TURING 图灵程序设计丛书

[PACKT] PUBLISHING

【荷】Sakis Kasampalis 著 夏永锋 译

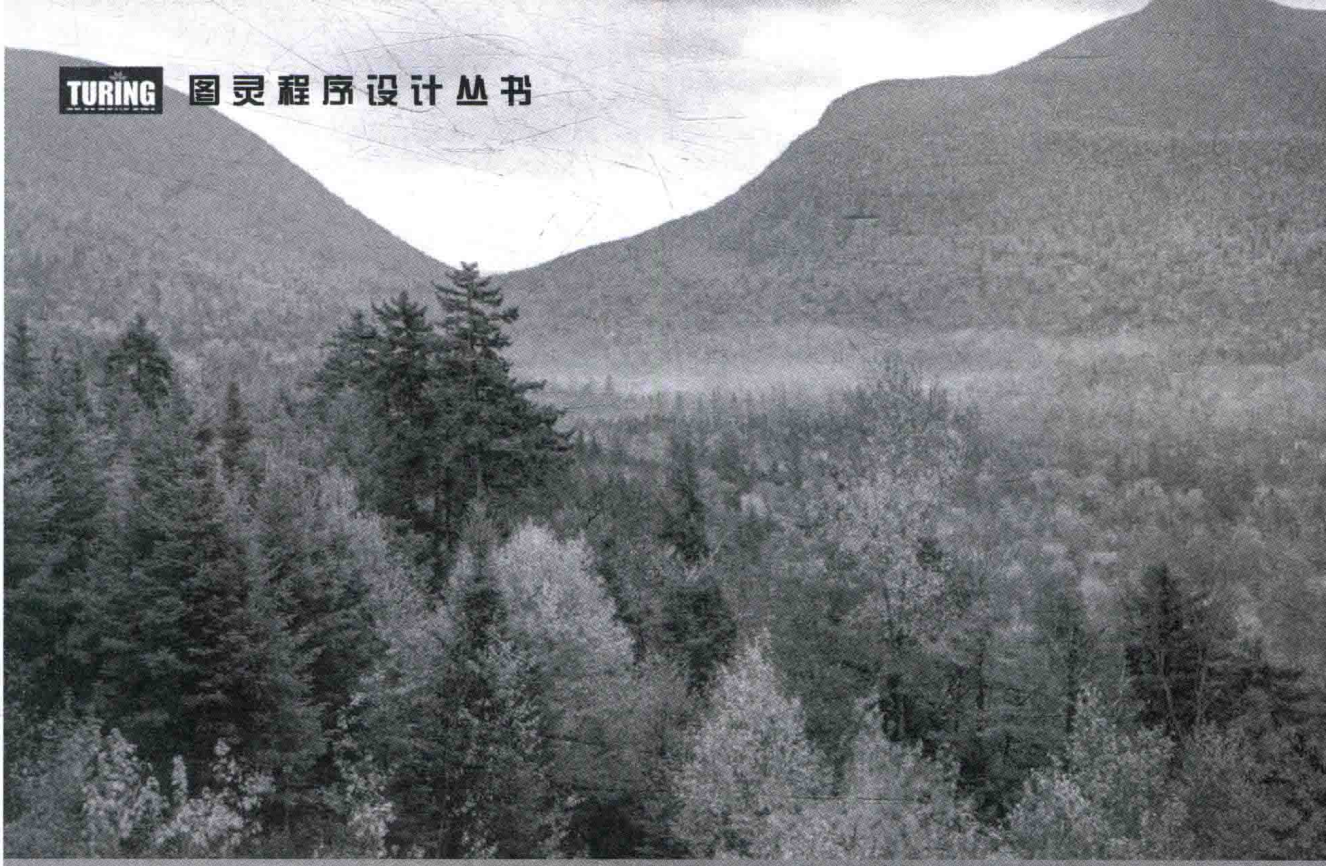
精通Python设计模式

Mastering Python Design Patterns

16种基本设计模式，轻松解决软件设计常见问题
借力高效的Python语言，用现实例子展示各模式关键特性

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS



【荷】Sakis Kasampalis 著 夏永锋 译

精通 Python设计模式

Mastering Python Design Patterns

人民邮电出版社
北 京

图书在版编目（CIP）数据

精通Python设计模式 /（荷）萨基斯·卡萨姆帕里斯(Sakis Kasampalis) 著 ; 夏永锋译. -- 北京 : 人民邮电出版社, 2016.7
（图灵程序设计丛书）
ISBN 978-7-115-42803-5

Ⅰ. ①精… Ⅱ. ①萨… ②夏… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2016)第139755号

内 容 提 要

本书分三部分、共16章，介绍一些常用的设计模式。第一部分介绍处理对象创建的设计模式，包括工厂模式、建造者模式、原型模式；第二部分介绍处理一个系统中不同实体（类、对象等）之间关系的设计模式，包括外观模式、享元模式等；第三部分介绍处理系统实体之间通信的设计模式，包括责任链模式、观察者模式等。

本书的读者对象为有一定基础的Python程序员。

◆ 著 [荷] Sakis Kasampalis
译 夏永锋
责任编辑 朱 巍
执行编辑 杨 琳 赵瑞琳
责任印制 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：800×1000 1/16
印张：10
字数：236千字 2016年7月第1版
印数：1－3 500册 2016年7月北京第1次印刷
著作权合同登记号 图字：01-2015-8301号

定价：45.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第8052号

版权声明

译 者 序

在我读大学的那几年，设计模式可谓红极一时，各大公司校招面试也几乎都会考设计模式；反观现在，则似乎很少有人聊设计模式的话题。这是因为设计模式过时了吗？还是说它只是一个错误的概念？从个人这几年的开发经验来看，答案是否定的：设计模式并未过时，更不是一个错误的概念。从曾经的“红极一时”到如今的“门可罗雀”，只是说明软件开发行业以更加客观理性的态度来看待设计模式。软件开发领域的技术概念也似乎总是遵循这样的流行度变迁，最终一次又一次地证明不存在“银弹”。

正确看待设计模式的前提是明白什么是设计模式。正如本书一开始就强调的，“设计模式的本质是在已有的方案之上发现更好的方案（而不是全新发明）”。这是一种务实的态度，设计模式并非是某种高大上或者神秘的东西，而是一些常见的软件工程设计问题的最佳实践方案。

那么应该如何学习设计模式？个人认为软件开发技术的学习都应该以实践为前提，只有理解实践过程中遇到的种种问题，才能明白那些技术的本质和目的是什么，因为每种新技术都是因某个/某些问题而出现的；软件开发高手一般都反对新手一开始就一股脑地学习设计模式。有些新手学了点设计模式的理论后，甚至在软件开发过程中生搬硬套，结果适得其反。因此，软件开发人员应该在积累了一定的开发经验之后，再系统地学习设计模式，往往能事半功倍。

现在有些积累了一定开发经验的软件开发人员在谈起设计模式时一脸鄙夷。我想这也不是一种客观务实的态度。软件开发不是简单的累积代码，在实现业务功能的同时应该仔细考虑如何控制软件的复杂度。软件的复杂度分为两个层面：业务逻辑复杂度和代码实现复杂度。对同一个业务系统，不同的软件开发人员会有不同的实现，复杂度也不同；相应地，实现的易理解性、可维护性和可扩展性也不同。软件开发人员应该不断学习如何控制软件的复杂度，学习并恰当地使用设计模式是应对软件复杂度的有效方法。

然而，设计模式并非是固定不变的（如《设计模式：可复用面向对象软件的基础》一书总结的23种模式），使用不同的编程语言来编写代码，需要学习的设计模式也不一样。一方面是因为软件开发领域迅猛发展，一些新的软件工程问题也随之出现；另一方面则是因为新的语言、新的平台会把一些常见设计模式吸收为内置特性。所以，软件开发人员应以实际问题为驱动，不断更新设计模式方面的知识。

本书以Python编程语言为例，针对目前的软件开发领域，分三大类讲解了16种常用的设计模式。使用Python语言编写示例代码，我认为作者主要是考虑到Python的抽象层次高、应用范围广，读者不会被一些实现细节所干扰，从而能快速直接地掌握模式的要领。

全书始终保持务实的态度，列举了大量现实生活的例子和软件开发的例子，并为每个模式提供了完整可运行的示例代码。虽然在书中给出所有示例代码似乎没什么必要，但个人认为作者的用意是希望读者能亲自动手，照着示例代码写一遍并运行，然后看看结果，从而加强学习的效果。

虽然是示例，但作者还是坚持以地道的Python风格编写代码，以此说明不同语言和不同平台要求软件开发人员学习的设计模式也有所不同。另外，开发人员也能从示例代码中学习到一些Python语言的高级特性，所以把本书当作Python开发进阶图书也无不可。

本书是个人正式翻译的第一本书。虽然以前翻译过很多文章，其中有些译文还有一些影响，但毕竟与正式出版有些不同，所以接手本书的翻译工作，我内心是有些忐忑的。为保证翻译的质量，我将翻译过程分为以下几个阶段来进行。

(1) 大致地预读一遍全书，整体上把握原书内容。
(2) 将原书翻译成初稿，此阶段基本保证译文的正确性。
(3) 通读审校初稿，此阶段确保译文的流畅性，以及用词和逻辑的一致性。
(4) 对着译稿，翻译相关图表中的单词；整理示例代码，并确保运行无误。

希望能通过这种方式基本保证译稿的质量。不过因为个人精力有限、能力不足，译稿中可能还会存在疏漏甚至错误之处，敬请谅解。如发现有错误之处，请将问题反馈给出版社，以便在再版时更正。

另外，本书的示例代码已经保存到GitHub的一个代码库（https://github.com/youngsterxyf/mpdp-code）中，如有需要，可以下载。

因个人原因，本书推延了一段时间才得以翻译完成，感谢图灵朱巍老师的体谅。译书是一件费时费力的事情，感谢妻子郑荣的体谅和支持，也感谢公司领导贾磊和同事的支持，谢谢你们！

夏永锋
于上海百度研发中心

前　言

什么是设计模式

软件工程中，设计模式是指软件设计问题的推荐方案。设计模式一般是描述如何组织代码和使用最佳实践来解决常见的设计问题。需谨记在心的一点是：设计模式是高层次的方案，并不关注具体的实现细节，比如算法和数据结构（请参考［GOF95，第13页］和网页［t.cn/RP6HFwi］）。对于正在尝试解决的问题，何种算法和数据结构最优，则是由软件工程师自己把握。

如果你不了解［］中文字的含义，请暂时先跳到前言的“排版约定”部分，查看一下本书中的引用所遵循的格式。

设计模式最重要的部分可能就是它的名称。给模式起名的好处是大家相互交流时有共同的词汇（请参考［GOF95，第13页］）。因此，如果你提交一些代码进行评审，同行评审者的反馈中提到“我认为这个地方你可以使用一个策略模式来代替……”，即使你不知道或不记得策略模式是什么，也可以立即去查阅。

随着编程语言的演进，一些设计模式（如单例）也随之过时，甚至成了反模式（请参考网页［t.cn/zRoxwyD］），另一些则被内置在编程语言中（如迭代器模式）。另外，也有一些新的模式诞生（比如Borg/Monostate，请参考网页［t.cn/zWOOOZC］和［t.cn/RqrKbBe］）。

关于设计模式的常见误解

关于设计模式有一些误解。第一个误解是，一开始写代码就应该使用设计模式。我们经常能看到开发人员纠结在代码中应该使用哪种设计模式，他们甚至都还没有先尝试一下使用自己的方式解决问题（请参考网页［t.cn/RqrJNDw］和［t.cn/RqrJl0m］）。

这不仅是错误的，而且违背了设计模式的本质。设计模式是在已有的方案之上发现更好的方案（而不是全新发明）。若你一个方案都没有，又何谈找一个更好的呢？先行动起来，用你的技能尽可能漂亮地解决问题。若代码评审者没有反对意见，而你经过一段时间后还是觉得自己的方

案足够漂亮灵活，那也就意味着没必要浪费时间纠结使用哪种模式。你也许还能发现更好的设计模式。谁知道呢，关键是不要为了强迫自己使用已有的设计模式而限制了你的创造力。

第二个误解是设计模式应随处使用。这会导致方案很复杂，夹杂着多余的接口和分层，而其实往往一个更简单直接的方案就足够了。设计模式并不是万能的，仅当代码确实存在坏味道、难以扩展维护时，才有使用的必要。多思考思考你不会需要它（You Aren't Gonna Need It，YAGNI，请参考网页[t.cn/SGw9Ec]）和保持简单直白（Keep It Simple Stupid，KISS，请参考网页[t.cn/RqrKMW4]）原则。随处使用设计模式与过早优化一样，都是误入歧途（请参考网页 [t.cn/ShMKfD]）。

设计模式与 Python

本书主要介绍Python实现的设计模式。与畅销设计模式书籍中大多使用的常见编程语言（通常是Java，请参考[FFBS04]；或C++，请参考[GOF95]）不同，Python支持动态类型（duck-typing），函数是一等公民，并且一些模式（例如，迭代器和修饰器）是内置特性。本书旨在演示最基本的设计模式，并非历史记载的所有模式（请参考网页[t.cn/RqrKbBe]）。代码示例也使用合适的Python惯用写法（请参考网页 [t.cn/hTfLt]）。如果你还不熟悉Python之禅，那现在就打开Python交互模式，执行import this。Python之禅趣味十足又意义深远。

本书内容

第一部分，创建型模式，介绍处理对象创建的设计模式。

- 第1章，工厂模式　介绍如何使用工厂设计模式（工厂方法和抽象工厂）来初始化对象，并说明与直接实例化对象相比，使用工厂设计模式的优势。
- 第2章，建造者模式　对于由多个相关对象构成的对象，介绍如何简化其创建过程。
- 第3章，原型模式　介绍如何通过完全复制（也就是克隆）一个已有对象来创建一个新对象。

第二部分，结构型模式，介绍处理一个系统中不同实体（类、对象等）之间关系的设计模式。

- 第4章，适配器模式　介绍如何以最小的改变实现已有代码与外来接口（例如，一个外部代码库）的兼容。
- 第5章，修饰器模式　介绍如何无需使用继承也能增强对象的功能。
- 第6章，外观模式　介绍如何创建单个入口点来隐藏系统的复杂性。
- 第7章，享元模式　介绍如何通过复用一个对象池中的对象来提高内存利用率及应用性能。
- 第8章，模型-视图-控制器模式　介绍如何避免业务逻辑与用户界面代码的耦合，提高应用的可维护性。

- 第9章，代理模式　介绍如何增加额外的保护层，提高应用的安全性。

第三部分，行为型模式，介绍处理系统实体之间通信的设计模式。

- 第10章，责任链模式　介绍如何向多个接收者发送请求。
- 第11章，命令模式　介绍如何让应用能够取消已经执行的操作。
- 第12章，解释器模式　介绍如何基于Python创建一种简单的语言，便于领域专家使用，而无需学习Python编程。
- 第13章，观察者模式　介绍如何在对象发生变化时，通知已注册的相关者。
- 第14章，状态模式　介绍如何创建一个状态机以对问题进行建模，并说明这种技术的优势。
- 第15章，策略模式　介绍如何基于某些输入标准（例如，元素大小）在程序运行期间从多个可用算法中选择一个。
- 第16章，模板模式　介绍如何明确区分一个算法的通用与不通用部分，以避免不必要的代码复制。

阅读准备

书中的代码仅用Python 3编写。Python 3在很多方面与Python 2.x不兼容（请参考网页[t.cn/Rw8Ycjs]）。虽然代码是使用Python 3.4.0进行测试的，但Python 3.3.0应该也可以，因为Python 3.3.0和Python 3.4.0之间并没有语法上的差别（请参考网页[t.cn/RqrK1eX]）。一般来说，如果你从www.python.org下载安装最新的Python 3版本，那么运行示例代码应该不会有问题。示例代码中使用的多数模块/库是Python 3自带的。如果有示例要求安装额外的模块，在相关代码之前会给出如何安装的说明。

读者对象

本书适合具备一定经验同时又对以地道的Python代码实现设计模式感兴趣的Python开发人员。使用其他语言的开发人员，如果对Python感兴趣，也能从中获益不少，但最好先阅读一些材料，了解一下Python的基本知识（请参考网页[t.cn/hTfLt]和网页[t.cn/hp20G]）。

排版约定

在书中，你会发现许多文本样式，用于区分不同种类的信息。以下举例说明，并解释其含义。

代码、用户输入、推特用户定位会使用等宽的代码字体，如下面的句子中所示：“我们将使用Python发行版自带的两个库（`xml.etree.ElementTree`和`json`）来处理XML和JSON。”

代码块的版式如下所示。

```
@property
def parsed_data(self):
    return self.data
```

当希望你关注代码块中某个特别部分时，相关的行或项目会以粗体显示，如下所示。

```
@property
def parsed_data(self):
    return self.data
```

任何命令行输入输出都按如下方式编写。

```
>>> python3 factory_method.py
```

新术语和重要的单词以楷体显示。在菜单或对话框等屏幕上看到的单词会保留原英文，如“点击Next按钮转到下一屏”。

警告或重要的注意事项会这样显示。

提示和小窍门会这样显示。

书籍引用遵循格式［作者，页码］。例如，［GOF95，第10页］是指引用GOF（《设计模式：可复用面向对象软件的基础》）一书的第10页。

Web引用遵循格式［t.cn/shortened］。可以将这些缩短的URL地址键入或复制到Web浏览器中，按Enter键后会跳转到实际的Web引用（通常也更长，也许会更丑）。例如，在Web浏览器地址栏中键入t.cn/hTfLt，按Enter键后会跳转到http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html。

读者反馈

我们始终欢迎来自读者的反馈。关于本书，如果你有一些想法，无论是喜欢还是不喜欢，都可以告诉我们。我们非常重视读者反馈，因为这能帮助我们开发一些让读者充分受益的出版物。

一般的反馈，只要发送电子邮件到feedback@packtpub.com并在消息标题中提及书名即可。

如果你擅长某个主题，并有兴趣写本书或者为某本书作出贡献，请阅读作者指南www.packtpub.com/authors。

客户支持

既然你购买了一本Packt出版的书籍，我们会提供很多服务，让你获得最大的购买利益。

下载示例代码

凡是通过http://www.packtpub.com网站账户购买的Packt书籍，都可以在网上下载相应的示例代码文件。如果你是在其他地方购买本书的，则可以访问http://packtpub.com/support，注册之后，相关文件会直接通过电子邮件发送给你。

勘误

虽然我们会全力确保书籍内容的准确性，但错误仍然在所避免。如果你在某本书中发现错误，无论是文本错误还是代码错误，都请报告给我们，对此我们将万分感激。这样不仅能消除其他读者的疑虑，也能帮助我们提升本书后续版本的质量。如果你发现任何错误，请访问http://www.packtpub.com/submit-errata进行报告，选择相应图书，单击Errata Submission Form链接，并输入勘误的详细信息。一旦勘误得到证实，我们就会接受你的提交，并将勘误上传到站点或添加到对应书名的勘误一节下面的已有勘误表中。

要查看之前提交的勘误，可以访问https://www.packtpub.com/books/content/support并在搜索框内输入书名。需要的信息会显示在Errata部分的下面。

反盗版

对所有媒体来说，互联网盗版行为都是一直存在的问题。在Packt，我们严格保护版权和许可证。如果你在互联网上发现任何形式的我们出版物的非法复制品，请立即把网址或站点名称提供给我们，以便我们进行补救。

请通过copyright@packtpub.com联系我们，提供可疑盗版资料的链接。

感谢你帮助保护我们的作者，让我们能够为你提供有价值的内容。

疑问解答

对于本书的任何方面，如果你有问题，可以通过question@packtpub.com联系我们，我们将尽力解决。

电子书

扫描如下二维码，即可购买本书电子版。

目　录

第一部分　创建型模式

第二部分　结构型模式

第三部分 行为型模式

Part 1 第一部分

创建型模式

本部分内容

第1章 工厂模式

创建型设计模式处理对象创建相关的问题（请参考网页［t.cn/RqBoSiu］），目标是当直接创建对象（在Python中是通过__init__()函数实现的，请参考网页［t.cn/RqB3mDM］和［Lott14，第26页］）不太方便时，提供更好的方式。

在工厂设计模式中，客户端[①]可以请求一个对象，而无需知道这个对象来自哪里；也就是，使用哪个类来生成这个对象。工厂背后的思想是简化对象的创建。与客户端自己基于类实例化直接创建对象相比，基于一个中心化函数来实现，更易于追踪创建了哪些对象（请参考［Eckel08，第187页］）。通过将创建对象的代码和使用对象的代码解耦，工厂能够降低应用维护的复杂度（请参考［Zlobin13，第30页］）。

工厂通常有两种形式：一种是*工厂方法*（Factory Method），它是一个方法（或以地道的Python术语来说，是一个函数），对不同的输入参数返回不同的对象（请参考网页［t.cn/RqB1yx2］）；第二种是抽象工厂，它是一组用于创建一系列相关事物对象的工厂方法（请参考［GOF95，第100页］和网页［t.cn/RqB1tZS］）。

1.1 工厂方法

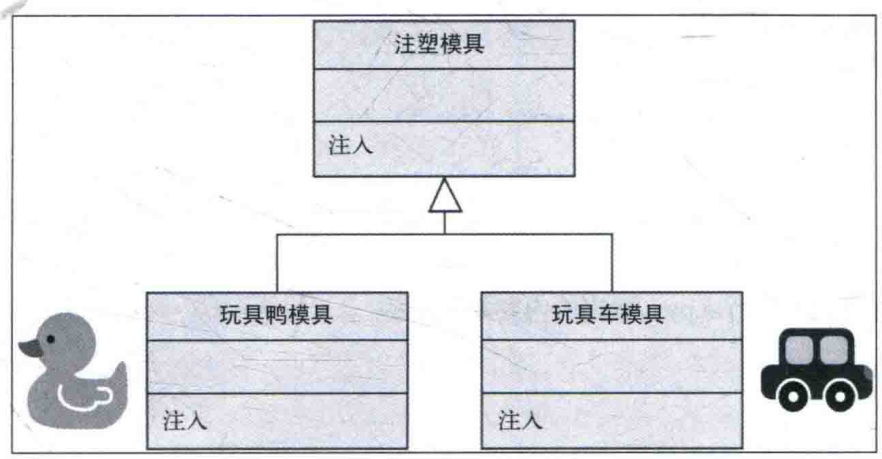
在工厂方法模式中，我们执行单个函数，传入一个参数（提供信息表明我们想要什么），但并不要求知道任何关于对象如何实现以及对象来自哪里的细节。

1.1.1 现实生活的例子

现实中用到工厂方法模式思想的一个例子是塑料玩具制造。制造塑料玩具的压塑粉都是一样的，但使用不同的塑料模具就能产出不同的外形。比如，有一个工厂方法，输入是目标外形（鸭子或小车）的名称，输出则是要求的塑料外形。下图展示的是玩具制造案例，该案例源自网站www.sourcemaking.com，请参考网页［t.cn/RqB1yx2］。

---

① 本书中涉及的“客户端”一词是指调用方，并非网络CS结构中的C。——译者注



1.1.2 软件的例子

Django框架使用工厂方法模式来创建表单字段。Django的`forms`模块支持不同种类字段（`CharField`、`EmailField`）的创建和定制（`max_length`、`required`），请参考网页[t.cn/Rqr9qD7]。

1.1.3 应用案例

如果因为应用创建对象的代码分布在多个不同的地方，而不是仅在一个函数/方法中，你发现没法跟踪这些对象，那么应该考虑使用工厂方法模式（请参考[Eckel08，第187页]）。工厂方法集中地在一个地方创建对象，使对象跟踪变得更容易。注意，创建多个工厂方法也完全没有问题，实践中通常也这么做，对相似的对象创建进行逻辑分组，每个工厂方法负责一个分组。例如，有一个工厂方法负责连接到不同的数据库（MySQL、SQLite），另一个工厂方法负责创建要求的几何对象（圆形、三角形），等等。

若需要将对象的创建和使用解耦，工厂方法也能派上用场。创建对象时，我们并没有与某个特定类耦合/绑定到一起，而只是通过调用某个函数来提供关于我们想要什么的部分信息。这意味着修改这个函数比较容易，不需要同时修改使用这个函数的代码（请参考[Zlobin13，第30页]）。

另外一个值得一提的应用案例与应用性能及内存使用相关。工厂方法可以在必要时创建新的对象，从而提高性能和内存使用率（请参考[Zlobin13，第28页]）。若直接实例化类来创建对象，那么每次创建新对象就需要分配额外的内存（除非这个类内部使用了缓存，一般情况下不会这样）。用行动说话，下面的代码（文件id.py）对同一个类`A`创建了两个实例，并使用函数`id()`比较它们的内存地址。输出中也会包含地址，便于检查地址是否正确。内存地址不同就意味着创建了两个不同的对象。