

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：

```
class Student(object):
    def __init__(self, name):
        self.name = name

s = Student('Bob')
s.score = 90
```

但是，如果 `Student` 类本身需要绑定一个属性呢？可以直接在 `class` 中定义属性，这种属性是类属性，归 `Student` 类所有：

```
class Student(object):
    name = 'Student'
```

当我们定义了一个类属性后，这个属性虽然归类所有，但类的所有实例都可以访问到。来测试一下：

```
>>> class Student(object):
...     name = 'Student'
...
>>> s = Student() # 创建实例s
>>> print(s.name) # 打印name属性，因为实例并没有name属性，所以会继续查找class的name属性
Student
>>> print(Student.name) # 打印类的name属性
Student
>>> s.name = 'Michael' # 给实例绑定name属性
>>> print(s.name) # 由于实例属性优先级比类属性高，因此，它会屏蔽掉类的name属性
Michael
>>> print(Student.name) # 但是类属性并未消失，用Student.name仍然可以访问
Student
>>> del s.name # 如果删除实例的name属性
>>> print(s.name) # 再次调用s.name，由于实例的name属性没有找到，类的name属性就显示出来了
Student
```

从上面的例子可以看出，在编写程序的时候，千万不要把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。

感觉本站内容不错，读后有收获？

¥ 我要小额赞助，鼓励作者写出更好的教程

还可以分享给朋友



珠峰NODE.JS全栈开发

技术陪伴成长社区 珠峰培训

百万级python导师亲身指导

保你120天 变身python大牛

有时候，你需要的只是一句点拨

立即咨询

掘金

一个只有高手分享
的技术社区

立即加入

深度学习在线课程

通向无人驾驶的必经之路

◀ [获取对象信息](#)

[面向对象高级编程](#) ▶

评论



请教：如何让函数传入参数，在函数内部生成以该参数为实例名称的类实例？

飞飞机的沐沐风 created at 10-21 11:30, Last updated at 9小时前

如题

在此插入代码

```
class a_class(object):
    def __init__(self, name):
        self.name = name

def a_function(arg, class_name):
    arg = a_class(class_name)
```

在这个函数中，我希望每次调用a_function的时候，能够让a_function生成一个以arg里的参数 为名

称的实例，但是上面代码的\

```
arg = a_class()
```

只能够生成一个以arg为名称的实例，它并不知道形参arg传入了什么。换言之我调用多次之后a_function之后并不能得到多个实例，应该则怎样实现这种功能呢？



enchantedl

Created at 10-24 19:11, Last updated at 10-24 19:11



```
class Person():  
    pass
```

```
def create(a):  
    ...    return a();  
    ...  
aa=create(Person)
```



summons_M

Created at 9小时前, Last updated at 9小时前

exec

```
class a_class(object):  
    name=' a_class'  
  
arg=' abc'  
exec(arg+'=a_class()')  
print(abc.name)
```

不建议这么做

View Full Discuss

Reply This Topic



测试代码

久疤_796 created at 2015-12-5 20:38, Last updated at 11-23 16:15

```
>>> class test(object):  
    num = 10  
  
>>> n1 = test()  
>>> n2 = test()  
>>> n1.num  
10  
>>> n2.num  
10  
>>> n1.num = 15  
>>> n2.num  
10
```

```
10
>>> n1.num
15
>>> test.num = 20
>>> n2.num
20
>>> n1.num
15
>>>
```

▼ [Read More](#)



[vforbox](#) [关注网络安全](#)

Created at 1-27 10:16, Last updated at 1-27 10:16

``
早已会! ^_^
``



[珉瑞斯特](#)

Created at 8-4 8:06, Last updated at 8-4 8:06

```
> > >echo '厉害'
666
```



[用户7032722563](#)

Created at 10-25 16:31, Last updated at 10-25 16:31

测试下，效果还不错~~~

```
print('Hello world!!!')
```



[我是YoungSir](#)

