

# Python基础教程（第三版）（七）再谈抽象



c747190cc2f5 (/u/c747190cc2f5)

0.3 2019.04.26 23:31\* 字数 2350 阅读 14 评论 0 喜欢 2

(/u/c747190cc2f5)

编辑文章 (/writer#/notebooks/36261492/notes/45694560)

菜鸡的学习笔记。

## 7.1 对象魔法

- 多态：可对不同类型的对象执行相同的操作，但是操作将随对象所属的类型而异；
- 封装：对外隐藏对象内部工作原理的细节；
- 继承：可基于通用类创建出专用类。

按作者的意思，多态最易懂，但也最有趣，就先讲多态。

### 7.1.1 多态

假如要实现查询物品价格的函数，但是有人用元组来存放（'item',price）数据,有人用字典来存放 {'item': price} 数据,还有人用别的来对象存放数据，那么在查询价格时，就要先判断对象的类型，再根据索引或者键值去获得价格。现在假设每种对象都有一种能查询价格的方法，那么就不用再去判断对象类型，只用调用方法即可，例如 Object.get\_price(), 这就是多态（也有一定程度上封装）。

### 7.1.2 多态和方法

有很多函数都用了多态，例如+函数，repr()函数，等等。

关于鸭子类型。(https://links.jianshu.com/go?

to=https%3A%2F%2Fblog.csdn.net%2Fu013573654%2Farticle%2Fdetails%2F51180566)就是只要你会走得像鸭子，叫得像鸭子，就可以把你当成鸭子来处理，只要你有某种方法，就可以用这种方法去实现某种功能，而不管你是什么（自己的理解）。

### 7.1.3 封装

主要理解封装和多态的区别。封装和多态很像，都是**抽象**。但是二者的区别是**多态让你无需知道对象所属的类就能调用其方法，而封装就让你无需知道对象的构造就能使用它。**

书中的🐥：

(https://yex.yo  
slot=30  
58c5-4  
a150d  
csqwC  
kja2lln  
JV08fc  
tf5ez4  
click.y  
58c5-4  
a150d  
77412



比如有两个相同类的对象，都有get\_name(),set\_name()的方法，但是都关联到了一个全局变量name，这样更改其中任何一个，另一个就会变；这就需要封装，把name封装到对象内部，让每个对象都有一个自己的name，这样就会互不影响。

### 7.1.4 继承

让通用类衍生出专用类，并且不会丢失其中的必要方法。

## 7.2 类

### 7.2.1 类到底是什么

类就是类型，对象是类的实例化。人是个类，你就是个对象。  
**超类和子类**：如麻雀是鸟的子类，鸟是麻雀的超类。  
子类可以增加新的方法，也可以重写超类的方法，比如鸟并不都吃肉，但是老鹰会吃肉，可以给子类老鹰新加方法eat\_meat();又如鸟有方法fly(),但是子类企鹅不会飞，就应该重写fly(),让fly()起不到什么作用。

### 7.2.2 创建自定义类

和JAVA类似。别忘了self，创建对象似乎不需要new。

### 7.2.3 属性、函数和方法

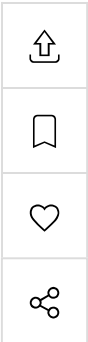
主要就是理解self (<https://links.jianshu.com/go?to=https%3A%2F%2Fwww.cnblogs.com%2Fjessonluo%2Fp%2F4717140.html>)

### 7.2.4 再谈隐藏

```
>>>c.name
'Sam'
>>>c.name = 'Tom'
>>>c.get_name()
'Tom'
```

上面的代码不通过set\_name()方法直接修改了变量。有的程序员声称这样违背了封装原则，他们认为应该对外完全隐藏对象的状态。为什么他们会如此极端？直接访问name不就好了，为什么非要用set\_name，get\_name呢？

(https://yex.yo slot=3( 58c5-4 a150dl csqwC kja2lln JV08fc tf5ez4\ click.yc 58c5-4 a150dl 77412:



