Object Oriented Programming in Python

By Serena Killion

What is an object? What is a class?

- An object is a collection of data and methods that act on the data
- Think of objects as a way of representing properties and behaviors of real world things
- A class is a user-defined blueprint or prototype from which objects are created
- __init__ method initializes attributes to a given object

Example:

```
class Animal:
  def __init__(self, species, name, color):
      self.species = species
      self.name = name
      self.color = color
```

Instantiating: Creating an Instance of a Class

- An instance is a copy of the class with actual values
- We can create multiple instances of an object
- Call the class using the class name and pass in the arguments in the __init__ method
- Recall: An argument is a value passed through a function

```
my_cat = Animal("cat", "Fritz", "orange")
my_dog = Animal("dog", "Louie", "brown")
my_llama = Animal("llama", "Fred", "white")
```

Functions versus Methods

- **User defined function**: Python allows us to define our own functions

```
>>> def add(a,b):
        return a+b
>>> add(3,-3)
```

- **Built-in-functions**: Functions provided by Python such as print(), open(), round()

  Ex: **__repr__** is a built-in function used to compute the "official" string reputation of an object

- **Methods**: A function called on an object and belong to a class

Example: talks() belongs to class Animal

We call talks on the object my_bird, an instance of Animal

```
class Animal:
  def __init__(self, species, name, color):
      self.species = species
      self.name = name
      self.color = color

  def __repr__(self):
    return "{}: {} \nColor: {}".format(self.species, self.name, self.color)


  def talks(self, sound):
    self.sound = sound
    return "{} is {}ing".format(self.name, self.sound)

if __name__ == '__main__':
  my_bird = Animal("bird", "Edith", "green")
  print(repr(my_bird))
  print(my_bird.talks("tweet"))
```

What is self?

- Used when we define a method and variable initialization
- Self is used to represent the instance of the class
- Used to access attributes and methods of a class
- Binds attributes with given arguments
- Helps differentiate between local (think regular) variables and instance attributes and methods

In our Animal class example we created multiple instance of an object named dog and cat. Each object has its own species, name, and color. The self argument is necessary to hold information for multiple instances of an object.

Resource: https://www.programiz.com/article/python-self-why

The Main Method: Convention or Requirement?

```
if __name__ == '__main__':
```

- Name identifier is bound to the name of any module (file of code)
- When a file is executed name is set to “__main__”
- Used to separate functionality (definitions) and functions (execution) in your code
- Java, C, C++ have a defined entry point, but Python does not
- Python does not require a main() function, but it’s a useful convention
- If you import a module as a library, the main method of that module is not executed

Additional Reading:
http://python.berkeley.edu/events/assets/newbie_nugget_Oct2_2013.html

Import Script As A Library

- I import all methods from the file animal.py to animal_main.py
- When I execute (run) animal_main.py, what gets printed?
- What happens when I execute animal.py?

animal.py — Edited

```
class Animal:
  def __init__(self, species, name, color):
      self.species = species
      self.name = name
      self.color = color

  def __repr__(self):
    return "{}: {} \nColor:
{}".format(self.species, self.name, self.color)


  """def talks(self, sound):
    self.sound = sound
    return "{} is {}ing".format(self.name,
self.sound)"""

if __name__ == '__main__':
  my_bird = Animal("bird", "Edith", "green")
  print(repr(my_bird))
```

animal_main.py

```
from animal import *


if __name__ == '__main__':
  my_cat = Animal("cat", "Fritz", "orange")
  my_dog = Animal("dog", "Louie", "brown")
  my_llama = Animal("llama", "Fred", "white")

  print(repr(my_cat))
  print(repr(my_dog))

  #print(my_cat.talks("meow"))
  #print(my_dog.talks("bark"))
```

We have imported built in libraries from Python already, but you can create your own libraries too.
*Library is a collection of functions and methods