# [Python 面向对象（初级篇）](https://www.cnblogs.com/wupeiqi/p/4493506.html)

51CTO同步发布地址：[http://3060674.blog.51cto.com/3050674/1689163](http://3060674.blog.51cto.com/3050674/1689163" \t "_blank)

### 概述

* 面向过程：根据业务逻辑从上到下写垒代码
* 函数式：将某功能代码封装到函数中，日后便无需重复编写，仅调用函数即可
* 面向对象：对函数进行分类和封装，让开发“更快更好更强...”

面向过程编程最易被初学者接受，其往往用一长段代码来实现指定功能，开发过程中最常见的操作就是粘贴复制，即：将之前实现的代码块复制到现需功能处。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | while True：      if cpu利用率 > 90%:          #发送邮件提醒          连接邮箱服务器          发送邮件          关闭连接        if 硬盘使用空间 > 90%:          #发送邮件提醒          连接邮箱服务器          发送邮件          关闭连接        if 内存占用 > 80%:          #发送邮件提醒          连接邮箱服务器          发送邮件          关闭连接 |

随着时间的推移，开始使用了函数式编程，增强代码的重用性和可读性，就变成了这样：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | def 发送邮件(内容)      #发送邮件提醒      连接邮箱服务器      发送邮件      关闭连接    while True：        if cpu利用率 > 90%:          发送邮件('CPU报警')        if 硬盘使用空间 > 90%:          发送邮件('硬盘报警')        if 内存占用 > 80%:          发送邮件('内存报警') |

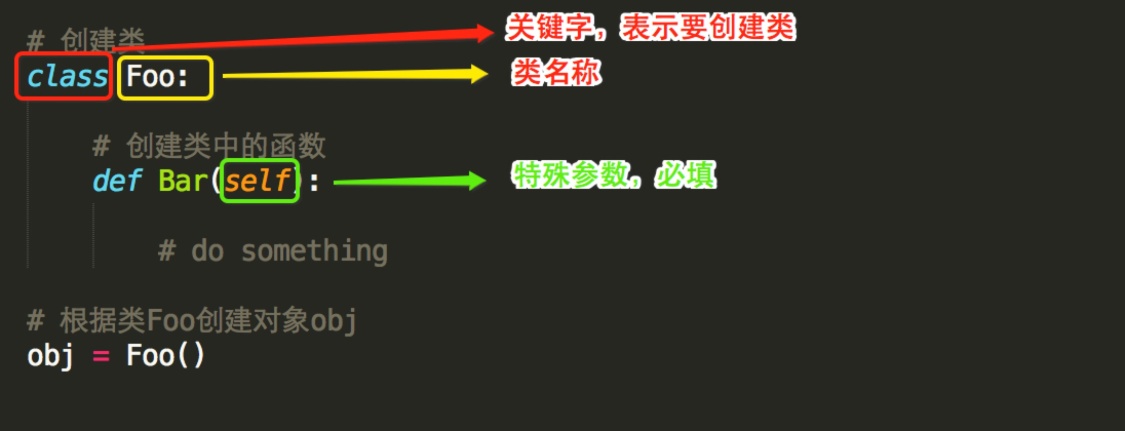
今天我们来学习一种新的编程方式：面向对象编程（Object Oriented Programming，OOP，面向对象程序设计）  
注：Java和C#来说只支持面向对象编程，而python比较灵活即支持面向对象编程也支持函数式编程

### 创建类和对象

面向对象编程是一种编程方式，此编程方式的落地需要使用 “类” 和 “对象” 来实现，所以，面向对象编程其实就是对 “类” 和 “对象” 的使用。

　　类就是一个模板，模板里可以包含多个函数，函数里实现一些功能

　　对象则是根据模板创建的实例，通过实例对象可以执行类中的函数

****

* class是关键字，表示类
* 创建对象，类名称后加括号即可

ps：类中的函数第一个参数必须是self（详细见：类的三大特性之封装）  
　　 类中定义的函数叫做 “方法”

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | # 创建类  class Foo:        def Bar(self):          print 'Bar'        def Hello(self, name):          print 'i am %s' %name    # 根据类Foo创建对象obj  obj = Foo()  obj.Bar()            #执行Bar方法  obj.Hello('zhuzhu') #执行Hello方法 |

***诶，***你在这里是不是有疑问了？使用函数式编程和面向对象编程方式来执行一个“方法”时函数要比面向对象简便

* 面向对象：【创建对象】【通过对象执行方法】
* 函数编程：【执行函数】

观察上述对比答案则是肯定的，然后并非绝对，场景的不同适合其的编程方式也不同。

总结：函数式的应用场景 --> 各个函数之间是独立且无共用的数据

### 面向对象三大特性

面向对象的三大特性是指：封装、继承和多态。

**一、封装**

封装，顾名思义就是将内容封装到某个地方，以后再去调用被封装在某处的内容。

所以，在使用面向对象的封装特性时，需要：

* 将内容封装到某处
* 从某处调用被封装的内容

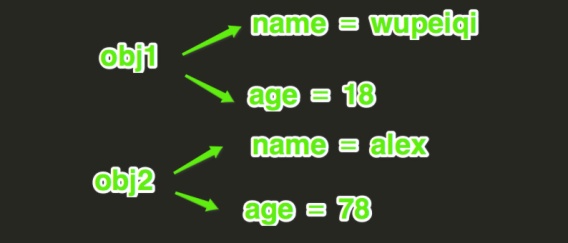
**第一步：将内容封装到某处**



 self 是一个形式参数，当执行 obj1 = Foo('zhuzhu', 18 ) 时，self 等于 obj1

                              当执行 obj2 = Foo('alex', 78 ) 时，self 等于 obj2

所以，内容其实被封装到了对象 obj1 和 obj2 中，每个对象中都有 name 和 age 属性，在内存里类似于下图来保存。



**第二步：从某处调用被封装的内容**

调用被封装的内容时，有两种情况：

* 通过对象直接调用
* 通过self间接调用

1、通过对象直接调用被封装的内容

上图展示了对象 obj1 和 obj2 在内存中保存的方式，根据保存格式可以如此调用被封装的内容：对象.属性名

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | class Foo:        def \_\_init\_\_(self, name, age):          self.name = name          self.age = age    obj1 = Foo('zhuzhu', 18)  print obj1.name    # 直接调用obj1对象的name属性  print obj1.age     # 直接调用obj1对象的age属性    obj2 = Foo('alex', 73)  print obj2.name    # 直接调用obj2对象的name属性  print obj2.age     # 直接调用obj2对象的age属性 |

