Types of Inheritance

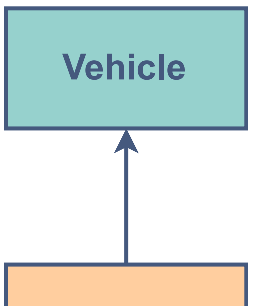
In this lesson, you will learn about the various types of inheritance in Python.

Based upon parent classes and child classes, there are the following **five** types of inheritance:

1. **Single**
2. **Multi-level**
3. **Hierarchical**
4. **Multiple**
5. **Hybrid**

Single Inheritance

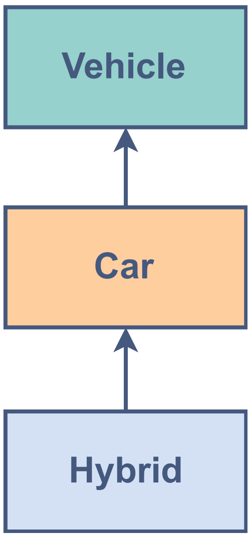
In single inheritance, there is only a single class extending from another class. We can take the example of the `Vehicle` class, as the parent class, and the `Car` class, as the child class. Let's implement these classes below:



```
class Vehicle:  # parent class
    def setTopSpeed(self, speed):  # defining the set
        self.topSpeed = speed
        print("Top speed is set to", self.topSpeed)


class Car(Vehicle):  # child class
    def openTrunk(self):
        print("Trunk is now open.")


corolla = Car()  # creating an object of the Car class
corolla.setTopSpeed(220)  # accessing methods from the parent class
corolla.openTrunk()  # accessing method from its own class
```

Multi-level Inheritance

When a class is derived from a class which itself is derived from another class, it’s called Multilevel Inheritance. We can extend the classes to as many levels as we want to.



Let’s implement the three classes illustrated above:

- A `Car` **IS A** `Vehicle`
- A `Hybrid` **IS A** `Car`

```
class Vehicle:  # parent class
    def setTopSpeed(self, speed):  # defining the set
        self.topSpeed = speed
        print("Top speed is set to", self.topSpeed)
```

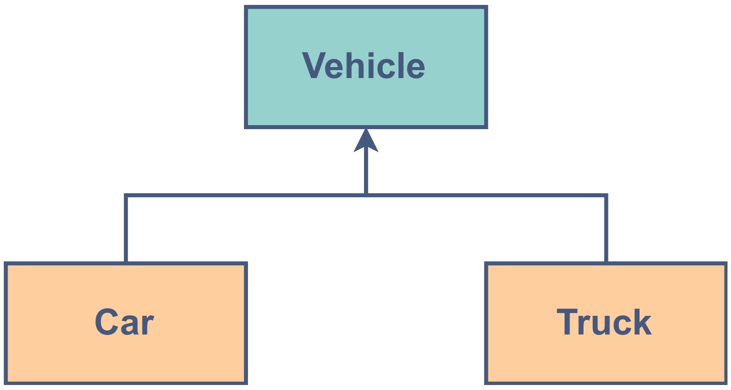
```
class Car(Vehicle):  # child class of Vehicle

    def openTrunk(self):
        print("Trunk is now open.")


class Hybrid(Car):  # child class of Car
    def turnOnHybrid(self):
        print("Hybrid mode is now switched on.")


priusPrime = Hybrid()  # creating an object of the Prius class
priusPrime.setTopSpeed(220)  # accessing methods from the parent class
priusPrime.openTrunk()  # accessing method from the parent class
priusPrime.turnOnHybrid()  # accessing method from the parent class
```

Hierarchical Inheritance

When more than one class inherits from the same class, it’s referred to as hierarchical inheritance. In hierarchical inheritance, more than one class extends, as per the requirement of the design, from the same base class. The common attributes of these child classes are implemented inside the base class.

Example:

- A `Car` **IS A** `Vehicle`
- A `Truck` **IS A** `Vehicle`



Below is a code example of hierarchal inheritance.

```
class Vehicle:  # parent class
    def setTopSpeed(self, speed):  # defining the set
        self.topSpeed = speed
        print("Top speed is set to", self.topSpeed)
```