Created at 10-25 23:23, Last updated at 10-25 23:23

这是**Markdown**的语法吧

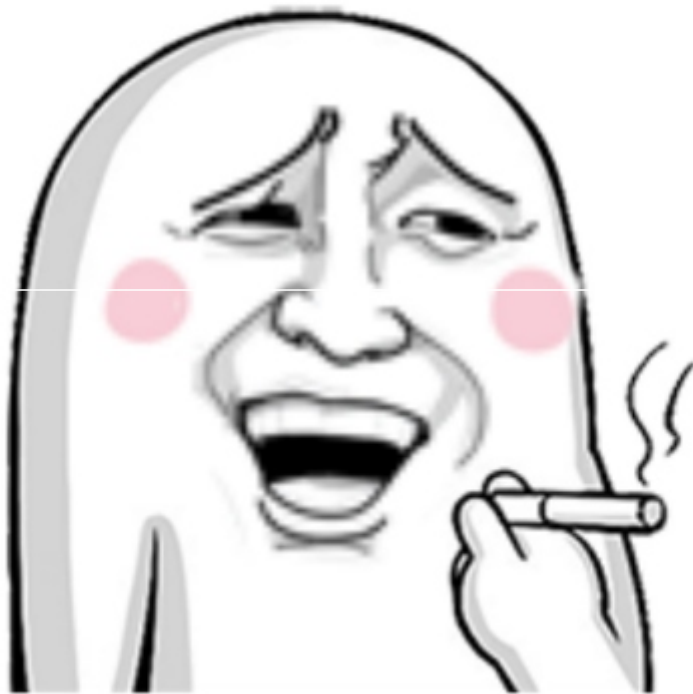


[桑桑喵喵i](#)

Created at 10-29 19:54, Last updated at 10-29 19:54

- [说的没错](#)
- [markdow](#)语法

莫装逼，装逼遭雷劈




小张小张从不慌张

Created at 11-23 16:15, Last updated at 11-23 16:15

666

蟹蟹这位大兄弟~

 View Full Discuss

 Reply This Topic



这个例子可以便于理解本节和上一节内容

xianlan6062356806 created at 11-3 16:12, Last updated at 11-21 23:06

这个例子可以便于理解本节和上一节内容

```
# 学生
class Student(object):
    # 用于记录已经注册学生数
    student_number = 0

    def __init__(self, name):
        self.name = name

# 注册一个学生:注册必填项名字, 选填项利用关键字参数传递。注册完成, 学生数+1
def register(name, **kw):
    a = Student(name)
    for k, v in kw.items():
        setattr(a, k, v)
    Student.student_number += 1
```

| return a |

▼ [Read More](#)



[NOTHINGBUTHARDWORK](#)

Created at 11-8 15:22, Last updated at 11-8 15:22



```
def register(name, **kw):  
    a = Student(name)  
    for k, v in kw.items():  
        setattr(a, k, v)  
    Student.student_number += 1  
    return a
```

请问这里的setattr(a, k, v)是什么意思?
是指赋了三个属性给register这个函数么?



[看海的狮子](#)

Created at 11-12 17:41, Last updated at 11-12 17:41

应该是a为self对象,
setattr(a, k, v)
如果对应
bob = register('bob', score = 99)
是
a = 'bob', k = score, v = 99.
对于python函数参数:args和*kw有一些特殊用法, 你可以去查一查。



[靠谱男青年](#)

Created at 11-21 23:06, Last updated at 11-21 23:06

```
def register(name, **kw):  
    a = Student(name)  
    for k, v in kw.items():  
        setattr(a, k, v)  
    Student.student_number += 1  
    return a
```

举例说明吧:

```
bob = register('Bob', score=90)
```

用'Bob'实例化Student类, 使用setattr把传进来的{'score':90}添加给之前实例化bob, 所以
bob.name = 'Bob', bob.score = 90

View Full Discuss

Reply This Topic



[关于实例和类的name属性](#)

[byebicycle](#) created at 8-8 10:25, Last updated at 10-25 16:25

```
class mvobject(object):
```

```
class myobject(object):
    def __init__(self, name):
        self.name = '123' #这里是不是相当于实例的name属性
        name = '456'
mary = myobject('mary')
print(mary.name)
print(myobject.name)
```

输出123和456



用户7032722563

Created at 10-25 16:25, Last updated at 10-25 16:25

在此插入代码

init方法是在对象实例化的时候才会调用，**mary**是一个**myobject**类的实例对象，实例化的时候调用了**init**方法，所以**mary.name**为123，但**print(myobject.name)**时并没有调用实例化**init**方法，所以**myobject.name**是456了~~~

View Full Discuss

Reply This Topic



请帮忙解惑一下！

用户5181363849 created at 6-21 11:28, Last updated at 10-15 20:49

```
class Word(object):
    def __init__(self, text):
        self.text=text
    def equals(self, word2):
        return self.text.lower()==word2.text.lower()

#这是正确的代码。
-----
class Word(object):
    def __init__(self, text):
        self.text=text
    def equals(self, word2):
        return self.text.lower()==word2.lower()
#这是错误的代码，为什么最后一句是word2.text.louwer()
```



houbo111

Created at 6-24 20:41, Last updated at 6-24 20:41

你这里**equals**方法是把两个实例进行比较，**word2**是该类的一个实例。因为是比较，所以**word2**应该和**self**是相同类型才能进行比较