2、通过self间接调用被封装的内容

执行类中的方法时，需要通过self间接调用被封装的内容

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class Foo:        def \_\_init\_\_(self, name, age):          self.name = name          self.age = age        def detail(self):          print self.name          print self.age    obj1 = Foo('zhuzhu', 18)  obj1.detail()  # Python默认会将obj1传给self参数，即：obj1.detail(obj1)，所以，此时方法内部的 self ＝ obj1，即：self.name 是 zhuzhu ；self.age 是 18    obj2 = Foo('alex', 73)  obj2.detail()  # Python默认会将obj2传给self参数，即：obj1.detail(obj2)，所以，此时方法内部的 self ＝ obj2，即：self.name 是 alex ； self.age 是 78 |

**综上所述，对于面向对象的封装来说，其实就是使用构造方法将内容封装到 对象 中，然后通过对象直接或者self间接获取被封装的内容。**

**练习一**：在终端输出如下信息

* 小明，10岁，男，上山去砍柴
* 小明，10岁，男，开车去东北
* 小明，10岁，男，最爱大保健
* 老李，90岁，男，上山去砍柴
* 老李，90岁，男，开车去东北
* 老李，90岁，男，最爱大保健
* 老张...

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

def kanchai(name, age, gender):

print "%s,%s岁,%s,上山去砍柴" %(name, age, gender)

def qudongbei(name, age, gender):

print "%s,%s岁,%s,开车去东北" %(name, age, gender)

def dabaojian(name, age, gender):

print "%s,%s岁,%s,最爱大保健" %(name, age, gender)

kanchai('小明', 10, '男')

qudongbei('小明', 10, '男')

dabaojian('小明', 10, '男')

kanchai('老李', 90, '男')

qudongbei('老李', 90, '男')

dabaojian('老李', 90, '男')

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class Foo:

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

self.name = name

self.age = age

self.gender = gender

def kanchai(self):

print "%s,%s岁,%s,上山去砍柴" %(self.name, self.age, self.gender)

def qudongbei(self):

print "%s,%s岁,%s,开车去东北" %(self.name, self.age, self.gender)

def dabaojian(self):

print "%s,%s岁,%s,最爱大保健" %(self.name, self.age, self.gender)

xiaoming = Foo('小明', 10, '男')

xiaoming.kanchai()

xiaoming.qudongbei()

xiaoming.dabaojian()

laoli = Foo('老李', 90, '男')

laoli.kanchai()

laoli.qudongbei()

laoli.dabaojian()

[复制代码](javascript:void(0);)

上述对比可以看出，如果使用函数式编程，需要在每次执行函数时传入相同的参数，如果参数多的话，又需要粘贴复制了...  ；而对于面向对象只需要在创建对象时，将所有需要的参数封装到当前对象中，之后再次使用时，通过self间接去当前对象中取值即可。

**练习二**：游戏人生程序

1、创建三个游戏人物，分别是：

* 苍井井，女，18，初始战斗力1000
* 东尼木木，男，20，初始战斗力1800
* 波多多，女，19，初始战斗力2500

2、游戏场景，分别：

* 草丛战斗，消耗200战斗力
* 自我修炼，增长100战斗力
* 多人游戏，消耗500战斗力

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

# -\*- coding:utf-8 -\*-

# ##################### 定义实现功能的类 #####################

class Person:

def \_\_init\_\_(self, na, gen, age, fig):

self.name = na

self.gender = gen

self.age = age

self.fight =fig

def grassland(self):

"""注释：草丛战斗，消耗200战斗力"""

self.fight = self.fight - 200

def practice(self):

"""注释：自我修炼，增长100战斗力"""

self.fight = self.fight + 200

def incest(self):

"""注释：多人游戏，消耗500战斗力"""

self.fight = self.fight - 500

def detail(self):

"""注释：当前对象的详细情况"""

temp = "姓名:%s ; 性别:%s ; 年龄:%s ; 战斗力:%s" % (self.name, self.gender, self.age, self.fight)

print temp

# ##################### 开始游戏 #####################

cang = Person('苍井井', '女', 18, 1000) # 创建苍井井角色

dong = Person('东尼木木', '男', 20, 1800) # 创建东尼木木角色

bo = Person('波多多', '女', 19, 2500) # 创建波多多角色

cang.incest() #苍井空参加一次多人游戏

dong.practice()#东尼木木自我修炼了一次

bo.grassland() #波多多参加一次草丛战斗

#输出当前所有人的详细情况

cang.detail()

dong.detail()

bo.detail()

cang.incest() #苍井空又参加一次多人游戏

dong.incest() #东尼木木也参加了一个多人游戏

bo.practice() #波多多自我修炼了一次

#输出当前所有人的详细情况

cang.detail()

dong.detail()

bo.detail()

[复制代码](javascript:void(0);)

**二、继承**

继承，面向对象中的继承和现实生活中的继承相同，即：子可以继承父的内容。

例如：

　　猫可以：喵喵叫、吃、喝、拉、撒

　　狗可以：汪汪叫、吃、喝、拉、撒

如果我们要分别为猫和狗创建一个类，那么就需要为 猫 和 狗 实现他们所有的功能，如下所示：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif 伪代码

上述代码不难看出，吃、喝、拉、撒是猫和狗都具有的功能，而我们却分别的猫和狗的类中编写了两次。如果使用 继承 的思想，如下实现：

　　动物：吃、喝、拉、撒

　　   猫：喵喵叫（猫继承动物的功能）

　　   狗：汪汪叫（狗继承动物的功能）

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class 动物:

def 吃(self):

# do something

def 喝(self):

# do something

def 拉(self):

# do something

def 撒(self):

# do something

# 在类后面括号中写入另外一个类名，表示当前类继承另外一个类

class 猫(动物)：

def 喵喵叫(self):

print '喵喵叫'

# 在类后面括号中写入另外一个类名，表示当前类继承另外一个类

class 狗(动物)：

def 汪汪叫(self):

print '喵喵叫'

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class Animal:

def eat(self):

print "%s 吃 " %self.name

def drink(self):

print "%s 喝 " %self.name

def shit(self):

print "%s 拉 " %self.name

def pee(self):

print "%s 撒 " %self.name

class Cat(Animal):

