

Python面向对象编程指南

作者：[美] Steven F. Lott 洛特

目 录

[版权信息](#)

[作者简介](#)

[版权声明](#)

[内容提要](#)

[译者简介](#)

[前言](#)

[本书涵盖的内容](#)

[阅读本书你需要准备什么](#)

[本书的目标读者](#)

[约定](#)

[读者反馈](#)

[客服支持](#)

[下载本书的示例代码](#)

[勘误](#)

[版权](#)

[其他问题](#)

[审阅者简介](#)

[一些预备知识](#)

[关于21点游戏](#)

[玩21点游戏](#)

[21点游戏策略](#)

[21点游戏模拟器对象的设计](#)

[性能——timeit模块](#)

[测试——unittest和doctest](#)

[单元测试与技术探究](#)

[Docstring——RST标记和文档工具](#)

[IDE的选择](#)

[关于特殊方法名](#)

[总结](#)

[第1部分 用特殊方法实现Python风格的类](#)

[用特殊方法实现](#)

[第1章 init\(\)方法](#)

[1.1 隐式的基类——object](#)

[1.2 基类中的__init__\(\)方法](#)

[1.3 在基类中实现init\(\)方法](#)

[1.4 使用__init__\(\)方法创建常量清单](#)

[1.5 通过工厂函数调用__init__\(\)](#)

[1.5.1 错误的工厂设计和模糊的else语句](#)

[1.5.2 使用c初始化设计来获得一致性](#)

[1.5.3 使用映射和类来简化设计](#)

[1.6 在每个子类中实现__init__\(\)方法](#)

[1.7 简单的组合对象](#)

[1.7.1 封装集合类](#)

[1.7.2 扩展集合类](#)

[1.7.3 可适应更多需求的另一种设计](#)

[1.8 复合的组合对象](#)

[完成组合对象的初始化](#)

[1.9 不带__init__\(\)方法的无状态对象](#)

[1.10 一些其他的类定义](#)

[1.11 多策略的__init__\(\)方法](#)

[1.11.1 更复杂的初始化方式](#)

[1.11.2 静态函数的初始化](#)

[1.12 更多的__init__\(\)技术](#)

[1.12.1 带有类型验证的初始化](#)

[1.12.2 初始化、封装和私有化](#)

[1.13 总结](#)

[第2章 与Python无缝集成——基本特殊方法](#)

[2.1 __repr__\(\)和__str__\(\)方法](#)

[2.1.1 非集合对象的__str__\(\)和__repr__\(\)](#)

[2.1.2 集合中的__str__\(\)和__repr__\(\)](#)

[2.2 __format__\(\)方法](#)

[2.2.1 内嵌格式规范](#)

[2.2.2 集合和委托格式规范](#)

[2.3 __hash__\(\)方法](#)

[2.3.1 决定哈希的对象](#)

[2.3.2 有关不可变对象和继承的默认行为](#)

[2.3.3 重载不可变对象](#)

[2.3.4 重载可变对象](#)

[2.3.5 从可变的Hand类中生成一个不可变的Hand类](#)

[2.4 __bool__\(\)方法](#)

[2.5 __bytes__\(\)方法](#)

[2.6 比较运算符方法](#)

[2.6.1 设计比较运算](#)

[2.6.2 实现同一个类的对象比较](#)

[2.6.3 实现不同类的对象比较](#)

[2.6.4 硬总和、软总和和多态](#)

[2.6.5 不同类比较的例子](#)

[2.7 __del__\(\)方法](#)

[2.7.1 引用计数和对象销毁](#)

[2.7.2 循环引用和垃圾回收](#)

[2.7.3 循环引用和weakref模块](#)

[2.7.4 __del__\(\)和close\(\)方法](#)

[2.8 __new__\(\)方法和不可变对象](#)

[2.9 __new__\(\)方法和元类型](#)

[2.9.1 元类型示例1——有序的属性](#)

[2.9.2 元类型示例2——自引用](#)

[2.10 总结](#)

[第3章 属性访问、特性和修饰符](#)

[3.1 属性的基本操作](#)

[特性和__init__\(\)方法](#)

[3.2 创建特性](#)

[3.2.1 主动计算特性](#)

[3.2.2 setter和deleter特性](#)

[3.3 使用特殊方法完成属性访问](#)

[3.3.1 使用__slots__创建不可变对象](#)

[3.3.2 使用tuple子类创建不可变对象](#)

[3.3.3 主动计算的属性](#)

[3.4 __getattr__方法](#)

[3.5 创建修饰符](#)

[3.5.1 使用非数据修饰符](#)

[3.5.2 使用数据修饰符](#)

[3.6 总结、设计要素和折中方案](#)

[3.6.1 特性与属性对比](#)

[3.6.2 修饰符的设计](#)

[3.6.3 展望](#)

[第4章 抽象基类设计的一致性](#)

[4.1 抽象基类](#)

[4.2 基类和多态](#)

[4.3 可调用对象](#)

[4.4 容器和集合](#)

[4.5 数值类型](#)

[4.6 其他的一些抽象基类](#)

[4.6.1 迭代器的抽象基类](#)

[4.6.2 上下文和上下文管理器](#)

[4.7 abc模块](#)

[4.8 总结、设计要素和折中方案](#)

[展望](#)

[第5章 可调用对象和上下文的使用](#)

[5.1 使用ABC可调用对象来进行设计](#)

[5.2 提高性能](#)

[记忆化与缓存](#)

[5.3 使用functools完成记忆化](#)

[使用可调用API简化](#)

[5.4 可调用API和复杂性](#)

[5.5 管理上下文和with语句](#)

[5.5.1 使用小数上下文](#)

[5.5.2 其他上下文](#)

[5.6 定义__enter__\(\)和__exit__\(\)方法](#)

[异常处理](#)

[5.7 上下文管理器工厂](#)

[上下文管理器的清理](#)

[5.8 总结](#)

[5.8.1 可调用对象的设计要素和折中方案](#)

[5.8.2 上下文管理器的设计要素和折中方案](#)

[5.8.3 展望](#)

[第6章 创建容器和集合](#)

[6.1 集合的抽象基类](#)

[6.2 特殊方法示例](#)

[6.3 使用标准库的扩展](#)

[6.3.1 namedtuple\(\)函数](#)

[6.3.2 deque类](#)

[6.3.3 使用ChainMap](#)

[6.3.4 OrderedDict集合](#)

[6.3.5 defaultdict子类](#)

[6.3.6 counter集合](#)

[6.4 创建新集合](#)

[6.5 定义一种新的序列](#)

[6.5.1 一个用于统计的list](#)

[6.5.2 主动计算vs延迟计算](#)

[6.5.3 使用__getitem__\(\), __setitem__\(\), __delitem__\(\)和slice操作](#)

[6.5.4 实现__getitem__\(\), __setitem__\(\)和__delitem__\(\)](#)

[6.5.5 封装list和委托](#)

[6.5.6 用__iter__\(\)创建迭代器](#)

[6.6 创建一种新的映射](#)

[6.7 创建一种新的集合](#)

[6.7.1 一些设计原则](#)

[6.7.2 定义Tree类](#)

[6.7.3 定义TreeNode类](#)

[6.7.4 演示二叉树集合](#)

[6.8 总结](#)

[6.8.1 设计要素和折中方案](#)

[6.8.2 展望](#)

[第7章 创建数值类型](#)

[7.1 numbers的抽象基类](#)

[7.1.1 决定使用哪种类型](#)

[7.1.2 方法解析和运算符映射](#)

[7.2 算术运算符的特殊方法](#)

[7.3 创建一个数字类](#)

[7.3.1 FixedPoint的初始化](#)

[7.3.2 定义固定小数位数的二进制算术运算符](#)

[7.3.3 定义FixedPoint一元算术运算符](#)

[7.3.4 实现FixedPoint反向运算符](#)

[7.3.5 实现FixedPoint比较运算符](#)

[7.4 计算一个数字的哈希值](#)

[7.5 实现其他的特殊方法](#)

[7.6 原地运算符的优化](#)

[7.7 总结](#)

[7.7.1 设计要素和折中方案](#)

[7.7.2 展望](#)

[第8章 装饰器和mixin——横切方面](#)

[8.1 类和描述](#)

[8.1.1 创建函数](#)

[8.1.2 创建类](#)

[8.1.3 一些类设计原则](#)

[8.1.4 面向方面编程](#)

[8.2 使用内置的装饰器](#)

[使用标准库中的装饰器](#)

[8.3 使用标准库中的mixin类](#)

[8.3.1 使用上下文管理器的mixin类](#)

[8.3.2 禁用类的一个功能](#)

[8.4 写一个简单的函数装饰器](#)

[创建独立的日志记录器](#)

[8.5 带参数的装饰器](#)

[8.6 创建方法函数装饰器](#)

[8.7 创建类装饰器](#)

[8.8 向类中添加方法函数](#)

[8.9 将装饰器用于安全性](#)

[8.10 总结](#)

[8.10.1 设计要素和折中方案](#)

[8.10.2 展望](#)

[第2部分 持久化和序列化](#)

[第9章 序列化和保存——JSON、YAML、Pickle、CSV和XML](#)

[9.1 持久化、类、状态以及数据表示](#)

[Python常用的术语](#)

[9.2 文件系统和网络的考虑](#)

[9.3 定义用于持久化的类](#)

[渲染博客与文章列表](#)

[9.4 使用JSON进行转储和加载](#)

[9.4.1 在类中支持JSON](#)

[9.4.2 自定义JSON编码](#)

[9.4.3 自定义JSON解码](#)

[9.4.4 安全性和eval\(\)](#)

[9.4.5 重构编码函数](#)

[9.4.6 日期字符串的标准化](#)

[9.4.7 将JSON写入文件](#)

[9.5 使用YAML进行转储和加载](#)

[9.5.1 YAML文件的格式化](#)

[9.5.2 扩展YAML的表示](#)

[9.5.3 安全性与安全加载](#)

[9.6 使用pickle进行转储和加载](#)

[9.6.1 针对可靠的pickle处理进行类设计](#)

[9.6.2 安全性和全局性问题](#)

[9.7 转储和加载CSV](#)

[9.7.1 将简单的序列转储为CSV](#)

[9.7.2 从CSV文件中加载简单的序列](#)

[9.7.3 处理集合与复杂的类](#)

[9.7.4 在一个CSV文件中转储并从多类型的行中加载数据](#)

[9.7.5 使用迭代器筛选CSV中的行](#)

[9.7.6 从CSV文件中转储和加载连接的行](#)

[9.8 使用XML转储和加载](#)

[9.8.1 使用字符串模板转储对象](#)

[9.8.2 使用xml.etree.ElementTree转储对象](#)

[9.8.3 加载XML文档](#)

[9.9 总结](#)

[9.9.1 设计要素和折中方案](#)

[9.9.2 模式演化](#)

[9.9.3 展望](#)

[第10章 用Shelve保存和获取对象](#)

[10.1 分析持久化对象用例](#)

[ACID属性](#)

[10.2 创建shelf](#)

[10.3 设计适于存储的对象](#)

[10.3.1 为我们的对象设计键](#)

[10.3.2 为对象生成代理键](#)

[10.3.3 设计一个带有简单键的类](#)

[10.3.4 为容器和集合设计类](#)

[10.3.5 用外键引用对象](#)

[10.3.6 为复杂对象设计CRUD操作](#)

[10.4 搜索、扫描和查询](#)

[10.5 为shelve设计数据访问层](#)

[编写演示脚本](#)

[10.6 用索引提高性能](#)

[创建顶层索引](#)

[10.7 有关更多的索引维护工作](#)

[10.8 用writeback代替更新索引](#)

[模式演变](#)

[10.9 总结](#)

[10.9.1 设计要素和折中方案](#)

[10.9.2 应用软件层](#)

[10.9.3 展望](#)

[第11章 用SQLite保存和获取对象](#)

[11.1 SQL数据库、持久化和对象](#)

[11.1.1 SQL数据模型——行和表](#)

[11.1.2 使用SQL的DML语句完成CRUD](#)

[11.1.3 使用SQL中SELECT语句执行查询](#)

[11.1.4 SQL事务和ACID属性](#)

[11.1.5 设计数据库中的主键和外键](#)

[11.2 使用SQL处理程序中的数据](#)

[在纯SQL中实现类似于类的处理方式](#)

[11.3 从Python对象到SQLite BLOB列的映射](#)

[11.4 手动完成从Python对象到数据库中行的映射](#)

[11.4.1 为SQLite设计一个访问层](#)

[11.4.2 实现容器的关系](#)

[11.5 使用索引提高性能](#)

[11.6 添加ORM层](#)

[11.6.1 设计ORM友好的类](#)

[11.6.2 使用ORM层创建模型](#)

[11.6.3 使用ORM层操作对象](#)

[11.7 通过指定标签字符串查询文章对象](#)

[11.8 通过创建索引提高性能](#)

[模型演化](#)

[11.9 总结](#)

[11.9.1 设计要素和折中方案](#)

[11.9.2 映射的方法](#)

[11.9.3 键和键的设计](#)

[11.9.4 应用软件的层](#)

[11.9.5 展望](#)

[第12章 传输和共享对象](#)

[12.1 类、状态和表示](#)

[12.2 用HTTP和REST传输对象](#)

[12.2.1 用REST实现CRUD操作](#)

[12.2.2 实现非CRUD操作](#)

[12.2.3 REST协议和ACID](#)

[12.2.4 选择一种表示方法——JSON、XML或者YAML](#)

[12.3 实现一个REST服务器——WSGI和mod_wsgi](#)

[12.3.1 创建简单的REST应用程序和服务](#)

[12.3.2 实现REST客户端](#)

[12.3.3 演示RESTful服务并创建单元测试](#)

[12.4 使用可回调类创建WSGI应用程序](#)

[12.4.1 设计RESTful对象标识符](#)

[12.4.2 多层REST服务](#)

[12.4.3 创建roulette服务器](#)

[12.4.4 创建roulette客户端](#)

[12.5 创建安全的REST服务](#)

[WSGI验证程序](#)

[12.6 用Web应用程序框架实现REST](#)

[12.7 用消息队列传输对象](#)

[12.7.1 定义进程](#)

[12.7.2 创建队列和提供数据](#)

[12.8 总结](#)

[12.8.1 设计要素和折中方案](#)

[12.8.2 模式演变](#)

[12.8.3 应用程序软件层次](#)

[12.8.4 展望](#)

[第13章 配置文件和持久化](#)

[13.1 配置文件的使用场景](#)

[13.2 表示、持久化、状态和可用性](#)

[13.2.1 应用程序配置的设计模式](#)

[13.2.2 使用对象的构造完成配置](#)

[13.2.3 实现具有层次结构的配置](#)

[13.3 使用INI文件保存配置](#)

[13.4 使用eval\(\)完成更多的文字处理](#)

[13.5 使用PY文件存储配置](#)

[13.5.1 使用类定义进行配置](#)

[13.5.2 通过SimpleNamespace进行配置](#)

[13.5.3 在配置中使用Python的exec\(\)](#)

[13.6 为什么执行exec\(\)没有问题](#)

[13.7 为默认值和重写使用链映射](#)

[13.8 使用JSON或YAML文件存储配置](#)

[13.8.1 使用压平的JSON配置](#)

[13.8.2 加载YAML配置](#)

[13.9 使用特性文件存储配置](#)

[13.9.1 解析特性文件](#)

[13.9.2 使用特性文件](#)

[13.10 使用XML文件——PLIST以及其他格式保存配置](#)

[自定义XML配置文件](#)

[13.11 总结](#)

[13.11.1 设计要素和折中方案](#)

[13.11.2 创建共享配置](#)

[13.11.3 模式演化](#)

[13.11.4 展望](#)

[第3部分 测试、调试、部署和维护](#)

[测试、调试、部署和维护](#)

[第14章 Logging和Warning模块](#)

[14.1 创建基本日志](#)

[14.1.1 创建共享的类级记录器](#)

[14.1.2 配置日志记录器](#)

[14.1.3 开始和关闭日志记录系统](#)

[14.1.4 使用命名的日志记录器](#)

[14.1.5 扩展日志等级](#)

[14.1.6 定义指向多个目标输出的handler](#)

[14.1.7 管理传播规则](#)

[14.2 理解配置](#)

[14.3 为控制、调试、审计和安全创建专门的日志](#)

[14.3.1 创建调试日志](#)

[14.3.2 创建审计和安全日志](#)

[14.4 使用warnings模块](#)

[14.4.1 用警告信息显示API变化](#)

[14.4.2 用警告信息显示配置问题](#)

[14.4.3 用警告信息显示可能存在的软件问题](#)

[14.5 高级日志——最后一些信息和网络目标地址](#)

[14.5.1 创建自动的tail缓冲区](#)

[14.5.2 发送日志消息到远程的进程](#)

[14.5.3 防止队列溢出](#)

[14.6 总结](#)

[14.6.1 设计要素和折中方案](#)

[14.6.2 展望](#)

[第15章 可测试性的设计](#)

[15.1 为测试定义并隔离单元](#)

[15.1.1 最小化依赖](#)

[15.1.2 创建简单的单元测试](#)

[15.1.3 创建一个测试组件](#)

[15.1.4 包含边界值测试](#)

[15.1.5 为测试模仿依赖](#)

[15.1.6 为更多的行为使用更多的模仿对象](#)

[15.2 使用doctest来定义测试用例](#)

[15.2.1 将doctest与unittest相结合](#)

[15.2.2 创建一个更完整的测试包](#)

[15.3 使用安装和卸载](#)

[15.3.1 使用OS资源进行安装和卸载](#)

[15.3.2 结合数据库进行安装和卸载](#)

[15.4 TestCase的类层次结构](#)

[15.5 使用外部定义的期望结果](#)

[15.6 自动化集成和性能测试](#)

[15.7 总结](#)

[15.7.1 设计要素和折中方案](#)

[15.7.2 展望](#)

[第16章 使用命令行](#)

[16.1 操作系统接口和命令行](#)

[参数和选项](#)

[16.2 用argparse解析命令行](#)

[16.2.1 简单的on/off选项](#)

[16.2.2 带参数选项](#)

[16.2.3 位置参数](#)

[16.2.4 所有其他参数](#)

[16.2.5 --version的显示和退出](#)

[16.2.6 --help的显示和退出](#)

[16.3 集成命令行选项和环境变量](#)

[16.3.1 提供更多的可配置默认值](#)

[16.3.2 用环境变量覆盖配置文件设置](#)

[16.3.3 用配置文件覆盖环境变量](#)

[16.3.4 让配置文件理解None](#)

[16.4 自定义帮助文档的输出](#)

[16.5 创建顶层main\(\)函数](#)

[16.5.1 确保配置遵循DRY原则](#)

[16.5.2 管理嵌套的配置上下文](#)

[16.6 大规模程序设计](#)

[16.6.1 设计命令类](#)

[16.6.2 添加用于分析的命令子类](#)

[16.6.3 向应用程序中添加更多的功能](#)

[16.6.4 设计更高级的复合命令](#)

[16.7 其他的复合命令设计模式](#)

[16.8 与其他应用程序集成](#)

[16.9 总结](#)

[16.9.1 设计要素和折中方案](#)

[16.9.2 展望](#)

[第17章 模块和包的设计](#)

[17.1 设计一个模块](#)

[17.1.1 一些模块设计的方法](#)

[17.1.2 模块VS类](#)

[17.1.3 模块中应该包含的内容](#)

[17.2 全局模块VS模块项](#)

[17.3 包的设计](#)

[17.3.1 包与模块的混合设计](#)

[17.3.2 使用多种实现进行包的设计](#)

[17.4 主脚本和__main__模块的设计](#)

[17.4.1 创建可执行脚本文件](#)

[17.4.2 创建__main__模块](#)

[17.4.3 大规模编程](#)

[17.5 设计长时间运行的应用](#)

[17.6 使用src、bin和test来组织代码](#)

[17.7 安装Python模块](#)

[17.8 总结](#)

[17.8.1 设计要素和折中方案](#)

[17.8.2 展望](#)

[第18章 质量和文档](#)

[18.1 为help\(\)函数编写docstrings](#)

[18.2 用pydoc编写文档](#)

[18.3 通过RST标记提供更好的输出](#)

[18.3.1 文本块](#)

[18.3.2 RST内联标记](#)

[18.3.3 RST指令](#)

[18.3.4 学习RST](#)

[18.4 编写有效的文档字符串](#)

[18.5 编写文件级别的文档字符串——包括模块和包](#)

[18.5.1 用RST标记编写详细的API文档](#)

[18.5.2 编写类和方法函数的文档字符串](#)

[18.5.3 编写函数文档字符串](#)

[18.6 更复杂的标记技术](#)

[18.7 用Sphinx生成文档](#)

[18.7.1 使用Sphinx的快速启动](#)

[18.7.2 编写Sphinx文档](#)

[18.7.3 在文档中加入4+1视图](#)

[18.7.4 编写实现文档](#)

[18.7.5 用Sphinx创建交叉引用](#)

[18.7.6 将Sphinx文件重构为目录](#)

[18.8 编写文档](#)

[18.9 大纲式编程](#)

[18.9.1 大纲式编程用例](#)

[18.9.2 使用大纲式编程工具](#)

[18.10 总结](#)

[设计要素和折中方案](#)

[看完了](#)