OOPs

.class

.objects

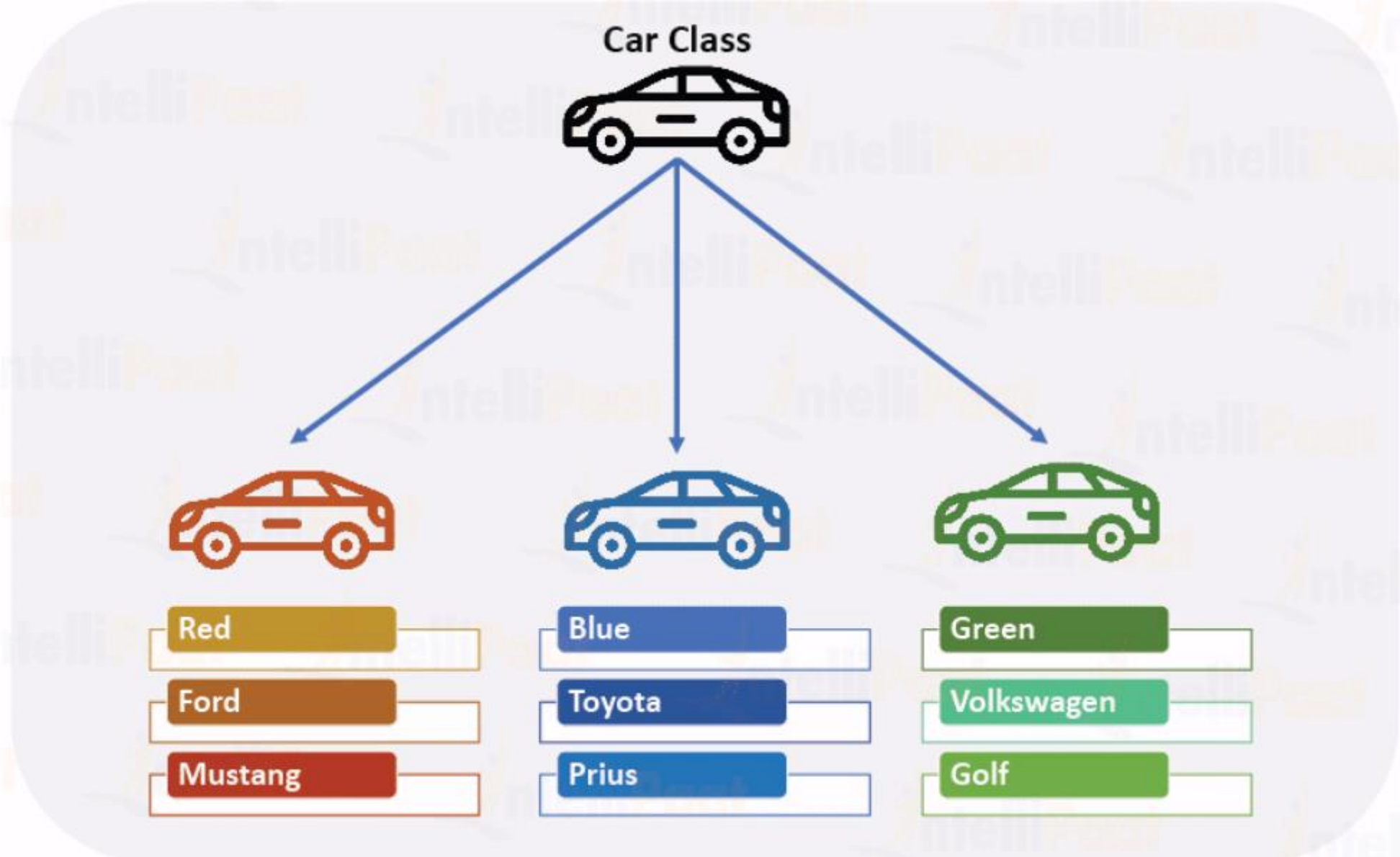
What is a Class and Objects in Python?

- **Class**: The class is a user-defined data structure that binds the data members and methods into a single unit. Class is a **blueprint or code template for object creation**. Using a class, you can create as many objects as you want.
- **Object**: An **object is an instance of a class**. It is a collection of attributes (variables) and methods. We use the object of a class to perform actions.

Objects have two characteristics: They have states and behaviors (object has attributes and methods attached to it) Attributes represent its state, and methods represent its behavior. Using its methods, we can modify its state.