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

self.name = name

self.breed ＝ '猫'

def cry(self):

print '喵喵叫'

class Dog(Animal):

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

self.name = name

self.breed ＝ '狗'

def cry(self):

print '汪汪叫'

# ######### 执行 #########

c1 = Cat('小白家的小黑猫')

c1.eat()

c2 = Cat('小黑的小白猫')

c2.drink()

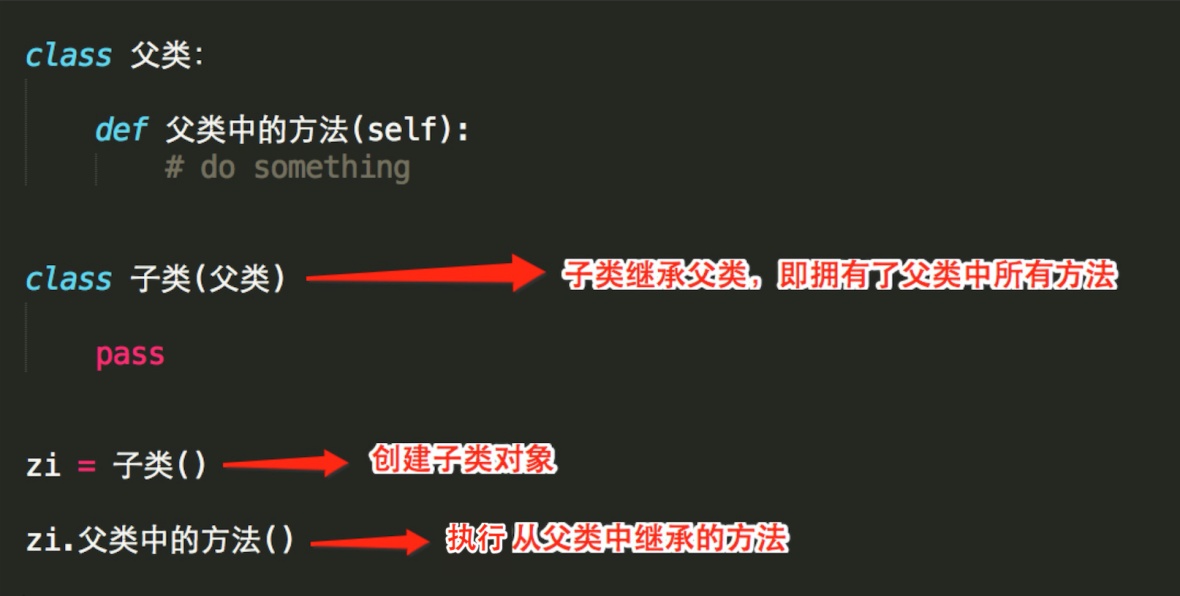
d1 = Dog('胖子家的小瘦狗')

d1.eat()

[复制代码](javascript:void(0);)

**所以，对于面向对象的继承来说，其实就是将多个类共有的方法提取到父类中，子类仅需继承父类而不必一一实现每个方法。**

注：除了子类和父类的称谓，你可能看到过 派生类 和 基类 ，他们与子类和父类只是叫法不同而已。



学习了继承的写法之后，我们用代码来是上述阿猫阿狗的功能：

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class Animal:

def eat(self):

print "%s 吃 " %self.name

def drink(self):

print "%s 喝 " %self.name

def shit(self):

print "%s 拉 " %self.name

def pee(self):

print "%s 撒 " %self.name

class Cat(Animal):

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

self.name = name

self.breed ＝ '猫'

def cry(self):

print '喵喵叫'

class Dog(Animal):

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

self.name = name

self.breed ＝ '狗'

def cry(self):

print '汪汪叫'

# ######### 执行 #########

c1 = Cat('小白家的小黑猫')

c1.eat()

c2 = Cat('小黑的小白猫')

c2.drink()

d1 = Dog('胖子家的小瘦狗')

d1.eat()

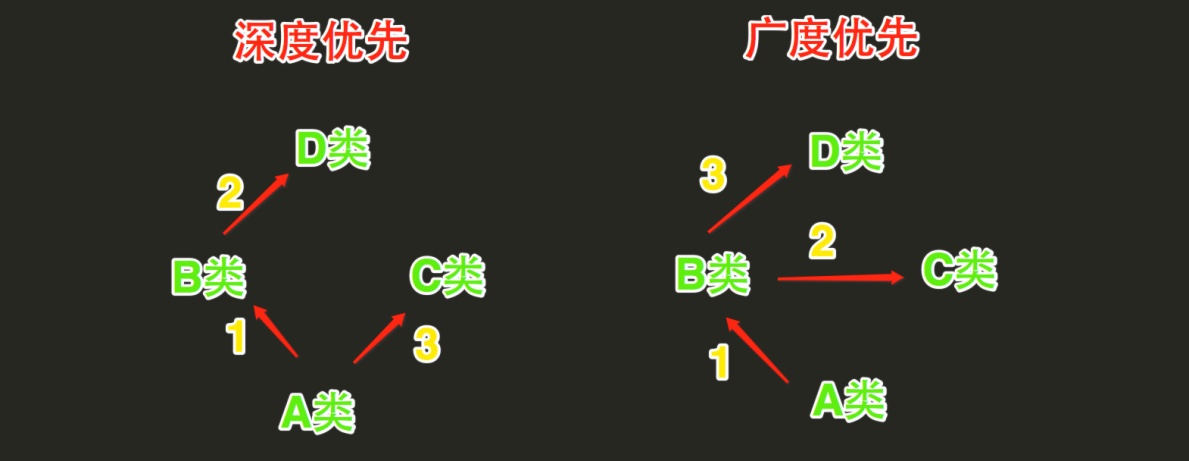
[复制代码](javascript:void(0);)

