User Defined Classes

CS100 2017F

Why Classes?

- Sometimes it is natural to bundle data and behaviors together (technical term: *encapsulation*)
- Python supports this bundling by allowing the programmer to define a *class* (like, for example, the str class)
- A class may have arbitrarily many *instances*
  - this allows all members of the class to have uniform characteristics
  - for example, the class *Dog* might have instances *fido*, *barker*, and *scrawny*

Example: The Point Class

- We can define a class that describes points in the plane with this code
  - *class* is a Python keyword (required)
  - *Point* is the name of the class (by convention, it is capitalized)
  - the *colon* (:) introduces the indented body of the class definition
  - the first thing in the class definition should be a short *docstring*

```
class Point:
    ''' Represents a point in a Euclidean plane '''
```

Creating a Point

- A class typically has a method (function) named __init__ that is called to create an object in the class
  - the method name `__init__` by convention has two leading and and two trailing underscores to distinguish it from other identifiers
  - `self` refers to the object created. The other parameters define the initial state of the object

```
class Point:
    ''' Represents a point in a Euclidean plane '''
    def __init__(self, x_coor, y_coor):
        self.x = x_coor
        self.y = y_coor
```

Creating a Module for a Class

- Though it is not required, a class definition is usually saved in a separate .py file that can be imported as a module
  - e.g., the class Point would be saved in a file named point.py
  - the point module could then be imported and used to create points (think turtle module and turtle.Turtle())
  - the module may contain related classes (Turtle and Screen)
  - methods and data are accessed by the dot (.) operator

```
>>> import point
>>> point.Point
<class 'point.Point'>
>>> center = point.Point(0,0)
>>> center.x
0
```

Creating a Point

- Additional behaviors for a point object can be defined by adding methods to the class

```
class Point:
    ''' Represent a point in a Euclidean plane '''

    def __init__(self, x_coor, y_coor):
        self.x = x_coor
        self.y = y_coor

    def coordinates(self):
        ''' Return a tuple of the x,y coordinates of a point '''
        return (self.x, self.y)

    def move_to(self, x_coor, y_coor):
        ''' Assign new coordinates to a point '''
        self.x = x_coor
        self.y = y_coor
```

Extend the Point Class

- Write a method named `move` that moves a point relative to its current location (follow the pattern of the `move_to` method)
- Write a method named `distance_to` that calculates the distance between the current point and some other point.
- The code below shows correct input and output for these methods

```
>>> import point
>>> center = point.Point(0,0)
>>> point_a = point.Point(1,1)
>>> center.distance_to(point_a)
1.4142135623730951
>>> point_a.move(5,3)
>>> point_a.coordinates()
(6, 4)
```

Class Data

- A class may have data associated with it that applies to every object in the class
- For example, the dimension of every point is 0. This should be defined inside the class, but outside any method.

```
class Point:
    ''' Represents a point in a Euclidean plane '''
    dimension = 0
```

```
>>> import point
>>> a_point = point.Point(2,2)
>>> point.Point.dimension
0
>>> a_point.dimension
0
```

Sugar Knows Frisbee