CS 157 - Introduction to Programming and Modeling

Object-oriented Programming

Dr. Stephen P. Carl

Quote of the Day

The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures...The magic of myth and legend has come true in our time.

- Fred Brooks

The Mythical Man-Month (1975)

Programming is Modeling

Software is a model of some "real" thing or process

- Audio programs model sound waveforms digitally
- Flight Simulators model the "look and feel" of piloting an aircraft
- Artificial Neural Networks model some abstraction of the brain
- Today's realistic games model military, physical, and biological processes

Modeling involves Abstraction

Abstraction means to whittle away details in order to distill a problem to its essence.

Software is complex; abstraction helps to tame the complexity. It allows us to focus on the *problem* rather than the details of the *solution.*

What is an Object?

Any value that can be *operated on* like numbers, strings, and lists, are considered objects. The *type* of an object dictates the possible range of values and legal operations for a particular set of values.

In programming language design terms, anything that can be assigned to a variable or passsed as an argument to a function is considered a *first-class* object.

In Python, functions and modules are also first-class objects!

In general, an object has a set of attributes and operations though not all objects necessarily have both.

SEWANEE
The University of the South

Components of an Object

An **attribute** is just a characteristic of the object. For example,

- Number: value, whole number or decimal, etc.
- List: length, values stored, etc.
- Car: price, engine type, body style, two- or four-door
- Graphical shape: kind of shape, color, size, location

Functions and modules also have attributes, such as __name__ and docstring (when defined).

SEWANEE
The University of the South

Components of an Object

Operations are things the object can do, or be done to the object:

- Number: add, subtract, multiply, etc.
- List: append, slice, etc.
- Car: accelerate, turn, stop, report speed
- Graphical shape: draw, move, rotate

SEWANEE
The University of the South

What is a Class?

All objects are an *instance* **of some** *class*

A class describes the attributes and operations which are specific to a particular set of objects of a particular type. You can think of a class as a *template* from which specific objects can be made.

We can use the `class` feature to create our own *types,* and therefore our own kinds of objects.

For example, the `graphics` module defines each component as a class with its own attributes and operations, which we've been using all this time.

SEWANEE
The University of the South

Class Hierarchies

A *class hierarchy* organizes the relationships between different classes of objects

Humans characterize objects into *class hierarchies* all the time:

- Biologists organize living things in *taxonomies*
- Sociologists organize people into *categories*
- We organize movies, music, and books into genres, and so forth

SEWANEE
The University of the South

The 3-Legged Stool of OOP

Encapsulation

- A class combines attributes and operations into a single unit.

Inheritance

- Sharing functionality between related classes.

Polymorphism

- Literally, "many shapes". Variables can refer to objects from related classes.

SEWANEE
The University of the South

Programming using Classes and Objects

We've just been through the most basic concepts of any object-oriented program. Now we'll begin to talk about object-oriented features of the Python language.

In the Python class construct, attributes are called instance variables and operations are called methods.

Each class is defined by code which describes the valid instance variables and the computations to be performed by the methods.

SEWANEE
The University of the South

Example Python Class

```
class Point:
   '''Class for representing (x,y) coordinates'''

   def __init__(self, x, y):
      ''' Create a new point at (x, y) '''
      self.x = float(x)
      self.y = float(y)
      self.setFill = 'black'
      self.setOutline = 'black'
```

SEWANEE
The University of the South

Example Python Class

```
# more methods for class Point:

    def move(self, dx, dy):
        ''' move the point by the given amounts dx, dy '''
        self.x = self.x + dx;
        self.y = self.y + dy;

    def getX(self):
        ''' return the x-coordinate '''
        return self.x

    def getY(self):
        ''' return the y-coordinate '''
        return self.y
```

SEWANEE
The University of the South

Creating Instances

A new instance of a class (an object) is created by invoking the *class constructor,* called using the class name. This automatically calls __init__ to create and initialize the instance variables for the new object. For example:

```
pt = Point(100, 100)  # creates a new Point: implicitly calls __init__
```

In constrast to *local variables* in regular functions, instance variables are shared by all methods of the class.

SEWANEE
The University of the South

Note to self

The keyword **`self`** in __init__ refers to the object being created. You can think of it as a special parameter specific to classes; it always comes first in the parameter list in each method.

When calling a method, **`self`** is passed implicitly; it takes on the value of the object making the method call. We call this the *invoking object.*

```
y = pt.getY()        # pt implicitly passed to getY method
```

In **`getY`** the parameter **`self`** is bound to the object **`pt`**

SEWANEE
The University of the South

Example from the Textbook

```
from random import randrange
class MSDie:
   '''Class for representing dice rolls'''
   def __init__(self, sides):
      self.sides = sides
      self.value = 1

   def roll(self):
      self.value = randrange(1, self.sides+1)

   def getValue(self):
      return self.value
```

SEWANEE
The University of the South