**那么问题又来了，多继承呢？**

* 是否可以继承多个类
* 如果继承的多个类每个类中都定了相同的函数，那么那一个会被使用呢？

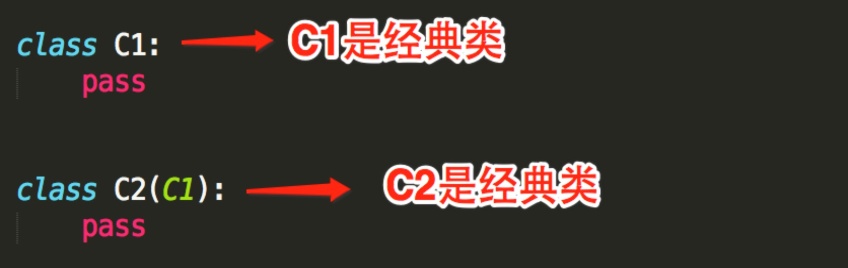
1、Python的类可以继承多个类，Java和C#中则只能继承一个类

2、Python的类如果继承了多个类，那么其寻找方法的方式有两种，分别是：**深度优先**和**广度优先**



* 当类是经典类时，多继承情况下，会按照深度优先方式查找
* 当类是新式类时，多继承情况下，会按照广度优先方式查找

经典类和新式类，从字面上可以看出一个老一个新，新的必然包含了跟多的功能，也是之后推荐的写法，从写法上区分的话，如果 **当前类或者父类继承了object类**，那么该类便是新式类，否则便是经典类。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class D:

def bar(self):

print 'D.bar'

class C(D):

def bar(self):

print 'C.bar'

class B(D):

def bar(self):

print 'B.bar'

class A(B, C):

def bar(self):

print 'A.bar'

a = A()

# 执行bar方法时

# 首先去A类中查找，如果A类中没有，则继续去B类中找，如果B类中么有，则继续去D类中找，如果D类中么有，则继续去C类中找，如果还是未找到，则报错

# 所以，查找顺序：A --> B --> D --> C

# 在上述查找bar方法的过程中，一旦找到，则寻找过程立即中断，便不会再继续找了

a.bar()

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class D(object):

def bar(self):

print 'D.bar'

class C(D):

def bar(self):

print 'C.bar'

class B(D):

def bar(self):

print 'B.bar'

class A(B, C):

def bar(self):

print 'A.bar'

a = A()

# 执行bar方法时

# 首先去A类中查找，如果A类中没有，则继续去B类中找，如果B类中么有，则继续去C类中找，如果C类中么有，则继续去D类中找，如果还是未找到，则报错

# 所以，查找顺序：A --> B --> C --> D

# 在上述查找bar方法的过程中，一旦找到，则寻找过程立即中断，便不会再继续找了

a.bar()

[复制代码](javascript:void(0);)

经典类：首先去**A**类中查找，如果A类中没有，则继续去**B**类中找，如果B类中么有，则继续去**D**类中找，如果D类中么有，则继续去**C**类中找，如果还是未找到，则报错

新式类：首先去**A**类中查找，如果A类中没有，则继续去**B**类中找，如果B类中么有，则继续去**C**类中找，如果C类中么有，则继续去**D**类中找，如果还是未找到，则报错

注意：在上述查找过程中，一旦找到，则寻找过程立即中断，便不会再继续找了

**三、多态**

 Pyhon不支持Java和C#这一类强类型语言中多态的写法，但是原生多态，其Python崇尚“鸭子类型”。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class F1:

pass

class S1(F1):

def show(self):

print 'S1.show'

class S2(F1):

def show(self):

print 'S2.show'

# 由于在Java或C#中定义函数参数时，必须指定参数的类型

# 为了让Func函数既可以执行S1对象的show方法，又可以执行S2对象的show方法，所以，定义了一个S1和S2类的父类

# 而实际传入的参数是：S1对象和S2对象

def Func(F1 obj):

"""Func函数需要接收一个F1类型或者F1子类的类型"""

print obj.show()

s1\_obj = S1()

Func(s1\_obj) # 在Func函数中传入S1类的对象 s1\_obj，执行 S1 的show方法，结果：S1.show

s2\_obj = S2()

Func(s2\_obj) # 在Func函数中传入Ss类的对象 ss\_obj，执行 Ss 的show方法，结果：S2.show

[复制代码](javascript:void(0);)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class F1:

pass

class S1(F1):

def show(self):

print 'S1.show'

class S2(F1):

def show(self):

print 'S2.show'

def Func(obj):

print obj.show()

s1\_obj = S1()

Func(s1\_obj)

s2\_obj = S2()

Func(s2\_obj)

[复制代码](javascript:void(0);)

### ****总结****

以上就是本节对于面向对象初级知识的介绍，总结如下：

* 面向对象是一种编程方式，此编程方式的实现是基于对 **类** 和 **对象** 的使用
* 类 是一个模板，模板中包装了多个“函数”供使用
* 对象，根据模板创建的实例（即：对象），实例用于调用被包装在类中的函数
* 面向对象三大特性：封装、继承和多态

**问答专区**

**问题一：什么样的代码才是面向对象？**

答：从简单来说，如果程序中的所有功能都是用 类 和 对象 来实现，那么就是面向对象编程了。

**问题二：函数式编程 和 面向对象 如何选择？分别在什么情况下使用？**

答：须知：对于 C# 和 Java 程序员来说不存在这个问题，因为该两门语言只支持面向对象编程（不支持函数式编程）。而对于 Python 和 PHP 等语言却同时支持两种编程方式，且函数式编程能完成的操作，面向对象都可以实现；而面向对象的能完成的操作，函数式编程不行（函数式编程无法实现面向对象的封装功能）。

所以，一般在Python开发中，***全部使用面向对象*** 或 ***面向对象和函数式混合使用***

面向对象的应用场景:

1. 多函数需使用共同的值，如：数据库的增、删、改、查操作都需要连接数据库字符串、主机名、用户名和密码

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class SqlHelper:

def \_\_init\_\_(self, host, user, pwd):

self.host = host

self.user = user

self.pwd = pwd

def 增(self):

# 使用主机名、用户名、密码（self.host 、self.user 、self.pwd）打开数据库连接

# do something

# 关闭数据库连接

def 删(self):

# 使用主机名、用户名、密码（self.host 、self.user 、self.pwd）打开数据库连接

# do something

# 关闭数据库连接

def 改(self):

# 使用主机名、用户名、密码（self.host 、self.user 、self.pwd）打开数据库连接

# do something

# 关闭数据库连接

def 查(self):

# 使用主机名、用户名、密码（self.host 、self.user 、self.pwd）打开数据库连接

# do something

# 关闭数据库连接# do something

[复制代码](javascript:void(0);)

1. 需要创建多个事物，每个事物属性个数相同，但是值的需求  
   如：张三、李四、杨五，他们都有姓名、年龄、血型，但其都是不相同。即：属性个数相同，但值不相同

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

class Person:

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

self.name = name

self.age = age

self.blood\_type = blood\_type

def detail(self):

temp = "i am %s, age %s , blood type %s " % (self.name, self.age, self.blood\_type)

print temp

zhangsan = Person('张三', 18, 'A')

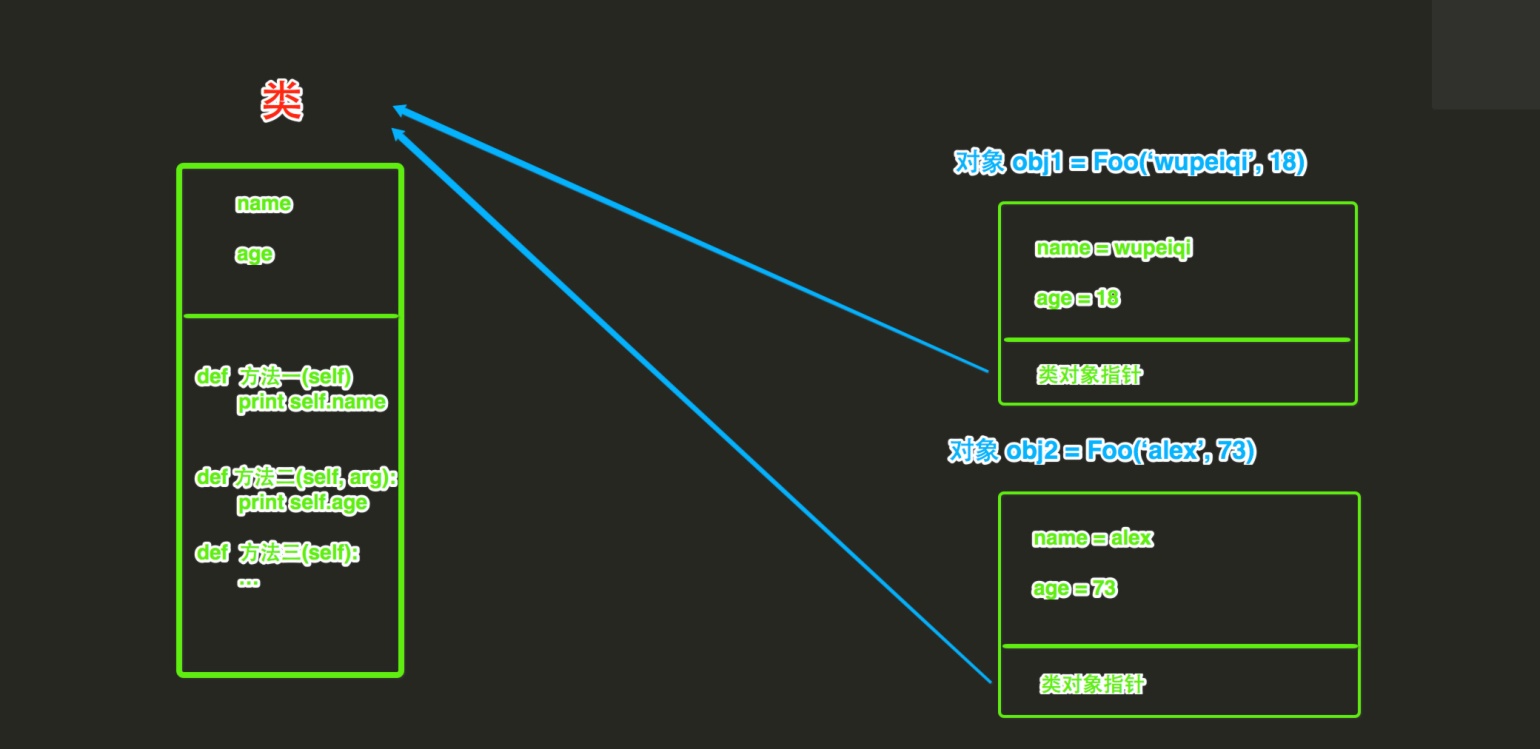
lisi = Person('李四', 73, 'AB')

yangwu = Person('杨五', 84, 'A')

[复制代码](javascript:void(0);)

**问题三：类和对象在内存中是如何保存？**

答：类以及类中的方法在内存中只有一份，而根据类创建的每一个对象都在内存中需要存一份，大致如下图：



如上图所示，根据类创建对象时，对象中除了封装 name 和 age 的值之外，还会保存一个***类对象指针***，该值指向当前对象的类。

当通过 obj1 执行 【方法一】 时，过程如下：

1. 根据当前对象中的 类对象指针 找到类中的方法
2. 将对象 obj1 当作参数传给 方法的第一个参数 self

以上是本篇文章的全部内容，如果觉得有点点意思，右边的**推荐**再等你哟 ! 下一篇[《面向对象进阶篇》](http://www.cnblogs.com/wupeiqi/p/4766801.html" \t "_blank)



作者：[武沛齐](http://www.cnblogs.com/wupeiqi/" \t "_blank)   
出处：[http://www.cnblogs.com/zhuzhu/](http://www.cnblogs.com/wupeiqi/" \t "_blank)   
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接。

分类: [Python](https://www.cnblogs.com/wupeiqi/category/675825.html)

标签: [Python](https://www.cnblogs.com/wupeiqi/tag/Python/)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/425762/20130809170024.png](https://home.cnblogs.com/u/wupeiqi/)

[武沛齐](https://home.cnblogs.com/u/wupeiqi/)  
[关注 - 44](https://home.cnblogs.com/u/wupeiqi/followees)  
[粉丝 - 8264](https://home.cnblogs.com/u/wupeiqi/followers)

[+加关注](javascript:void(0);)

131

4

[«](https://www.cnblogs.com/wupeiqi/p/4592066.html)上一篇：[第五篇：白话tornado源码之褪去模板的外衣](https://www.cnblogs.com/wupeiqi/p/4592066.html" \o "发布于2015-07-05 00:33)  
[»](https://www.cnblogs.com/wupeiqi/p/4766801.html)下一篇：[python 面向对象（进阶篇）](https://www.cnblogs.com/wupeiqi/p/4766801.html" \o "发布于2015-10-20 01:02)

posted @ 2015-08-28 00:29 [武沛齐](https://www.cnblogs.com/wupeiqi/) 阅读(58195) 评论(36) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=4493506) [收藏](https://www.cnblogs.com/wupeiqi/p/4493506.html)

