The AI-Native Refactorer, Topic 1: Long Methods	Student Assignment: Refactor a Second Long Method on Your Own

Overview

In the hands-on, the instructor walked you through refactoring process_order with AI assistance. Now it's your turn to apply the same five-step workflow to a different long method, on your own.

Estimated time: 10 minutes.

WHAT YOU'LL PRACTICE

Driving the SCAN → PROTECT → EXTRACT → VERIFY loop without an instructor. The goal is not perfection — it's repetition. The second time through this workflow is when it starts to feel automatic instead of effortful.

The Code You'll Work With

Below is a 60-line function called generate_invoice_pdf_report. It mixes data retrieval, formatting, currency conversion, PDF generation stubs, and file naming — five responsibilities in one function. Copy it into a file called assignment1.py before you start.

```
def generate_invoice_pdf_report(invoice_id, db, currency_service, output_dir):
    # fetch invoice
    invoice = db.find_invoice(invoice_id)
    if invoice is None:
        raise ValueError(f"Invoice {invoice_id} not found")
    if invoice.status == "voided":
        raise ValueError(f"Invoice {invoice_id} is voided")

    # fetch customer
    customer = db.find_customer(invoice.customer_id)
    if customer is None:
        raise ValueError(f"Customer for invoice {invoice_id} missing")

    # currency conversion
    target_currency = customer.preferred_currency or "USD"
    if invoice.currency != target_currency:
        rate = currency_service.get_rate(invoice.currency, target_currency)
        converted_total = round(invoice.total * rate, 2)
        converted_lines = []
        for line in invoice.lines:
            converted_lines.append({
```

```
                "description": line.description,
                "amount": round(line.amount * rate, 2),
            })
    else:
        rate = 1.0
        converted_total = invoice.total
        converted_lines = [
            {"description": l.description, "amount": l.amount}
            for l in invoice.lines
        ]

    # format header
    header = (
        f"INVOICE {invoice.number}\n"
        f"Date: {invoice.date}\n"
        f"Customer: {customer.name}\n"
        f"Currency: {target_currency}\n"
    )

    # format body
    body_lines = []
    for line in converted_lines:
        body_lines.append(f"  {line['description']:30s} {line['amount']:>10.2f}")
    body = "\n".join(body_lines)

    # format footer
    footer = f"\n  {'TOTAL':30s} {converted_total:>10.2f} {target_currency}"

    # build full document
    document_text = header + "\n" + body + footer

    # write file
    filename = f"invoice_{invoice.number}_{target_currency}.pdf"
    path = f"{output_dir}/{filename}"
    with open(path, "w") as f:
        f.write(document_text)

    return path
```

Your Task

Apply the five-step workflow from the lecture. The deliverable is two files plus a short write-up. Follow these steps in order:

Step 1 — SCAN

Ask your AI assistant to identify the distinct responsibilities inside generate_invoice_pdf_report and name each one in four words or fewer. Write the list in your README.md.

Step 2 — PROTECT

Ask the AI to generate pytest characterization tests that cover the happy path, the "invoice not found" error, the "voided invoice" error, and the currency-conversion branch. Save them as assignment1_tests.py. Run them — they should all pass against the original code.

Step 3 — EXTRACT

Drive the AI through one Extract Function refactoring at a time. Use scope-limiting prompts like the ones from the hands-on. Run the test suite after each extraction. Stop when the top-level function is at most 10 lines and reads like a recipe.

Step 4 — VERIFY

All tests must still pass. Then ask the AI the final question: "Where would I add support for a new output format like HTML?" If you can answer in one sentence, you're done. If you can't, refactor further.

Deliverables

- assignment1.py — your refactored version of generate_invoice_pdf_report.
- assignment1_tests.py — the characterization tests, all passing.
- README.md — a short write-up (under 200 words) listing the responsibilities you identified in Step 1, the names of the helper functions you extracted in Step 3, and your answer to the "new output format" question from Step 4.

Self-Assessment Rubric

How to know you're done

✓ The top-level function is ≤ 10 lines.

✓ Every helper function has a single clear responsibility and a name that says so.

✓ All characterization tests pass.

✓ You can answer the "new output format" question in one sentence.

✓ No magic numbers or stringly-typed configuration left in the helpers (preview of Topic 3).

Common mistakes to watch for

✗ Refactoring before writing tests. If you change behavior, you won't know until it's too late.

✗ Asking the AI to "refactor the whole function" instead of one section at a time.

✗ Letting the AI rename your helpers to its own preferences without pushing back.

✗ Skipping the test run between extractions — small failures compound into big ones.