```
# Import the library for drawing the objects, first:
import matplotlib.pyplot as plt
%matplotlib inline

(33) # Create a class Circle
class Circle(object):
    def __init__(self, radius=3, color='blue'):  # Constructor
        self.radius = radius
        self.color = color
    def add_radius(self, r):  # Method
        self.radius = self.radius + r
        return(self.radius)
    def drawCircle(self):  # Method
        plt.gca().add_patch(plt.Circle((0, 0), radius=self.radius,
fc=self.color))
        plt.axis('scaled')
        plt.show()

(2) # Create an object RedCircle
RedCircle = Circle(10, 'red')

(3)
dir(RedCircle)
# Returns:
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
etc.

(4) RedCircle.radius
RedCircle.color

(5) RedCircle.radius = 1

(6) RedCircle.drawCircle()

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

Result: Radius of object: 10
Radius of object of after applying the method add_radius(2): 12

(8) # 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))
```

Create class Circle: (1)
// Data attributes: radius, colour

Notes:
- Notice __init__, a special method called a CONSTRUCTOR, used to initialize the object.
- The term self contains all the attributes in the set. For example the self.color gives the value of the attribute color.
- We also have the method add_radius() with the parameter r, the method adds the value of r to the attribute radius.

Create the object RedCircle of type Circle: (2)

We can use the dir command to get a list of the object's methods. Many of them are default Python methods: (3)

Print the object attributes radius and colour: (4)

We can change the object's data attributes: (5)

Call the method drawCircle that draws the object: (6)

Use method to change the object attribute radius: (7)

Create rectangle class with 3 attributes, like height, width and color: (8)

5. OBJECTS & CLASSES

PYTHON FOR DATA SCIENCE
PROGRAMMING FUNDAMENTALS

Ana-María Dobre

based on EDX Course

SEP 2023

OBJECTS

OBJECT = instance of a particular type.

An OBJECT has:
- type
- blueprint or internal data representation
- methods

Example: every time we are creating a list, we are creating an instance of type list or a list object.

Find type of object with type() command
// type(1) => integer

Methods:
Ratings = [1, 3, 2]
Ratings.sort() => [1, 2, 3]
Ratings.reverse() => [3, 2, 1]

CLASS

CLASS has DATA ATTRIBUTES and METHODS

class Circle(object):
class: tells Python you are defining a class
Circle: name of the class
object: name of the parent

Use the data attributes to initialise each instance of the class, via the c-tor:

```
def __init__(self, radius, color):
    self.radius = radius
    self.color = color
```

- The function __init__ is a special method or C-TOR used to initialise the data attributes.
- The radius and colour PARAMS are used to initialise the data attributes of the class instance.
- The **self** parameter refers to the newly created instance of the class.

It is helpful to think of self as a box containing all the data attributes of the object

The dir function is useful for obtaining the list of data attributes and methods associated with a class:
dir(nameOfObject)
Note: The attribute surrounded by underscores are for internal use