Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

AY: 2021-2022

PYTHON'S OBJECT ORIENTED PROGRAMMING - 1

Guido Van Rossum

Agenda:

1. Basic Idea about Class, Object and Reference Variable.

1. Basic Idea about Class, Object and Reference Variable

First, you need to get much clarity about the following Key terms of Object Oriented Programming (OOP) and then the remaining things will become very easy.

1. **Class**
2. **Object**
3. **Reference Variable**

We will try to understand the above three terms with some examples.

Example 1: Building Plan

Suppose, **we want to construct a building of 100 floors.** Can we construct directly this building?

Ans: No

So, What we need to do?

- First, we need to consult an Architect and ask that can you please provide plan to construct a building of100 floors.
- This Plan contains complete information about the building (i.e., How many floors, Space and how many rooms, How many windows and how many doors etc.,).
- Plan describes complete template about our building.
- Once the Plan is ready, we can start construction of the building.

Let us assume that,

- One building is constructed based on this Plan in Nandyal.

Now, can we construct the same building in another place based on this Plan or not?

Ans: Yes

- That means, Once Plan is ready, we can construct '**n**' number of buildings.

For example, Assume that a Gated community contains 100 villas, every villa having the same plan. Here, **Plan acts as a Blue print for the building**.

In this example, **Plan will be considered as Class** and **each Building is considered as an Object** [i.e., Object nothing but a real world entity].

What is a Class?

- Class is a blue print / template / plan / design.
- To create objects we required some Model or Plan or Blue print, which is nothing but class.
- We can write a class to represent properties (attributes) and actions (behaviour) of object.
- Properties can be represented by Variables.
- Actions can be represented by Methods.
- Hence, Class contains both Variables and Methods.

What is an Object?

- A Physical existence of a class (or) An Instance of a class.
- For one class, we can create multiple objects.

Example 2: TV Model

Suppose I want to buy a SONY KD 65 model TV.

- I went to some shop1 at Kurnool and ask about the price of that model TV and they said 50000/-.
- I went to another shop 2 at Kurnool and ask about the price of that model TV and they said 48900/-.
- I went to another shop 3 at Kurnool and ask about the price of that model TV and they said 47900/-.
- Then I checked in the Amazon and found that the price of that model TV is 40000/-.

Now, you can take the TV from anywhere. Once the model is same then TV object is always same.

In this case, **TV Model is considered as Class** and each **TV is considered as an Object**.

Example 3: Ganesh Chaturthi

- In our country, we are going to celebrate Ganesh festival. We require a small Ganesh statue and we do some pooja to that statue.
- Suppose in our house if the Ganesh statue is not there, then we will go to the market. In the market the person who is going to prepare Ganesh statues have some mould.
- In this mould he is going to keep clay and prepares a single Ganesh statue at a time.
- If you want to buy a Ganesh Idol, Based on the mould, he can create 'n' number of Ganesh statues.

Here, this **mould acts as class** and **each Ganesh statue coming from this mould is considered as an object.**

What is a Reference variable?

- Suppose, if the SONY people have the design of SONY K 65 Model TV. Based on this design, they are going to create multiple TV's.
- Here, the design is considered as a class and each TV is considered as an object.
- I brought the TV to my home and now I want to operate the TV. To perform Operations on my TV, I require a small device called as remote.
- By using this remote, I can perform required operations on the TV.

In this case, the remote itself is considered as a **reference variable** and TV is considered as **Object**.

Def: The Variable which can be used to refer the Object is called “reference variable”.

The main purpose of the reference variable:

- By using this reference variable, we can perform required operations on the Object.
- By using a reference variable we can access properties and methods of an object.

Note:

- For one class, you may create multiple objects.
- For one object, any number of reference variables may be there. (Like one TV, any number of remotes may be there).
- If there is an object without reference variable, such objects are useless.

How to define a class?

- We can define a class by using **class** keyword.

Syntax:

```
class className:
    ''' Documentation string'''
    Variables ==> Properties / Attributes
    Methods ==> Actions / Behaviors
```

- **Documentation string** represents description of the class.
- Within the class doc string is always optional.

We can get 'doc string' of the class by using the following 2 ways.

