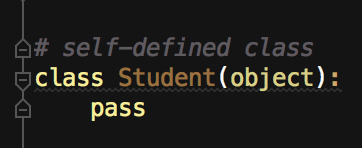
# Python - OOP

1. Class and Instance in python

## 类class –抽象的模版

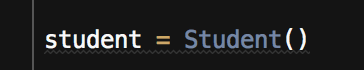
## 实例 instance – 根据类创建出来的一个具体的对象，每一个对象都从类中继承相同的方法，但各自的数据都有可能不同

* Python中自定义类：



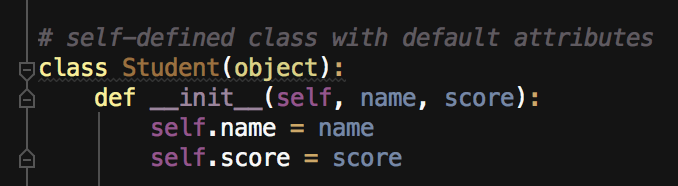
(object)表示从哪个类继承下来的 【object类是所有类都会继承的类】

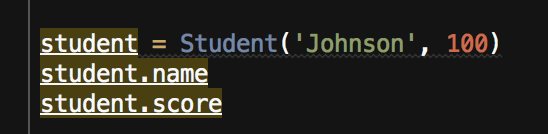
* 创建实例：



在定义类时可以绑定必须的类属性【使用\_\_init\_\_方法】：

【~initialize default attribute】

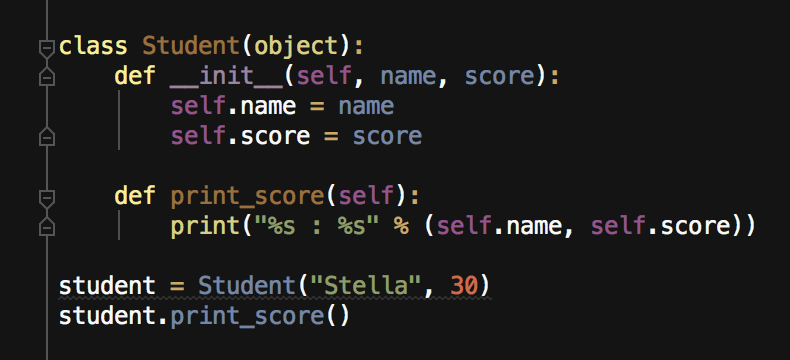




这里注意， 1. \_\_init\_\_方法第一参数永远是self，表示创建的实例本身，所以在\_\_init\_\_内部可以把属性绑定到self，因为self就指向创建的实例本身；2. 有\_\_init\_\_方法，在创建实例的时候就要按照\_\_init\_\_的参数列表来传递参数创建实例【不需传self】

类中的函数：1. 第一参数永远是self并且调用时不需要显式传递该参数 2.其它参数可以是默认参数 | 可变参数 | 关键字参数 🡪 （） | \*args 是可变参数，\*args 接收的是tuple | \*\*kw 是关键字参数， \*\*kw接收的是 dict

