**[python中子类调用父类的初始化方法](https://www.cnblogs.com/nerrissa/articles/5607291.html)**

<http://www.crazyant.net/1303.html>

python和其他面向对象语言类似，每个类可以拥有一个或者多个父类，它们从父类那里继承了属性和方法。如果一个方法在子类的实例中被调用，或者一个属性在子类的实例中被访问，但是该方法或属性在子类中并不存在，那么就会自动的去其父类中进行查找。

继承父类后，就能调用父类方法和访问父类属性，而要完成整个集成过程，子类是需要调用的构造函数的。

**子类不显式调用父类的构造方法，而父类构造函数初始化了一些属性，就会出现问题**

如果子类和父类都有构造函数，子类其实是重写了父类的构造函数，如果不显式调用父类构造函数，父类的构造函数就不会被执行，导致子类实例访问父类初始化方法中初始的变量就会出现问题。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class A:      def \_\_init\_\_(self):          self.namea="aaa"        def funca(self):          print "function a : %s"%self.namea    class B(A):      def \_\_init\_\_(self):          self.nameb="bbb"        def funcb(self):          print "function b : %s"%self.nameb    b=B()  print b.nameb  b.funcb()    b.funca() |

结果：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | bbb  function b : bbb  Traceback (most recent call last):    File "D:\workbench\python\MyPythonProject\test\study\overwrite\_method.py", line 19, in <module>      print b.funca()    File "D:\workbench\python\MyPythonProject\test\study\overwrite\_method.py", line 6, in funca      print "function a : %s"%self.namea  AttributeError: B instance has no attribute 'namea' |

在子类中，构造函数被重写，但新的构造方法没有任何关于初始化父类的namea属性的代码，为了达到预期的效果，子类的构造方法必须调用其父类的构造方法来进行基本的初始化。有两种方法能达到这个目的：调用超类构造方法的未绑定版本，或者使用super函数。

**方法一：调用未绑定的超类构造方法**

修改代码，多增一行：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | class A:      def \_\_init\_\_(self):          self.namea="aaa"        def funca(self):          print "function a : %s"%self.namea    class B(A):      def \_\_init\_\_(self):          #这一行解决了问题          A.\_\_init\_\_(self)          self.nameb="bbb"        def funcb(self):          print "function b : %s"%self.nameb    b=B()  print b.nameb  b.funcb()    b.funca() |

如上有注释的一行解决了该问题，直接使用父类名称调用其构造函数即可。

这种方法叫做调用父类的未绑定的构造方法。在调用一个实例的方法时，该方法的self参数会被自动绑定到实例上（称为绑定方法）。但如果直接调用类的方法（比如A.\_\_init），那么就没有实例会被绑定。这样就可以自由的提供需要的self参数，这种方法称为未绑定unbound方法。

通过将当前的实例作为self参数提供给未绑定方法，B类就能使用其父类构造方法的所有实现，从而namea变量被设置。

**方法二：使用super函数**

修改代码，这次需要增加在原来代码上增加2行：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #父类需要继承object对象  class A(object):      def \_\_init\_\_(self):          self.namea="aaa"        def funca(self):          print "function a : %s"%self.namea    class B(A):      def \_\_init\_\_(self):          #这一行解决问题          super(B,self).\_\_init\_\_()          self.nameb="bbb"        def funcb(self):          print "function b : %s"%self.nameb    b=B()  print b.nameb  b.funcb()    b.funca() |

如上有注释的为新增的代码，其中第一句让类A继承自object类，这样才能使用super函数，因为这是python的“新式类”支持的特性。当前的雷和对象可以作为super函数的参数使用，调用函数返回的对象的任何方法都是调用超类的方法，而不是当前类的方法。

super函数会返回一个super对象，这个对象负责进行方法解析，解析过程其会自动查找所有的父类以及父类的父类。

**方法一更直观，方法二可以一次初始化所有超类**

super函数比在超累中直接调用未绑定方法更直观，但是其最大的有点是如果子类继承了多个父类，它只需要使用一次super函数就可以。然而如果没有这个需求，直接使用A.\_\_init\_\_(self)更直观一些。