Classes and Objects in Python

Estimated time needed: **40** minutes

Objectives

After completing this lab you will be able to:

- Work with classes and objects
- Identify and define attributes and methods

Table of Contents

Introduction to Classes and Objects

Creating a Class

The first step in 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, which we call *attributes*. 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, which are variables. The class Circle has the attribute radius and color, while the Rectangle class 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.

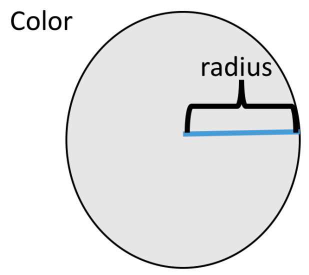
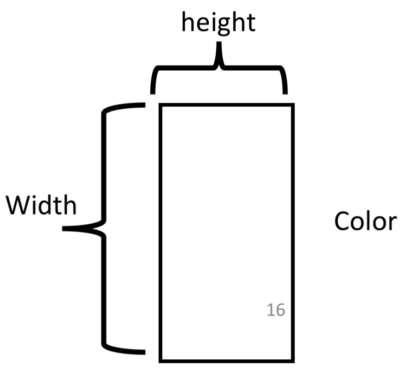
Class Circle	**Class Rectangle**
Attributes: radius, Color	Attributes: Color, height and Width
	

Figure 1: Classes circle and rectangle, and each has their own attributes. The class Circle has the attribute radius and colour, the class Rectangle has the attributes 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 color attribute for each object.

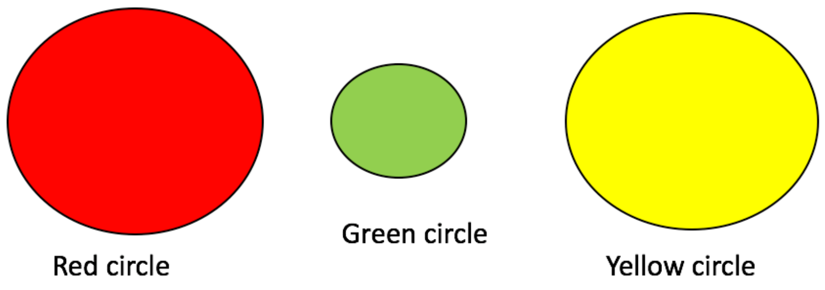


Figure 2: Three instances of the class Circle, or three objects of type Circle.

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 of a circle by a specified amount. 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.

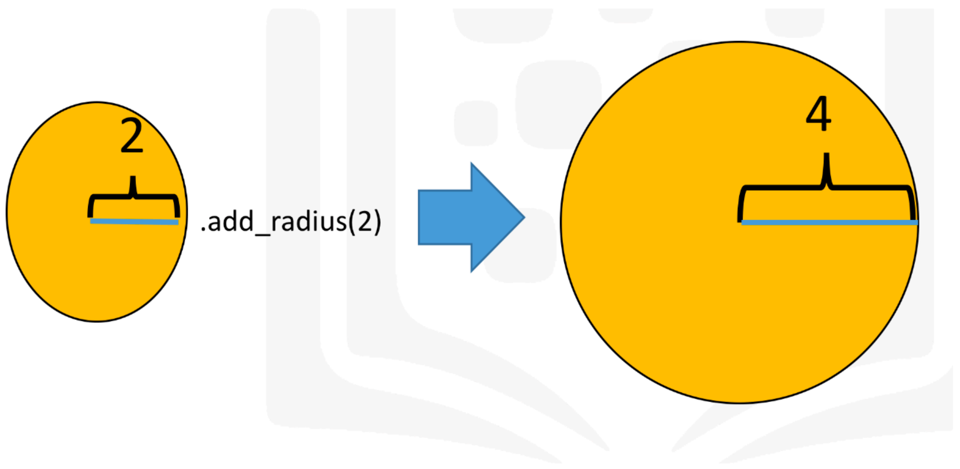


Figure 3: Applying the method "add_radius" to the object orange circle object.

