例

Class Student(object):

Def \_\_init\_\_(self, name, score):

Self.name = name

Self.score = score

Def print\_score(self):

Printf('%s:%s' %(self.name, self.score))

仍以Student类为例，在Python中，定义类是通过class关键字：

class Student(object):  
    pass

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。

IMG_256 访问限制

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：

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))

IMG_257 静态语言 vs 动态语言

对于静态语言（例如Java）来说，如果需要传入Animal类型，则传入的对象必须是Animal类型或者它的子类，否则，将无法调用run()方法。

对于Python这样的动态语言来说，则不一定需要传入Animal类型。我们只需要保证传入的对象有一个run()方法就可以了：

class Timer(object):  
    def run(self):  
        print('Start...')

这就是动态语言的“鸭子类型”，它并不要求严格的继承体系，一个对象只要“看起来像鸭子，走起路来像鸭子”，那它就可以被看做是鸭子。

Python的“file-like object“就是一种鸭子类型。对真正的文件对象，它有一个read()方法，返回其内容。但是，许多对象，只要有read()方法，都被视为“file-like object“。许多函数接收的参数就是“file-like object“，你不一定要传入真正的文件对象，完全可以传入任何实现了read()方法的对象

IMG_258 获取对象信息

使用type()

1）判断对象类型，使用type()函数：

基本类型都可以用type()判断：

>>> type(123)  
<class 'int'>  
>>> type('str')  
<class 'str'>  
>>> type(None)  
<type(None) 'NoneType'>

如果一个变量指向函数或者类，也可以用type()判断：

>>> type(abs)  
<class 'builtin\_function\_or\_method'>  
>>> type(a)  
<class '\_\_main\_\_.Animal'>

2）判断一个对象是否是函数怎么办？可以使用types模块中定义的常量：

>>> import types  
>>> def fn():  
...     pass  
...  
>>> type(fn)==types.FunctionType  
True  
>>> type(abs)==types.BuiltinFunctionType  
True  
>>> type(lambda x: x)==types.LambdaType  
True  
>>> type((x for x in range(10)))==types.GeneratorType  
True

使用isinstance()

对于class的继承关系来说，使用type()就很不方便。我们要判断class的类型，可以使用isinstance()函数。

使用dir()

如果要获得一个对象的所有属性和方法，可以使用dir()函数，它返回一个包含字符串的list，比如，获得一个str对象的所有属性和方法。

IMG_259 实例属性和类属性

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。