Inheritance in Python (OOP)

Inheritance: Behaviors and Characteristics

Certain objects have some things in common: their behavior and characteristics.

For example, I inherited some characteristics and behaviors from my father. I inherited his eyes and hair as characteristics, and his impatience and introversion as behaviors.

In object-oriented programming, classes can inherit common characteristics (data) and behavior (methods) from another class.

Let's see another example and implement it in Python.

Imagine a car. Number of wheels, seating capacity and maximum velocity are all attributes of a car. We can say that an **ElectricCar** class inherits these same attributes from the regular **Car** class.

```
class Car:
   def __init__(self, number_of_wheels, seating_capacity,
maximum_velocity):
       self.number_of_wheels = number_of_wheels
       self.seating_capacity = seating_capacity
       self.maximum_velocity = maximum_velocity
```

Our **Car** class implemented:

```
my_car = Car(4, 5, 250)
print(my_car.number_of_wheels)
print(my_car.seating_capacity)
print(my_car.maximum_velocity)
```

Once initiated, we can use all instance variables created. Nice.

In Python, we apply a parent class to the child class as a parameter. An **ElectricCar** class can inherit from our **Car** class.

```
class ElectricCar(Car):
   def __init__(self, number_of_wheels, seating_capacity,
maximum_velocity):
       Car.__init__(self, number_of_wheels, seating_capacity,
maximum_velocity)
```

Simple as that. We don't need to implement any other method, because this class already has it (inherited from **Car** class). Let's prove it:

```
my_electric_car = ElectricCar(4, 5, 250)
print(my_electric_car.number_of_wheels) # => 4
print(my_electric_car.seating_capacity) # => 5
print(my_electric_car.maximum_velocity) # => 250
```

That's it!

We learnt a lot of things about Python object oriented programming:

- Encapsulation: hiding information
- Inheritance: behaviors and characteristics