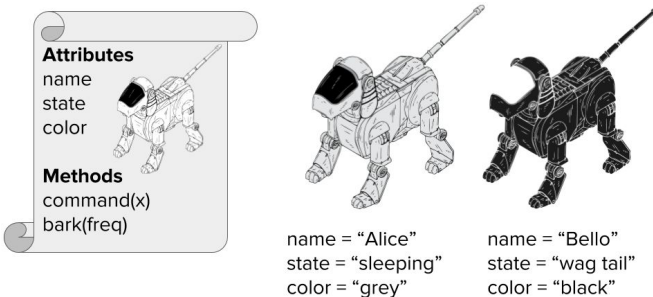


Python Cheat Sheet - Classes

“A puzzle a day to learn, code, and play” → Visit finxter.com

	Description	Example
Classes	<p>A class encapsulates data and functionality - data as attributes, and functionality as methods. It is a blueprint to create concrete instances in the memory.</p> <p>Class Instances</p> <div style="display: flex; align-items: flex-start;"><div style="border: 1px solid grey; padding: 5px; margin-right: 20px;"><p>Attributes name state color</p><p>Methods command(x) bark(freq)</p></div><div style="text-align: center;"><p>name = "Alice" state = "sleeping" color = "grey"</p><p>name = "Bello" state = "wag tail" color = "black"</p></div></div>	<pre>class Dog: """ Blueprint of a dog """ # class variable shared by all instances species = ["canis lupus"] def __init__(self, name, color): self.name = name self.state = "sleeping" self.color = color def command(self, x): if x == self.name: self.bark(2) elif x == "sit": self.state = "sit" else: self.state = "wag tail" def bark(self, freq): for i in range(freq): print("[" + self.name + "]: Woof!") bello = Dog("bello", "black") alice = Dog("alice", "white") print(bello.color) # black print(alice.color) # white bello.bark(1) # [bello]: Woof! alice.command("sit") print("[alice]: " + alice.state) # [alice]: sit bello.command("no") print("[bello]: " + bello.state) # [bello]: wag tail alice.command("alice") # [alice]: Woof! # [alice]: Woof! bello.species += ["wulf"] print(len(bello.species) == len(alice.species)) # True (!)</pre>
Instance	<p>You are an instance of the class human. An instance is a concrete implementation of a class: all attributes of an instance have a fixed value. Your hair is blond, brown, or black - but never unspecified.</p> <p>Each instance has its own attributes independent of other instances. Yet, class variables are different. These are data values associated with the class, not the instances. Hence, all instance share the same class variable species in the example.</p>	
Self	<p>The first argument when defining any method is always the self argument. This argument specifies the instance on which you call the method.</p> <p>self gives the Python interpreter the information about the concrete instance. To <i>define</i> a method, you use self to modify the instance attributes. But to <i>call</i> an instance method, you do not need to specify self.</p>	
Creation	<p>You can create classes “on the fly” and use them as logical units to store complex data types.</p> <pre>class Employee(): pass employee = Employee() employee.salary = 122000 employee.firstname = "alice" employee.lastname = "wonderland" print(employee.firstname + " " + employee.lastname + " " + str(employee.salary) + "\$") # alice wonderland 122000\$</pre>	