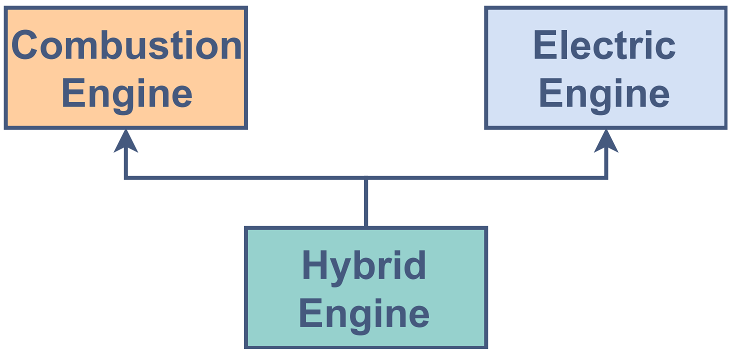
```
class Car(Vehicle):  # child class of Vehicle
    pass


class Truck(Vehicle):  # child class of Car
    pass


corolla = Car()  # creating an object of the Prius class
corolla.setTopSpeed(220)  # accessing methods from the parent class

volvo = Car()  # creating an object of the Prius class
volvo.setTopSpeed(180)  # accessing methods from the parent class
```

Multiple Inheritance

When a class is derived from more than one base class, i.e., when a class has more than one immediate parent class, it is called Multiple Inheritance.

Example:

- `HybridEngine` **IS A** `ElectricEngine`.
- `HybridEngine` **IS A** `CombustionEngine` as well.



Below is a code example of multiple inheritance.

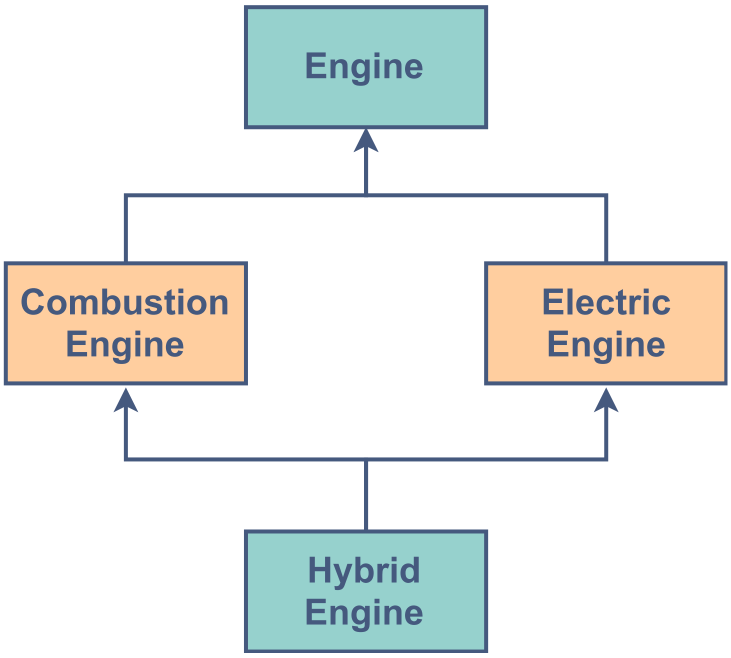
```
class CombustionEngine():  # Child class inherited from Engine
    def setTankCapacity(self, tankCapacity):
        self.tankCapacity = tankCapacity


class ElectricEngine():  # Child class inherited from Engine
    def setChargeCapacity(self, chargeCapacity):
        self.chargeCapacity = chargeCapacity

# Child class inherited from CombustionEngine and ElectricEngine
```

```
class HybirdEngine(CombustionEngine, ElectricEngine):
    def printDetails(self):
        print("Tank Capacity:", self.tankCapacity)
        print("Charge Capacity:", self.chargeCapacity)


car = HybirdEngine()
car.setChargeCapacity("250 W")
car.setTankCapacity("20 Litres")
car.printDetails()
```

Hybrid Inheritance

A type of inheritance which is a combination of **Multiple** and **Multi-level** inheritance is called *hybrid inheritance*.

- `CombustionEngine` **IS A** `Engine`.
- `ElectricEngine` **IS A** `Engine`.
- `HybridEngine` **IS A** `ElectricEngine` and a `CombustionEngine`.



Below is the code implementation of an example of Hybrid inheritance.

```
class Engine:  # Parent class
    def setPower(self, power):
        self.power = power


class CombustionEngine(Engine):  # Child class inherited from Engine
    def setTankCapacity(self, tankCapacity):
        self.tankCapacity = tankCapacity


class ElectricEngine(Engine):  # Child class inherited from Engine
    def setChargeCapacity(self, chargeCapacity):
        self.chargeCapacity = chargeCapacity

# Child class inherited from CombustionEngine and ElectricEngine


class HybirdEngine(CombustionEngine, ElectricEngine):
    def printDetails(self):
        print("Power:", self.power)
        print("Tank Capacity:", self.tankCapacity)
        print("Charge Capacity:", self.chargeCapacity)


car = HybirdEngine()
car.setPower("2000 CC")
car.setChargeCapacity("250 W")
car.setTankCapacity("20 Litres")
car.printDetails()
```

This lesson was about different types of inheritance. In the next lesson, we’ll discuss the advantages of inheritance.