MSCS & MSDS OOP WITH PYTHON ASSIGNMENT 3, ADVENT 2025

Instruction: Submit a GitHub repository link and a well-documented Jupyter Notebook (.ipynb) file and/or .py file(s) on Canvas or Moodle. Ensure the notebook runs without errors and includes clear explanations, code, and outputs. Where APIs are simulated, include your JSON data samples. Ensure your code demonstrates OOP structure and error handling.

Q1. Create a class hierarchy that models vehicles registered in Uganda.

Attempt the following tasks:

1.1. Create a parent class Vehicle with attributes: plate_number, owner, engine_cc, base_tax.

1.2. Create subclasses:
- Car – adds passenger_capacity.
- Truck – adds load_capacity.
- Motorbike – adds type (e.g., boda, private).

1.3. Implement a method calculate_tax():
- Cars: base_tax + (engine_cc × 0.05)
- Trucks: base_tax + (load_capacity × 0.1)
- Motorbikes: base_tax + 20,000

1.4. Demonstrate polymorphism by calling calculate_tax() for all types.

1.5. Save all records to a JSON file named vehicle_registry.json.

Q2. Model a SACCO (Savings and Credit Cooperative) that evaluates members for loan approval.

Attempt the following tasks:

2.1. Create an abstract base class Member with: name, id_no, savings_balance.

2.2. Create subclasses:
- FarmerMember
- TraderMember

2.3. Implement loan_eligibility():
- Trader: 4 × savings balance
- Farmer: 6 × savings balance

2.4. Add static method get_exchange_rate() that retrieves USD rate using requests.get().

2.5. Display equivalent loan eligibility in UGX and USD.

2.6. Include exception handling for network/JSON errors.

Q3. Design a system that models traffic light management.

Attempt the following tasks:

3.1. Base class TrafficLight with turn_green(), turn_red(), and status().

3.2. Subclass SmartTrafficLight that:
- Connects to a simulated sensor API returning car density.
- Adjusts green/red duration based on car count.

3.3. Demonstrate encapsulation using private attribute __current_state.

3.4. Simulate multiple cycles printing logs like:

```
[09:15] Jinja Road: Green for 60s, Red for 30s.
```

Q4. Build a student grading system.

Requirements:

4.1. Base class Student with: student_id, name, marks (dictionary of subjects).

4.2. Methods:
- Compute total and average marks.
- Return grade (A ≥80, B ≥70, etc.).

4.3. Subclass PostgraduateStudent adds research_topic and evaluate_thesis() returning 'Pass' or 'Revise'.

4.4. Save and load data to/from JSON.

4.5. Use polymorphism to display coursework and thesis evaluations together.

Q5. Using the Spotify API (spotipy) or a mock JSON file of Ugandan artists:

Attempt the following tasks:

5.1. Create a class UgandaMusicAnalytics with methods to fetch top tracks of Ugandan artists (eg. Azawi, Sheebah etc.).

5.2. Use pandas to create a DataFrame of track names, play counts, and popularity.

5.3. Plot a Seaborn bar chart showing track popularity.

5.4. Save results to a JSON file.

5.5. Add error handling for invalid tokens and API limits.

5.6. Create a subclass LocalArtistAnalytics that filters tracks produced in Uganda only.

Q6. The Bank of Uganda provides JSON-based daily exchange rate data (try to locate this online) for USD, GBP, and EUR.

Attempt the following tasks:

6.1. Write a class ExchangeRatePipeline that fetches or loads data from local exchange_rates.json.

6.2. Clean and transform data using pandas.

6.3. Calculate monthly averages using numpy.

6.4. Plot exchange rate trends using Matplotlib.

6.5. Export processed data to both CSV and JSON.

6.6. Include exception handling for missing data or HTTP errors.