IIT Madras
BSc Degree

Home Lesson-8.2

Object Oriented Programming

Objects and Classes

Objects are at the core of object oriented programming. With minimum jargon, let us try to understand what it all means. The simplest way of understanding the idea of objects is with the help of this wonderful phrase:

> Unity in diversity

What unites all of us? We are all humans. We call this planet home.

And yet, each of us is different. Each individual is unique. For example, height and weight are two obvious properties that make each individual stand out. In more philosophical terms, there are two forces in operation here: a global force that unites all of us and a local force that gives each one of us our own special identity. In the terminology of object oriented programming, each human being is an **object**. But all these objects belong to the **class** called "Humanity".

This idea can be extended further. As a more mundane example, think about cars on the road. No two cars are the same. The brand of a car is one obvious point of difference. Even two cars of the same brand could be moving at different speeds. The speed of a car is another point of difference. However, a car is not the same as a train. We know a car when we see one. There are certain global features that are common to all cars and there is no mistaking that. In the terminology of object oriented programming, each car is an object. But all these objects belong to the class called "Car".

We are now ready to move from the concrete to the abstract.

> Objects are entities that have certain attributes along with operations associated with them.

For example, cars on the road could have the following attributes: speed, fuel level. The operations associated with it could be: start, stop, accelerate, decelerate, fill fuel tank. Given this basic understanding of what we mean by objects, we are now ready to define a class:

> A class is a blueprint or a template that is used to create objects.

The specification of what constitutes a car is present in a class, say `Car`, note that capital "C". The specification of what makes a human is present in another class, say `Human`. Think about a class as a Google form. A form is nothing but a template. The template is created once and then it is distributed. Each of us fills this form differently hence creating different objects.

> Object Oriented Programming (OOP) is a paradigm that looks at the world as a collection of objects and the interactions among them.

Rather than focusing on more definitions, let us jump in and look at OOP in action.

OOP in Python: an example

Consider a very simple template for a student that has following information:

- Name
- Marks

We want to perform the following operations:

- Update the marks of the student
- Print student details

```
class Student:
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks

    def update_marks(self, marks):
        self.marks = marks

    def print_details(self):
        print(f'{self.name}')
```

`class` is a keyword in Python to define classes. It is similar to the `def` keyword that is used to define functions. `Student` is the name of the class that we are creating. Within the class, we see that there are three functions: `__init__`, `update_marks` and `print_details`. Functions defined inside a class are given a special name, they are called methods of the class. Among the methods that can be defined for a class, `__init__` holds a special place and is called the constructor. Let us first see how to create an object of type `Student`:

```
anish = Student('Anish', 95)
```

`anish` is now an object of type `Student`. To verify this, run the following command:

```
print(type(anish)) # output should be: <class '__main__.Student'>
```

We shall take up this example in the next lesson and understand various features of a class.