Assignment Redesign Example

Introduction to Python / Computer Science

From AI-Vulnerable to AI-Resistant

BEFORE: Vulnerable to AI

Task 1: The Four Fours Challenge

40 points; individual-only

To practice arithmetic expressions in Python, you will come up with four arithmetic expressions that will produce the integer values 1, 2, 3, and 4.

In this problem you will write a simple Python program that computes the integers 0 through 4 using expressions involving exactly four fours and no other numbers. For example:

```
zero = 4 + 4 - 4 - 4
```

Your expressions may use any of the following operators: `+, -, *, // (integer division), ** (power)`, and parentheses.

We have given you an example expression for computing the value 0. You should add in the code needed to compute the integers 1 through 4 using four fours. For each integer, you should assign the result of the computation to an appropriately named variable.

Notes:

- There are multiple ways to compute a given integer.
- We encourage you to try to come up with each of the expressions on your own. However, the real point of this problem is getting comfortable with Python, so if you have trouble coming up with an expression for a given number, feel free to ask a classmate or one of the course staff members, or to consult information about the four fours challenge that is available online.

Why This Is Vulnerable

- **Well-known puzzle** — The Four Fours challenge has been around for decades; AI (and Google) can produce correct solutions instantly.
- **Code-only submission** — If it runs and outputs the right numbers, it passes. No thinking required.
- **No explanation** — Students don't have to explain why their expressions work or what they learned.
- **Encourages outsourcing** — The assignment itself says "feel free to ask a classmate... or consult information available online."

AFTER: AI-Resistant Redesign

Task 1: The Four Fours Challenge + Reflection

40 points; individual-only

This assignment has two goals: (1) practice arithmetic expressions in Python, and (2) start building the habit of explaining your code — a skill that matters more than getting the right answer.

Part 1: Solve the Puzzle (20 points)

Write a Python program that computes the integers 0 through 4 using exactly four fours and no other numbers. You may use: `+, -, *, //, **,` and parentheses.

We've given you an example for 0:

```
zero = 4 + 4 - 4 - 4
```

Add expressions for 1 through 4.

Part 2: Explain Your Thinking (15 points)

In comments directly in your code, answer these questions for **two** of your expressions (your choice):

```
# Expression for [number]:
# 1. What was your first idea? Did it work?
#    If not, what went wrong?
# 2. Walk through the order of operations step
#    by step. Why does this produce the result?
# 3. What's one other way you could have written
#    this? Why did you choose the version you did?
```

Part 3: Predict and Test (5 points)

Before running your code, write down (in comments at the bottom of your file) what you expect to see in the console. Then run the code.

```
# PREDICTION: I expect to see...
# ACTUAL RESULT: [fill in after running]
# If different, what surprised you?
```

Submission:

- Your .py file with working expressions, Part 2 explanations, and Part 3 prediction/reflection
- Be prepared to walk through one of your expressions aloud in lab this week

What Changed and Why

Metacognition: "What was your first idea? Did it work?" asks them to trace their own problem-solving, not just produce an answer.
Process Over Product: Documenting what didn't work and why makes the thinking visible.
Explanation Required: Walking through order of operations proves they understand the code, not just that it runs.
Predict and Test: Requires engagement before seeing output — AI can't predict what the student expects.
Live Component: "Be prepared to walk through one expression aloud" — they need to actually understand what they submit.