1. print(className.__doc__)

2. help(className)

Eg:

```
class student:
    '''This class developed by Karthi for demo purpose'''
    # Variables
    # Methods
print(student.__doc__)     # doc is the predefined variable, which gives the
                           # description of the documentation string.
```

```
This class developed by Karthi for demo purpose
```

Another way:

```
class student:
    '''This class developed by Karthi for demo purpose'''
    # Variables
    # Methods
help(student)
```

```
Help on class student in module __main__:

class student(builtins.object)
 |  This class developed by Karthi for demo purpose
 |
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

- Within the Python class we can represent data by using variables. There are 3 types of variables are allowed inside a python class.

    1. **Instance Variables (Object Level Variables)**
    2. **Static Variables (Class Level Variables)**
    3. **Local variables (Method Level Variables)**

Note:

We will discuss later about these variables.

- Within the Python class, we can represent operations by using methods. The following are various types of allowed methods inside a python class.

    1. **Instance Methods**
    2. **Class Methods [Not there in Java class]**
    3. **Static Methods**

Note:

We will discuss later about these methods.

Demo program to define and use a class in Python

```
class student:
    '''class developed by Karthi for demo purpose''' # Optional
    def __init__(self):
        self.name = 'Karthi'
        self.rollno = 101
        self.marks = 78
                                                # It is a template
    def talk(self):
        print("Hello I am ",self.name)
        print("My roll number is :",self.rollno)
        print("My marks are :",self.marks)          # Syntacically correct
```

Interpretation of above example:

- Functions/Procedures is the functional or procedural programming terminology, but in Object oriented programming terminology we call them as methods [i.e., functions which are defined inside a class].
- We are taking one special method **__init__** and one argument self is passing to this method. We will discuss about these in next level.
- We declared 3 variables inside a class.
- Above class contains a docstring, variables(3) and methods(2).
- Once class is ready, for this class, you can create object and that object contain a reference variable.
- By using that reference variable, you can perform required operations on that object.

How to create an Object?

Syntax to create object:

Reference_Variable = className()

Eg:

```
s = student( )
```

Now, student class object is ready with the reference variable **s**.

- Once we create the object, automatically the special method (i.e., **__init__**) will be executed. This special method also known as constructor [Note: We will discuss about this later].
- The variable which can be used to refer object is called reference variable.
- By using reference variable, we can access properties and methods of object.

Program: Write a Python program to create a Student class and Create an object to it. Call the method talk() to display student details.

```
class student:
    '''class developed by Karthi for demo purpose''' # Optional
    def __init__(self):
        self.name = 'Karthi'
        self.rollno = 101
        self.marks = 78

    def talk(self):
        print("Hello I am ",self.name)
        print("My roll number is :",self.rollno)
        print("My marks are :",self.marks)
s = student()
print(s.name)
print(s.rollno)   # Accessing variables by using reference variables
print(s.marks)
s.talk() # For the self argument we need not required to pass any thing.
         # The value internally PVM is going to take care
```

```
Karthi
101
78
Hello I am  Karthi
My roll number is : 101
My marks are : 78
```

Another Example:

In the above example, name,rollno and marks are assigned. But these properties are varied from student to student. How you can do this?

```
class student:
    '''class developed by Karthi for demo purpose''' # Optional
    def __init__(self,name,rollno,marks):
        self.name = name
        self.rollno = rollno    # Whatever values are paasing, these values are assigne
        self.marks = marks      # name,rollno and marks.

    def talk(self):
        print("Hello I am ",self.name)
        print("My roll number is :",self.rollno)
        print("My marks are :",self.marks)
s = student("Karthi",202,76)
print(s.name)
print(s.rollno)
print(s.marks)
s.talk()
```

```
Karthi
202
76
Hello I am  Karthi
My roll number is : 202
My marks are : 76
```

```
class student:
    '''class developed by Karthi for demo purpose''' # Optional
    def __init__(self,name,rollno,marks):
        self.name = name
        self.rollno = rollno   # Whatever values are paasing, these values are assigne
        self.marks = marks     # name,rollno and marks.

    def talk(self):
        print("Hello I am ",self.name)
        print("My roll number is :",self.rollno)
        print("My marks are :",self.marks)
s = student("Karthi",202,76)
s1 = student("Sourav",203,87)     # You can create any number of objects for a class
print(s.name)
print(s.rollno)
print(s.marks)
s.talk()
print(s1.name)
print(s1.rollno)
print(s1.marks)
s1.talk()
```

```
Karthi
202
76
Hello I am  Karthi
My roll number is : 202
My marks are : 76
Sourav
203
87
Hello I am  Sourav
My roll number is : 203
My marks are : 87
```

```
class student:
    '''class developed by Karthi for demo purpose''' # Optional
    def __init__(self,name,rollno,marks):
        self.name = name
        self.rollno = rollno    # Whatever values are paasing, these values are assigne
        self.marks = marks      # name,rollno and marks.

    def talk(self):
        print("Hello I am ",self.name)
        print("My roll number is :",self.rollno)
        print("My marks are :",self.marks)
s = student("Karthi",202,76)
s1 = student("Sourav",203,87)     # You can create any number of objects for a class
#print(s.name)
#print(s.rollno)
#print(s.marks)
#s.talk()
print(s1.name)
print(s1.rollno)
print(s1.marks)
s1.talk()
```

```
Sourav
203
87
Hello I am  Sourav
My roll number is : 203
My marks are : 87
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You