### 🚀 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**.