14.S: OBJECT-ORIENTED PROGRAMMING (SUMMARY)

Contributed by Chuck Severance
Clinical Associate Professor (School of Information) at University of Michigan

This is a very quick introduction to object-oriented programming that focuses mainly on terminology and the syntax of defining and using objects. Let's quickly review the code that we looked at in the beginning of the chapter. At this point you should fully understand what is going on.

```
stuff = list()
stuff.append('python')
stuff.append('chuck')
stuff.sort()
print (stuff[0])

print (stuff.__getitem__(0))
print (list.__getitem__(stuff,0))

# Code: http://www.py4e.com/code3/party1.py
```

The first line constructs a `list` *object*. When Python creates the `list` object, it calls the *constructor* method (named `__init__`) to set up the internal data attributes that will be used to store the list data. Due to *encapsulation* we neither need to know, nor need to care about how these internal data attributes are arranged.

We are not passing any parameters to the *constructor* and when the constructor returns, we use the variable `stuff` to point to the returned instance of the `list` class.

The second and third lines are calling the `append` method with one parameter to add a new item at the end of the list by updating the attributes within `stuff` . Then in the fourth line, we call the `sort` method with no parameters to sort the data within the `stuff` object.

Then we print out the first item in the list using the square brackets which are a shortcut to calling the `__getitem__` method within the `stuff` *object*. And this is equivalent to calling the `__getitem__` method in the `list` *class* passing the `stuff` object in as the first parameter and the position we are looking for as the second parameter.

At the end of the program the `stuff` object is discarded but not before calling the *destructor* (named `__del__`) so the object can clean up any loose ends as necessary.

Those are the basics and terminology of object oriented programming. There are many additional details as to how to best use object oriented approaches when developing large applications and libraries that are beyond the scope of this chapter.[3]