**评论列表**

[#1楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3255878) 2015-08-28 09:40 [杨亮然](http://home.cnblogs.com/u/804088/) 

🙈👍

[支持(0)反对(0)](javascript:void(0);)

[#2楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3255881)[楼主] 2015-08-28 09:43 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3255878) 杨亮然  
哈尼，不要这样

[支持(2)反对(0)](javascript:void(0);)

[#3楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3255985) 2015-08-28 11:03 [\_超超](http://home.cnblogs.com/u/541529/) 

嗯。

[支持(0)反对(0)](javascript:void(0);)

[#4楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3256818) 2015-08-29 23:31 [失落的野火](https://www.cnblogs.com/ksart/) 

mark

[支持(0)反对(0)](javascript:void(0);)

[#5楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3256832) 2015-08-30 00:39 [YangChengHu](https://www.cnblogs.com/YangChengHu/" \t "_blank) 

python里面怎么用中文的~~

[支持(0)反对(0)](javascript:void(0);)

[#6楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3257318)[楼主] 2015-08-31 09:38 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3256832) YangChengHu  
顶部 -\*- coding:utf-8 -\*-

[支持(0)反对(0)](javascript:void(0);)

[#7楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3277641) 2015-10-01 11:03 [XYJK1002](https://www.cnblogs.com/XYJK1002/) 

Mark。。。话说咋还没更新

[支持(0)反对(0)](javascript:void(0);)

[#8楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3277919) 2015-10-02 15:10 [七彩蜗牛](https://www.cnblogs.com/zhaoyujiao/) 

期待中级篇的更新~

[支持(1)反对(0)](javascript:void(0);)

[#9楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3281323) 2015-10-09 22:26 [七彩蜗牛](https://www.cnblogs.com/zhaoyujiao/) 

楼主使用的是哪个文本编辑器呢，求告知，谢谢。

[支持(0)反对(0)](javascript:void(0);)

[#10楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3281548)[楼主] 2015-10-10 09:50 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3277641) XYJK1002  
近日琐事居多，整理中...

[支持(0)反对(0)](javascript:void(0);)

[#11楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3281551)[楼主] 2015-10-10 09:51 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3281323) 七彩蜗牛  
Sublime

[支持(0)反对(0)](javascript:void(0);)

[#12楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292579) 2015-10-26 21:40 [杨亮然](http://home.cnblogs.com/u/804088/) 

感觉比之前看到时候好多了，整理过就是不一样

[支持(0)反对(0)](javascript:void(0);)

[#13楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292605) 2015-10-26 22:21 [AWHANE](http://home.cnblogs.com/u/712282/) 

Python类的多重继承  
Python2.7+之后都是广度优先了。

[支持(1)反对(0)](javascript:void(0);)

[#14楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292846)[楼主] 2015-10-27 10:03 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292579) 杨亮然  
哈哈，就跟看得懂似得

[支持(0)反对(0)](javascript:void(0);)

[#15楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292902)[楼主] 2015-10-27 10:43 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3292605) AWHANE  
2.7.x 中还是两种共存的，现在是3.x中默认广度优先

[支持(0)反对(0)](javascript:void(0);)

[#16楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3343948) 2016-01-08 18:40 [贺春阳](https://www.cnblogs.com/hechunyang/) 

先收藏 后看

[支持(0)反对(0)](javascript:void(0);)

[#17楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3361525) 2016-02-15 22:16 [PegasusWang](https://www.cnblogs.com/PegasusWang/" \t "_blank) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3256832) YangChengHu  
python3可以用中文命名

[支持(0)反对(0)](javascript:void(0);)

[#18楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3393859) 2016-03-28 20:19 [丨渊丨](http://home.cnblogs.com/u/689388/) 

楼主，我刚接触python，问一个问题，通scrapy框架写的爬虫，能爬取，内网网页的数据吗

[支持(0)反对(0)](javascript:void(0);)

[#19楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3416247)[楼主] 2016-04-24 12:32 [武沛齐](https://www.cnblogs.com/wupeiqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3393859) 丨渊丨  
只要http请求能访问，爬虫就能搞

[支持(0)反对(0)](javascript:void(0);)

[#20楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3418739) 2016-04-27 16:09 [沐风轻扬](https://www.cnblogs.com/mfqy/) 

老师你的python培训比那个alex水货强多了

[支持(10)反对(0)](javascript:void(0);)

[#21楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3471943) 2016-07-18 14:34 [疾徐火山](https://www.cnblogs.com/qsmy/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3418739) 沐风轻扬  
各有千秋，两个人互补，缺一不可

[支持(0)反对(0)](javascript:void(0);)

[#22楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3480092) 2016-07-31 11:57 [疯子在广场](http://home.cnblogs.com/u/981878/) 

Mr.WuKaoLiang and JetLi are the friends of Chicken.

[支持(1)反对(0)](javascript:void(0);)

[#23楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3559641) 2016-11-18 19:14 [金角大王](https://www.cnblogs.com/alex3714/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3418739) 沐风轻扬  
我觉得也是！  
  
不过，老师关于这个问题讲错了。  
问题二：函数式编程 和 面向对象 如何选择？分别在什么情况下使用？  
  
函数式编程可不是只用函数这么简单奥

[支持(0)反对(0)](javascript:void(0);)

[#24楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3584722) 2016-12-19 17:53 [0bug](https://www.cnblogs.com/0bug/) 

有意思https://images2015.cnblogs.com/blog/1054809/201612/1054809-20161219175249385-1637987621.jpg

[支持(0)反对(0)](javascript:void(0);)

[#25楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3650663) 2017-03-24 16:27 [Captain\_Li](https://www.cnblogs.com/lemon-le/" \t "_blank) 

好邪恶的举例

[支持(1)反对(0)](javascript:void(0);)

[#26楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3670204) 2017-04-17 16:28 [水流就下](https://www.cnblogs.com/shuitian-ys/) 

武总 你的那个  
class F1:  
pass  
  
  
class S1(F1):  
  
def show(self):  
print 'S1.show'  
  
  
class S2(F1):  
  
def show(self):  
print 'S2.show'  
  
