More on Subclasses
Subclasses represent hierarchical information. A subclass inherits all of the properties -- the instance variables and the methods -- of its parent class.

Because of this inheritance, we usually start creating a subclass instance by running the parent class constructor. That way, if we change something about the parent class, the change is automatically passed down to the subclass.
But that makes a problem: how do we call the parent class constructor?
Suppose we have a parent class defined:

class Person:
    def __init__(self, name):
        self.name = name
        self.age = 0

and we want to construct a subclass:

class Student(Person):
    def __init__(self, name):
How does class call class Person's constructor? Only one of these makes sense:

A)   def __init__(self, name):
       __init__(self, name)

B)   def __init__(self, name):
       self.__init__(name)

C)   def __init__(self, name):
       Person.__init__(name)

D)   def __init__(self, name)
       Person.__init__(self, name)
In general, when you are inside a subclass and you want to call one of the parent class's methods that has been overridden in the subclass, you can do that with

<parent class name>.__str__(self)

For example,

    Person.__str__(self)
This is getting a bit weird, but if you have an object x of a subclass and you want to call one of its parent class's methods that has been overridden in the subclass, you can do it with

super(<subclass name>, x).<method name>(args)

as in

super(Student, x).Print( )
And if that isn't weird enough, Python allows *multiple inheritance*: a subclass can have multiple parent classes. This means we can define a class C as

```python
class C(A, B)
```

which means that C inherits all of the instance variables and methods of both class A and class B.
There are rules for what happens if class C is a subclass of both class A and class B, both of which happen to have methods with the same name that do different things. The best rule is

**DON'T USE MULTIPLE INHERITANCE**

Multiple inheritance is a nice idea gone bad. If you ever get into a situation where it seems like a good idea, go get some sleep and then redesign your code so you don't need to inherit from more than one class.