碉你妹的堡

Created at 8-3 13:01, Last updated at 8-3 13:01

我运行你下面那个才正确啊 刚好跟你相反



Miles顽石不化

Created at 8-9 17:22, Last updated at 8-9 17:22

这和你world2的类型有关吧



我有一只小绵羊从来也不骑

Created at 10-15 20:49, Last updated at 10-15 20:49

这要看你需要比较对象的类型呀，看你比较对象是Word类还是一个str，两个都没什么错

```
class Word(object):
    def __init__(self, text):
        self.text = text
    def equals(self, word2):
        print(self.text.lower() == word2.lower())
    def euqals2(self, word3):
        print(self.text.lower() == word3.text.lower())

word1 = Word('ABC')
word2 = 'abc'
word3 = Word('abc')
word1.equals(word2)
word1.euqals2(word3)
```

View Full Discuss

Reply This Topic



交课堂笔记

陈祯飞 created at 10-7 22:49, Last updated at 10-7 22:49

```
>>> class Student(object):#object 竟然不是大写开头
...     name = 'Student'
...
>>> s = Student()#创建示例s
>>> print(s.name)#打印name属性，因为实例没有name属性，所以继续查找class的name属性
Student
>>> print(Student.name)#打印class属性
Student
>>> s.name = 'Yang' #给实例绑定name属性
>>> print(s.name)
Yang
>>> print(Student.name)
Student
>>> del s.name #注意del的用法
>>> print(s.name)
Student
>>>
```

View Full Discuss

Reply This Topic



对类属性进行限制后 用methodtype对类和实例分别动态添加方法 实例属性怎么分析



对类属性进行限制后，用methodtype对类属性添加动态添加方法，类属性是动态添加
用户5601807279 created at 7-28 13:22, Last updated at 9-4 13:02



```
# -*- coding: utf-8 -*-
from types import MethodType

# case1 :限制类的属性，对类进行动态添加方法
def set_age(self):
    self.age=16
class Student(object):
    __slots__ = ('name', 'score', 'set_age')
    pass

Student.set_age=MethodType(set_age, Student)
s=Student()
s.set_age()
print(s.age)
#.....

# case2 :限制类的属性，对实例进行动态添加方法
def set_age(self):
    self.age=16
class Student(object):
    __slots__ = ('name', 'score', 'set_age')
    pass
```



▼ [Read More](#)



[yikedaxibuoijiy](#)

Created at 9-4 13:02, Last updated at 9-4 13:02

在case1中添加的age属性，实际上是类属性：

```
from types import MethodType

def set_age(self):
    self.age = 16

class Student(object):
    __slots__ = ('name', 'score', 'set_age')
    pass

Student.set_age = MethodType(set_age, Student)
s = Student()
s.set_age()
print(id(s.age))
print(id(Student.age))
```

输出

▼ [Read More](#)

[View Full Discuss](#)

[Reply This Topic](#)



java转过来的毫无压力啊

云梦的微博 created at 6-28 13:34, Last updated at 7-1 17:20

用了很长时间java，用python作第二语言，感觉毫无压力啊= =



不完整的幽灵

Created at 7-1 17:20, Last updated at 7-1 17:20

哈哈，同...

View Full Discuss

Reply This Topic



关于【千万不要把实例属性和类属性使用相同的名字】

CloudAtlasN created at 5-11 17:21, Last updated at 5-28 19:42

如果不使用相同的名字，怎么给实例对象的属性赋值？



Drak_C

Created at 5-28 19:42, Last updated at 5-28 19:42

例子里实例属性是和类属性使用相同的 name 做名称

```
>>> s.name = 'Michael' # 给实例绑定name属性
>>> print(s.name) # 由于实例属性优先级比类属性高，因此，它会屏蔽掉类的name属性
Michael
>>> print(Student.name) # 但是类属性并未消失，用Student.name仍然可以访问
Student
>>> del s.name # 如果删除实例的name属性
>>> print(s.name) # 再次调用s.name，由于实例的name属性没有找到，类的name属性就显示出来了
Student
```

绑定类和绑定实例的属性名不一样就可以吧。

View Full Discuss

Reply This Topic



当有类属性时，再定义实例属性是因为实例属性的变量名指向了新的值，类属性并没有被改变

philaradox created at 5-9 22:44, Last updated at 5-9 22:44


```
class Person(object):
    """定义一个具有类属性的类"""
    _die = True
    def __init__(self, age):
        self.age = age


pa = Person(20)
```

`print(Person._die, id(Person._die))`
#输出值: True 1531278160
`print(pa._die, id(pa._die))`
#输出值: True 1531278160

`pa._die = 1`
`print(pa._die, id(pa._die))`
#输出值: 1 1531459344

`del pa._die`
`print(pa._die, id(pa._die))`
#输出值: True 1531278160

 [View Full Discuss](#)

 [Reply This Topic](#)

发表评论

[Sign In to Make a Comment](#)



友情链接: [中华诗词](#) - [阿里云](#) - [SICP](#) - [4closure](#)