关键是其他程序猿可能不知道（也不应该知道）对象内部发生的情况。例如，ClosedObject可能在对象修改其名称时向管理员发送电子邮件。这种功能可能包含在方法set\_name中。但如果直接设置c.name，结果将如何呢？什么都不会发生——根本不会发送电子邮件。为避免这类问题，可将属性定义为私有。私有属性不能从对象外部访问，而只能通过存取器方法（如get\_name和set\_name）来访问。

可以用两个下划线让方法或者属性变为私有。

这种方法是变相的变为私有，只是对其名称做了转换，在开头加个下划线和类名即可访问，但是不推荐。

```
>>>s = Secretive()
>>>s.__inaccessible() # 会报错
```

```
s._Secretive__inaccessible() #不会报错
```

## 7.2.5 类的命名空间

额外的知识——下面两条语句等价：

```
def foo(x): return x * x
foo = lambda x: x * x
```

进入正题：

```
class MemberCounter:
    members = 0
    def init(self):
        MemberCounter.members += 1
m1 = MemberCounter()
m2 = MemberCounter()
m1.init()
print(MemberCounter.members)
m2.init()
print(MemberCounter.members)
```

结果：

上述代码在类作用域定义了一个变量，所有的成员（实例）都可访问它。

(https://  
yex.yo  
slot=3(  
58c5-4  
a150d(  
csqwC  
kja2lln  
JV08fc  
tf5ez4  
click.y  
58c5-4  
a150d(  
77412:



自己体会一下，不难理解，但要注意，定义init()时，写成members +=1 会报错，必须要加上类名。

有个特殊情况：

```
m1.members = 'Two'
print(m1.members)
print(m2.members)
```

结果：

相当于m1写入了一个新属性，覆盖了类级变量

## 7.2.6 指定超类

在class语句的类名后面通过用圆括号把超类名称括起来来制定超类。

一个栗子：

```
class Filter:
    def init(self):
        self.blocked = []

    def filter(self,sequence):
        return [x for x in sequence if x not in self.blocked]

class SPAMFilter(Filter):
    def init(self):
        self.blocked = ['SPAM']

f = Filter()
f.init()
print(f.filter([1, 2, 3]))

s = SPAMFilter()
s.init()
print(s.filter(['SPAM', 1, 2, 3]))
```

结果：

- 对于超类中的函数可以重写，如例子中的init；
- 可以从超类中继承方法，如filter，不需要再次定义，提高了效率。

## 7.2.7 深入讨论继承

(https://  
yex.yo  
slot=3(  
58c5-4  
a150d  
csqwC  
kja2lln  
JV08fc  
tf5ez4  
click.y  
58c5-4  
a150d  
77412)



```
print(issubclass(SPAMFilter, Filter))
print(SPAMFilter.__bases__)
print(isinstance(s, SPAMFilter))
print(isinstance(s, Filter))
```

结果：

- 判断一个类是否是另一个类的子类，可以用方法issubclass；
- 想要知道某个类的基类，访问它的特殊属性**bases**；
- 想要知道某个对象是否是某个类的实例，用方法isinstance；如果一个对象是某个特定的类的实例，那么也是该类的超类的实例。

## 7.2.8 多个超类

即多重继承，在圆括号中增加要继承的类即可。

**注意：**如果继承的类中有方法重名，那么圆括号中前面的类的方法会覆盖后面的类的方法；

慎用超类，容易带来Bug。

## 7.2.9 接口和内省

暂时没有理解什么是接口、内省。

等以后结合java的接口理解一下。

掌握两种方法

hasattr(tc, 'talk')

callable(getattr(tc, 'talk', None))

- 第一种检查某个对象是否有某个属性，返回True或者False；
- 第二种getattr，可以在属性不存在时设置返回的值。

## 7.2.10 抽象基类

使用模块abc可以创建抽象基类。抽象基类用于指定子类必须提供哪些功能，却不实现这些功能。

书中的这段话理解的还不透彻，先记录下来：

(https://  
yex.yo  
slot=3(  
58c5-4  
a150dl  
csqwC  
kja2lln  
JV08fc  
tf5ez4  
click.y  
58c5-4  
a150dl  
77412)



