPT, NO CHECKS

```
requests → HTTP 200
look for div.quote → 0 quotes
"Done."
```

The quotes aren't in the HTML markup. JavaScript draws them later.

✓ CLAUDE WITH THE SKILL

```
run 1  requests → data in the page
       100 quotes · 0 missing · recipe saved
run 2  recipe → checks ✓ 100 quotes
```

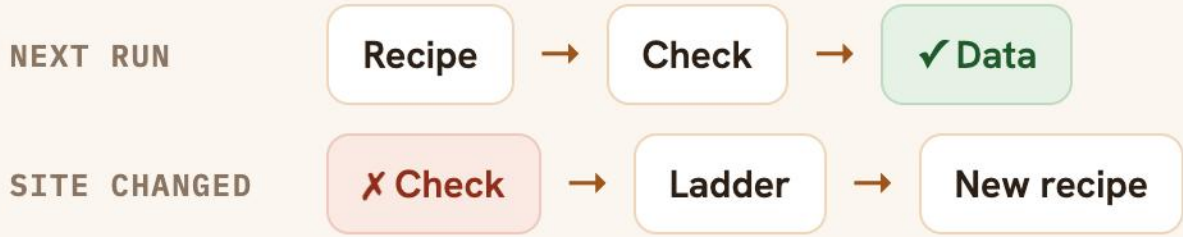
No browser needed: the check led it to the data, then it remembered how.

Real runs of Claude Code with this skill, on a public practice site for scraping.

STEP 4 · SAVE THE RECIPE

# Solve each site once

```
# targets/<site>/
scrape.py    # what worked
recipe.md    # method, selectors,
             # normal count
```



THE WHOLE SKILL · SKILL.MD

# One file, ready to use

```
---
name: scrape-public-pages
description: Build and verify small
  scrapers for permitted public pages,
  and save a reusable recipe per site.
---
## Known sites first
Recipe in targets/<host>/? Run it, check it.
A check fails? Back to the ladder.

## Before coding
Public page? robots.txt noted?
Login, CAPTCHA, denial: stop and ask.
```

```
## Use this order
requests → curl_cffi → Playwright → stop

## Verify after each attempt
Fields found · enough records · none empty
Samples look right · same schema twice
Saved file opened and counted

## Deliver
scraper · requirements.txt · data · command

## Save the recipe
targets/<host>/scrape.py + recipe.md
```

TAKE IT WITH YOU

# Everything is linked below

THE SKILL

### SKILL.md

Drop it in ~/.claude/skills/ and use it on any public page.

A REAL EXAMPLE

### targets/quotes.../

The recipe Claude saved: scrape.py and recipe.md.

THE LESSON

### Slides + script

Every step from this video, to read at your own pace.

Need Google results instead? **scrapebeaver.com** · clean JSON, try it free