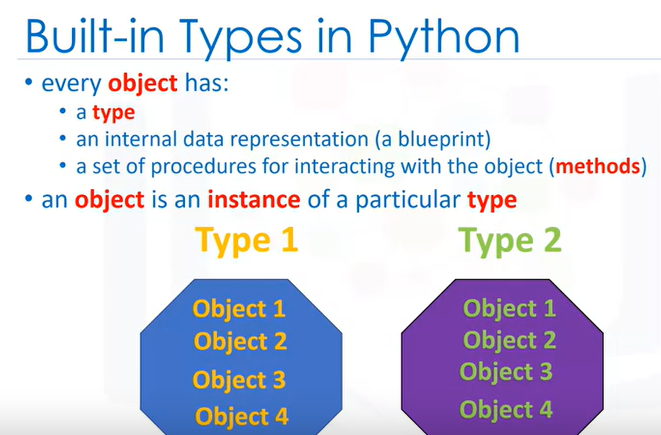
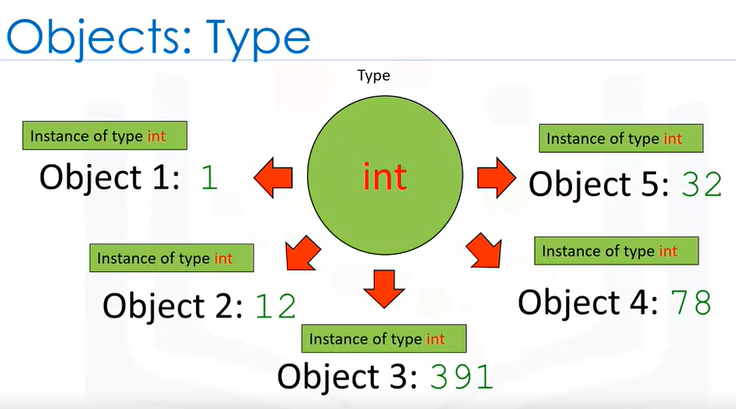
In this module, we're going to talk about objects and classes. Python has many different kinds of data types: integers, floats, strings, lists, dictionaries, booleans. In Python, each is an object.

Every object has the following: a type, internal representation, a set of functions called methods to interact with the data. An object is an instance of a particular type.

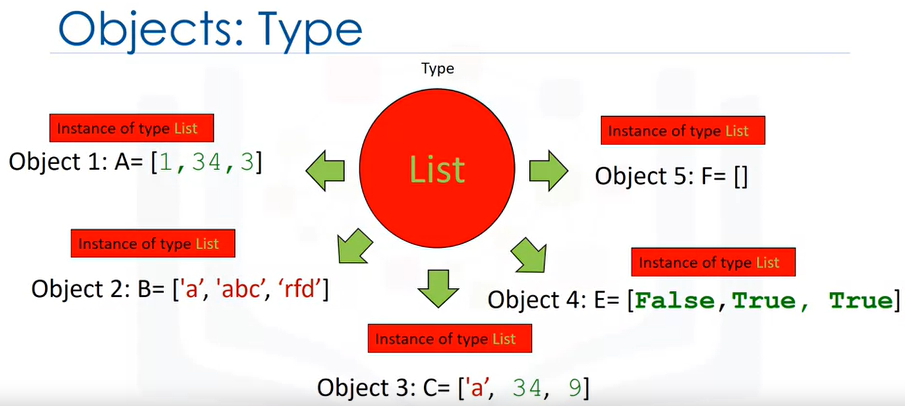


For example, we have two types, type one and type two. We can have several objects of type one as shown in yellow. Each object is an instance of type one. We also have several objects of type two shown in green. Each object is an instance of type two.

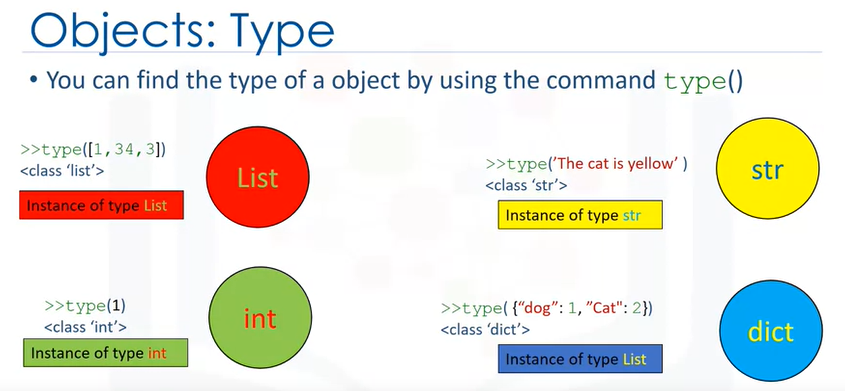
Let's do several less abstract examples. Every time we create an integer, we are creating an instance of type integer, or we are creating an integer object. In this case, we are creating five instances of type integer or five integer objects.



