Chapter 9 — Object Oriented Programming

Introduction

Object Oriented Programming is a paradigm used in many languages, including: c++, java, JavaScript, and Python. It allows for programming code and data to be connected in an easy to use structure. This structure (called a class) allows for code re-use in many applications.

You have already seen classes and objects, when we were using the various methods, of strings, dates, random, and in many other places. When we follow a variable or a value by a period and then a name, we are accessing attributes or methods of that object.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Describe the concept of a class, an object, attributes of an object, and methods that act upon that object.
- Create a class to describe a "real world" class of things.
- Set and Use attributes from within an object's initializer and other methods.
- Implement methods to set default attribute value and to return a string representation of the object.
- Use the concept of inheritance to extend a base class into a more specific or specialized class.

Prerequisites

This Chapter requires topics through Chapter 8.

Definitions

Before we get started with this topic, we need to define and discuss several new ideas. The first is the class definition. A class defines a structure that can contain actions, known as methods, and places to store values, known as attributes. An object is an instance of a specific class.
A sample class could be "Automobile". All automobiles have common attributes (manufacturer, year, model, serial number...) and have common methods (made, sold, wrecked, licensed...). Each automobile in the parking lot could be represented by an object of the Automobile class. Every car is different and can be bought, sold, built, and wrecked independently of each other instance of

Automobile.

We could create a class called Vehicle that contains all the common information and methods that an Automobile, Truck, and Motorcycle have in common. When defining the class for a specific vehicle type, we would inherit the base class Vehicle and only add the attributes and methods that make it different from the base class.

`class`	Concept
A ***class*** is the definition of the structure that will be used to create the actual things (object). A class may include how values will be stored (attributes) and the actions that the things will do (methods). https://docs.python.org/3/tutorial/classes.html	

`object`	Concept
An ***object*** is a unique thing (either real or imagined). All objects of the same type share a common structure. In technical language, an object is an instance of a class. https://docs.python.org/3/tutorial/classes.html#a-word-about-names-and-objects	

`method`	Concept
A ***method*** is an action that an object can do. A method is actually a `def` that is embedded in object. https://docs.python.org/3/tutorial/classes.html#class-objects	

`attribute`	Concept
An ***attribute*** (also known a property) is a value that is connected to an object. Most attributes are unique to an instance of an object, but sometimes may be shared across all objects of a class. https://docs.python.org/3/tutorial/classes.html#class-objects	

inheritance	Concept
Inheritance is when a class extends a base class. The methods and attributes of the original class are also part of the new class. https://docs.python.org/3/tutorial/classes.html#inheritance	

Creating a Simple Class

In this first example, we will create a simple class called People. It is recommended that the first letter of the class' name be capitalized. This way it might not be confused with other variables.

class *Class_name*: *suite*	Statement
The class statement... https://docs.python.org/3/reference/compound_stmts.html#class-definitions	

```
1| class People():
2|     pass
3|
4| jim=People()
5| jim.name = "Jim Reneau"
6| em=People()
7| em.name = "Emanuel Goldstein"
8| print(jim.name,"likes to read about",em.name)
```

```
Jim Reneau likes to read about Emanuel Goldstein
```

As you can see in the code the class definition contains nothing but a `pass`. As is defined, now, the People class defines nothing (attributes or methods). In the code, we create two objects (stored in the variables jim and em). We then set the attribute name, on each of them, to an individual's name. Lastly we print them out.

Adding a Method to a Class

Taking the simple People class from the previous unit, we can change it.

```
1| class People():
2|     def fullName(self):
3|         return self.lastName + ", " + self.firstName
4|
5| jim=People()
6| jim.lastName = "Reneau"
7| jim.firstName = "Jim"
8| em=People()
9| em.lastName = "Goldstein"
10| em.firstName = "Emanuel"
11| print(jim.firstName,"likes to read about",em.fullName())
```

```
Jim likes to read about Goldstein, Emanuel
```

In this example we still have two objects, but we define two attributes: lastName and firstName. To get a person's full name we can concatenate the two name parts together, but that would be a line of code used several times throughout a large program. You can see that we have created a def called fullName. When creating a method the first argument is always the free variable "self". This allows the method to access the attributes for that specific object. Without "self" the variables would be local to the def and would cease to exist when it returned control to the calling code.

```
1| class People():
2|     def fullName(self):
3|         return self.lastName + ", " + self.firstName
4|     def titledName(self, title):
5|         return title + " " + self.firstName + " " +
   self.lastName
6|
7| jim=People()
8| jim.lastName = "Reneau"
9| jim.firstName = "Jim"
10| em=People()
11| em.lastName = "Goldstein"
12| em.firstName = "Emanuel"
13| print(jim.firstName,"likes to read about",em.titledName("Mr."))
```

