Classes, Initialization, and Equality

Structure of a class

    def class <name>

        # assign data members

        # methods (functions)

            # constructor and other bookkeeping methods

            # computational methods

Let's take a look at a sample class, Time. See (cs.usfca.edu/~wolber/courses/110/python/time0.py)

    class Time:
        hour=0
        minute=0
        second=0

        # main
        t = Time() # create an instance of time
        t.hour=18
        t.minute=3
        t.second=5
        print t.hour
        print t.minute
        print t.second

Teacher Demonstration

1. add a function outside the class Time called 'isNight', with a parameter of type Time and a call in main to it.

2. Rewrite the function as a method subordinate to the class Time.
Object Initialization (Constructors)

When we create an instance of a class, we often want to initialize its data members (fields).

We could create an instance of time set to 12:30:30 with the following code in main:

```python
# main
t = Time()
hour=12
minute=30
second=30
```

This is quite laborious. A constructor is a special method for initializing fields on creation.

In Python, a constructor is a method with the special name `__init__` (two underscores around "init")

Using a constructor, the code in the sample main can be reduced to one line.

Here's a modified class Time and modified main:

```python
class Time:
    def __init__(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds

# main
t = Time(12, 30, 30)
```
Object Equality

Consider the following:

```python
# main
t1 = Time(4,5,22)
t2 = Time(4,5,22)
if (t1==t2)
    print "t1 is same as t2"
```

Will anything be printed?

With objects, "==" is defined to compare the addresses of the two objects.

A programmer must define a method to compare the content of two objects. Python allows a programmer to redefine == for a class by using the special name `__eq__`.

```python
class Time:
    def __init__(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds
    def __eq__(self, other):
        if ((self.hours==other.hours) and
            (self.minutes==other.minutes) and
            (self.seconds==other.seconds)):
            return true
        else
            return false;
```

Given this `__eq__`, we'll get a different result for:

```python
# main
t1 = Time(4,5,22)
t2 = Time(4,5,22)
if (t1==t2)
    print "t1 is same as t2"
```
**String Representation**

When we call print on an object, Python does not print the contents of the object.

A programmer can write a method with the name \_\_str\_\_ to provide a string representation for an object (which can be printed)

class Time:
    def \_\_init\_\_(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds

    def \_\_eq\_\_(self, other):
        if ((self.hours==other.hours) and
            (self.minutes==other.minutes) and
            (self.seconds==other.seconds)):
            return true
        else
            return false

    def \_\_str\_\_(self):
        return str(self.hours) + ":" + str(self.minutes) + ":" + str(self.seconds)

*In-Class Problems*

1. Copy your Coordinate code into another file, then modify it so that it has a constructor (\_\_init\_\_) which allows the x and y coordinates to be set in the creation statement. Modify the main program so that it makes use of the constructor.

2. Write a \_\_eq\_\_ and \_\_str\_\_ method for Coordinate. Add code in the main that shows that these methods work.