The AI-Native Refactorer, Topic 3: Magic Numbers and Strings	Student Assignment: Name Every Magic Value at the Right Scope

Overview

In the hands-on, the instructor walked through a subscription service riddled with magic values. Now it's your turn — a different file, a different domain, but the same exercise: find every literal that doesn't carry its meaning, name it at the right scope, and prove the behavior didn't change.

Estimated time: 10 minutes.

WHAT YOU'LL PRACTICE

Distinguishing the four shapes of magic values from each other; choosing scope deliberately (function-local vs module vs settings); and using lookup tables to collapse parallel if-chains.

The Code You'll Work With

Below is a 60-line shipping calculator. It has at least eight magic numbers, four magic strings, one format string, and a hardcoded configuration value. Copy it into a file called assignment3.py.

```
from datetime import datetime, timedelta


class Shipment:
    def __init__(self, weight_kg, destination, priority, created_at):
        self.weight_kg = weight_kg
        self.destination = destination
        self.priority = priority
        self.created_at = created_at
        self.status = "pending"


def calculate_shipping_cost(shipment):
    # Base by zone
    if shipment.destination == "domestic":
        base = 5.00
    elif shipment.destination == "international":
        base = 25.00
    elif shipment.destination == "express":
        base = 15.00
```

```
    else:
        base = 50.00

    # Weight surcharge — $1.50 per kg over 2 kg
    if shipment.weight_kg > 2:
        base += (shipment.weight_kg - 2) * 1.50

    # Priority multiplier
    if shipment.priority == "overnight":
        base *= 2.5
    elif shipment.priority == "two_day":
        base *= 1.5

    # Tax
    base *= 1.08

    return round(base, 2)


def estimate_delivery_date(shipment):
    if shipment.priority == "overnight":
        delivery_days = 1
    elif shipment.priority == "two_day":
        delivery_days = 2
    elif shipment.destination == "international":
        delivery_days = 10
    else:
        delivery_days = 5

    delivery = shipment.created_at + timedelta(days=delivery_days)
    return delivery.strftime("%Y-%m-%d")


def can_ship_today(shipment):
    # Same-day cutoff is 3 PM
    now_hour = datetime.utcnow().hour
    if now_hour >= 15:
        return False
    if shipment.weight_kg > 30:  # over 30 kg requires special handling
        return False
    return True
```

Your Task

Apply the five-step workflow.

Step 1 — SCAN

Ask the AI to find every magic value, classified by shape (numeric, string, format, config). Have it suggest a name and scope for each.

Step 2 — PROTECT

Generate characterization tests covering the domestic/international/express base costs, the weight surcharge above 2 kg, the priority multipliers, the tax, the delivery-date logic for each priority, and the can-ship-today rules.

Step 3 — EXTRACT

Extract by shape:

- Numeric constants at module level: BASE_WEIGHT_KG, WEIGHT_SURCHARGE_PER_KG, TAX_RATE, SAME_DAY_CUTOFF_HOUR, SPECIAL_HANDLING_WEIGHT_KG.
- Enums: Destination (DOMESTIC/INTERNATIONAL/EXPRESS), Priority (OVERNIGHT/TWO_DAY/STANDARD), ShipmentStatus.
- Lookup tables: BASE_COSTS keyed by Destination, PRIORITY_MULTIPLIERS keyed by Priority, DELIVERY_DAYS that combines both.
- Format constant: DATE_FORMAT for the strftime call.

Step 4 — VERIFY

Run the tests after each extraction step. All must still pass. At the end, ask the AI to scan again and confirm no magic values remain.

Deliverables

- assignment3.py — your refactored shipping calculator with all magic values named.
- assignment3_tests.py — the characterization tests, all passing.
- README.md — under 200 words: list every magic value you found grouped by shape, the constant name and scope you chose, and your answer to one question: "Where would I add a new 'standard' destination zone with a $10 base cost?"

Self-Assessment Rubric

How to know you're done

✓ No numeric literals other than 0 and 1 appear in business logic.

✓ All destination and priority strings are replaced with Enums.

✓ Base costs and delivery days live in lookup tables, not if-chains.

✓ All characterization tests pass.

✓ Adding a new destination zone is a one-line config change.

Common mistakes to watch for

✗ Extracting 1 and 2 (kg threshold) — 2 is magic, 1 is the loop step. Be deliberate about which need names.

✗ Putting TAX_RATE in a function-local scope when it's used in multiple places.

✗ Replacing the if-chain with constants but leaving the if-chain itself — use a lookup table instead.

✗ Forgetting that 15 is an HOUR (not a percent or a count) — always confirm units before naming.