Object-Oriented Programming

Basic class operations ("glorified struct")

In [1]:
```
class MyClass(object):
    a = 5
    b = 6
print MyClass
print MyClass.a
```

```
<class '__main__.MyClass'>
5
```

In [3]:
```
obj = MyClass()
print obj.a
print obj.b
```

```
5
6
```

In [4]:
```
obj.a = 'new value for a'
print obj.a
```

```
new value for a
```

In [5]:
```
print MyClass.a
```

```
5
```

Basic methods

In [9]:
```
class MyClass(object):
    def say(self, something):
        print 'MyClass says', something
```

In [11]:
```
myobj = MyClass()
myobj.say('Hello')
```

```
MyClass says Hello
```

In [14]:
```
# Classes can have a constructor

class MyClass(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

obj = MyClass('avalue', 'bvalue')
print obj.a
```

```
avalue
```

Method access and visibility

By convention, a single leading underscore indicates that a method or variable is "protected" and should not be modified outside the class:

In [16]:
```
class MyClass(object):
    def __init__(self, a, b):
        self._a = a
        self._b = b
    def value(self):
        return self._a, self._b

obj = MyClass('avalue', 'bvalue')
print obj.value()
print obj._a
```

```
('avalue', 'bvalue')
avalue
```

To "enforce" private methods / variables, we can use a *double* leading underscore:

In [17]:
```
class MyClass(object):
    def __init__(self, a, b):
        self.__a = a
        self.__b = b
    def value(self):
        return self.__a, self.__b

obj = MyClass('avalue', 'bvalue')
print obj.value()
```

```
('avalue', 'bvalue')
```

In [18]:
```
print obj.__a
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/vagrant/<ipython-input-18-383b7dc3f3c4> in <module>()
----> 1 print obj.__a

AttributeError: 'MyClass' object has no attribute '__a'
```

In [21]:
```
print obj._MyClass__a
```

```
avalue
```

Exercise

Write a class to manage a telephone directory. The directory should be stored in a dict as an instance variable. The class should have methods `add_number(name, number)`, `remove_number(name)`, and `lookup_number(name)`.