Grade D Python coding exercise

# Coding Exercise: Flight Data Processor

### Objective

Implement a robust Python class that processes flight data and demonstrates clean coding practices while incorporating unit testing.

### Task Description

You are given a list of flight data in JSON format. Each flight entry consists of several details. Your goal is to implement a FlightDataProcessor class with advanced features that include handling data transformations and deriving insights from the data.

1. **Attributes**:
   * flights: A list of dictionaries, where each dictionary represents a flight with the following keys:
     + flight\_number (string)
     + departure\_time (string in "YYYY-MM-DD HH:MM" format)
     + arrival\_time (string in "YYYY-MM-DD HH:MM" format)
     + duration\_minutes (integer)
     + status (enum, e.g. "ON\_TIME", "DELAYED", "CANCELLED")
2. **Methods**:
   * add\_flight(data: dict) -> None: Adds a new flight to the list, ensuring no duplicates (by flight number).
   * remove\_flight(flight\_number: str) -> None: Removes a flight based on the flight number.
   * flights\_by\_status(status: str) -> List[dict]: Returns all flights with a given status.
   * get\_longest\_flight() -> dict: Returns the flight with the longest duration in minutes.

### Requirements

* Implement the class with clean coding practices.
* Use Python's type hinting feature across all methods and method arguments.

### Unit Testing Requirements

* Use the unittest library to write comprehensive tests for all methods, covering a range of inputs and scenarios.
* Ensure the tests validate the correctness, performance, and stability of each method in the FlightDataProcessor class.

### Example Data

flight\_data = [

{"flight\_number": "AZ001", "departure\_time": "2025-02-19 15:30", "arrival\_time": "2025-02-20 03:45", "status": "ON\_TIME"},

{"flight\_number": "AZ002", "departure\_time": "2025-02-21 11:00", "arrival\_time": "2025-02-21 16:00", "status": "DELAYED"},

]

### Bonus Challenge

* Implement a method, update\_flight\_status(flight\_number: str, new\_status: str) -> None, that updates the status of a flight and ensure it reflects in the overall data.

### Instructions for Submission

1. Implement the FlightDataProcessor class in a script named flight\_data\_processor.py.
2. Write unit tests in a separate script test\_flight\_data\_processor.py.
3. Ensure tests run and pass without errors.
4. Submit both scripts for review. If you are conducting this test in an interview, demonstrate your code to your interviewer. If you are conducting this exercise prior to interview, please push it to your personal GitHub account and give the repo URL to your recruiter.