Similarly, every time we create a list, we are creating an instance of type list, or we are creating a list object. In this case, we are creating five instances of type list or five list objects.



We could find out the type of an object by using the type command. In this case, we have an object of type list, we have an object of type integer, we have an object of type string. Finally, we have an object of type dictionary.



type(["a"])

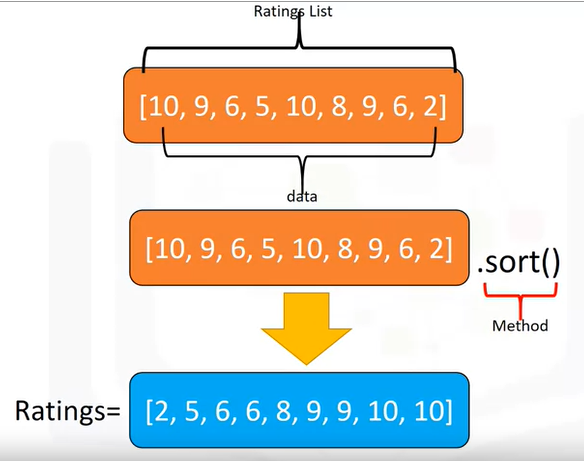
= list

**CLASSES**

A class or type's methods are functions that every instance of that class or type provides.

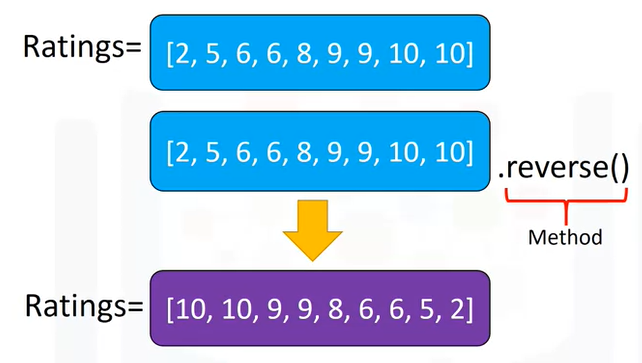
It's how you interact with the object. We have been using methods all this time, for example, on lists.

Sorting is an example of a method that interacts with the data in the object. Consider the list ratings, the data is a series of numbers contained within the list. The method sort will change the data within the object. We call the method by adding a period at the end of the object's name, and the method's name we would like to call with parentheses.



We have the rating's list represented in orange. The data contained in the list is a sequence of numbers. We call the sort method, this changes the data contained in the object. You can say it changes the state of the object.

We can call the reverse method on the list, changing the list again. We call the method, reversing the order of the sequence within the object. In many cases, you don't have to know the inner workings of the class and its methods,

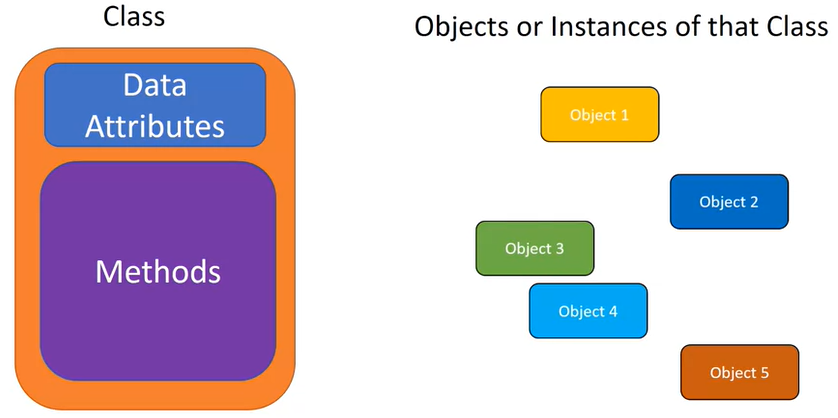


you just have to know how to use them.