然而，有比手工检查各个方法更好的选择。在历史上的大部分时间内，Python几乎都只依赖于鸭子类型，即假设所有对象都能完成其工作，同时偶尔使用`hasattr`来检查所需的方法是否存在。很多其他语言（如JAVA和Go）都采用显式指定接口的理念，而有些第三方模块提供了这种理念的各种实现。最终，Python通过引入模块`abc`提供了官方解决方案。这个模块为所谓的抽象基类提供了支持。一般而言，抽象类是不能（至少是不应该）实例化的类，其职责是定义子类应实现的一组抽象方法。

```
class Talker(ABC):
    @abstractmethod
    def talk(self):
        pass
```

上面的类是抽象基类ABC派生出的抽象子类，是不能够实例化的，它的作用只是为了规定其子类必须有`talk`方法。

**实例化 Talker类会报错：**

如果定义这么一个子类：

```
class Knigget(Talker):
    pass
```

由于没有重写方法`talk`，这个子类也是抽象类，也是无法实例化的。

如果重写了方法`talk`：

```
class Knigget(Talker):
    def talk(self):
        print("yeah!")
```

现在就可以实例化。从另一个角度说，如果一个对象确实是Talker对象，那么它一定有方法`talk`。

```
k = Knigget()
k.talk()
print(isinstance(k, Talker))
```

结果：

(https://  
yex.yo  
slot=3(  
58c5-4  
a150d  
csqwC  
kja2lln  
JV08fc  
tf5ez4  
click.y  
58c5-4  
a150d  
77412:



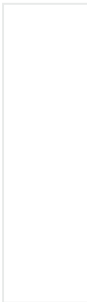
下图中Herring虽然有talk方法，但是isinstance判False。用Talker派生可以解决问题。如果是引用的类，无法自己定义，可以用register。但是**register会把没有talk方法的类也弄成Talker的子类。**

书上说，只要实现了方法talk，即使不是Talker的子类，依然能够通过类型检查。（这句话不明白是啥意思，在register之前isinstance的结果明明是False）

```
class Herring:
    def talk(self):
        print("em...")

h = Herring()
h.talk()
print(isinstance(h, Talker))
Talker.register(Herring)
print(isinstance(h, Talker))
```

结果：



(https://yex.yo  
slot=3(58c5-4  
a150d( csqwC  
kja2lln  
JV08fc  
tf5ez4\  
click.yo  
58c5-4  
a150d(  
77412:

### 7.3 关于面向对象设计的一些思考

- 将相关的东西放在一起。如果一个函数操作一个全局变量，最好将它们作为一个类的属性和方法；
- 不要让对象之间过于亲密。方法应只关心其所属实例的属性，对于其他实例的状态，让它们自己去管理就好了；
- 慎用继承，尤其是多态继承。继承有时很有用，但在有些情况下可能带来不必要的复杂性。要正确地使用多重继承很难，要排除其中的Bug更难。
- 保持简单，让方法短小紧凑。一般而言，应确保大多数方法都能在30秒内读完并理解。对于其余的方法，尽可能将其篇幅控制在一页或一屏内。



确定哪些类以及这些类应该包含哪些方法时，尝试这样做：

- 1. 将有关问题的描述（程序需要做什么）记录下来，并给所有的名词、动词、形容词加上标记；
- 2. 在名词中找出可能的类；
- 3. 在动词中找出可能的方法；
- 4. 在形容词中找出可能的属性；
- 5. 将找出的方法和属性分配给个各类。

Python基础教程（第三版）（八）异常  
(<https://www.jianshu.com/writer#/notebooks/36261492/notebooks/47029047/preview>)

小礼物走一走，来简书关注我

赞赏支持

 python基础教程（第三版） (/nb/36261492) © 著作权归作者所有




c747190cc2f5 (/u/c747190cc2f5)  
写了 25958 字，被 1 人关注，获得了 3 个喜欢  
(/u/c747190cc2f5)

一个菜鸡的挣扎...

喜欢 | 2








更多分享

被以下专题收入，发现更多相似内容


 投稿管理


+ 收入我的专题


Google推出了Python最牛逼的编辑器 (/p/7373948964fa?utm\_campaign=...


