**Python Activity 19: Understanding and Applying Liskov Substitution Principle (LSP)**

**Objective:**

In this activity, you will analyze a Python class hierarchy and identify violations of the **Liskov Substitution Principle (LSP)**. You will then refactor the code to ensure that both parent and subclass objects can be used interchangeably without breaking the program's behavior.

**Instructions:**

1. **Review the Code Snippet:**

**Code Snippet:**

class Car:

def \_\_init\_\_(self, fuel: int):

self.fuel = fuel

def drive(self):

if self.fuel > 0:

self.fuel -= 1

print("Car is driving!")

else:

raise Exception("Not enough fuel to drive!")

def refuel(self, amount: int):

self.fuel += amount

print(f"Car refueled by {amount}.")

class ElectricCar(Car):

def \_\_init\_\_(self, battery: int):

self.fuel = battery # Incorrectly reusing 'fuel' for the battery

self.battery = battery

def drive(self):

if self.battery > 0:

self.battery -= 1

print("Electric car is driving!")

else:

raise Exception("Not enough battery to drive!")

def refuel(self, amount: int):

raise NotImplementedError("Electric cars can't refuel with fuel!")

def recharge(self, amount: int):

self.battery += amount

print(f"Electric car recharged by {amount}.")

1. **Task 1 - Identify the LSP Violation:**
   * Carefully review the code above and **identify the LSP violation**.
2. **Task 2 - Refactor the Code to Fix the LSP Violation:**

After identifying the violation, **refactor the code** to ensure it can be used interchangeably **without violating the LSP**.

1. **Submission:**
   * Once you’ve completed the tasks, **save your file** as python\_activity19.py.
   * Submit the following:
     + **An analysis of the LSP violation** (why the original code violates LSP).
     + **Your refactored code** that respects LSP.