**CLASSES**

* *Object-oriented programming* is one of the

most effective approaches to writing software.

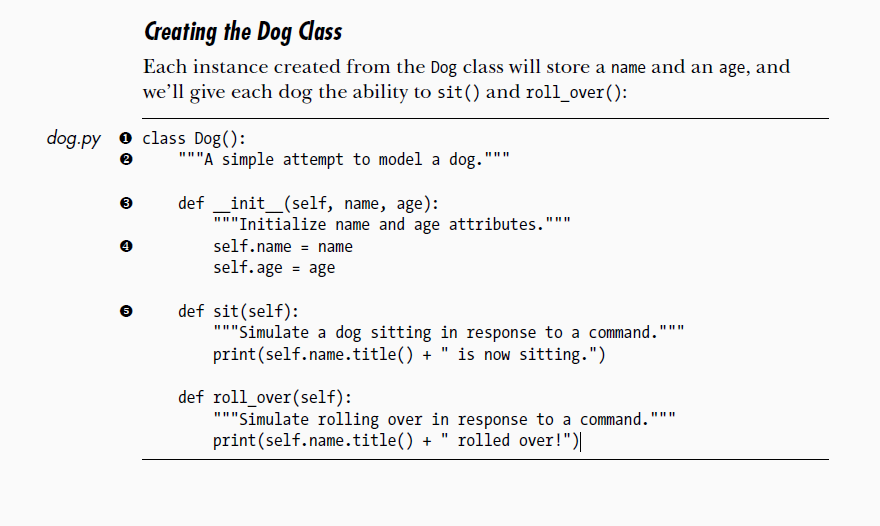
In object-oriented programming you

write *classes* that represent real-world things

and situations, and you create *objects* based on these

classes. When you write a class, you define the general

behavior that a whole category of objects can have



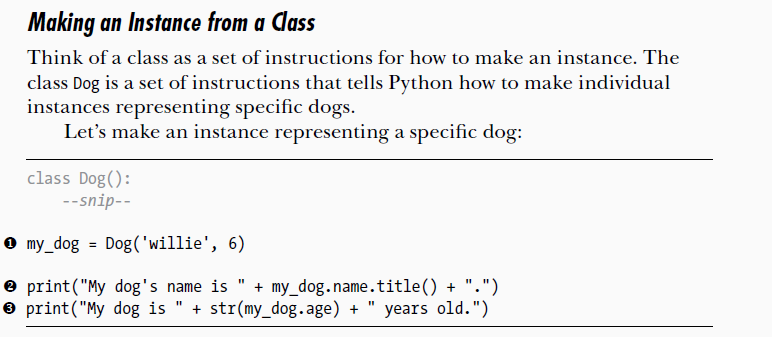
**The \_\_init\_\_ method**

1. A function that is a part of a class is a method.
2. The \_\_init\_\_ method is a special method that python runs automatically when we create the instance of the class.
3. We define three parameters: self, name, age. The self-parameter is required in the method definition and it must come before other parameters.
4. Python calls \_\_init\_\_ method later to create an instance of a class Dog, the method call automatically passes the self-argument.
5. Every method call associated with the class automatically passes self, which is a reference to the instance itself.
6. It gives individual instance access to the attributes and methods in the class.
7. The two variables defined at x each have the prefix self. Any variable prefixed with self is available to every method in the class, and we’ll also be able to access these variable through any instance created from the class.

self.name = name takes the value stored in the parameter name and stores it in the variable name, which is then attached to the instance being created.

The same process happens with self.age = age. Variables that are accessible through instances like this are called *attributes*.

1. The Dog class has two other methods defined: sit() and roll\_over() y. Because these methods don’t need additional information like a name or age, we just define them to have one parameter, self. The instances we create later will have access to these methods. In other words, they’ll be able to sit and roll over. For now, sit() and roll\_over() don’t do much. They simply print a message saying the dog is sitting or rolling over. But the concept can be extended to realistic situations: if this class were part of an actual computer game, these methods would contain code to make an animated dog sit and roll over. If this class was written to control a robot, these methods would direct movements that cause a dog robot to sit and roll over.



* **Accessing Attributes**
* **Calling Methods**
* **Creating Multiple Instances**

**Inheritance**

You don’t always have to start from scratch when writing a class. If the class

you’re writing is a specialized version of another class you wrote, you can

use *inheritance*. When one class *inherits* from another, it automatically takes

on all the attributes and methods of the first class. The original class is

called the *parent class*, and the new class is the *child class*. The child class

inherits every attribute and method from its parent class but is also free to

define new attributes and methods of its own.