环境配置对于大多数人来说都是拦路虎，我们小白往往不知道：怎么正确的安装 不知道选择什么 怎么安装常用的第三方库。。。Google推出了一个在线的网站 <https://colab.research.google.com>，这些问题现在...

(https://yex.yoslot=3(58c5-4a150dlcsqwCkja2llnJV08fctf5ez4\click.yc58c5-4a150dl77412:











 Python小老弟 (/u/d6a000a7da9e?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/d4c49ed14f19?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend

**为什么 Python 4.0 会与 Python 3.0 不同？ (/p/d4c49ed14f19?utm\_camp...**

不管我们如何希望PHP永远天下第一，亦或是Java永久无敌，更或者希望C语言永远是最好的语言。然而，笔者今天搜索百度指数得知，Python的指数，已经高于Java和PHP的指数之和。而Python的版本迭代也是...

 Python小老弟 (/u/d6a000a7da9e?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/360a9bf5fb14?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend

**Python骚操作：Python控制Excel实现自动化办公！ (/p/360a9bf5fb14?ut...**

1.安装 2.操作一个简单的Excel文档 操作注释及代码： 操作完成后，数据存储结果如下： 3. 操作简单Excel文档并添加数据格式 操作代码如下： 附带数据格式的定义 操作效果如图所示： 4.Excel中添加不同类型的...


 浪里小白龙q (/u/1a076099f916?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

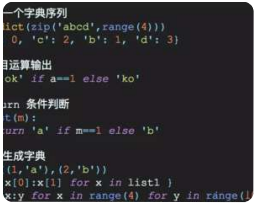
(/p/7e110b2fef4?

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend

**我珍藏的一些好的Python代码，技巧 (/p/7e110b2fef4...**

01.简洁的表达式 点评：Python因为简洁高效而出名，就是因为语法非常简单，而且内置了很多强大的数据结构：比如我们可以大量用推导列表来生成很多...

 我爱学python (/u/8f2987e2f9fb?



utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/a6e2879a1651?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend

**34个最优秀好用的Python开源框架 (/p/a6e2879a1651?utm\_campaign=m...**

免费学习海量AI课程，请访问七月在线官网：www.julyedu.com .....以下是正文.....

(/p/7145536b3ced?)



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**NBA最佳新秀已被本-西蒙斯预定？未必 (/p/7145536b3ced?utm\_campaig...**

1996年出生于澳大利亚的本-西蒙斯，七岁开始打正式篮球比赛，2013年开始就读美国的蒙特沃德学院，并两次带队获得全美高中锦标赛冠军。在大学期间，获得全美年度最佳新人、全美大学篮球最佳阵容一队等...

奋斗星人\_小帆

(/u/881f1029429e?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

**2019-04-09 闹 (/p/7230f12d442f?utm\_campaign=maleskine&utm\_conte...**

兜兜的闹腾开始逐日增加，抱着他，他会扭来扭去，放下他他会爬来爬去，吃饭时，小手迅捷无论，稍不注意就会打饭小勺。

懿和的小屋

(/u/8ac8334d2d23?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

**问问心+W16+88紫凝 (/p/fac38abc3144?utm\_campaign=maleskine&utm...**

工作忙的要死，加工资却不如别人多，别人闲的侃大山，你的心里什么感受？同样当父母，妈妈各种学习如何育儿，还要学会沟通，爸爸直接葛优躺，妈妈内心会不会觉得不公，为什么我要学那么多？培训市场杂...

满身书香气张小梅

(/u/32f3ccd3d699?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

**开始漫长的简书路 (/p/e855b963ef72?utm\_campaign=maleskine&utm\_c...**

这是第一次在简书写，有可能不会得到世界任何的回应。自己目前0文章、0粉丝、0关注。自己希望自己能坚持写下去。我很喜欢这里的氛围。第一次来到这里看免流教程来的。当然现在自己是搞懂免流了。...

孤独带不走的风车

(/u/1f70266bbdd1?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

↑

🔖

❤

🔗