Object-Oriented Programming

Classes and Instances

Gia Khang Huỳnh

February 2026

0.1 Classes:

In programming, when we start upscaling our variables, we begin to define their **attributes** and **behaviors**. For example we want to describe the main features of a car, it consists of a brand, the model , the year it is manufactured.

Usually in C , a struct or a class contains only the data itself. But when transtioned to a more powerful language such as Python, a class could contain both the data and the methods of an object, how does this work?

Class in Python:

In Python, a Class is a blueprint for creating objects. It creates a new "type" where you can bundle data (attributes) and the functionality (methods) that acts on that data together. (Defined by calling: "class object: ")

Instance attributes:

Associated with classes, it is specific object created from the blueprint that holds actual data in memory. You can have thousands of instances generated from a single class, and each one is independent.

Example:

To clarify the previous theories, we are going to use the exact example mentioned above: a car consisting of a few features (Instances):

- Car named A with a brand like Lamborghini and manufactured in 2024
- Car named B with a brand like Honda and manufactured in 2017

After described a car , we are going to add a few methods to these cars like a car sound (Vroom vroom)

```
1  class car:
2      # This acts like a constructor. It sets up the initial state.
3      def __init__ (self, name ,brand , year): #Self is referencing the data
       ↪  itself
4
5          self.name = name #setting up the name unique to the instance
6          self.brand = brand
7          self.year = year
8      def sound(self): #Defining a method within the class
9          return "Vroom Vroom!"
10 #Applying the instances:
11 carA = car('A', "Lamborghini" , 2024)
12 carB = car('B', "Honda", 2017)
13
14 #Accessing a certain data:
15
```

```python
print(f"Car A is named {carA.name}, with the brand {carA.brand} and manufactured in {carA.year}")
print (f"Car B is named {carB.name}, with the brand {carB.brand} and manufactured in {carB.year}" )
#Both share the same class definition but holds different datas

#Implementation of the methods within the class:
print(carA.sound())


```

Output:

```
Car A is named A, with the brand Lamborghini and manufactured in 2024
Car B is named B, with the brand Honda and manufactured in 2017
Vroom Vroom!
```

Figure 1: Output of the Car Class execution

Notices: This is only a brief introduction to classes and there is a few functions included in the example code that these theories cannot fully cover like "self" and "__init __", these functions will be properly introduced in the next articles.