## python class 用法

 (2011-12-26 17:26:47)

[IMG_256转载▼](http://blog.sina.com.cn/s/javascript:;)

|  |  |
| --- | --- |
| 标签： [杂谈](http://search.sina.com.cn/?c=blog&q=%D4%D3%CC%B8&by=tag" \t "http://blog.sina.com.cn/s/_blank) | 分类： [python](http://blog.sina.com.cn/s/articlelist_1995001121_6_1.html" \t "http://blog.sina.com.cn/s/_blank) |

# **Python 类的定义、继承及使用对象**

作者：保⑩洁

本文转载：<http://www.17jo.com/program/python/base/ClassUse.html>

Python编程中类的概念可以比作是某种类型集合的描述，如“人类”可以被看作一个类，然后用人类这个类定义出每个具体的人——你、我、他等作为其对象。类还拥有属性和功能，属性即类本身的一些特性，如人类有名字、身高和体重等属性，而具体值则会根据每个人的不同；功能则是类所能实现的行为，如人类拥有吃饭、走路和睡觉等功能。具体的形式如下：

# 例：类的概念

class 人类:

名字 = '未命名' # 成员变量

def 说话(内容): # 成员函数

print 内容 # 成员变量赋初始值

某人 = 人类() # 定义一个人类对象某人

某人.名字 = "路人甲"

某人.说话 ('大家好') # 路人甲说话

>>> 大家好！ # 输出

示例程序一（**类的定义**）：

>>> class pp:

...     pass

...

>>> p = pp()

>>> print p

<\_\_main\_\_.pp instance at 0x00CA77B0>

>>>

打印了这个变量的类型。它告诉我们我们已经在\_\_main\_\_模块中有了一个Person类的实例。

示例程序二（**\_\_init\_\_用法）**：

说明：\_\_init\_\_方法在类的对象被建立时，马上运行。该方法用来对对象进行初始化。

>>> class Person:

...     def \_\_init\_\_(self, name):

...             self.name = name

...     def sayHi(self):

...             print 'Hello, my name is', self.name

...

>>> p = Person('Swaroop')

>>> p.sayHi()

Hello, my name is Swaroop

>>>

示例程序三（**\_\_del\_\_方法**）：

说明：\_\_del\_\_方法是在程序退出时调用的。

>>> class Person:

...     population = 0

...     def \_\_init\_\_(self, name):

...             self.name = name

...             print '(Initializing %s)' % self.name

...     def \_\_del\_\_(self):

...             print '%s says bye.' % self.name

...             Person.population -= 1

...

...     def howMany(self):

...             if Person.population == 1:

...                     print 'I am the only person here.'

...             else:

...                     print 'We have %d persons here.' % Person.population

...

>>> A = Person('aa')

(Initializing aa)

>>> A.howMany()

We have 0 persons here.

>>> B = Person('bb')

(Initializing bb)

>>> B.howMany()

We have 0 persons here.

>>> ^Z

aa says bye.

bb says bye.

Python中定义和使用类的形式为：class 类名[(父类名)]:[成员函数及成员变量]，类名为这个类的名称，而父类名为可选，但定义父类名后，子类则拥有父类的相应属性和方法。在用类定义成对象时，会先调用\_\_init\_\_构造函数，以初始化对象的各属性，类的各属性（成员变量）均可以在构造函数中定义，定义时只要加上对象指针就好了。而在对象销毁时，则会调用\_\_del\_\_析构函数，定义类的成员函数时，必须默认一个变量（类似于C++中的this指针）代表类定义的对象本身，这个变量的名称可自行定义，下面例子将使用self变量表示类对象变量。

# 例：类定义及使用

class CAnimal:

name = 'unname' # 成员变量

def \_\_init\_\_(self,voice='hello'): # 重载构造函数

self.voice = voice # 创建成员变量并赋初始值

def \_\_del\_\_(self): # 重载析构函数

pass # 空操作

def Say(self):

print self.voice

t = CAnimal() # 定义动物对象t

t.Say() # t说话

>> hello # 输出

dog = CAnimal('wow') # 定义动物对象dog

dog.Say() # dog说话

>> wow # 输出

Python编程中类可以承继父类属性，形式为class 类名（父类），子类可以继承父类的所有方法和属性，也可以重载父类的成员函数及属性，须注意的是子类成员函数若重载父类（即名字相同），则会使用子类成员函数

# 例：类的继承

class CAnimal:

def \_\_init\_\_(self,voice='hello'): # voice初始化默认为hello

self.voice = voice

def Say(self):

print self.voice

def Run(self):

pass # 空操作语句（不做任何操作）

class CDog(CAnimal): # 继承类CAnimal

def SetVoice(self,voice): # 子类增加函数

SetVoice self.voice = voice

def Run(self,voice): # 子类重载函数Run

print 'Running'

bobo = CDog()

bobo.SetVoice('My Name is BoBo!') # 设置child.data为hello

bobo.Say()

bobo.Run()

>> My Name is BoBo!

>> Running