Pseudocode: Energy Management Simulation Program

---

1. Import Libraries

plaintext

Copy code

IMPORT libraries:

- random: For generating random weather patterns.
- datetime: For handling date and time.
- pandas: For managing data structures and analysis.
- tkinter: For creating graphical user interface (GUI).
- matplotlib: For data visualization.
- logging: For debugging and information tracking.
- platform, os: For system-level operations.

SWITCH matplotlib backend to 'TkAgg' for GUI integration.

---

2. Configure Logging

plaintext

Copy code

CONFIGURE logging:

- LEVEL: DEBUG for detailed tracking of all events.
- FORMAT: Include timestamp, log level, and message.
- HANDLER: StreamHandler to display logs in the console.

---

3. Define Constants and Initialize Data

plaintext

Copy code

DEFINE CONSTANTS:

- DEFAULT energy capacities for hydro, solar, wind, and battery.

- DEFAULT initial battery level, health, and grid electricity cost per kWh.

INITIALIZE:

- Lists `weather_data` and `energy_data` to store simulation results.

- Battery metrics: `battery_health`, `charge_cycles`, and `battery_age`.

- Grid metrics: `grid_usage`, `grid_usage_cost`, and `total_savings`.

---

4. Initialize Capacities

plaintext

Copy code

DEFINE FUNCTION `initialize_capacities`:

- CHECK if a configuration file (`config.py`) exists:

  - IF exists, IMPORT custom capacities from the file.

  - HANDLE ImportError or missing values, LOG a warning, and fall back to default values.

- ELSE, LOG a warning and USE default capacities.

- RETURN capacities as a tuple: Hydro, Solar, Wind, Battery, and Initial Battery Level.

CALL `initialize_capacities` and STORE the values in respective variables.

---

5. Simulate Weather Patterns

plaintext

Copy code

DEFINE FUNCTION `simulate_weather`:

- GET current date and time.
- SIMULATE temperature:
  - BASED on the current month, define seasonal ranges.
- SIMULATE wind speed:
  - HIGHER during the day (6:00–18:00).
  - LOWER during the night.
- SIMULATE solar radiation:
  - ONLY during the day (6:00–18:00), with random intensity.

LOG the generated weather patterns.

RETURN simulated temperature, wind speed, and solar radiation as a tuple.

---

6. Simulate Energy Usage

plaintext

Copy code

DEFINE FUNCTION `simulate_energy_usage`:

- INPUT: Current temperature (°C).
- DEFINE a base energy usage (e.g., 1000 kWh).
- ADD random variation to simulate daily fluctuations.
- INCREASE energy usage for extreme temperatures (below 0°C or above 30°C).
- LOG the calculated energy usage.
- RETURN energy usage (kWh).

---

7. Simulate Energy Generation

plaintext

Copy code

DEFINE FUNCTION `simulate_hydroelectricity`:

- CHECK if hydro power is active:
  - IF not active, RETURN 0 power generation.
- SIMULATE water flow rate.
- CALCULATE hydroelectric power:
  - Use water flow, efficiency, gravity, and hydro capacity.
- LIMIT hydro power output to the maximum capacity.
- LOG the simulated hydro power.
- RETURN hydroelectric power (kW).

DEFINE FUNCTION `simulate_solar_power`:

- INPUT: Solar radiation (W/m^2).
- CHECK if solar power is active:
  - IF not active, RETURN 0 power generation.
- CALCULATE solar power:
  - Use solar radiation, efficiency, and solar capacity.
- LIMIT solar power output to the maximum capacity.
- LOG the simulated solar power.
- RETURN solar power (kW).

DEFINE FUNCTION `simulate_wind_power`:

- INPUT: Wind speed (m/s).
- CHECK if wind power is active:
  - IF not active, RETURN 0 power generation.
- CALCULATE wind power:

- Use wind speed, efficiency, and wind capacity.

- LIMIT wind power output to the maximum capacity.

- LOG the simulated wind power.

- RETURN wind power (kW).

---

8. Update Simulation Data

plaintext

Copy code

DEFINE FUNCTION `update_data`:

- GLOBAL VARIABLES:

  - Weather and energy data lists.

  - Battery level, health, age, and charge cycles.

  - Grid usage, grid cost, and total savings.

- SIMULATE:

  - Weather (temperature, wind speed, solar radiation).

  - Energy usage based on temperature.

  - Power generation for hydro, solar, and wind.

- CALCULATE:

  - Total energy generation.

  - Net energy balance (generation - usage).

- UPDATE Battery and Grid:

  - IF battery is active:

    - IF surplus energy and battery isn't full:

        - Charge battery based on health and available capacity.

        - INCREMENT charge cycles if fully charged.

      - IF energy deficit:

        - Discharge battery based on health and capacity.

    - UPDATE grid usage for unmet energy demands.

  - UPDATE Metrics:

    - LOG the updated values.

    - Simulate battery degradation over time.

  - APPEND results to data lists.

  - REMOVE old entries if lists exceed 100 items.

---

9. GUI and Visualization

plaintext

Copy code

SET UP GUI:

  - CREATE main window using `tkinter`.

  - CONFIGURE grid layout and embed a `matplotlib` figure for visualization.

  - ADD scrollbars to support dynamic content.

  - EMBED frames for buttons and renewable energy controls.

ADD CONTROLS:

  - Buttons for toggling energy sources (hydro, solar, wind, battery).

  - Start/stop buttons for controlling animations.

  - Text fields to display real-time simulation data.

INTEGRATE MATPLOTLIB ANIMATION:

- DEFINE `animate` function to refresh graphs.
- LINK it to `FuncAnimation` for real-time updates.

START MAIN EVENT LOOP:

- CALL `root.mainloop` to keep the GUI responsive.