14.3: USING OBJECTS

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

It turns out we have been using objects all along in this class. Python provides us with many built-in objects. Here is some simple code where the first few lines should feel very simple and natural to you.

CODE 14.3.1 (PYTHON):

```
%%python3

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
```

run restart

But instead of focusing on what these lines accomplish, lets look at what is really happening from the point of view of object-oriented programming. Don't worry if the following paragraphs don't make any sense the first time you read them because we have not yet defined all these terms.

The first line is *constructing* an object of type *list*, the second and third lines are calling the `append()` *method*, the fourth line is calling the `sort()` *method*, and the fifth line is retrieving the item at position 0.

The sixth line is calling the `__getitem__()` method in the `stuff` list with a parameter of zero.

```
print (stuff.__getitem__(0))
```

The seventh line is an even more verbose way of retrieving the 0th item in the list.

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

In this code, we care calling the `__getitem__` method in the `list` class and passing in the list (`stuff`) and the item we want retrieved from the list as parameters.

The last three lines of the program are completely equivalent, but it is more convenient to simply use the square bracket syntax to look up an item at a particular position in a list.

We can take a look into the capabilities of an object by looking at the output of the `dir()` function:

```
>>> stuff = list()
>>> dir(stuff)
['__add__', '__class__', '__contains__', '__delattr__',
'__delitem__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__',
'__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__',
'__setitem__', '__sizeof__', '__str__', '__subclasshook__',
'append', 'clear', 'copy', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

The precise definition of `dir()` is that it lists the *methods* and *attributes* of a Python object.

The rest of this chapter will define all of the above terms so make sure to come back after you finish the chapter and re-read the above paragraphs to check your understanding.