Monroe Township Library Coding Bootcamp

6/23 Class Notes

* Classes & Object Oriented Programming
* Init method
* Instances
* Defining class methods
* Overriding ‘magic’ methods
* Subclasses/Inheritance

**Classes & Instances:**

* Classes in Python give us a way to define our own objects with specific functionality
  + We’ve technically been using classes all along, all the data types we looked at are defined as classes with their own specific attributes and methods
    - For instance, there is a ‘string’ class in Python and methods like find() or join() are defined in that class
* Classes are a feature of Object Oriented Programming which has some advantages
  + Objects are encapsulated and abstracted, meaning that important information is stored inside the object and only certain things are ‘exposed’
    - This makes code more organized and makes it less likely that you’ll have issues where changing a bit of information will affect your whole program
* To create a new class in Python, use the class keyword followed by the name of your class with a colon at the end
  + Inside of this code block is where you’ll define any variables or methods associated with your class

**Init Method:**

* A class is just a definition but if we want to be able to create actual instances of a class that is usable we need to define an init method
  + \_\_init\_\_ is a magic method (or a dunder method) so it has 2 underscores before and after the name
* Methods are defined the same way as any other function, using the def keyword followed by parentheses containing any parameters (arguments), with a colon at the end
  + All methods require self to be the first parameter, but this does not need to passed in as an argument when calling the method
    - self simply represents the current instance that is calling the method
* The init method should set up any attributes that are unique to new instances of the class
  + For instance, if you are passing in arguments when initializing a new object, your init method should use a self.<variable name> = <argument> format
    - Using self specifies that this variable definition applies *only* to this specific instance and not the class as a whole

**Instances:**

* Once a class is defined, it can be used to create an indefinite number of instances, similar to the way a function can be called multiple times after it is defined
* To create a new instance, use the name of your class followed by parentheses with any arguments that your init method takes
  + If your init method does not take arguments (besides self) you still need to include an empty parentheses

**Overriding ‘magic’ methods:**

* Magic methods are methods in Python that are not meant to be called directly, but are technically called when using different operators
  + For example when using the (+) operator the \_\_add\_\_ method is called
* You can overwrite these methods in your class definition the same way we did with \_\_init\_\_
  + By default, trying to add 2 objects together with the (+) operator will throw an error, but you can overwrite the \_\_add\_\_ method to produce a different result
* Some other useful magic methods:
  + \_\_sub\_\_ (-) operator, subtraction
  + \_\_lt\_\_ (<) operator (should return True or False)
  + \_\_gt\_\_ (>) operator (should return True or False)
  + \_\_eq\_\_ (==) operator (should return True or False)
  + \_\_str\_\_ converts object to string, and is also used when printing object

**Subclasses/Inheritance:**

* You can create a subclass by including a previously created class in parentheses in the new class’ definition
* A subclass inherits all properties and methods associated with its parent class
  + For example, if we created the method func() in a class, and we created a second class that inherits from the first, we would be able to call the func() method on any instance of that class
  + Parent class methods can also be replaced by creating a method of the same name in the subclass
* You can use the super() function to get the class directly above
  + For example, if we want to implement a similar \_\_init\_\_ method, we can call super() and grab the \_\_init\_\_ method from the parent class so that we don’t have to retype all the same code

**Check class files at** [**github.com/monroecoding**](https://github.com/monroecoding)

Chris DiFazio

Adult Services Librarian

cdifazio@monroetpl.org