**CONSTRUCT YOUR OWN CLASSES**

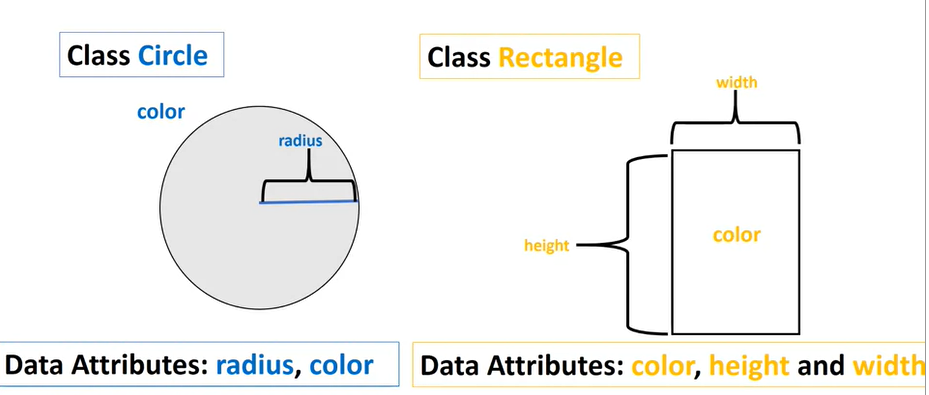
Next, we will cover how to construct your own classes. You can create your own type or class in Python. In this section, you'll create a class.

The class has data attributes. The class has methods. We then create instances or instances of that class or objects.



The class data attributes define the class. Let's create two classes.

The first class will be a circle, the second will be a rectangle. Let's think about what constitutes a circle. Examining this image, all we need is a radius to define a circle, and let's add color to make it easier to distinguish between different instances of the class later.



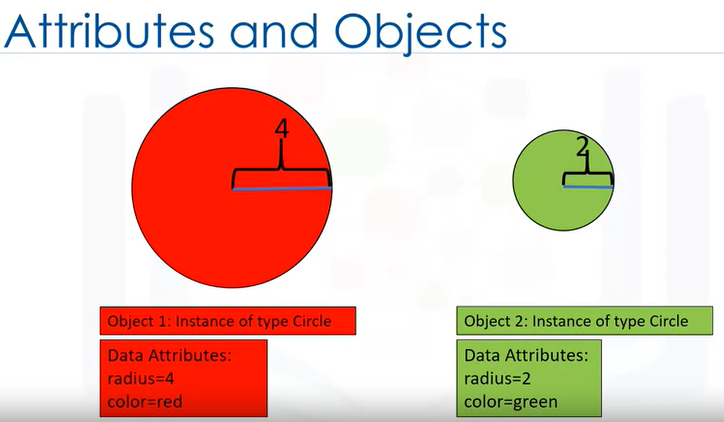
Therefore, our class data attributes are radius and color. Similarly, examining the image in order to define a rectangle, we need the height and width. We will also add color to distinguish between instances later. Therefore, the data attributes are color, height, and width.

To create the class circle, you will need to include the class definition. This tells Python you're creating your own class, the name of the class. For this course in parentheses, you will always place the term object, this is the parent of the class.

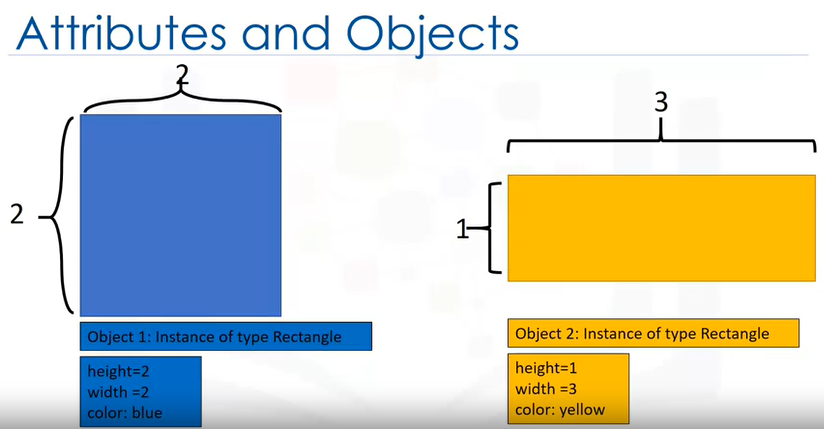


For the class rectangle, we changed the name of the class, but the rest is kept the same. Classes are outlines we have to set the attributes to create objects.