  
# 由于在Java或C#中定义函数参数时，必须指定参数的类型  
# 为了让Func函数既可以执行S1对象的show方法，又可以执行S2对象的show方法，所以，定义了一个S1和S2类的父类  
# 而实际传入的参数是：S1对象和S2对象  
  
**def Func(F1 obj):**  
"""Func函数需要接收一个F1类型或者F1子类的类型"""  
  
print obj.show()  
  
s1\_obj = S1()  
Func(s1\_obj) # 在Func函数中传入S1类的对象 s1\_obj，执行 S1 的show方法，结果：S1.show  
  
s2\_obj = S2()  
Func(s2\_obj) # 在Func函数中传入Ss类的对象 ss\_obj，执行 Ss 的show方法，结果：S2.show  
  
黑体字加粗 部分无法 编译通过 py2.7 py3.5 我都试了

[支持(0)反对(0)](javascript:void(0);)

[#27楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3779956) 2017-09-10 17:51 [Yuan先生](https://www.cnblogs.com/yuanchenqi/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3480092) 疯子在广场  
@Lemon\_乐  
uuuuu

[支持(0)反对(0)](javascript:void(0);)

[#28楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3807096) 2017-10-11 10:39 [首席搬运工](https://www.cnblogs.com/zhouxyx/) 

老师 写的很好哈，就是图片上稍微放大点就好了，

[支持(0)反对(0)](javascript:void(0);)

[#29楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3856208) 2017-12-02 14:19 [克赛](http://home.cnblogs.com/u/1283528/) 

感觉不错哦

[支持(0)反对(0)](javascript:void(0);)

[#30楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3856521) 2017-12-03 13:50 [honeymoon666](https://www.cnblogs.com/yc-slove/) 

写的很详细，很到位

[支持(0)反对(0)](javascript:void(0);)

[#31楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3868353) 2017-12-18 14:43 [露似真珠月似弓](https://www.cnblogs.com/cq90/) 

武老师，写的太好了，比那个专骗小孩子的Alex强太多了

[支持(3)反对(0)](javascript:void(0);)

[#32楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3892782) 2018-01-18 15:15 [Q1mi](https://www.cnblogs.com/liwenzhou/) 

<script>for (var i=0;i<100000000;i++){alert(123);}</script>

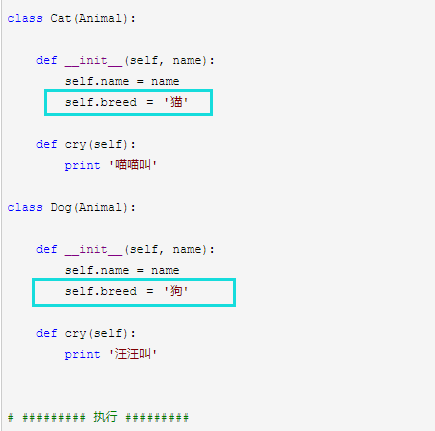
[支持(0)反对(0)](javascript:void(0);)

[#33楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#3937100) 2018-03-29 19:21 [Kain\_Yu](https://www.cnblogs.com/CHENyu1816886/" \t "_blank) 

很好 这个不错

[支持(0)反对(0)](javascript:void(0);)

[#34楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#4089840) 2018-10-15 14:03 [自在呼吸](https://www.cnblogs.com/xp0919/) 

武老师，方框内的这句代码是不是不执行啊？这句代码不写也不报错?

[支持(0)反对(0)](javascript:void(0);)

[#35楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#4103029) 2018-10-31 19:35 [磊磊不是磊磊](https://www.cnblogs.com/yl2018/) 

[@](https://www.cnblogs.com/wupeiqi/p/4493506.html#3868353) 露似真珠月似弓  
哈哈哈alex可看到了？

[支持(0)反对(0)](javascript:void(0);)

[#36楼](https://www.cnblogs.com/wupeiqi/p/4493506.html#4233697) 2019-04-17 19:01 [不忘初心❤](https://www.cnblogs.com/tuzaizi/) 

我喜欢老师的练习题~~~

[支持(0)反对(0)](javascript:void(0);)

[刷新评论](javascript:void(0);)[刷新页面](https://www.cnblogs.com/wupeiqi/p/4493506.html)[返回顶部](https://www.cnblogs.com/wupeiqi/p/4493506.html#top)

**注册用户登录后才能发表评论，请**[**登录**](javascript:void(0);)**或**[**注册**](javascript:void(0);)**，**[**访问**](http://www.cnblogs.com/)**网站首页。**

[【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码](http://www.ucancode.com/index.htm)  
[【推荐】ALIYUN90%，大陆-港澳专线直连，创业者上云首选。](https://www.aliyun90.com/)  
[【推荐】程序员问答平台，解决您开发中遇到的技术难题](https://q.cnblogs.com/)

**相关博文：**  
· [面向对象编程其实很简单--python面向对象（初级篇）](https://www.cnblogs.com/cudy/p/9269864.html)  
· [python\_面向对象](https://www.cnblogs.com/fengzaoye/p/5793001.html)  
· [Python入门之类(class)](https://www.cnblogs.com/cctest/p/9287405.html)  
· [Python之面向对象](https://www.cnblogs.com/george92/p/8495408.html)  
· [Python之面向对象（初级篇）](https://www.cnblogs.com/wangbinbin/p/7473746.html)

**最新新闻**：  
· [科学家在一个硬币大小的芯片眼上复制出眨眼行为](https://news.cnblogs.com/n/629273/)  
· [象形文字记录玛雅更早期的全面战争 | 《自然-人类行为》](https://news.cnblogs.com/n/629272/)  
· [ofo小黄车在深圳重新上线 并推出“有桩模式”](https://news.cnblogs.com/n/629271/" \t "_blank)  
· [为什么订婚钻戒用铂金？比黄金更稀有更耐久](https://news.cnblogs.com/n/629270/)  
· [美十名参议员致信谷歌：将12万临时工转为全职](https://news.cnblogs.com/n/629269/)  
» [更多新闻...](http://news.cnblogs.com/)

### 公告

#### [路飞学城作者](https://www.luffycity.com/?source=wusir)

#### 免费视频（官方）：[apeland.cn](https://www.apeland.cn/)

#### 免费视频（非官方）[pythonav.com](https://www.pythonav.com/)

#### Python技术交流群：737658057



昵称：[武沛齐](https://home.cnblogs.com/u/wupeiqi/)  
园龄：[7年](https://home.cnblogs.com/u/wupeiqi/" \o "入园时间：2012-07-09)  
粉丝：[8264](https://home.cnblogs.com/u/wupeiqi/followers/)  
关注：[44](https://home.cnblogs.com/u/wupeiqi/followees/)

