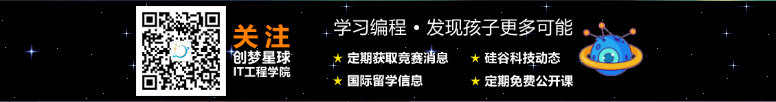
****

1. **python基础：如何使用类和对象**

一：概述

**Python从设计之初就已经是一门面向对象的语言，正因为如此，在Python中创建一个类和对象是很容易的。本章节我们将详细介绍Python的面向对象编程。**

**如果你以前没有接触过面向对象的编程语言，那你可能需要先了解一些面向对象语言的一些基本特征，在头脑里头形成一个基本的面向对象的概念，这样有助于你更容易的学习Python的面向对象编程。**

二：学习目标

**1：如何根据一系列事物来创建一些类，并用这些类来生产类的对象。**

**2：学会子类如何继承父类中的函数的**

**3：了解面向对象的三大特点之一：继承**

三：开始学习

* **面向对象的简介：**

**1：类(class):用来描述具有相同属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。对象是类的实例。**

**2：类变量：类变量在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。**

**3：数据成员：类变量或者实例变量用于处理类及其实例对象的相关的数据。**

**4：方法重写：如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖，也称为方法的重写**

**5:实例变量：定义在方法中的变量，只作用于当前实例的类。**

**6：继承：即一个派生类继承基类的字段和方法。继承也许把一个派生类的对象作为一个基类对象对待。**

* **类定义**

class ClassName:

<statement-1>

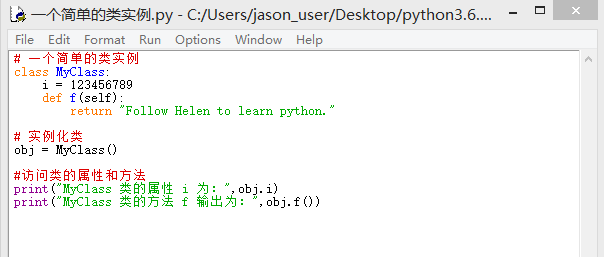
. . .

<statement-N>

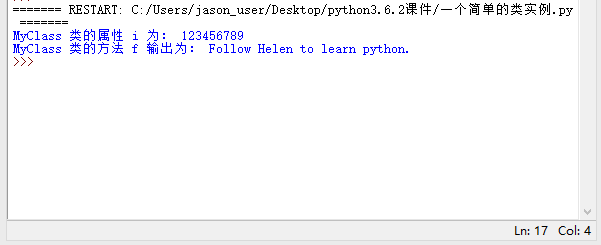
**类实例化后，可以使用其属性。实际上，创建一个类之后，可以通过类名访问其属性。**