We can create an object that is an instance of type circle. The color data attribute is red, and the data attribute radius is four. We could also create a second object that is an instance of type circle. In this case, the color data attribute is green, and the data attribute radius is two.



We can also create an object that is an instance of type rectangle. The color data attribute is blue, and the data attribute of height and width is two. The second object is also an instance of type rectangle. In this case, the color data attribute is yellow, and the height is one, and the width is three. We now have different objects of class circle or type circle.



We also have different objects of class rectangle or type rectangle. Let us continue building the circle class in Python.

We define our class. We then initialize each instance of the class with data attributes, radius, and color using the class constructor.



# Create a class Circle

class Circle(object):

# Constructor

def \_\_init\_\_(self, radius=3, color='blue'):

self.radius = radius

self.color = color

# Method

def add\_radius(self, r):

self.radius = self.radius + r

return(self.radius)

# Method

def drawCircle(self):

plt.gca().add\_patch(plt.Circle((0, 0), radius=self.radius, fc=self.color))

plt.axis('scaled')

plt.show()

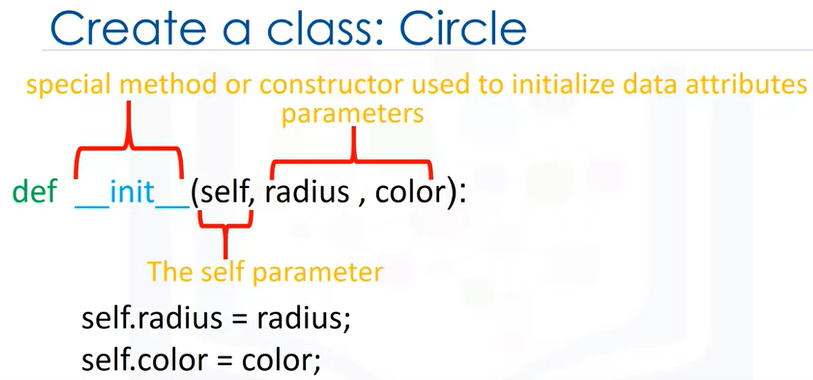
**Init Constructor**

The function init is a constructor. It's a special function that **tells Python you are making a new class.**

There are other special functions in Python to make more complex classes. The radius and color parameters are used to initialize the radius and color data attributes of the class instance.

**SELF Parameter**

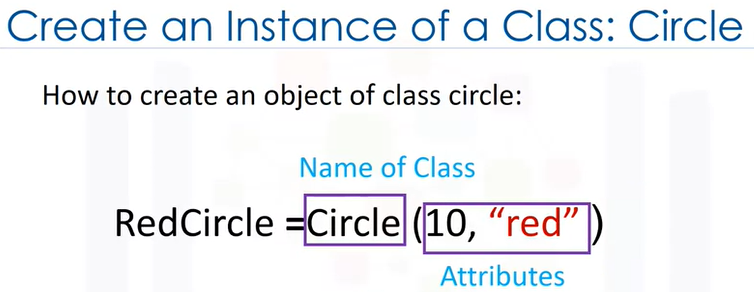
The self parameter refers to the newly created instance of the class.



The parameters, radius, and color can be used in the constructors body to access the values passed to the class constructor when the class is constructed.

We could set the value of the radius and color data attributes to the values passed to the constructor method. Similarly, we can define the class rectangle in Python. The name of the class is different. This time, the class data attributes are color, height, and width.

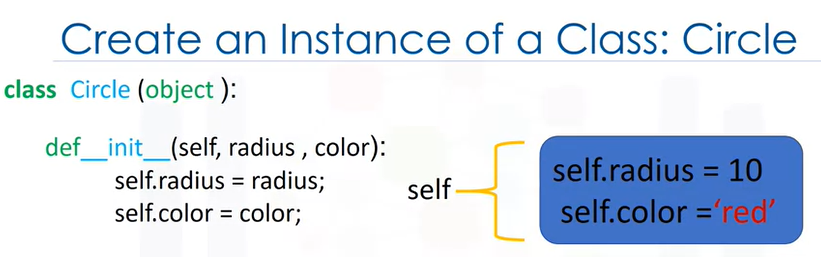
After we've created the class, in order to create an object of class circle, we introduce a variable. This will be the name of the object. We create the object by using the object constructor.



