**Python学习笔记**

**第二部分：面向对象编程（10.4-10.11）**

10.4

## 一、引入内容

1、面向对象最重要的概念就是类（Class）和实例（Instance）

* 类，类型的简称，相当于自定制的新的类型
* 类是模板，是抽象的
* 而实例是具体的，根据类创建出来的一个个具体的“对象”
* 每个对象都拥有相同的方法，但各自的数据可能不同。
* 类和函数一样，可以大大地减少代码的重复编写

2、def \_\_init\_\_：\_\_init\_\_(self) 初始化，\_\_new\_\_实例化方法,两者执行的顺序，先有实例，才能初始化。**init**()方法意义重大的原因有两个。第一个原因是在对象生命周期中初始化是最重要的一步；每个对象必须正确初始化后才能正常工作。第二个原因是**init**()参数值可以有多种形式。

## 二、类与实例

1、class后面紧接着是类名，即Student，类名通常是大写开头的单词，紧接着是(object)，表示该类是从哪个类继承下来的，继承的概念我们后面再讲，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 [可以不写]。

2、定义好了Student类（它是一个新类型），就可以根据Student类（型）创建出Student的实例（变量），创建实例是通过类名+()实现的：

3、由于类可以起到模板的作用，因此，可以在创建实例的时候，把一些我们认为必须绑定的属性强制填写进去。通过定义一个特殊的\_\_init\_\_方法，在创建实例的时候，就把name，score等属性绑上去：

* 在python中方法名如果是\_\_xxxx\_\_()的，那么就有特殊的功能，因此叫做“魔法”方法
* 当使用print输出对象的时候，只要自己定义了\_\_str\_\_(self)方法，那么就会打印从在这个方法中return的数据
* \_\_str\_\_方法需要返回一个字符串，当做这个对象的描写
* \_\_repr\_\_ 也是用来描述对象的，一般和 \_\_str\_\_方法是一样的

4、**数据封装** 面向对象编程的一个重要特点就是数据封装。在上面的Student类中，每个实例就拥有各自的name和score这些数据。我们可以通过函数来访问这些数据，比如打印一个学生的成绩：

既然Student实例本身就拥有这些数据，要访问这些数据，就没有必要从外面的函数去访问，可以直接在Student类的内部定义访问数据的函数，这样，就把“数据”给封装起来了。这些封装数据的函数是和Student类本身是关联起来的，我们称之为**类的方法**：

**小结：**

* 类是创建实例的模板，是抽象的，而实例则是一个一个具体的对象，各个实例拥有的数据都互相独立，互不影响；
* 方法就是与实例绑定的函数，和普通函数不同，方法可以直接访问实例的数据；
* 通过在实例上调用方法，我们就直接操作了对象内部的数据，但无需知道方法内部的实现细节。
* 和静态语言不同，Python允许对实例变量绑定任何数据，也就是说，对于两个实例变量，虽然它们都是同一个类的不同实例，但拥有的变量名称都可能不同：

## 三、访问限制

在Class内部，可以有属性和方法，而外部代码可以通过直接调用实例变量的方法来操作数据，这样，就隐藏了内部的复杂逻辑。但是，从前面Student类的定义来看，外部代码还是可以自由地修改一个实例的name、score属性；

如果要让内部属性不被外部访问，可以把属性的名称前加上两个下划线\_\_，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)，只有内部可以访问，外部不能访问，所以，我们把Student类改一改；这样就确保了外部代码不能随意修改对象内部的状态，这样通过访问限制的保护，代码更加健壮。

## 四、继承和多态

**1、继承**

在OOP程序设计中，当我们定义一个class的时候，可以从某个现有的class继承，新的class称为子类（Subclass），而被继承的class称为基类、父类或超类（Base class、Super class）。继承可以把父类的所有功能都直接拿过来，这样就不必从零做起，子类只需要新增自己特有的方法，也可以把父类不适合的方法覆盖重写。

**2、多态**

要理解什么是多态，我们首先要对数据类型再作一点说明。当我们定义一个class的时候，我们实际上就定义了一种数据类型。我们定义的数据类型和Python自带的数据类型，比如str、list、dict没什么两样。

新增一个Animal的子类，不必对run\_twice()做任何修改，实际上，任何依赖Animal作为参数的函数或者方法都可以不加修改地正常运行，原因就在于多态。多态的好处就是，当我们需要传入Dog、Cat、Tortoise……时，我们只需要接收Animal类型就可以了，因为Dog、Cat、Tortoise……都是Animal类型，然后，按照Animal类型进行操作即可。由于Animal类型有run()方法，因此，传入的任意类型，只要是Animal类或者子类，就会自动调用实际类型的run()方法，这就是多态的意思：

对于一个变量，我们只需要知道它是Animal类型，无需确切地知道它的子类型，就可以放心地调用run()方法，而具体调用的run()方法是作用在Animal、Dog、Cat还是Tortoise对象上，由运行时该对象的确切类型决定，这就是多态真正的威力：调用方只管调用，不管细节，而当我们新增一种Animal的子类时，只要确保run()方法编写正确，不用管原来的代码是如何调用的。甚至于，连Animal的子类都不用，只要它有一个run方法就行！！！这就是著名的“开闭”原则：

对扩展开放：允许新增Animal子类；允许“类似”Animal类

对修改封闭：不需要修改依赖Animal类型的run\_twice()等函数。

## 五、获取对象信息

**1、函数获取**

type()

isinstance()

dir()：返回一个对象的所有属性和方法

getattr()

setattr()

hasattr()

类似\_\_ xxx\_\_的属性和方法在Python中都是有特殊用途的，比如\_\_len\_\_方法返回长度。 在Python中，如果你调用len()函数试图获取一个对象的长度，实际上，在len()函数内部， 它自动去调用该对象的\_\_ len\_\_()方法。仅仅把属性和方法列出来是不够的，配合getattr()、setattr()以及hasattr()，我们可以直接操作一个对象的状态

## 六、实例属性和类属性

由于Python是动态语言，根据类创建的实例可以任意绑定属性。给实例绑定属性的方法是通过实例变量，或者通过self变量。

从上面的例子可以看出，在编写程序的时候，千万不要对实例属性和类属性使用相同的名字，因为相同名称的实例属性将屏蔽掉类属性，但是当你删除实例属性后，再使用相同的名称，访问到的将是类属性。

注：

实例属性属于各个实例所有，互不干扰；

类属性属于类所有，所有实例共享一个属性；

不要对实例属性和类属性使用相同的名字，否则将产生难以发现的错误。

## 七、面向对象高级编程

1、\_\_slots\_\_：限制class实例能添加的属性；

2、多重继承