* **类对象**

**类对象支持两种操作：属性引用和实例化**

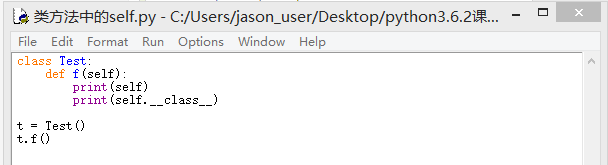


输出结果：

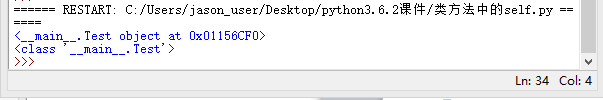


**以上就体现了属性引用和实例化两个操作，其中类方法中self代表类的实例，而非类。类的方法与普通的函数（方法）只有一个特别的区别：类方法必须有一个额外的第一个参数名称，按照惯例它的名称是self。**

**我们在来看一个案例来理解类方法的这个额外参数：**



**输出结果：**

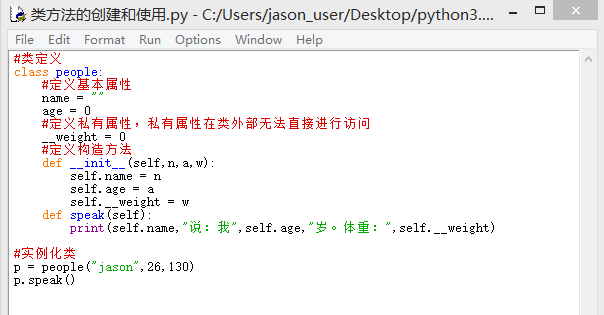


**从执行结果可以很明显的看出，self代表的是类的实例，代表当前对象的地址，而self.class则指向类。**

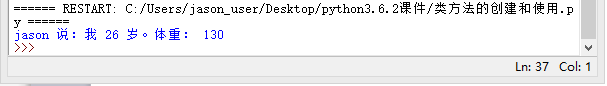
**要注意self不是python关键字，我们把他换成Jason也是可以正常执行的。**

* **类的方法**

**在类的内部，使用def关键字来定义一个方法，与一般函数定义不同，类方法必须包含一个额外参数self，且为第一个参数，self代表的是类的实例。**



**运行结果如下：**



* **继承**

**python类继承的语法格式如下**

class DerivedClassName(BaseClassName1):

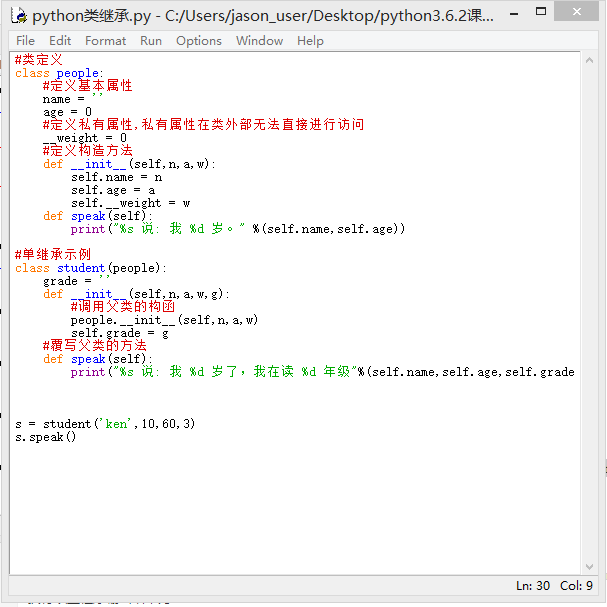
<statement-1>

. . .

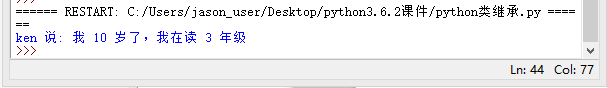
<statement-N>

**需要注意圆括号中基类的顺序，若是基类中有相同的方法名，而在子类使用时未指定，python从左至右搜索 即方法在子类中未找到时，从左到右查找基类中是否包含方法。**

**BaseClassName（示例中的基类名）必须与派生类定义在一个作用域内。**

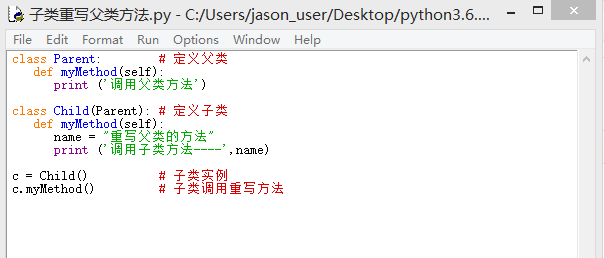


**输出结果：**

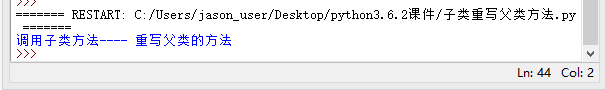


* **方法重写**

**如果你的父类方法的功能不能满足你的需求，你可以在子类重写你父类的方法实例如下**



运行结果如下:



四：作业