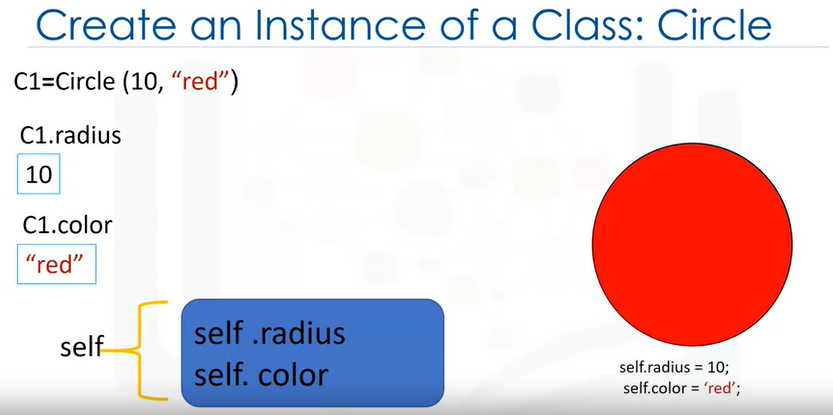
The object constructor consists of the name of the class as well as the parameters. These are the data attributes.

When we create a circle object, we call the code like a function. The arguments passed to the circle constructor are used to initialize the data attributes of the newly created circle instance.

It is helpful to think of self as a box that contains all the data attributes of the object.

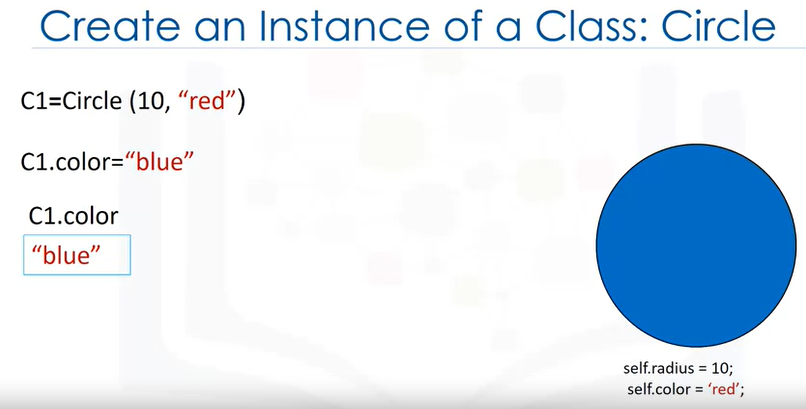


Typing the object's name followed by a dot and the data attribute name gives us the data attribute value, for example, radius. In this case, the radius is 10.



We can do the same for color. We can see the relationship between the self parameter and the object.

In Python, we can also set or change the data attribute directly.



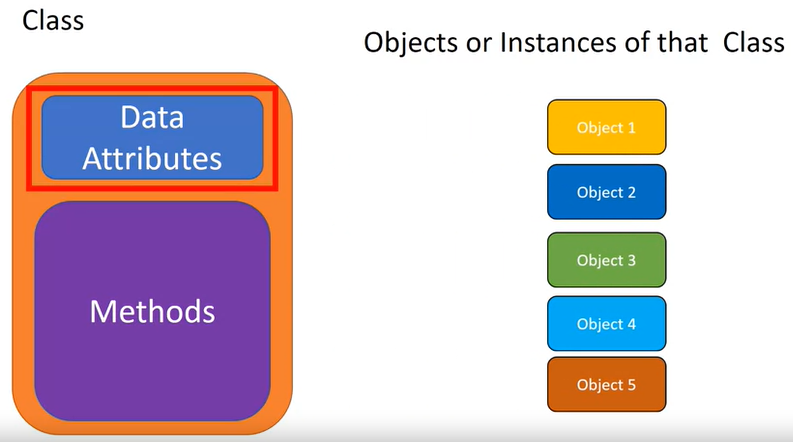
Typing the object's name followed by a dot and the data attribute name, and set it equal to the corresponding value. We can verify that the color data attribute has changed.

Usually, in order to change the data in an object, we define methods in the class.

