PY0101EN-3-4-Classes

June 17, 2019

<a href="https://cocl.us/topNotebooksPython101Coursera">
    <img src="https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/PY0101EN/Ad/To
</a>

Classes and Objects in Python

<strong>Welcome!</strong>
Objects in programming are like objects in real life. Like life, there are different classes of objects. In this notebook, we
<ul>
    <li>what a class is</li>
    <li>what an attribute is</li>
    <li>what a method is</li>
</ul>

Don't worry if you don't get it the first time, as much of the terminology is confusing. Don't forget to do the practice tests in the notebook.

Table of Contents

<p>
    Estimated time needed: <strong>40 min</strong>
</p>

Introduction to Classes and Objects

Creating a Class

The first part of creating a class is giving it a name: In this notebook, we will create two classes, Circle and Rectangle. We need to determine all the data that make up that class, and we call that an attribute. Think about this step as creating a blue print that we will use to create objects. In figure 1 we see two classes, circle and rectangle. Each has their attributes, they are variables. The class circle has the attribute radius and color, while the rectangle has the attribute height and width. Let's use the visual examples of these shapes before we get to the code, as this will help you get accustomed to the vocabulary.

Figure 1: Classes circle and rectangle, and each has their own attributes. The class circle has the attribute radius and colour, the rectangle has the attribute height and width.

Instances of a Class: Objects and Attributes

An instance of an object is the realisation of a class, and in Figure 2 we see three instances of the class circle. We give each object a name: red circle, yellow circle and green circle. Each object has different attributes, so let's focus on the attribute of colour for each object.

Figure 2: Three instances of the class circle or three objects of type circle.

The colour attribute for the red circle is the colour red, for the green circle object the colour attribute is green, and for the yellow circle the colour attribute is yellow.

Methods

Methods give you a way to change or interact with the object; they are functions that interact with objects. For example, let's say we would like to increase the radius by a specified amount of a circle. We can create a method called **add_radius(r)** that increases the radius by **r**. This is shown in figure 3, where after applying the method to the "orange circle object", the radius of the object increases accordingly. The "dot" notation means to apply the method to the object, which is essentially applying a function to the information in the object.

Figure 3: Applying the method "add_radius" to the object orange circle object.

Creating a Class

Now we are going to create a class circle, but first, we are going to import a library to draw the objects:

```
In [6]: # Import the library

        import matplotlib.pyplot as plt
        %matplotlib inline
```

The first step in creating your own class is to use the class keyword, then the name of the class as shown in Figure 4. In this course the class parent will always be object:

Figure 4: Three instances of the class circle or three objects of type circle.

The next step is a special method called a constructor __init__, which is used to initialize the object. The input are data attributes. The term self contains all the attributes in the set. For example the self.color gives the value of the attribute color and self.radius will give you the radius of the object. We also have the method add_radius() with the parameter r, the method adds the value of r to the attribute radius. To access the radius we use the syntax self.radius. The labeled syntax is summarized in Figure 5:

Figure 5: Labeled syntax of the object circle.

The actual object is shown below. We include the method drawCircle to display the image of a circle. We set the default radius to 3 and the default colour to blue:

```
In [7]: # 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()
```

Creating an instance of a class Circle
Let's create the object RedCircle of type Circle to do the following:

```
In [8]: # Create an object RedCircle

        RedCircle = Circle(10, 'red')
```

We can use the dir command to get a list of the object's methods. Many of them are default Python methods.

```
In [9]: # Find out the methods can be used on the object RedCircle

        dir(RedCircle)
```

```
Out[9]: ['__class__',
         '__delattr__',
         '__dict__',
         '__dir__',
         '__doc__',
         '__eq__',
         '__format__',
         '__ge__',
         '__getattribute__',
         '__gt__',
         '__hash__',
         '__init__',
         '__init_subclass__',
         '__le__',
         '__lt__',
         '__module__',
         '__ne__',
         '__new__',
         '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'add_radius',
 'color',
 'drawCircle',
 'radius']
```

We can look at the data attributes of the object:

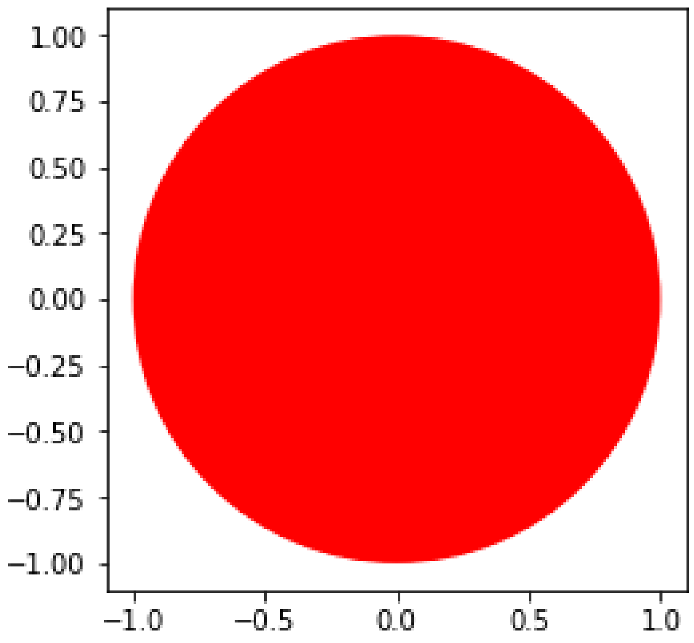
```
In [10]: # Print the object attribute radius

         RedCircle.radius

Out[10]: 10

In [11]: # Print the object attribute color

         RedCircle.color

Out[11]: 'red'
```

We can change the object's data attributes:

```
In [12]: # Set the object attribute radius

         RedCircle.radius = 1
         RedCircle.radius

Out[12]: 1
```

We can draw the object by using the method drawCircle():

```
In [13]: # Call the method drawCircle

         RedCircle.drawCircle()
```



We can increase the radius of the circle by applying the method add_radius(). Let increases the radius by 2 and then by 5:

```
In [14]: # Use method to change the object attribute radius

         print('Radius of object:',RedCircle.radius)
         RedCircle.add_radius(2)
         print('Radius of object of after applying the method add_radius(2):',RedCircle.radius)
         RedCircle.add_radius(5)
         print('Radius of object of after applying the method add_radius(5):',RedCircle.radius)
```

```
Radius of object: 1
Radius of object of after applying the method add_radius(2): 3
Radius of object of after applying the method add_radius(5): 8
```

Let's create a blue circle. As the default colour is blue, all we have to do is specify what the radius is:

```
In [16]: # Create a blue circle with a given radius

         BlueCircle = Circle(radius=100)
```

As before we can access the attributes of the instance of the class by using the dot notation:

```
In [17]: # Print the object attribute radius

         BlueCircle.radius
```

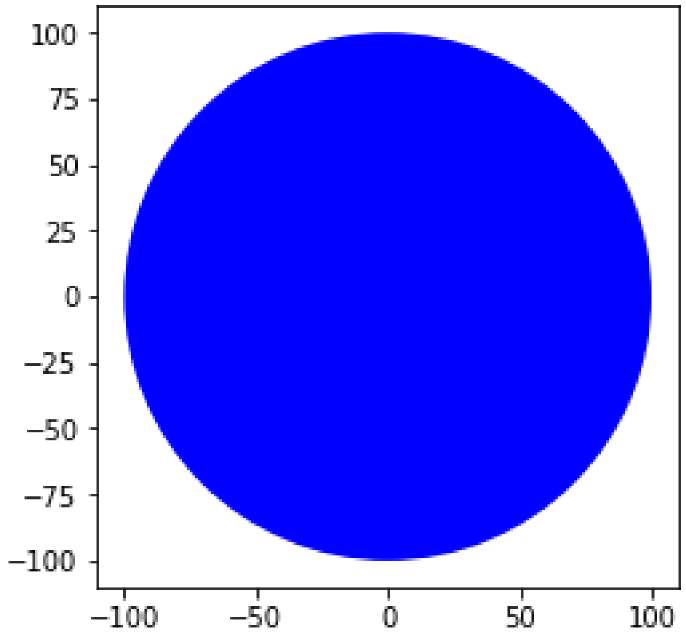
Out[17]: 100

In [18]:
```
# Print the object attribute color

BlueCircle.color
```

Out[18]: 'blue'

