

lab

April 3, 2026

1 Lesson 03 — Prompt Engineering Fundamentals

1.1 Lab Notebook

Work through Exercises 1 and 2 as described in `lab_guide.html`.

Prerequisites: `.env` file with `ANTHROPIC_API_KEY` in the project root.

```
[ ]: # Setup - run this cell first
import os
import json
from dotenv import load_dotenv
import anthropic

load_dotenv()
client = anthropic.Anthropic()
MODEL = "claude-3-5-haiku-20241022"

def call_claude(prompt: str, system: str = None) -> str:
    """Simple wrapper: send a prompt, return the text response."""
    kwargs = {
        "model": MODEL,
        "max_tokens": 1024,
        "messages": [{"role": "user", "content": prompt}]
    }
    if system:
        kwargs["system"] = system
    response = client.messages.create(**kwargs)
    return response.content[0].text

def compare_prompts(original: str, improved: str, label: str = "") -> None:
    """Call Claude with both prompts and print outputs side by side."""
    print(f"{'='*60}")
    if label:
        print(f" {label}")
    print(f"{'='*60}")
    print(f"\n--- ORIGINAL PROMPT ---\n{original}\n")
    original_out = call_claude(original)
    print(f"--- ORIGINAL OUTPUT ---\n{original_out}\n")
```

```

print(f"--- IMPROVED PROMPT ---\n{improved}\n")
improved_out = call_claude(improved)
print(f"--- IMPROVED OUTPUT ---\n{improved_out}\n")

print("Setup complete. Client ready.")

```

1.2 Exercise 1 — RTCF Rewrites

Five poor prompts are provided below. For each one: 1. Rewrite it using the RTCF framework (Role, Task, Context, Format) 2. Call `compare_prompts()` to see the improvement

See `lab_guide.html` for guidance on each prompt.

```

[ ]: # The 5 poor prompts - do not change these
POOR_PROMPTS = [
    "write something about climate",
    "summarize this",
    "be helpful",
    "fix my code",
    "what do you think about AI?",
]

print(f"Loaded {len(POOR_PROMPTS)} poor prompts.")

```

```

[ ]: # TODO: Rewrite each poor prompt using the RTCF framework.
# improved_prompts should be a list of 5 strings, one per poor prompt.
# Each improved prompt must include Role, Task, Context, and Format.

improved_prompts = None # TODO: rewrite each using RTCF; then call
    ↪ compare_prompts()

if improved_prompts is None:
    raise NotImplementedError("Complete this TODO")

```

```

[ ]: # Run this cell after completing the TODO above
for i, (poor, improved) in enumerate(zip(POOR_PROMPTS, improved_prompts),
    ↪ start=1):
    compare_prompts(poor, improved, label=f"Prompt {i}")

```

Reflection (fill in after running):

For each prompt, write one sentence explaining what changed and why it matters:

1. Prompt 1: *(your note here)*
2. Prompt 2: *(your note here)*
3. Prompt 3: *(your note here)*
4. Prompt 4: *(your note here)*
5. Prompt 5: *(your note here)*

1.3 Exercise 2 — Format Control

The text block below contains names and email addresses. Write **3 separate prompts** that extract the same data in three different formats: - Format A: JSON array - Format B: Markdown table - Format C: Plain bullet list

```
[ ]: # Input text - do not change
CONTACT_TEXT = """The project kickoff meeting was attended by Sarah Chen (sarah.
↪chen@techcorp.io),
lead engineer Marcus Webb at m.webb@devstudio.com, and the client contact
Priya Nair - her email is priya.nair@clientco.org. Also cc'd were Jordan Ellis
(jellis@techcorp.io) and the PM, Dana Okafor, reachable at d.okafor@pm.io."""

print("Contact text loaded.")
```

```
[ ]: # TODO: Write 3 format-control prompts.
# Each prompt must extract ALL 5 names and emails from CONTACT_TEXT.

json_prompt = None # TODO: prompt that produces [{"name":..., "email":...}, .
↪...]
table_prompt = None # TODO: prompt that produces a markdown table
list_prompt = None # TODO: prompt that produces a plain bullet list

if json_prompt is None or table_prompt is None or list_prompt is None:
    raise NotImplementedError("Complete this TODO")
```

```
[ ]: # Run this cell after completing the TODO above
print("=== FORMAT A: JSON ===")
json_output = call_claude(json_prompt)
print(json_output)

# Validate JSON parses correctly
try:
    parsed = json.loads(json_output)
    assert len(parsed) == 5, f"Expected 5 contacts, got {len(parsed)}"
    print(f"\nParsed successfully: {len(parsed)} contacts found.")
except (json.JSONDecodeError, AssertionError) as e:
    print(f"Validation failed: {e}")

print("\n=== FORMAT B: MARKDOWN TABLE ===")
print(call_claude(table_prompt))

print("\n=== FORMAT C: BULLET LIST ===")
print(call_claude(list_prompt))
```