Python类函数、实例函数、静态函数

**实例方法**

定义：第一个参数必须是实例对象，该参数名一般约定为“self”，通过它来传递实例的属性和方法（也可以传类的属性和方法）；

调用：只能由实例对象调用。

**类方法**

定义：使用装饰器@classmethod。第一个参数必须是当前类对象，该参数名一般约定为“cls”，通过它来传递类的属性和方法（不能传实例的属性和方法）；

调用：实例对象和类对象都可以调用。

**静态方法**

定义：使用装饰器@staticmethod。参数随意，没有“self”和“cls”参数，但是方法体中不能使用类或实例的任何属性和方法；

调用：实例对象和类对象都可以调用。

**class** **A**:

count = 0

**def** \_\_init\_\_(self, inst\_name):

self.inst\_name = inst\_name

self.\_\_class\_\_.count += 1

**def** inst\_method(self):

**print** '实例（**%s**）：**%s**' % (self.\_\_class\_\_.count, self.inst\_name)

@classmethod

**def** class\_method(cls):

**print** cls.count

@staticmethod

**def** static\_method():

**print** 'hello'

a1,a2,a3,a4 = A('a1'),A('a2'),A('a3'),A('a4')

a1.inst\_method()

a1.class\_method() # 或 A.class\_method()

a1.static\_method() # 或 A.static\_method()

**类实例方法：第一个参数强制为类实例对象，可以通过这个类实例对象访问类属性，可以通过类实例对象的\_\_class\_\_属性访问类属性。**

def inst\_method(self):

print '实例（%s）：%s' % (self.\_\_class\_\_.count, self.inst\_name)

类实例方法不需要标注，第一个参数必不可少，解析器自动会将**类实例对象**传给方法的第一个参数。

类的初始化方法\_\_init\_\_也是实例方法，在实例创建的时候自动调用。在这里每当初始化一个实例，就通过\_\_class\_\_来访问类属性count，是它加一，用来统计类的实例数。

def \_\_init\_\_(self, inst\_name):

self.inst\_name = inst\_name

self.\_\_class\_\_.count += 1

**类方法：第一个参数强制为类对象，可以通过这个类对象访问类属性，由于没有传入类实例对象，所以不能访问类实例属性。**

@classmethod

def class\_method(cls):

print cls.count

类方法需要使用classmethod标注，第一个参数必不可少，解析器自动会将类对象传给方法的第一个参数。

**类静态方法：无法访问类属性、类实例属性、没有默认的第一个参数，其实跟类没什么关系，只是绑定在类命名空间下的函数而已。**

@staticmethod

def static\_method():

print 'hello'

类静态方法通常用来定义一些和类主题相关的函数。

**通过类对象可以调用类方法、类静态方法，但不可以调用类实例方法；通过类实例对象可以调用以上三种方法**

a1,a2,a3,a4 = A('a1'),A('a2'),A('a3'),A('a4')

a1.inst\_method()

a1.class\_method() # 或 A.class\_method()

a1.static\_method() # 或 A.static\_method()

#类函数和静态函数

class People(object):

# 类变量

total= 0

def \_\_init\_\_(self, name, age):

# 调用父类的初始化函数

super(People, self).\_\_init\_\_()

#初始化当前类对象的一些属性

self.name = name

self.age = age

#对象函数，只能由对象调用

def eat(self):

print('吃。。。。。')

#1.类函数

#装饰器是以@开头，@结构的称之为语法糖,装饰器的作用主要是给一些现有的函数增加一些额外的功能

@classmethod

def work(cls,time,\*args,\*\*kwargs):

#cls class 如果是类调用该函数，cls指的是这个类

#如果是对象调用该函数，cls指的就是这个对象的类型

print(cls)

print(time)

@classmethod

def sleep(cls):

print('每一个类函数前必须添加装饰器@classmethod')

# 2.静态函数

# @staticmethod 描述的函数称为静态函数，静态可以由类和对象调用，函数中没有隐形参数

@staticmethod

def run(time):

print('跑步%s小时。。。'%time)

# 对象函数只能由对象调用

# 类函数由类调用，也可以用对象调用

People.work(10)

p1 = People('张三',22)

p1.work(20)

People.run(50)

p1.run(6)

People.sleep()

注意在类的内部，使用def关键字可以为类定义一个函数（方法），与一般函数定义不同，类方法必须包含参数self,且为第一个参数！

此外，可使用 \*\*kw定义关键参数，代表任意参数，python函数可变参数及关键字参数定义参考见下文。

python函数只能先定义再调用！

class person():

def \_\_init\_\_(self,name,gender,birth,\*\*kw):

self.name=name

self.gender=gender

self.birth=birth

for k,w in kw.iteritems():

setattr(self,k,w)

def sayhi(self):

print 'my name is',self.name

xiaoming = person('Xiao Ming', 'Male', '1991-1-1',job='student',tel='18089355',stdid='15010')

xiaohong = person('Xiao Hong', 'Female', '1992-2-2')

print xiaoming.name

print xiaohong.birth

print xiaoming.job

print xiaoming.tel

print xiaoming.stdid

print xiaoming.sayhi()

运行结果

Xiao Ming

1992-2-2

student

18089355

15010

my name is Xiao Ming

None

在本例中建立一个person的类，并赋值了两个实例：xiaoming和xiaohong , self指的是传入的实例（不同实例类的属性值不同以及方法执行结果不同）即xiaoming和xiaohong，类的属性即为name,gender,birth及其他kw，类的方法即为sayhi。