The method "titledName" is an example of passing values to a method. When it is defined the "self" free variable is included, but it is not included when the method is called. This may be confusing, calling a method with the first argument missing, but is important that you define it and leave it off when calling.

```
1| …
```

```
2|      def titledName(self, title = "Mr. "):
3|          return title + " " + self.firstName + " " +
    self.lastName
4| …
```

Optional arguments are also allowed when creating a method, just like when creating a stand alone `def`.

The Initializer (__init__)

There are a few `def` names that Python reserves for special purposes. The most common is `__init__`. The initializer is called automatically by Python when the object is first created and is a place for code that sets up the initial or default attribute values. You may also pass values to the initializer just like any def.

__init__	Concept
Defines a function that will be executed when a new instance of the class is created. Typically, this function initializes attributes of the object. https://docs.python.org/3/reference/datamodel.html#object.__init__	

```
1| class People():
2|     def __init__(self,first,last,title="Mr."):
3|         self.lastName = last
4|         self.firstName = first
5|         self.title = title
6|     def fullName(self):
7|         return self.lastName + ", " + self.firstName
8|     def titledName(self):
9|         return self.title + " " + self.firstName + " " +
    self.lastName
10|
11| jim=People("James","Reneau","Dr.")
12| em=People("Emanuel","Goldstein")
13| print(jim.firstName,"likes to read about",em.titledName())
```

```
James likes to read about Mr. Emanuel Goldstein
```

Object to String

Another special method that may come in handy is "__str__". This definition will return a string value for the class, automatically when a string is expected. In the example below you can see how the print statement (when printing the object) uses the __str__ method.

__str__	Concept
Defines a special method of a class that will return a string representing the specific object's attributes. https://docs.python.org/3/reference/datamodel.html#object.__str__	

```
1|  class Class():
2|      def __init__(self, name="unnamed", capacity=30):
3|          self.enrolled = []
4|          self.name = name
5|          self.capacity = 30
6|
7|      def __str__(self):
8|          return ( "The " + self.name + " class has "+
    str(len(self.enrolled))
9|                + " students, and has " + str(self.available())
10|               + " seats available.")
11|
12|     def available(self):
13|         return self.capacity - len(self.enrolled)
14|
15|     def register(self, student):
16|         # return true/false if successful
17|         if student in self.enrolled:
18|             return True  # already enrolled
19|         else:
20|             if len(self.enrolled) >= self.capacity:
21|                 return False
22|             else:
23|                 self.enrolled.append(student)
24|                 return True
25|
26|     def drop(self, student):
```

```
27|          # return true/false if successful
28|          if student in self.enrolled:
29|              self.enrolled.remove(student)
30|              return True
31|          else:
32|              return False
33|
34|
35|  buis3100 = Class("database")
36|
37|  for x in (123, 56, 43, 435345, 54645, 8787, 213123):
38|      print('register', x, buis3100.register(x))
39|  print(buis3100)
40|  print(buis3100.enrolled)
41|
42|  print('drop 43', buis3100.drop(43))
43|  print(buis3100)
44|  print(buis3100.enrolled)
```

```
register 123 True
register 56 True
register 43 True
register 435345 True
register 54645 True
register 8787 True
register 213123 True
The database class has 7 students, and has 23 seats available.
[123, 56, 43, 435345, 54645, 8787, 213123]
drop 43 True
The database class has 6 students, and has 24 seats available.
[123, 56, 435345, 54645, 8787, 213123]
```

Inheritance

NEED TO ADD
Extending one class from another
call base class method (init)

Summary

Goes here

Important Terms

- attribute
- base class
- class
- def
- inheritance
- initializer
- method
- object
- property
- self
- string

Exercises

Here

Word Search


```
a t t r i b u t e c c u ·
c n z p n q y m j l y s h
q f q r i i m e q a n t ·
p k t o t z g t z s r r j
m s a p i w r h c s x i i
j e · e a j n o s r w n h
h l d r l x l d m g r g i
d f g t i e x d e f l m z
l r u y z i x c v e x x w
n b a s e · c l a s s d g
i n h e r i t a n c e d i
o b j e c t o s a u x t c
f n y o · · y k t p j t q
```

attribute, base class, class, def, inheritance, initializer, method, object, property, self, string

References