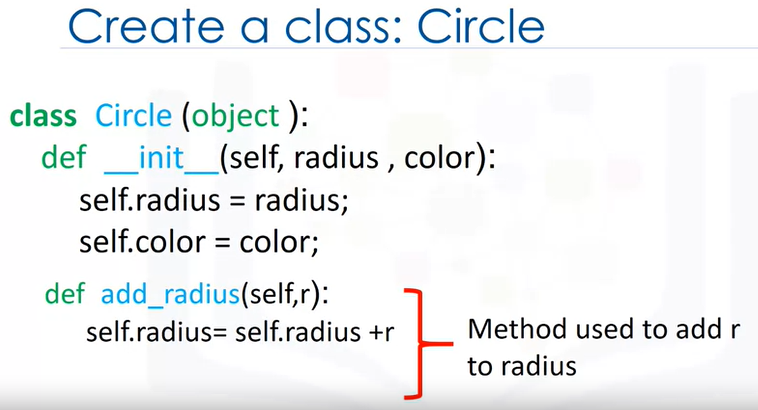
**METHODS**

Let's discuss methods. We have seen how data attributes consist of the data defining the objects.

**Methods are functions that interact and change the data attributes**, **changing or using the data attributes of the object**.



Let's say we would like to change the size of a circle. This involves changing the radius attribute. We add a method, add radius to the class circle.

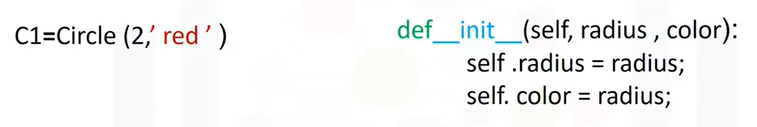


The method has a function that requires the self as well as other parameters.

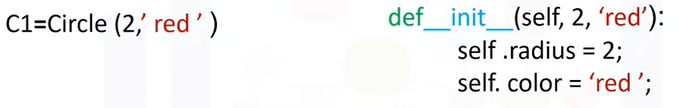
In this case, we are going to add a value to the radius, We denote that value as r. We are going to add r to the data attribute radius.

Let's see how this part of the code works when we create an object and call the add\_radius method.

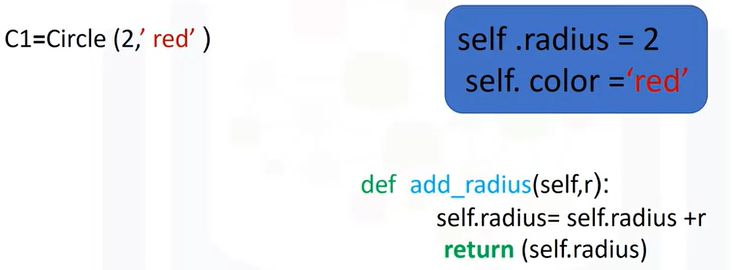
As before, we create an object with the object constructor. We pass two arguments to the constructor. The radius is set to two and the color is set to red.



In the constructor's body, the data attributes are set.



We can use the box analogy to see the current state of the object.

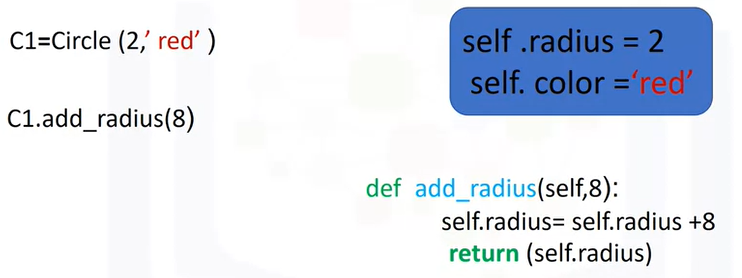


We call the method by adding a dot followed by the method, name, and parentheses.