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:

```
# 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:

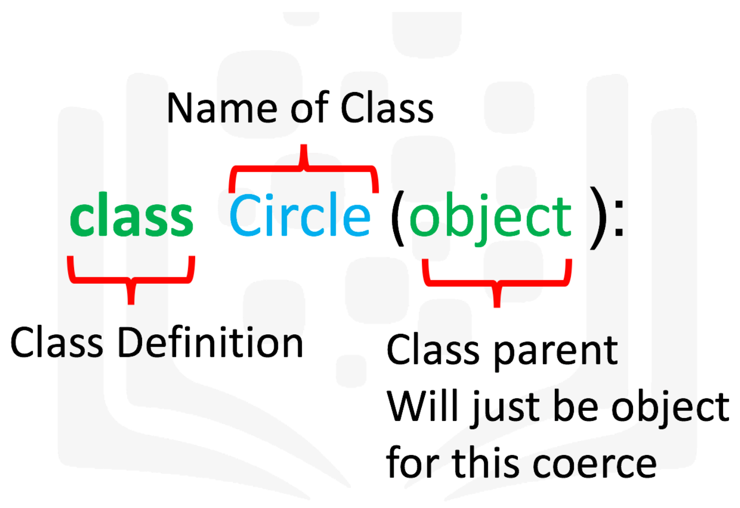
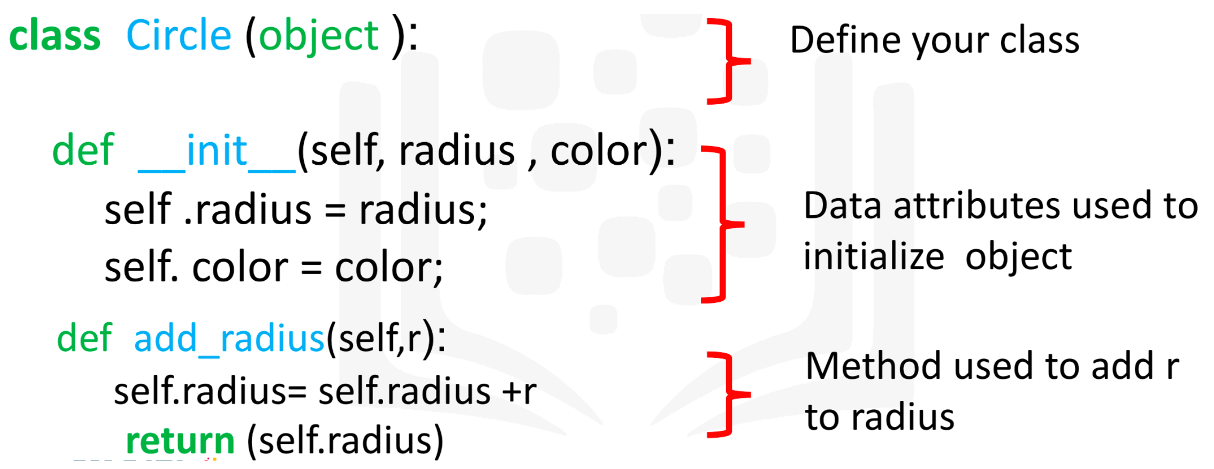


Figure 4: Creating a class Circle.

The next step is a special method called a constructor `_init_\`, which is used to initialize the object. The inputs 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:



```
class  Circle (object ):

  def  __init__(self, radius , color):
      self .radius = radius;
      self. color = color;
  def  add_radius(self,r):
       self.radius= self.radius +r
        return (self.radius)
```

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:

```
# 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,
        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:

```
# 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.

