书籍

使用 LaTeX 排版

一种良好的风格

作者[1] | 编者[2]

版本：1.0

[1]作者：
[2]编者：邮箱：a358003542@gmail.com。

前言

这里说明你写这个 python 项目的基本思路和想法。

目 录

前言 **i**

目录 **ii**

1 classtools.py **1**

1.1 AttrDisplay 1

1.2 测试代码 2

1.3 输出结果 2

2 person.py **3**

2.1 Person 3

2.1.1 Manager 4

2.2 测试代码 6

2.3 输出结果 6

classtools.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

############################
```

AttrDisplay

```
class AttrDisplay:
    def gatherattrs(self):
        attrs = []
        for key in sorted(self.__dict__):
            attrs.append('%s=%s' % (key,getattr(self,key)))
        return ', '.join(attrs)

    def __repr__(self):
        return '[%s: %s]' % (self.__class__.__name__,
        self.gatherattrs())
```

测试代码

```
if __name__ == '__main__':
    class TopTest(AttrDisplay):
        count=0
        def __init__(self):
            self.attr1 = TopTest.count
            self.attr2 = TopTest.count+1
            TopTest.count +=2

    class SubTest(TopTest):
        pass

    X,Y=TopTest(),SubTest()
    print(X)
    print(Y)
```

输出结果

person.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

###################################
# 序言部分
```

Person

```
from classtools import AttrDisplay
class Person(AttrDisplay):
    def __init__(self, name, job=None, pay=0):
        self.name = name
        self.job = job
        self.pay = pay
    def lastname(self):
        return self.name.split()[-1]
    def giveraise(self,percent):
        self.pay = int(self.pay * (1+percent))
```

```
#    def __repr__(self):
#        return '[Person: %s,%s]' % (self.name,self.pay)
```

类名一般都大写。

特殊的__init__ 用于这个类具体创建 instance 实例的时候执行的动作。self 表示创建的那个实例，self.name 表示实例的名字，self.name = name 表示接受的 name 将会传递值给 self.name，同时创建的那个实例将会拥有一个自己的 name 属性。其他属性操作类似。__init__ 这里实际上也是进行了函数重载。

和一般函数的做法一样，job=None，表示 job 这个参数是一个可选参数，它有一个默认值 None。

这里新建了一个 Person 类，这个类有三个属性：name，姓名，job，工作，和 pay，薪酬。

新建了 lastname 方法，将会该实例的名字的最后的姓氏。

新建了 giveRaise 方法，还需要一个参数 percent，这样该实例的 pay 属性将会提高这么多百分比。这里的 int 函数是将数值转化为整数。

重新定义__repr__，将会影响 print 函数的行为。

Manager

```
class Manager(Person):
    def __init__(self,name,pay):
        Person.__init__(self,name,'mgr',pay)
```

```
4     def giveraise(self,percent,bonus=0.10):
5         Person.giveraise(self,percent+bonus)
```

这里定义了一个类 Manager，它还接受一个参数 Person，表示它是 Person 的子类，即一切 Person 类的属性它都将继承。

这里重新定义了 giveraise 方法，用一种巧妙的方式。直接借用 Person 类原有的 giveraise 方法，对参数输入稍作修正。

重载`__init__`方法，提供更加灵活的本地方案。

在 python 中：

```
instance.method(args...)
```

都会化成这样的形式：

```
class.method(instance, args...)
```

这种转换只针对实例。

python 中超类，子类，实例的重载是由一种搜索机制实现的：

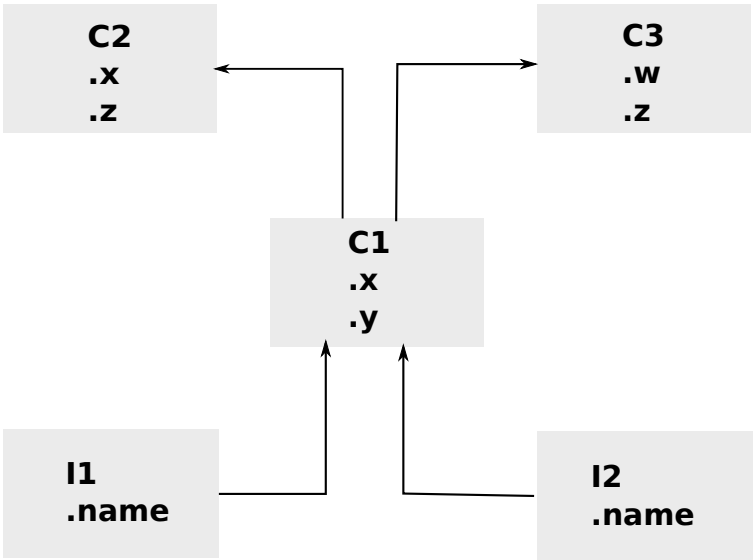

图 2-1: 类搜索结构图

python 首先搜索 self 有没有这个属性或者方法，如果没有，就向上搜索。比如说实例没有，就向上搜索那个子类，子类没有就向上搜索那个超类。

测试代码

```
if __name__ == '__main__':
    bob = Person('Bob Smith')
    sue = Person('Sue Jones', job='dev', pay=100000)
    print(bob)
    print(sue)
    print(bob.lastname())
    sue.giveraise(0.10)
    print(sue)
    tom=Manager('Tom Jones',50000)
    tom.giveraise(0.10)
    print(tom.lastname())
    print(tom)
```

如果 py 文件是 import 的形式，那么这段代码将不会执行。只有以脚本 python3 test.py 之类的形式才执行。

可见继承过来的类将不会有自己默认定义的`__init__` 等方法。

输出结果