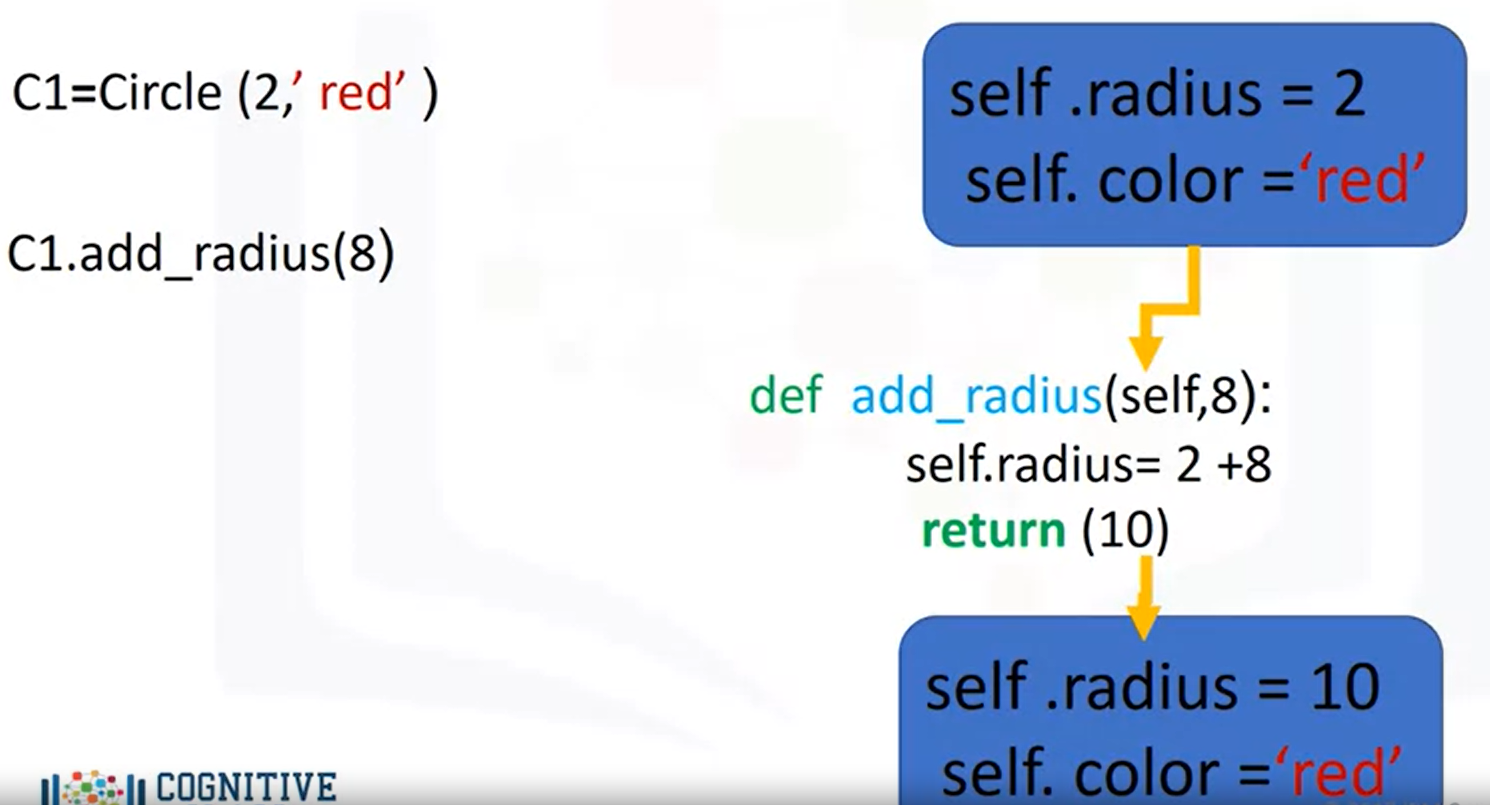
In this case, the argument of the function is the amount we would like to add. We do not need to worry about the self parameter when calling the method.

Just like with the constructor, Python will take care of that for us. In many cases, there may not be any parameters other than self specified in the method's definition. So we don't pass any arguments when calling the function.

Internally, the method is called with a value of eight, and the proper self object.

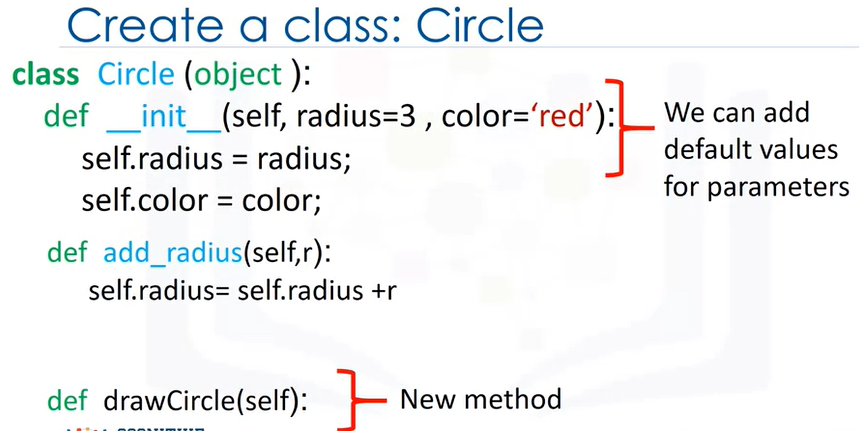


The method assigns a new value to self radius. This changes the object, in particular, the radius data attribute.