封装 🡪 在类中定义对类属性进行操作的方法，类的方法



封装是一种思想：调用很容易，而无需关注内部实现细节

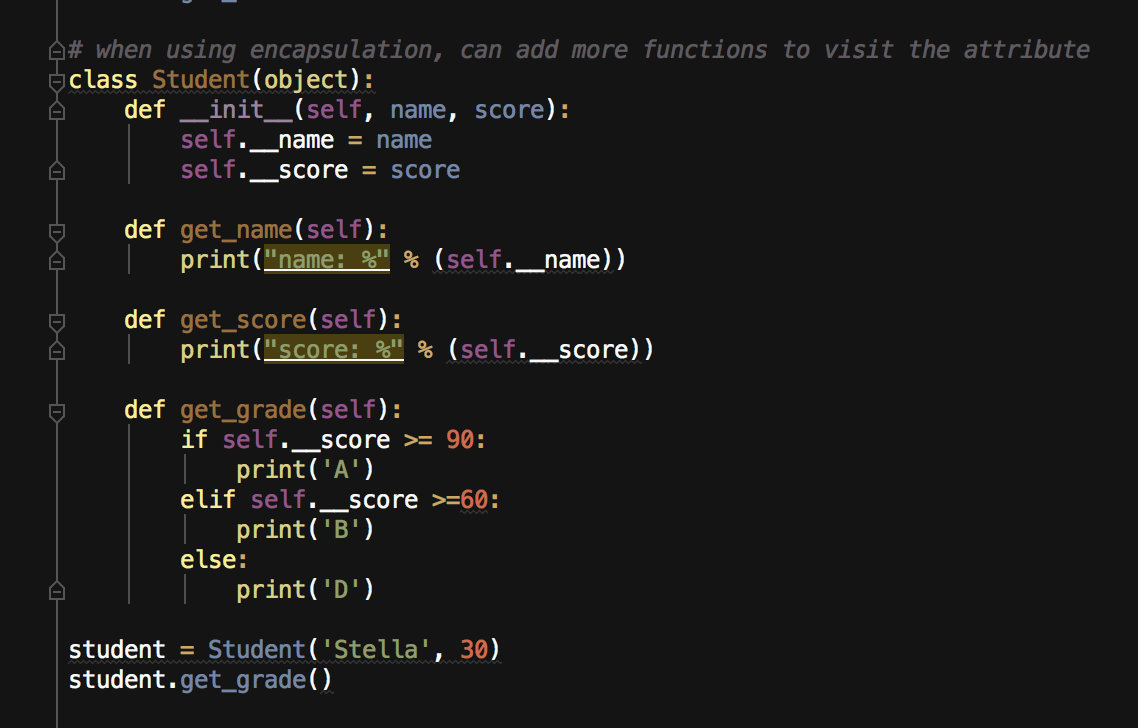
Python中的私有(private)变量：

变量名以’\_\_’开头，对于私有变量，无法从外部直接访问【访问限制，代码健壮】



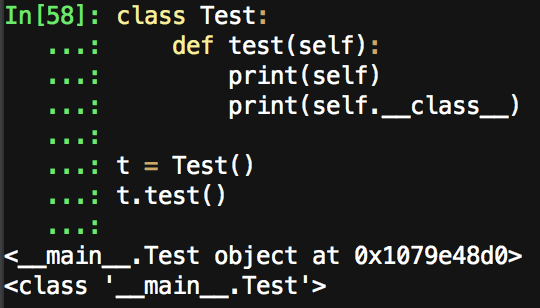
get()方法和 set()方法提供了访问以及修改内部属性的接口

注意：以\_\_开头和以\_\_结尾的变量是特殊变量【public属性】



关于self：

1. Self表示类的实例，而非类



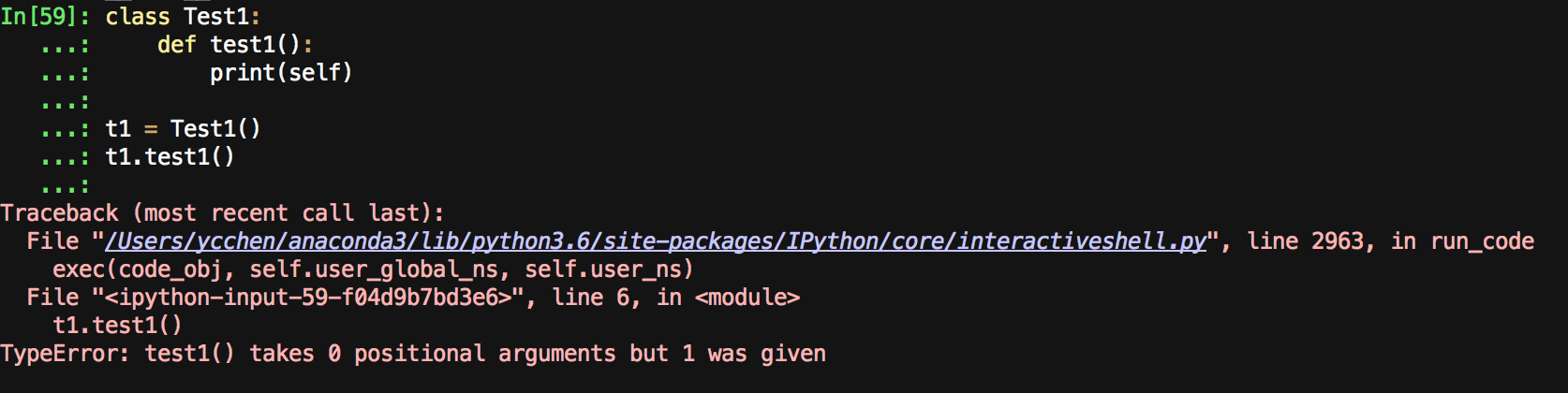
从输出中可以看出，self是类Test的一个实例对象

而这个实例的类是Test

所以self应该看作是类的一个实例

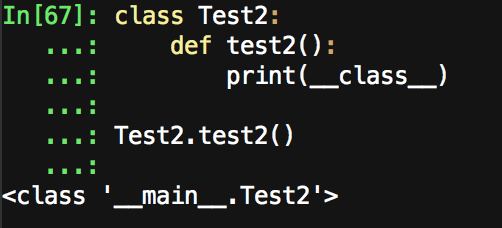
1. Self在定义方法时不可以省略，如果省略了调用时要注意不能传入类实例

定义函数时没有传入self，而调用时使用实例来调用就会报错：

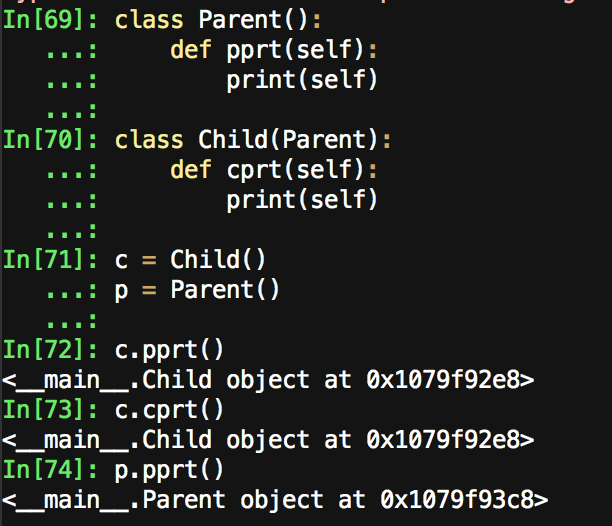


其实t1.test1()相当于是Test1.test1(t1)，所以会报错

定义时没有传入self，调用时直接使用类来调用是可以的【类方法】



1. 在继承时，传入的是哪个实例，就是那个传入的实例而不是定义了self的类的实例



C是Child的一个实例，所以self就是Child的一个实例好的对象

其实c.pprt()等同于Child.pprt(c) self指的是child类的一个实例