We can draw the object by using the method drawCircle():

In [19]:
```
# Call the method drawCircle

BlueCircle.drawCircle()
```



Compare the x and y axis of the figure to the figure for RedCircle; they are different.
The Rectangle Class
Let's create a class rectangle with the attributes of height, width and color. We will only add the method to draw the rectangle object:

In [21]:
```
# Create a new Rectangle class for creating a rectangle object

class Rectangle(object):

    # Constructor
    def __init__(self, width=2, height=3, color='r'):
        self.height = height
        self.width = width
        self.color = color

    # Method
    def drawRectangle(self):
        plt.gca().add_patch(plt.Rectangle((0, 0), self.width, self.height ,fc=self.color))
        plt.axis('scaled')
        plt.show()
```

Let’s create the object SkinnyBlueRectangle of type Rectangle. Its width will be 2 and height will be 3, and the color will be blue:

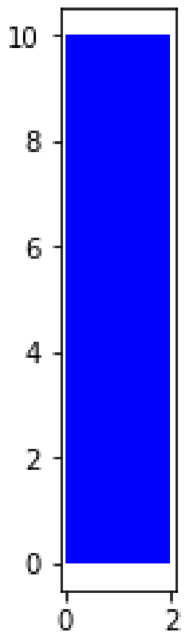
```
In [22]: # Create a new object rectangle

         SkinnyBlueRectangle = Rectangle(2, 10, 'blue')
```

As before we can access the attributes of the instance of the class by using the dot notation:

```
In [23]: # Print the object attribute height

         SkinnyBlueRectangle.height

Out[23]: 10

In [24]: # Print the object attribute width

         SkinnyBlueRectangle.width

Out[24]: 2

In [25]: # Print the object attribute color

         SkinnyBlueRectangle.color

Out[25]: 'blue'
```

We can draw the object:

```
In [26]: # Use the drawRectangle method to draw the shape

         SkinnyBlueRectangle.drawRectangle()
```



Let's create the object FatYellowRectangle of type Rectangle :

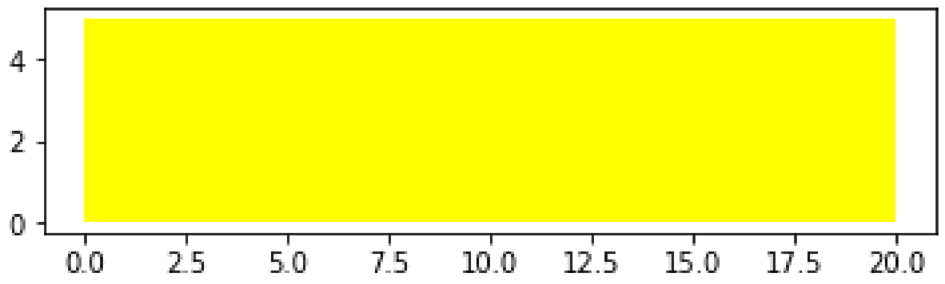
```
In [28]: # Create a new object rectangle

         FatYellowRectangle = Rectangle(20, 5, 'yellow')
```

We can access the attributes of the instance of the class by using the dot notation:

```
In [29]: # Print the object attribute height

         FatYellowRectangle.height

Out[29]: 5

In [30]: # Print the object attribute width

         FatYellowRectangle.width

Out[30]: 20

In [31]: # Print the object attribute color

         FatYellowRectangle.color

Out[31]: 'yellow'
```

We can draw the object:

```
In [32]: # Use the drawRectangle method to draw the shape

         FatYellowRectangle.drawRectangle()
```



0
2
4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0