Classes and Object

- A class is a user-defined blueprint or prototype from which objects are created.
- Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made.
- Each class instance can have attributes attached to it for maintaining its state.
- Class instances can also have methods (defined by their class) for modifying their state.

Syntax: Class Definition

```
class ClassName:
    # Statement
```

Syntax: Object Definition

```
obj = ClassName()
print(obj.atrr)
```

- The class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- A class is like a blueprint for an object.

❖ Some points on Python class:

- ➢ Classes are created by keyword class.
- ➢ Attributes are the variables that belong to a class.
- ➢ Attributes are always public and can be accessed using the dot (.) operator.
  Eg.: My class.Myattribute

❖ Creating a Python Class

Here, the class keyword indicates that you are creating a class followed by the name of the class

Example:

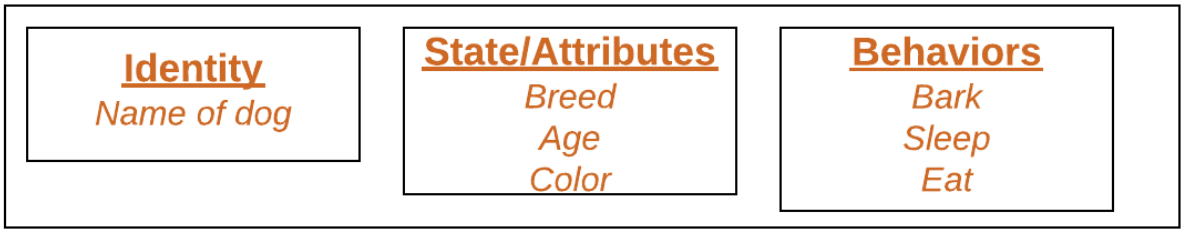
```
class Dog:
        sound = "bark"
```

❖ Object of Python Class

An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*.

An object consists of:

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

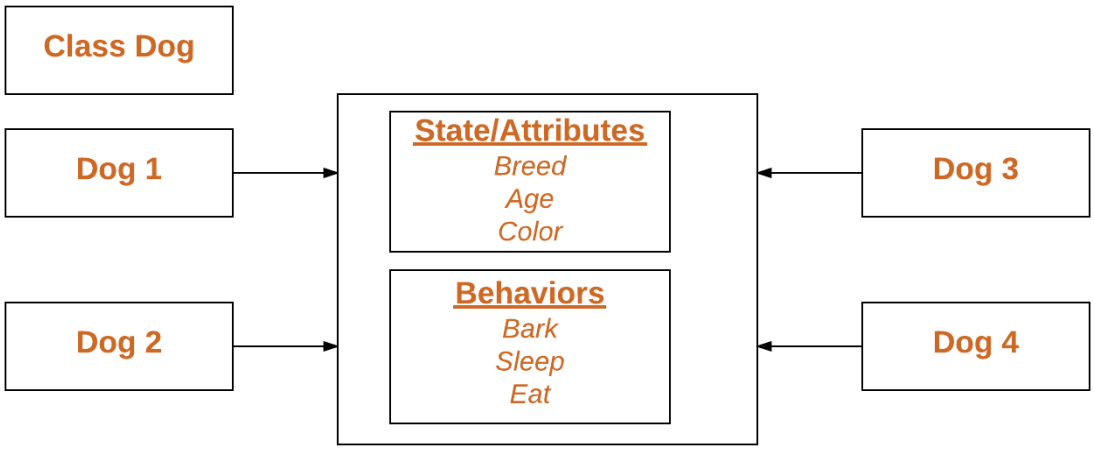


❖ Declaring Claas Objects (Also called instantiating a class)

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of

those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

Example:



❖ Example of Python Class and object

Creating an object in Python involves instantiating a class to create a new instance of that class. This process is also referred to as object instantiation.

```
# Python3 program to
# demonstrate instantiating
# a class
class Dog:
        # A simple class
        # attribute
        attr1 = "mammal"
        attr2 = "dog"
        # A sample method
        def fun(self):
                print("I'm a", self.attr1)
                print("I'm a", self.attr2)
```

Output:

```
mammal
I'm a mammal
I'm a dog
```

❖ Self Parameter

When we call a method of this object as myobject.method(arg1, arg2), this is automatically converted by Python into MyClass.method(myobject, arg1, arg2) – this is all the special self is about.

Example:

```
class GFG:
        def __init__(self, name, company):
                self.name = name
                self.company = company

        def show(self):
                print("Hello my name is " + self.name+" and I" +
                        " work in "+self.company+".")


obj = GFG("John", "Topper World")
obj.show()
```

Output:

```
Hello my name is John and I work in GeeksForGeeks.
```

❖ Pass Statement

- The program's execution is unaffected by the pass statement's inaction. It merely permits the program to skip past that section of the code without doing anything.
- It is frequently employed when the syntactic constraints of Python demand a valid statement but no useful code must be executed.

Example:

```
class MyClass:
        pass
```

❖ Class and Instance Variables

- Instance variables are for data, unique to each instance and class variables are for attributes and methods shared by all instances of the class.
- Instance variables are variables whose value is assigned inside a constructor or method with self whereas class variables are variables whose value is assigned in the class.