OOPs Practice Tasks - Classes, Objects & Inheritance

# Task 1: Class and Object - Basic

Task:  
Create a class called `Car` with attributes: `brand`, `model`, and `year`. Create an object of this class and display its attributes.  
  
Hint:  
Use the `\_\_init\_\_` method to set attributes. Use `print(object.attribute)` to display them.

# Task 2: Class Method

Task:  
Add a method `start\_engine()` to the `Car` class that prints "Engine started". Call the method using the object.  
  
Hint:  
Define a method using `def start\_engine(self):` and use `object.start\_engine()` to call it.

# Task 3: Constructor (Init method)

Task:  
Modify the `Car` class to use a constructor (`\_\_init\_\_`) to initialize the attributes when creating an object.  
  
Hint:  
Use `def \_\_init\_\_(self, brand, model, year):` inside the class.

# Task 4: Inheritance - Simple

Task:  
Create a base class `Animal` with a method `make\_sound()`. Create a derived class `Dog` that inherits from `Animal` and overrides `make\_sound()` to print "Bark".  
  
Hint:  
Use `class Dog(Animal):` to inherit. Define a new `make\_sound()` method in `Dog` to override.

# Task 5: Inheritance - Multiple Derived Classes

Task:  
Create another derived class `Cat` from `Animal` and override `make\_sound()` to print "Meow". Create objects of `Dog` and `Cat` and call their `make\_sound()` methods.  
  
Hint:  
Each class can have its own version of `make\_sound()` even if they inherit from the same parent.

# Task 6: Class with Method to Update Attribute

Task:  
Add a method `update\_year()` in the `Car` class to change the `year` attribute.  
  
Hint:  
Use a method like `def update\_year(self, new\_year):` to set `self.year = new\_year`.

# Task 7: Object Comparison (Simple Logic)

Task:  
Create two objects of `Car` and write a method `is\_newer()` to compare their `year` values.  
  
Hint:  
Use an `if` statement inside `is\_newer(self, other\_car)` to compare `self.year` and `other\_car.year`.

# Task 8: Use of `super()`

Task:  
Create a class `ElectricCar` that inherits from `Car`. Use `super()` to call the constructor of `Car` and add a new attribute `battery\_capacity`.  
  
Hint:  
In the constructor of `ElectricCar`, use `super().\_\_init\_\_(brand, model, year)` to reuse parent class constructor.

# Task 9: Method Overriding

Task:  
In `ElectricCar`, override the `start\_engine()` method to print "Electric engine started".  
  
Hint:  
Just define a method with the same name in the child class. It will replace the parent class version.

# Task 10: Class with a List of Objects

Task:  
Create a class `Garage` that contains a list of `Car` objects. Add a method to display all car brands in the garage.  
  
Hint:  
Use a list like `self.cars = []` and loop through it in a method using `for car in self.cars:`.