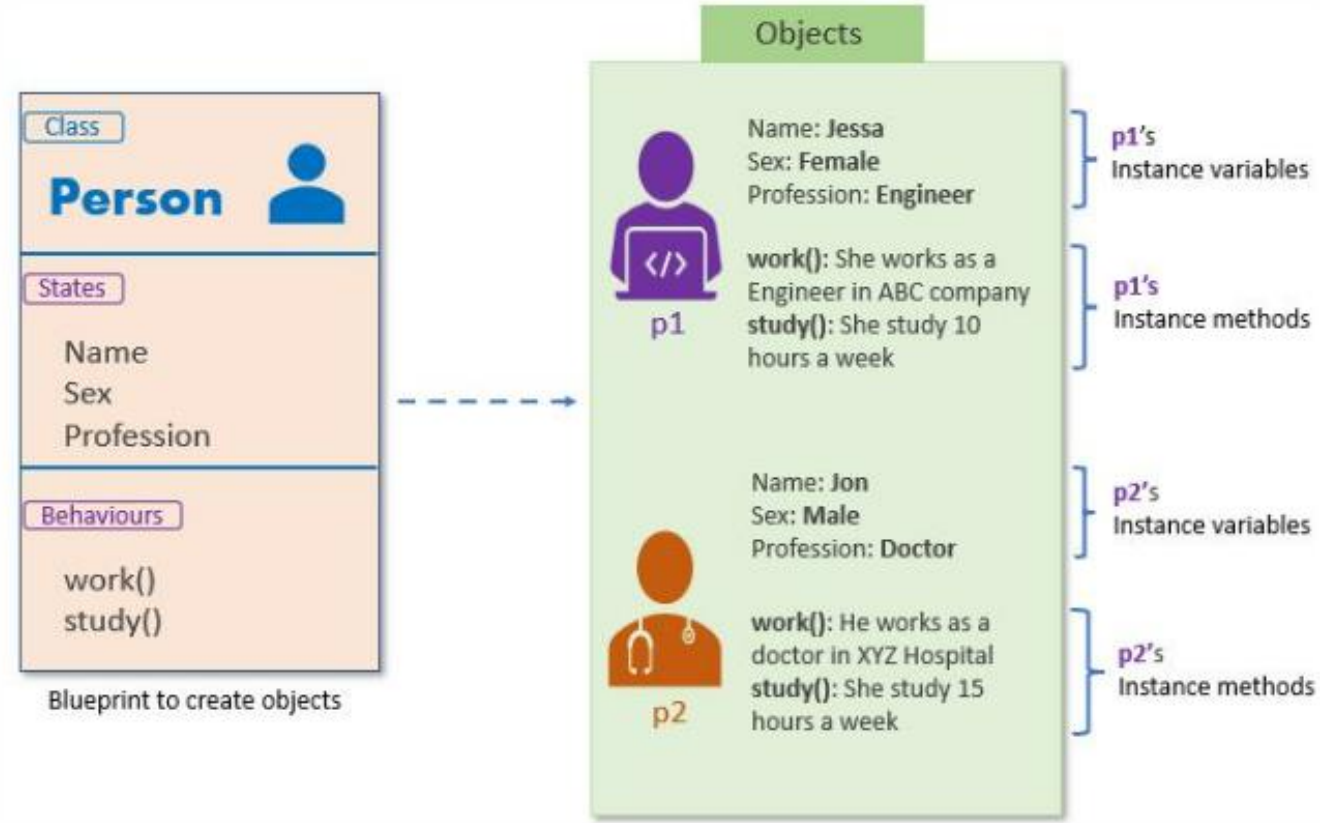
In short, Every object has the following property.

- **Identity**: Every object must be uniquely identified.
- **State**: An object has an attribute that represents a state of an object, and it also reflects the property of an object.
- **Behavior**: An object has methods that represent its behavior.


Car Class
Red
Ford
Mustang
Blue
Toyota
Prius
Green
Volkswagen
Golf

Python is an Object-Oriented Programming language, so everything in Python is treated as an object. An object is a real-life entity. It is the collection of various data and functions that operate on those data.

For example, If we design a class based on the states and behaviors of a Person, then States can be represented as instance variables and behaviors as class methods.



understand class and objects in Python

A real-life example of class and objects.

Class: Person

- **State**: Name, Sex, Profession
- **Behavior**: Working, Study

Using the above class, we can create multiple objects that depict different states and behavior.

Object 1: Jessa

- **State**:
  - Name: Jessa
  - Sex: Female
  - Profession: Software Engineer
- **Behavior**:
  - Working: She is working as a software developer at ABC Company
  - Study: She studies 2 hours a day

Object 2: Jon

- **State**:
  - Name: Jon
  - Sex: Male
  - Profession: Doctor
- **Behavior**:
  - Working: He is working as a doctor
  - Study: He studies 5 hours a day

As you can see, Jessa is female, and she works as a Software engineer. On the other hand, Jon is a male, and he is a lawyer. Here, both **objects are created from the same class, but they have different states and behaviors**.

Syntax

```
class class_name:
    '''This is a docstring. I have created a new class'''
    <statement 1>
    <statement 2>
    .
    .
    <statement N>
```

- `class_name` : It is the name of the class
- `Docstring` : It is the first string inside the class and has a brief description of the class. Although not mandatory, this is highly recommended.
- `statements` : Attributes and methods

Example: Define a class in Python

In this example, we are creating a Person Class with name, sex, and profession instance variables.

```
class Person:
    def __init__(self, name, sex, profession):
        # data members (instance variables)
        self.name = name
        self.sex = sex
        self.profession = profession

    # Behavior (instance methods)
    def show(self):
        print('Name:', self.name, 'Sex:', self.sex, 'Profession:', self.profession)

    # Behavior (instance methods)
    def work(self):
        print(self.name, 'working as a', self.profession)
```

Create Object of a Class

An object is essential to work with the class attributes. The object is created using the class name. When we create an object of the class, it is called instantiation. The object is also called the instance of a class.

A constructor is a special method used to create and initialize an object of a class. This method is defined in the class.

In Python, Object creation is divided into two parts in **Object Creation** and **Object initialization**

- Internally, the `__new__` is the method that creates the object
- And, using the `__init__()` method we can implement constructor to initialize the object.

```
<object-name> = <class-name>(<arguments>)
```

Below is the code to create the object of a Person class

```
jessa = Person('Jessa', 'Female', 'Software Engineer')
```

The complete example:

```
class Person:
    def __init__(self, name, sex, profession):
        # data members (instance variables)
        self.name = name
        self.sex = sex
        self.profession = profession

    # Behavior (instance methods)
    def show(self):
        print('Name:', self.name, 'Sex:', self.sex, 'Profession:', self.profession)

    # Behavior (instance methods)
    def work(self):
        print(self.name, 'working as a', self.profession)

# create object of a class
jessa = Person('Jessa', 'Female', 'Software Engineer')

# call methods
jessa.show()
jessa.work()
```