Object Oriented Programming (Classes)

05/09/2017

Objectives

- Define OOP
- State why it is used
- Describe important terms with objects and classes in Python
- Write your own classes
- Know when to use classes vs. functions
- Define inheritance, encapsulation, and polymorphism

What is OOP

● Object-oriented programming (OOP) is a programming paradigm based on the concept of objects

● In Python, **objects** are data structures that contain data, known as **attributes**; and procedures, known as **methods**.

Why use OOP?

OOP was developed to :

- Help build large-scale software
- Promote software reuse (keep well tested code)
- Decouple code; improve maintenance and stability of code
- Hide internal details away from those that use the code.

You already have some OOP experience

Lists, dictionaries and strings are all classes

Let’s look at some examples

Let’s look at a simple example

Important Terms

Class

The abstract concept of an object...

Object

An actual instance of a class, a variable for the instance/object being accessed -- it holds reference to the object itself

Instance

What Python returns when you tell it to create a class

Instantiation

A fancy way of talking about creating a class

Constructor

What we call to instantiate a class

self

Inside of a class

Attribute

A piece of data that a class has stored within it

self.my_attribute

Method

A block of code that is accessible via the class and act on the classes attributes. Really just a function within a class.

self.my_method(self, other_args)

Let’s look at another example

What makes code OOP? - for your reference

- Inheritance - When a class is based on another class, building off of the existing class to take advantage of existing behavior, while having additional specific behavior of its own.
- Encapsulation - The practice of hiding the inner workings of our class, and only exposing what is necessary to the outside world. This idea is effectively the same as the idea of abstraction, and allows users of our classes to only care about the what (i.e. what our class can do) and not the how (i.e. how our class does what it does).
- Polymorphism - The provision of a single interface to entities of different types. This enables us to use a shared interface for similar classes while at the same time still allowing each class to have its own specialized behavior.