**Object-Oriented Programming Defined**

In object-oriented programming, concepts are modeled as classes and objects. An idea is defined using a class, and an instance of this class is called an object. Almost everything in Python is an object, including strings, lists, dictionaries, and numbers. When we create a list in Python, we’re creating an object which is an instance of the list class, which represents the concept of a list. Classes also have attributes and methods associated with them. Attributes are the characteristics of the class, while methods are functions that are part of the class.

# Classes and Objects in Detail

We can use the **type()** function to figure out what class a variable or value belongs to. For example, **type(" ")** tells us that this is a string class. The only attribute in this case is the string value, but there are a bunch of methods associated with the class. We've seen the **upper()** method, which returns the string in all uppercase, as well as **isnumeric()** which returns a boolean telling us whether or not the string is a number. You can use the **dir()** function to print all the attributes and methods of an object. Each string is an instance of the string class, having the same methods of the parent class. Since the content of the string is different, the methods will return different values. You can also use the **help()** function on an object, which will return the documentation for the corresponding class. This will show all the methods for the class, along with parameters the methods receive, types of return values, and a description of the methods.

**Defining Classes (Optional)**

We can create and define our classes in Python similar to how we define functions. We start with the **class** keyword, followed by the name of our class and a colon. Python style guidelines recommend class names to start with a capital letter. After the class definition line is the class body, indented to the right. Inside the class body, we can define attributes for the class.

Let's take our Apple class example:

>>> class Apple:

...     color = ""

...     flavor = ""

...

We can create a new instance of our new class by assigning it to a variable. This is done by calling the class name as if it were a function. We can set the attributes of our class instance by accessing them using dot notation. Dot notation can be used to set or retrieve object attributes, as well as call methods associated with the class.

>>> jonagold = Apple()

>>> jonagold.color = "red"

>>> jonagold.flavor = "sweet"

We created an Apple instance called jonagold, and set the color and flavor attributes for this Apple object. We can create another instance of an Apple and set different attributes to differentiate between two different varieties of apples.

>>> golden = Apple()

>>> golden.color = "Yellow"

>>> golden.flavor = "Soft"

We now have another Apple object called golden that also has color and flavor attributes. But these attributes have different values.