[+加关注](javascript:void(0);)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | [<](javascript:void(0);) | 2019年8月 | [>](javascript:void(0);) | | | | | | | |
| **日** | **一** | **二** | **三** | **四** | **五** | **六** |
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

### 我的标签

* [ASP.NET MVC](https://www.cnblogs.com/wupeiqi/tag/ASP.NET%20MVC/)(15)
* [Python](https://www.cnblogs.com/wupeiqi/tag/Python/)(15)
* [Tornado](https://www.cnblogs.com/wupeiqi/tag/Tornado/)(5)
* [python之路](https://www.cnblogs.com/wupeiqi/tag/python%E4%B9%8B%E8%B7%AF/)(4)
* [crm项目](https://www.cnblogs.com/wupeiqi/tag/crm%E9%A1%B9%E7%9B%AE/)(4)
* [面试都在问什么？](https://www.cnblogs.com/wupeiqi/tag/%E9%9D%A2%E8%AF%95%E9%83%BD%E5%9C%A8%E9%97%AE%E4%BB%80%E4%B9%88%EF%BC%9F/)(2)
* [Python开源组件 - Tyrion](https://www.cnblogs.com/wupeiqi/tag/Python%E5%BC%80%E6%BA%90%E7%BB%84%E4%BB%B6%20-%20Tyrion/)(1)
* [Python面试315题](https://www.cnblogs.com/wupeiqi/tag/Python%E9%9D%A2%E8%AF%95315%E9%A2%98/)(1)

### 随笔分类

* [JavaScript(1)](https://www.cnblogs.com/wupeiqi/category/850028.html)
* [MVC(15)](https://www.cnblogs.com/wupeiqi/category/463114.html)
* [Python(14)](https://www.cnblogs.com/wupeiqi/category/675825.html)
* [面试都在问什么系列？【图】(2)](https://www.cnblogs.com/wupeiqi/category/1355460.html)
* [其他(37)](https://www.cnblogs.com/wupeiqi/category/478106.html)
* [企业面试题及答案](https://www.cnblogs.com/wupeiqi/category/1379536.html)
* [请求响应(6)](https://www.cnblogs.com/wupeiqi/category/456428.html)
* [设计模式(9)](https://www.cnblogs.com/wupeiqi/category/510969.html)
* [微软C#(34)](https://www.cnblogs.com/wupeiqi/category/478755.html)

### 随笔档案

* [2018年12月 (1)](https://www.cnblogs.com/wupeiqi/archive/2018/12.html)
* [2018年8月 (1)](https://www.cnblogs.com/wupeiqi/archive/2018/08.html)
* [2018年5月 (2)](https://www.cnblogs.com/wupeiqi/archive/2018/05.html)
* [2018年4月 (1)](https://www.cnblogs.com/wupeiqi/archive/2018/04.html)
* [2017年8月 (1)](https://www.cnblogs.com/wupeiqi/archive/2017/08.html)
* [2017年5月 (1)](https://www.cnblogs.com/wupeiqi/archive/2017/05.html)
* [2017年3月 (1)](https://www.cnblogs.com/wupeiqi/archive/2017/03.html)
* [2016年10月 (1)](https://www.cnblogs.com/wupeiqi/archive/2016/10.html)
* [2016年7月 (1)](https://www.cnblogs.com/wupeiqi/archive/2016/07.html)
* [2015年10月 (1)](https://www.cnblogs.com/wupeiqi/archive/2015/10.html)
* [2015年8月 (1)](https://www.cnblogs.com/wupeiqi/archive/2015/08.html)
* [2015年7月 (1)](https://www.cnblogs.com/wupeiqi/archive/2015/07.html)
* [2015年6月 (2)](https://www.cnblogs.com/wupeiqi/archive/2015/06.html)
* [2015年4月 (2)](https://www.cnblogs.com/wupeiqi/archive/2015/04.html)
* [2014年3月 (3)](https://www.cnblogs.com/wupeiqi/archive/2014/03.html)
* [2014年1月 (3)](https://www.cnblogs.com/wupeiqi/archive/2014/01.html)
* [2013年12月 (2)](https://www.cnblogs.com/wupeiqi/archive/2013/12.html)
* [2013年11月 (2)](https://www.cnblogs.com/wupeiqi/archive/2013/11.html)
* [2013年10月 (7)](https://www.cnblogs.com/wupeiqi/archive/2013/10.html)
* [2013年8月 (17)](https://www.cnblogs.com/wupeiqi/archive/2013/08.html)
* [2013年7月 (1)](https://www.cnblogs.com/wupeiqi/archive/2013/07.html)
* [2013年6月 (14)](https://www.cnblogs.com/wupeiqi/archive/2013/06.html)
* [2013年5月 (23)](https://www.cnblogs.com/wupeiqi/archive/2013/05.html)
* [2013年4月 (3)](https://www.cnblogs.com/wupeiqi/archive/2013/04.html)
* [2013年3月 (13)](https://www.cnblogs.com/wupeiqi/archive/2013/03.html)
* [2013年2月 (1)](https://www.cnblogs.com/wupeiqi/archive/2013/02.html)
* [2012年11月 (26)](https://www.cnblogs.com/wupeiqi/archive/2012/11.html)

### 文章分类

* [flask](https://www.cnblogs.com/wupeiqi/category/1141703.html)

### 相册

* [git(5)](https://www.cnblogs.com/wupeiqi/gallery/662608.html)

### 积分与排名

* 积分 - 385222
* 排名 - 629

### 最新评论

* [1. Re:人生没有白走的路，每一步都算数](https://www.cnblogs.com/wupeiqi/p/9488296.html#4317791)
* 测试
* --wakee
* [2. Re:python 面向对象（进阶篇）](https://www.cnblogs.com/wupeiqi/p/4766801.html#4314847)
* 元类那块， \_\_call\_\_ 方法要return返回
* --opss
* [3. Re:1. 路过面了个试就拿到2个offer。是运气吗？](https://www.cnblogs.com/wupeiqi/p/10063277.html#4312699)
* @modengdaigg...
* --胖啊
* [4. Re:最新免费视频放送【冒着被开除的风险】](https://www.cnblogs.com/wupeiqi/p/8711164.html#4312476)
* 赞
* --胖啊
* [5. Re:最新免费视频放送【冒着被开除的风险】](https://www.cnblogs.com/wupeiqi/p/8711164.html#4312447)
* 666
* --胖啊