Chapter 7 : Objects and Classes

Aviral Janveja

1 Object

Throughout this course in Python, we have seen various examples of data like numbers, strings, booleans and so on. All of these instances of data belong to a particular type like integer, float or string.

In Python, an object essentially represents any piece of data. Whether it's an integer, float or string, all data in Python are objects. This means that the term **object** serves a dual purpose : it refers to an instance of a particular data type and also represents the fundamental root of all data types in Python. This base object type provides core functionality in Python language, such as the ability to assign values to variables.

This is why it is often said that **everything is an object** in Python, and every object has a type. The type of an object defines how it is represented internally within the Python language and dictates how we can interact with it. For instance, a list is always represented via square brackets [ ] and we can manipulate list objects using various built-in methods like insert, append, remove and sort.

2 Class

The next question that naturally arises is : how are these data-types defined and can we define our own type ?

Answer : Yes, we can define our own data-type by defining a new **class**. Defining your own class is like defining a blue print for your own data-type, similar to integers, floats or lists. The class definition looks like this :

```
class class_name(object):
    # Define attributes here
```

We start with the **class** keyword, then we mention the **name** of our class and in the parenthesis we mention the **parent** class. The parent class here is ofcourse object. As we have already discussed, it is the parent of all types in Python. Inside the class, we define its attributes which are the data attributes and the procedural attributes that belong to the class.

2.1 Data

Data attributes define the data representation of this particular type. For example, a two dimensional coordinate object will be made up of two numbers, in a bracket, separated by a comma.

Coordinate object = (3,5)

One value for the x-coordinate and one value for the y-coordinate. We can further decide on whether these two numbers will be of type int or type float.

2.2 Methods

Procedural attributes, better known as methods are functions, that only work with this particular type. For example, you can define a distance method between two coordinate objects, but that method will have no meaning for list objects.

3 Defining a new type : Class Coordinate

Let us continue with the class definition above and implement our own class.

```
class Coordinate(object):
    # Data and methods
```

3.1 Special Method __init__

First, we define a method called **__init__** in our class. In Python __init__ is a special method known as the constructor. It is automatically called when a new instance object of a class is created. The __init__ method allows you to **initialize** the data attributes or variables of an object as soon as the object is formed.

```
class Coordinate(object):

    def __init__(self, x, y):
        self.x = x
        self.y = y
```

While defining this method, a default parameter named **self** is always used. The self parameter refers to the object of the class itself and is used for assigning values to the data members of the object. The two underscores on each side of its name imply that the method is invoked automatically and used internally by Python, without needing to be called explicitly by the object.

3.2 Creating Instances