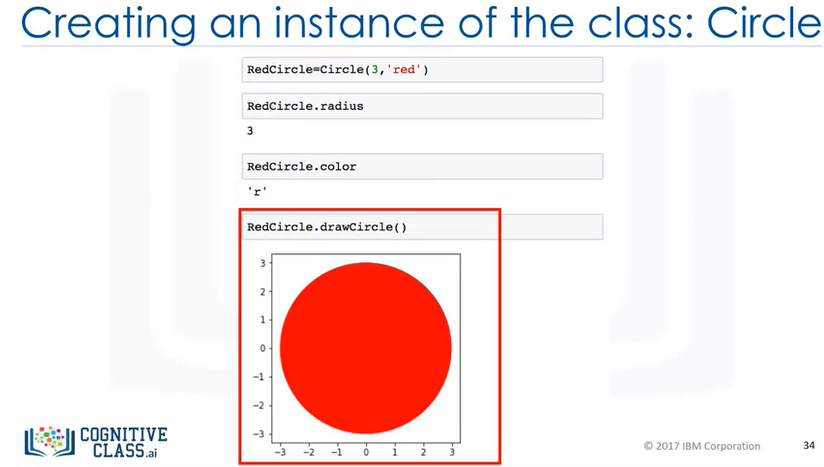


When we call the add\_radius method, this changes the object by changing the value of the radius data attribute. We can add default values to the parameters of a class as constructor.

In the labs, we also create the method called drawCircle. See the lab for the implementation of drawCircle.

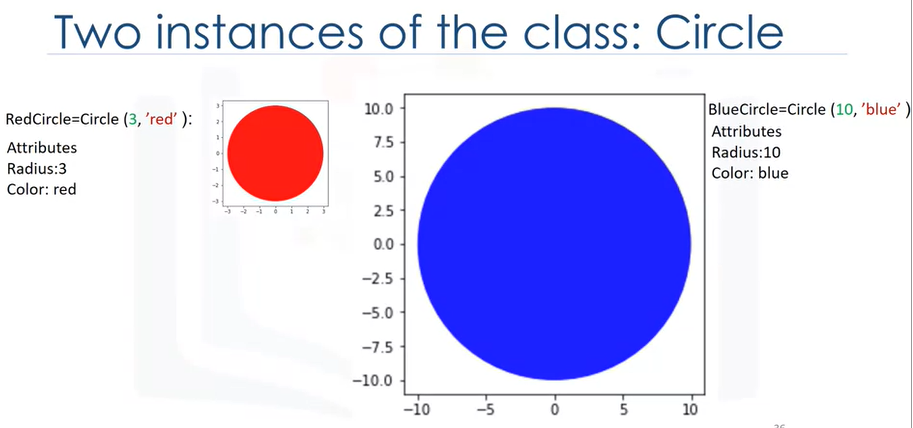


In the labs, we can create a new object of type circle using the constructor. The color will be red and the radius will be three. We can access the data attribute radius. We can access the attribute color. Finally, we can use the method drawCircle to draw the circle.



Similarly, we can create a new object of type circle. We can access the data attribute of radius. We can access the data attribute color. We can use the method drawCircle to draw the circle.

In summary, we have created an object of class circle called RedCircle with a radius attribute of three, and a color attribute of red.



We also created an object of class circle called BlueCircle, with a radius attribute of 10 and a color attribute of blue. In the lab, we have a similar class for rectangle. We can create a new object of type rectangle using the constructor. We can access a data attribute of height. We can also access the data attribute of width.

We could do the same for the data attribute of color. We can use the method drawRectangle to draw the rectangle. **So we have a class, an object that is a realization or instantiation of that class.**

For example, we can create two objects of class Circle, or two objects of class Rectangle.

**Dir Function**

The dir function is useful for obtaining the list of data attributes and methods associated with a class. The object you're interested in is passed as an argument. The return value is a list of the objects data attributes.



The attribute surrounded by underscores are for internal use, and you shouldn't have to worry about them. The regular looking attributes are the ones you should concern yourself with. These are the objects, methods, and data attributes. There is a lot more you can do with objects in Python. Check Python.org for more info.