面向对象编程

编程范式

- 面向对象编程——Object Oriented Programming,简称 OOP,把对象作为程序的基本单元,一个对象包含了数据和操作数据的函数。
- 面向过程把函数继续切分为子函数,来降低系统的复杂度。

基础概念

- **类:**在 Python 中,所有数据类型都可以视为对象,当然也可以自定义对象。自定义的对象数据类型就是面向对象中的类(Class)。
- OOP首选思考的不是程序的执行流程,而是某个数据类型应该被视为一个对象,这个对象拥有的属性(Property)。
- **方法:**给对象发消息实际上就是调用对象对应的关联函数,我们称之为对象的方法(Method)。

类

```
class Student(object):

    def __init__(self, name, score):

        self.name = name

        self.score = score

    def print_score(self):

        print '%s: %s' % (self.name, self.score)
```

类

- Class 是一种抽象概念,比如我们定义的 Class——Student,

是指学生这个概念;

- 实例(Instance)则是一个个具体的 Student;
- 面向对象的抽象程度又比函数要高,因为一个 Class 既包含数

据,又包含操作数据的方法。

类

- object表示该类是从哪个类继承下来的。通常,如果没有合适的继承类,就使用 object 类,这是所有类最终都会继承的类。
- 创建实例的方式：student1 = Student()
- 可以自由地给实例变量绑定属性，student1.name,student1.score;
- 可定义一个特殊的 __init__ 方法,在创建实例的时候,就把name，score等属性绑上去

类

- __init__ 方法的第一个参数永远是 self ,表示创建的实例本身;
- 在类中定义的函数只有一点不同,就是第一个参数永远是实例变量self ,

并且调用时,不用传递该参数。

数据封装

- 数据和逻辑被“封装”起来了,调用很容易,但却不用知道内部实现的细节。
- 封装的另一个好处是可以给类增加新的方法;

访问限制

- 在 Python 中,实例的变量名如果以 __ 开头,就变成了一个私有变量(private);
- 双下划线开头的实例变量是不是一定不能从外部访问呢?NO

继承和多态

```
class Animal(object):
    def run(self):
        print 'Animal is running...'
class Dog(Animal):
    pass
class Cat(Animal):
    pass
```

继承和多态

- 对于 Dog 来说,Animal 就是它的父类,对于 Animal 来说,Dog 就是它的子类;
- 继承最大的好处是子类获得了父类的全部功能。
- 继承的另一个好处:多态。子类的覆盖了父类的方法。

获取对象信息

- 使用 type();
- 使用 isinstance();
- 使用 dir()

over !