Python Lecture 4 Notes: Methods, Classes, and Libraries

Miles Martinez

10/6/22

1 Introduction and learning outcomes from class

Here's a quick summary of what we went over & learned in class!

- Know what classes are
- know what methods & attributes are, and how they relate to classes
- Understand how classes fit into 'object oriented programming'
- Understand the relationship between classes and libraries
- Know what packages/libraries are and why they're important
- Learn how to import a library in Jupyter Notebook
- Learn where to search for information about a library

2 Classes

Classes are the big building blocks of python. You can think of them as templates that you can use to create new variables. These individual new variables are called "instances".

Class = template

Instances = creations from template

We create a class using the following code:

```
class class_name:

    def __init__(self,attributes):
        self.attributes = attributes
```

and so on. Some examples of classes that we've seen already are lists and strings.

2.1 attributes

Attributes are variables that are intrinsic to classes. These describe the characteristics of each class. In class we went over the characteristics of the class "cat" (name, color, size, nickname). These characteristics are a part of the template, so each instance will also have all these attributes. We add them to our class as above: using dot notation, to assign these attributes to self. We then access these using dot notation:

```
instance.attribute
```

2.2 methods

methods are functions that belong to a class. As a result, each instance of the class will be able to use these methods. We define those using this syntax:

```
def class_name:

    ...

    def new_method(self,arguments):
        function code goes here
```

So we need an indented block, 'self' as the first argument, and then later arguments. Then, the rest of the code we'd normally use to define a function goes in the indented block!

We use a method using the dot notation. After the dot notation, its use is similar to any other function.

```
instance.method(arguments)
```

3 Libraries

Libraries are collections of functions and classes, that we can also view AS classes. We import these using this notation:

```
import package
```

This line goes at the top of our script, before we write any of our code. This lets us use any of the functions and classes IN that package in the rest of our code. For example:

```
import numpy as np
...
a = np.array([1,2,3])
b = np.random.choice(10)
```

np.array is a **class** from numpy, and np.random.choice is a **function** from the sublibrary "random" of numpy. We've renamed numpy as "np", so we can access any of these things using "np.function" instead of "numpy.function".

We install new libraries in jupyter notebooks using:

```
!pip install package
!pip install other_package --update
```

Or, we can use Anaconda!

If we look for **Documentation** for these packages, google is the right place to look. We can generally google things like "numpy.array" if we're interested in specific functions, even just general packages if we feel we don't ENTIRELY know what we're looking for. Some good packages that will be helpful for your projects:

a. matplotlib

b. numpy

c. pandas

d. seaborn

e. psychopy