🚀 Problem Statement: Submarine Nuclear Warhead Launch System

Background

You are a software engineer for the **Naval Strategic Weapons Division (NSWD)**, responsible for developing a secure nuclear warhead launch system for **submarine-launched ballistic missiles (SLBMs)**. The system must be highly secure, modular, and log critical actions.

Your task is to implement the **Launch Authorization System** (LAS) in **Python**, following security protocols used in real-world naval warfare.

🔹 Problem Requirements

1. Classes & Objects

- **`Warhead`**: Represents a nuclear warhead.
- **`Submarine`**: Controls the SLBM launch system.
- **`LaunchAuthorizationSystem`**: Validates authorization codes and logs actions.

2. User-defined Module

- A module named `authorization.py` will handle launch authorization.

3. Regular Expressions

- Validate **encrypted authorization codes** using a regex pattern.
- Example valid code: `AUTH-XYZ123-4567-SECURE`

4. Logging (Console Only, No Files)

- Use Python's `logging` module to track key actions such as:
  - System initialization
  - Authorization validation
  - Launch attempts (successful/failed)

5. JSON Data

- Store **warhead data** in JSON format, representing different payloads.

🔹 Stubbed Code

🔹 Step 1: Create `authorization.py` (User-defined Module)

```
# authorization.py
import re
import json
import logging

# Set up logging (console output only)
```

```
logging.basicConfig(level=logging.INFO, format="%(asctime)s -
%(levelname)s - %(message)s")

class LaunchAuthorizationSystem:
    """Handles nuclear launch authorization validation."""

    AUTH_PATTERN = r"^AUTH-[A-Z0-9]{3,6}-\d{4}-SECURE$"  # Regex for
security code validation

    @staticmethod
    def validate_code(code):
        """Validates the launch authorization code."""
        if re.match(LaunchAuthorizationSystem.AUTH_PATTERN, code):
            logging.info("Authorization Code Validated Successfully!")
            return True
        else:
            logging.warning("Invalid Authorization Code!")
            return False
```

Step 2: Implement the Main System

```
# main.py
import json
import logging
from authorization import LaunchAuthorizationSystem

# Set up logging (console output only)
logging.basicConfig(level=logging.INFO, format="%(asctime)s -
%(levelname)s - %(message)s")

class Warhead:
    """Represents a nuclear warhead with specific payload
information."""

    def __init__(self, warhead_id, type, yield_kt):
        self.warhead_id = warhead_id
        self.type = type
        self.yield_kt = yield_kt  # Yield in kilotons

    def get_info(self):
        return f"Warhead {self.warhead_id}: Type {self.type}, Yield
{self.yield_kt}kt"

class Submarine:
    """Controls the nuclear missile launch sequence."""

    def __init__(self, name, warhead_data):
        self.name = name
        self.warheads = [Warhead(**w) for w in warhead_data]
```

```
    def authorize_launch(self, auth_code):
        """Attempts to authorize and launch a missile."""
        if LaunchAuthorizationSystem.validate_code(auth_code):
            logging.info(f"Launch authorized for {self.name}.
Preparing to launch SLBM...")
            self.launch_missile()
        else:
            logging.error("Launch Authorization Failed! Access
Denied.")

    def launch_missile(self):
        """Simulates launching a missile."""
        if self.warheads:
            warhead = self.warheads.pop(0)  # Fire the first available
warhead
            logging.info(f"🚀 Missile launched carrying
{warhead.get_info()}!")
        else:
            logging.warning("No warheads available for launch.")

# JSON Data (Simulating a warhead payload inventory)
warhead_json = '''
[
    {"warhead_id": "W001", "type": "Thermonuclear", "yield_kt": 1000},
    {"warhead_id": "W002", "type": "Tactical", "yield_kt": 300}
]
'''

# Load warhead data
warhead_data = json.loads(warhead_json)

# Initialize submarine
submarine = Submarine("USS Trident", warhead_data)

# 🚀 Try launching with an incorrect code
submarine.authorize_launch("INVALID-123")

# 🚀 Try launching with a valid code
submarine.authorize_launch("AUTH-XYZ123-4567-SECURE")
```

🔍 Hints to Solve the Problem

1. **User-defined Module (`authorization.py`)**
   - Ensure the regex pattern correctly matches valid authorization codes.
   - Test it using different code formats.
2. **Regular Expressions (`re` module)**
   - Modify the regex pattern if needed.
   - Test different invalid and valid codes.

3. **Logging (Console)**
   - Use different levels (`INFO`, `WARNING`, `ERROR`) to categorize messages.
   - Ensure launch attempts are logged.
4. **JSON Handling**
   - Ensure warhead data is loaded properly.
   - Modify or add more warheads to test different scenarios.
5. **Class-Based Implementation**
   - Ensure each class has a clear responsibility.
   - Implement additional checks before launch if needed.

Expected Output

```
2025-03-03 10:00:00 - WARNING - Invalid Authorization Code!
2025-03-03 10:00:00 - ERROR - Launch Authorization Failed! Access Denied.
2025-03-03 10:00:00 - INFO - Authorization Code Validated Successfully!
2025-03-03 10:00:00 - INFO - Launch authorized for USS Trident. Preparing to
launch SLBM...
2025-03-03 10:00:00 - INFO - Missile launched carrying Warhead W001: Type
Thermonuclear, Yield 1000kt!
```

Time Limit: 30 Minutes

Success Criteria

✓Implement **class-based structure** (OOP).
✓Use **a separate module** (`authorization.py`).
✓Validate codes using **Regular Expressions**.
✓Log all actions using **logging (console only, no files)**.
✓Load **JSON warhead data** and use it dynamically.

Bonus Challenge: Add a **self-destruct mechanism** if unauthorized launch attempts exceed **3 times**.