```
# Find out the methods can be used on the object RedCircle

dir(RedCircle)
```

We can look at the data attributes of the object:

```
# Print the object attribute radius

RedCircle.radius
```

```
# Print the object attribute color

```

```
RedCircle.color
```

We can change the object's data attributes:

```
# Set the object attribute radius

RedCircle.radius = 1
RedCircle.radius
```

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

```
# Call the method drawCircle

RedCircle.drawCircle()
```

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

```
# 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.add_radius(5)
print('Radius of object of after applying the method add_radius(5)
```

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

```
# 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:

```
# Print the object attribute radius

BlueCircle.radius
```

```
# Print the object attribute color

BlueCircle.color
```

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

```
# 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:

```
# 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
        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:

```python
# Create a new object rectangle

SkinnyBlueRectangle = Rectangle(2, 3, 'blue')
```

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

```python
# Print the object attribute height

SkinnyBlueRectangle.height
```

```python
# Print the object attribute width

SkinnyBlueRectangle.width
```

```python
# Print the object attribute color

SkinnyBlueRectangle.color
```

We can draw the object:

```python
# Use the drawRectangle method to draw the shape

SkinnyBlueRectangle.drawRectangle()
```

Let's create the object `FatYellowRectangle` of type Rectangle:

```python
# 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:

```python
# Print the object attribute height

FatYellowRectangle.height
```

```python
# Print the object attribute width

FatYellowRectangle.width
```

```python
# Print the object attribute color

FatYellowRectangle.color
```

We can draw the object:

```python
# Use the drawRectangle method to draw the shape

FatYellowRectangle.drawRectangle()
```

Scenario: Car dealership's inventory management system

You are working on a Python program to simulate a car dealership's inventory management system. The system aims to model cars and their attributes accurately.

Task-1. You are tasked with creating a Python program to represent vehicles using a class. Each car should have attributes for maximum speed and mileage.

```python
#Type your code here
class Vehicle:
    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage
```

▶ Click here for the solution

Task-2. Update the class with the default color for all vehicles," white".

```python
#Type your code here
class Vehicle:
    color = "white"

    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage
```

▶ Click here for the solution

Task-3. Additionally, you need to create methods in the Vehicle class to assign seating capacity to a vehicle.

```
#Type your code here
class Vehicle:
    color = "white"

    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage
        self.seating_capacity = None

    def assign_seating_capacity(self, seating_capacity):
        self.seating_capacity = seating_capacity
```

▶ Click here for the solution

Task-4. Create a method to display all the properties of an object of the class.

```
#Type your code here
class Vehicle:
    color = "white"

    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage
        self.seating_capacity = None

    def assign_seating_capacity(self, seating_capacity):
        self.seating_capacity = seating_capacity

    def display_properties(self):
        print("Properties of the Vehicle:")
        print("Color:", self.color)
        print("Maximum Speed:", self.max_speed)
        print("Mileage:", self.mileage)
        print("Seating Capacity:", self.seating_capacity)
```

▶ Click here for the solution

Task-5. Additionally, you need to create two objects of the Vehicle class object that should have a max speed of 200kph and mileage of 50000kmpl with five seating capacities, and another car object should have a max speed of 180kph and 75000kmpl with four seating capacities.

```python
#Type your code here
class Vehicle:
    color = "white"

    def __init__(self, max_speed, mileage):
        self.max_speed = max_speed
        self.mileage = mileage
        self.seating_capacity = None

    def assign_seating_capacity(self, seating_capacity):
        self.seating_capacity = seating_capacity

    def display_properties(self):
        print("Properties of the Vehicle:")
        print("Color:", self.color)
        print("Maximum Speed:", self.max_speed)
        print("Mileage:", self.mileage)
        print("Seating Capacity:", self.seating_capacity)

# Creating objects of the Vehicle class
vehicle1 = Vehicle(200, 50000)
vehicle1.assign_seating_capacity(5)
vehicle1.display_properties()

vehicle2 = Vehicle(180, 75000)
vehicle2.assign_seating_capacity(4)
vehicle2.display_properties()
```

```
Properties of the Vehicle:
Color: white
Maximum Speed: 200
Mileage: 50000
Seating Capacity: 5
Properties of the Vehicle:
Color: white
Maximum Speed: 180
Mileage: 75000
Seating Capacity: 4
```

▶ Click here for the solution

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python.

Author

Joseph Santarcangelo

Other contributors

Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-05-16	2.2	Akansha Yadav	updated lab under maintenance
2022-01-10	2.1	Malika	Removed the readme for GitShare
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab
