

<<Python学习手册（第4版）>>

图书基本信息

书名：<<Python学习手册（第4版）>>

13位ISBN编号：9787111326533

10位ISBN编号：7111326539

出版时间：2011-4

出版时间：机械工业出版社

作者：[美] Mark Lutz

页数：889

译者：李军,刘红伟

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

内容概要

学习Python的主要内建对象类型：数字、列表和字典。
使用Python语句创建和处理对象，并且学习Python的通用语法模型。
使用函数构造和重用代码，函数是Python的基本过程工具。
学习Python模块：封装语句、函数以及其他工具，以便构建较大的组件。
学习Python的面向对象编程工具，用于组织程序代码。
学习异常处理模型，以及用于编写较大程序的开发工具。
了解高级Python工具，如装饰器、描述器、元类和Unicode处理等。

作者简介

鲁特兹(Mark Lutz)，世界级的Python培训讲师。他是畅销Python书籍的作者，从1992年起，他就是Python社区的先锋。他也是《Programming Python》、《Python Pocket Reference》和《Learning Python》等书的作者。

<<Python学习手册 (第4版)>>

书籍目录

前言
第一部分 使用入门
第1章 问答环节
人们为何使用Python
软件质量
开发效率
Python是“脚本语言”吗
好吧，Python的缺点是什么呢
如今谁在使用Python
使用Python可以做些什么
系统编程
用户图形接口
Internet脚本
组件集成
数据库编程
快速原型
数值计算和科学计算编程
游戏、图像、人工智能、XML、机器人等
Python如何获得支持
Python有哪些技术上的优点
面向对象
免费
可移植
功能强大
可混合
简单易用
简单易学
Python和其他语言比较起来怎么样
本章小结
本章习题
习题解答
Python是工程，不是艺术
第2章 Python如何运行程序
Python解释器简介
程序执行
程序员的视角
Python的视角
执行模块的变体
Python实现的替代者
执行优化工具
冻结二进制文件
其他执行选项
未来的可能性
本章小结
本章习题

习题解答

第3章 如何运行程序

交互提示模式下编写代码

交互地运行代码

为什么使用交互提示模式

使用交互提示模式

系统命令行和文件

第一段脚本

使用命令行运行文件

使用命令行和文件

UNIX可执行脚本(!)

UNIX env查找技巧

点击文件图标

在Windows中点击图标

input的技巧

图标点击的其他限制

模块导入和重载

模块的显著特性：属性

import和reload的使用注意事项

使用exec运行模块文件

IDLE用户界面

IDLE基础

使用IDLE

高级IDLE工具

其他的IDE

其他启动选项

嵌入式调用

冻结二进制的可执行性

文本编辑器启动的选择

其他的启动选择

未来的可能

我应该选用哪种

调试Python代码

本章小结

本章习题

习题解答

第一部分 练习题

第二部分 类型和运算

第4章 介绍Python对象类型

为什么使用内置类型

Python的核心数据类型

数字

字符串

序列的操作

不可变性

类型特定的方法

寻求帮助

编写字符串的其他方法

模式匹配

列表

序列操作

类型特定的操作

边界检查

嵌套

列表解析

字典

映射操作

重访嵌套

键的排序：for 循环

迭代和优化

不存在的键：if 测试

元组

为什么要用元组

文件

其他文件类工具

其他核心类型

如何破坏代码的灵活性

用户定义的类

剩余的内容

本章小结

本章习题

习题解答

第5章 数字

Python的数字类型

数字常量

内置数学工具和扩展

Python表达式操作符

在实际应用中的数字

变量和基本的表达式

数字显示的格式

比较：一般的和连续的

str和repr显示格式

除法：传统除法、Floor除法和真除法

整数精度

复数

十六进制、八进制和二进制记数

位操作

其他的内置数学工具

其他数字类型

小数数字

分数类型

集合

布尔型

数字扩展

本章小结

本章习题

习题解答

第6章 动态类型简介

缺少类型声明语句的情况

变量、对象和引用

类型属于对象，而不是变量

对象的垃圾收集

共享引用

共享引用和在原处修改

共享引用和相等

动态类型随处可见

本章小结

本章习题

习题解答

第7章 字符串

字符串常量

单双引号字符串是一样的

用转义序列代表特殊字节

raw字符串抑制转义

三重引号编写多行字符串块

实际应用中的字符串

基本操作

索引和分片

为什么要在意：分片

字符串转换工具

修改字符串

字符串方法

字符串方法实例：修改字符串

字符串方法实例：文本解析

实际应用中的其他常见字符串方法

最初的字符串模块（在Python 3.0中删除）

字符串格式化表达式

更高级的字符串格式化表达式

基于字典的字符串格式化

字符串格式化调用方法

基础知识

添加键、属性和偏移量

添加具体格式化

与%格式化表达式比较

为什么用新的格式化方法

通常意义下的类型分类

同样分类的类型共享其操作集合

可变类型能够在原处修改

本章小结

本章习题

习题解答

第8章 列表与字典

列表

实际应用中的列表

基本列表操作

列表迭代和解析

索引、分片和矩阵

原处修改列表

字典

实际应用中的字典

字典的基本操作

原处修改字典

其他字典方法

语言表

字典用法注意事项

为什么要在意字典接口

创建字典的其他方法

Python 3.0中的字典变化

本章小结

本章习题

习题解答

第9章 元组、文件及其他

元组

实际应用中的元组

为什么有了列表还要元组

文件

打开文件

使用文件

实际应用中的文件

其他文件工具

重访类型分类

为什么要在意操作符重载

对象灵活性

引用 VS 拷贝

比较、相等性和真值

Python 3.0的字典比较

Python中真和假的含义

Python的类型层次

Type对象

Python中的其他类型

内置类型陷阱

赋值生成引用，而不是拷贝

重复能够增加层次深度

留意循环数据结构

不可变类型不可以在原处改变

本章小结

本章习题

习题解答

第二部分练习题

第三部分 语句和语法

第10章 Python语句简介

重访Python程序结构

Python的语句

两个if的故事

Python增加了什么

Python删除了什么

为什么使用缩进语法

几个特殊实例

简短实例：交互循环

一个简单的交互式循环

对用户输入数据做数学运算

用测试输入数据来处理错误

用try语句处理错误

嵌套代码三层

本章小结

本章习题

习题解答

第11章 赋值、表达式和打印

赋值语句

赋值语句的形式

序列赋值

Python 3.0中的扩展序列解包

多目标赋值语句

增强赋值语句

变量命名规则

Python的废弃协议

表达式语句

表达式语句和在原处的修改

打印操作

Python 3.0的print函数

Python 2.6 print语句

打印流重定向

版本独立的打印

为什么要注意print和stdout

本章小结

本章习题

习题解答

第12章 if测试和语法规则

if语句

通用格式

基本例子

多路分支

Python语法规则

代码块分隔符

语句的分隔符

- 一些特殊情况
- 真值测试
- if/else三元表达式
- 为什么要在意布尔值
- 本章小结
- 本章习题
- 习题解答
- 第13章 while和for循环
- while循环
- 一般格式
- 例子
- break、continue、pass和循环else
- 一般循环格式
- pass
- continue
- break
- 循环else
- 为什么要在意“模拟C语言的while循环”
- for循环
- 一般格式
- 例子
- 为什么要在意“文件扫描”
- 编写循环的技巧
- 循环计数器：while和range
- 非完备遍历：range和分片
- 修改列表：range
- 并行遍历：zip和map
- 产生偏移和元素：enumerate
- 本章小结
- 本章习题
- 习题解答
- 第14章 迭代器和解析，第一部分
- 迭代器：初探
- 文件迭代器
- 手动迭代：iter和next
- 其他内置类型迭代器
- 列表解析：初探
- 列表解析基础知识
- 在文件上使用列表解析
- 扩展的列表解析语法
- 其他迭代环境
- Python 3.0中的新的可迭代对象
- range迭代器
- map、zip和filter迭代器
- 多个迭代器 VS 单个迭代器
- 字典视图迭代器
- 其他迭代器主题

本章小结

本章习题

习题解答

第15章 文档

Python文档资源

#注释

dir函数

文档字符串：__doc__

PyDoc：help函数

PyDoc：HTML报表

标准手册集

网络资源

已出版的书籍

常见编写代码的陷阱

本章小结

本章习题

习题解答

第三部分 练习题

第四部分 函数

第16章 函数基础

为何使用函数

编写函数

def语句

def语句是实时执行的

第一个例子：定义和调用

定义

调用

Python中的多态

第二个例子：寻找序列的交集

定义

调用

重访多态

本地变量

本章小结

本章习题

习题解答

第17章 作用域

Python作用域基础

作用域法则

变量名解析：LEGB原则

作用域实例

内置作用域

在Python 2.6中违反通用性

global语句

最小化全局变量

最小化文件间的修改

其他访问全局变量的方法

作用域和嵌套函数
嵌套作用域的细节
嵌套作用域举例
nonlocal语句
nonlocal基础
nonlocal应用
为什么使用nonlocal
本章小结
本章习题
习题解答
第18章 参数
传递参数
参数和共享引用
避免可变参数的修改
对参数输出进行模拟
特定的参数匹配模型
基础知识
匹配语法
细节
关键字参数和默认参数的实例
任意参数的实例
Python 3.0 Keyword-Only参数
min调用
满分
加分点
结论
一个更有用的例子：通用set函数
模拟Python 3.0 print函数
使用Keyword-Only参数
为什么要在意：关键字参数
本章小结
本章习题
习题解答
第19章 函数的高级话题
函数设计概念
递归函数
用递归求和
编码替代方案
循环语句VS递归
处理任意结构
函数对象：属性和注解
间接函数调用
函数自省
函数属性
Python 3.0中的函数注解
匿名函数：lambda
lambda表达式

为什么使用lambda
如何 (不要) 让Python代码变得晦涩难懂
嵌套lambda和作用域
为什么要在意：回调
在序列中映射函数：map
函数式编程工具：filter和reduce
本章小结
本章习题
习题解答
第20章 迭代和解析，第二部分
回顾列表解析：函数式编程工具
列表解析与map
增加测试和嵌套循环
列表解析和矩阵
理解列表解析
为什么要在意：列表解析和map
重访迭代器：生成器
生成器函数：yield VS return
生成器表达式：迭代器遇到列表解析
生成器函数 VS 生成器表达式
生成器是单迭代器对象
用迭代工具模拟zip和map
为什么你会留意：单次迭代
内置类型和类中的值生成
Python 3.0解析语法概括
解析集合和字典解析
针对集合和字典的扩展的解析语法
对迭代的各种方法进行计时
对模块计时
计时脚本
计时结果
计时模块替代方案
其他建议
函数陷阱
本地变量是静态检测的
默认和可变对象
没有return语句的函数
嵌套作用域的循环变量
本章小结
本章习题
习题解答
第四部分练习题
第五部分 模块
第21章 模块：宏伟蓝图
为什么使用模块
Python程序架构
如何组织一个程序

- 导入和属性
- 标准库模块
- import如何工作
- 1.搜索
- 2.编译 (可选)
- 3.运行
- 模块搜索路径
- 配置搜索路径
- 搜索路径的变动
- sys.path列表
- 模块文件选择
- 高级的模块选择概念
- 第三方工具: distutils
- 本章小结
- 本章习题
- 习题解答
- 第22章 模块代码编写基础
- 模块的创建
- 模块的使用
- import语句
- from语句
- from *语句
- 导入只发生一次
- import和from是赋值语句
- 文件间变量名的改变
- import和from的对等性
- from语句潜在的陷阱
- 模块命名空间
- 文件生成命名空间
- 属性名的点号运算
- 导入和作用域
- 命名空间的嵌套
- 重载模块
- reload基础
- reload实例
- 为什么要在意: 模块重载
- 本章小结
- 本章习题
- 习题解答
- 第23章 模块包
- 包导入基础
- 包和搜索路径设置
- __init__.py包文件
- 包导入实例
- 包对应的from语句和import语句
- 为什么要使用包导入
- 三个系统的传说

<<Python学习手册 (第4版) >>

包相对导入

Python 3.0中的变化

相对导入基础知识

为什么使用相对导入

相对导入的作用域

模块查找规则总结

相对导入的应用

为什么要在意：模块包

本章小结

本章习题

习题解答

第24章 高级模块话题

在模块中隐藏数据

最小化from *的破坏：__X__和__all__

启用以后的语言特性

混合用法模式：__name__和__main__

以__name__进行单元测试

使用带有__name__的命令行参数

修改模块搜索路径

Import语句和from语句的as扩展

模块是对象：元程序

用名称字符串导入模块

过渡性模块重载

模块设计理念

模块陷阱

顶层代码的语句次序的重要性

from复制变量名，而不是连接

from *会让变量语义模糊

reload不会影响from导入

reload、from以及交互模式测试

递归形式的from导入无法工作

本章小结

本章习题

习题解答

第五部分练习题

第六部分 类和OOP

第25章 OOP：宏伟蓝图

为何使用类

概览OOP

属性继承搜索

类和实例

类方法调用

编写类树

OOP是为了代码重用

本章小结

本章习题

习题解答

第26章 类代码编写基础

类产生多个实例对象

类对象提供默认行为

实例对象是具体的元素

第一个例子

类通过继承进行定制

第二个例子

类是模块内的属性

类可以截获Python运算符

第三个例子

为什么要使用运算符重载

世界上最简单的Python类

类与字典的关系

本章小结

本章习题

习题解答

第27章 更多实例

步骤1：创建实例

编写构造函数

在进行中测试

以两种方式使用代码

版本差异提示

步骤2：添加行为方法

编写方法

步骤3：运算符重载

提供打印显示

步骤4：通过子类定制行为

编写子类

扩展方法：不好的方式

扩展方法：好的方式

多态的作用

继承、定制和扩展

OOP：大思路

步骤5：定制构造函数

OOP比我们认为的要简单

组合类的其他方式

在Python 3.0中捕获内置属性

步骤6：使用自省工具

特殊类属性

一种通用显示工具

实例与类属性的关系

工具类的命名考虑

类的最终形式

步骤7（最后一步）：把对象存储到数据库中

Pickle和Shelve

在shelve数据库中存储对象

交互地探索shelve

更新shelve中的对象

未来方向

本章小结

本章习题

习题解答

第28章 类代码编写细节

class语句

一般形式

例子

方法

例子

调用超类构造函数

其他方法调用的可能性

继承

属性树的构造

继承方法的专有化

类接口技术

抽象超类

Python 2.6和Python 3.0的抽象超类

命名空间：完整的内容

简单变量名：如果赋值就不是全局变量

属性名称：对象命名空间

Python命名空间的“禅”：赋值将变量名分类

命名空间字典

命名空间链接

回顾文档字符串

类与模块的关系

本章小结

本章习题

习题解答

第29章 运算符重载

基础知识

构造函数和表达式：__init__和__sub__

常见的运算符重载方法

索引和分片：__getitem__和__setitem__

拦截分片

Python 2.6中的分片和索引

索引迭代：__getitem__

迭代器对象：__iter__和__next__

用户定义的迭代器

有多个迭代器的对象

成员关系：__contains__、__iter__和__getitem__

属性引用：__getattr__和__setattr__

其他属性管理工具

模拟实例属性的私有性：第一部分

__repr__和__str__会返回字符串表达形式

右侧加法和原处加法：__radd__和__iadd__

<<Python学习手册 (第4版) >>

原处加法

Call表达式：__call__

函数接口和回调代码

比较：__lt__、__gt__ 和其他方法

Python 2.6的__cmp__方法（已经从Python 3.0中移除了）

布尔测试：__bool__和__len__

Python 2.6中的布尔

对象析构函数：__del__

本章小结

本章习题

习题解答

第30章 类的设计

Python和OOP

通过调用标记进行重载（或不要）

OOP和继承：“是一个”关系

OOP和组合：“有一个”关系

重访流处理器

为什么要在意：类和持续性

OOP和委托：“包装”对象

类的伪私有属性

变量名压缩概览

为什么使用伪私有属性

方法是对象：绑定或无绑定

在Python 3.0中，无绑定方法是函数

绑定方法和其他可调用对象

为什么要在意：绑定方法和回调函数

多重继承：“混合”类

编写混合显示类

类是对象：通用对象的工厂

为什么有工厂

与设计相关的其他话题

本章小结

本章习题

习题解答

第31章 类的高级主题

扩展内置类型

通过嵌入扩展类型

通过子类扩展类型

新式类

新式类变化

类型模式变化

钻石继承变动

新式类的扩展

slots实例

类特性

__getattr__和描述符

元类

<<Python学习手册 (第4版) >>

静态方法和类方法
为什么使用特殊方法
Python 2.6和Python 3.0中的静态方法
静态方法替代方案
使用静态和类方法
使用静态方法统计实例
用类方法统计实例
装饰器和元类：第一部分
函数装饰器基础
装饰器例子
类装饰器和元类
更多详细信息
类陷阱
修改类属性的副作用
修改可变的类属性也可能产生副作用
多重继承：顺序很重要
类、方法以及嵌套作用域
Python中基于委托的类：__getattr__和内置函数
“过度包装”
本章小结
本章习题
习题解答
第六部分 练习题
为什么要在意：大师眼中的OOP
第七部分 异常和工具
第32章 异常基础
为什么使用异常
异常的角色
异常处理：简明扼要
默认异常处理器
捕获异常
引发异常
用户定义的异常
终止行为
为什么要在意：错误检查
本章小结
本章习题
习题解答
第33章 异常编码细节
try/except/else语句
try语句分句
try/else分句
例子：默认行为
例子：捕捉内置异常
try/finally语句
例子：利用try/finally编写终止行为
统一try/except/finally语句

统一try语句语法
通过嵌套合并finally和except
合并try的例子
raise语句
利用raise传递异常
Python 3.0异常链：raise from
assert语句
例子：收集约束条件（但不是错误）
with/as环境管理器
基本使用
环境管理协议
本章小结
本章习题
习题解答
第34章 异常对象
异常：回到未来
字符串异常很简单
基于类的异常
类异常例子
为什么使用类异常
内置Exception类
内置异常分类
默认打印和状态
定制打印显示
定制数据和行为
提供异常细节
提供异常方法
本章小结
本章习题
习题解答
第35章 异常的设计
嵌套异常处理器
例子：控制流程嵌套
例子：语法嵌套化
异常的习惯用法
异常不总是错误
函数信号条件和raise
关闭文件和服务器连接
在try外进行调试
运行进程中的测试
关于sys.exc_info
与异常有关的技巧
应该包装什么
捕捉太多：避免空except语句
捕捉过少：使用基于类的分类
核心语言总结
Python工具集

大型项目的开发工具
本章小结
第七部分练习题
第八部分 高级话题注
第36章 Unicode和字节字符串
Python 3.0中的字符串修改
字符串基础知识
字符编码方法
Python的字符串类型
文本和二进制文件
Python 3.0中的字符串应用
常量和基本属性
转换
编码Unicode字符串
编码ASCII文本
编码非ASCII文本
编码和解码非ASCII文本
其他Unicode编码技术
转换编码
在Python 2.6中编码Unicode字符串
源文件字符集编码声明
使用Python 3.0 Bytes对象
方法调用
序列操作
创建bytes对象的其他方式
混合字符串类型
使用Python 3.0 (和Python 2.6) bytearray对象
使用文本文件和二进制文件
文本文件基础
Python 3.0中的文本和二进制模式
类型和内容错误匹配
使用Unicode文件
在Python 3.0中读取和写入Unicode
在Python 3.0中处理BOM
Python 2.6中的Unicode文件
Python 3.0中其他字符串工具的变化
re模式匹配模块
Struct二进制数据模块
pickle对象序列化模块
XML解析工具
本章小结
本章习题
习题解答
第37章 管理属性
为什么管理属性
插入在属性访问时运行的代码
特性

基础知识

第一个例子

计算的属性

使用装饰器编写特性

描述符

基础知识

第一个示例

计算的属性

在描述符中使用状态信息

特性和描述符是如何相关的

`__getattr__`和`__getattribute__`

基础知识

第一个示例

计算属性

`__getattr__`和`__getattribute__`比较

管理技术比较

拦截内置操作属性

重访基于委托的Manager

示例：属性验证

使用特性来验证

使用描述符验证

使用`__getattr__`来验证

使用`__getattribute__`验证

本章小结

本章习题

习题解答

第38章 装饰器

什么是装饰器

管理调用和实例

管理函数和类

使用和定义装饰器

为什么使用装饰器

基础知识

函数装饰器

类装饰器

装饰器嵌套

装饰器参数

装饰器管理函数和类

编写函数装饰器

跟踪调用

状态信息保持选项

类错误之一：装饰类方法

计时调用

添加装饰器参数

编写类装饰器

单体类

跟踪对象接口

类错误之二：保持多个实例
装饰器与管理器函数的关系
为什么使用装饰器（重访）
直接管理函数和类
示例：“私有”和“公有”属性
实现私有属性
实现细节之一
公有声明的泛化
实现细节之二
开放问题
Python不是关于控制
示例：验证函数参数
目标
针对位置参数的一个基本范围测试装饰器
针对关键字和默认泛化
实现细节
开放问题
装饰器参数 VS 函数注解
其他应用程序：类型测试
本章小结
本章习题
习题解答
第39章 元类
要么是元类，要么不是元类
提高魔力层次
“辅助”函数的缺点
元类与类装饰器的关系：第一回合
元类模型
类是类型的实例
元类是Type的子类
Class语句协议
声明元类
编写元类
基本元类
定制构建和初始化
其他元类编程技巧
实例与继承的关系
示例：向类添加方法
手动扩展
基于元类的扩展
元类与类装饰器的关系：第二回合
示例：对方法应用装饰器
用装饰器手动跟踪
用元类和装饰器跟踪
把任何装饰器应用于方法
元类与类装饰器的关系：第三回合
“可选的”语言功能

本章小结

本章习题

习题解答

第九部分 附录注

附录A 安装和配置

附录B 各部分练习题的解答

章节摘录

版权页：插图：除了较大的结构性目标，类设计往往也必须解决名称用法。

在第五部分中，我们学到了每个在模块文件顶层赋值的变量名都会导出。

在默认情况下，类也是这样：数据隐藏是一个惯例，客户端可以读取或修改任何它们想要的类或实例的属性。

事实上，用C++术语来讲，属性都是“public”和“virtual”，在任意地方都可进行读取，并且在运行时进行动态查找。

如今依然如此。

然而，Python也支持变量名压缩（mangling，相当于扩张）的概念，让类内某些变量局部化。

压缩后的变量名有时会被误认为是“私有属性”，但这其实只是一种把类所创建的变量名局部化的方式而已：名称压缩并无法阻止类外代码对它的读取。

这种功能主要是为了避免实例内的命名空间的冲突，而不是限制变量名的读取。

因此，压缩的变量名最好称为“伪私有”，而不是“私有”。

伪私有变量名是高级且完全可选的功能，除非你开始在多人的项目中编写大型的类的层次，否则可能不会觉得有什么用处。

实际上，即便当它们可能应该使用的时候，也并非总是使用它们——更通俗地说，Python程序员用一个单个的下划线来编写内部名称（例如，`_X`），这只是一个非正式的惯例，让你知道这是一个不应该修改的名字（它对Python自身来说没有什么意义）。

媒体关注与评论

对于那些想要开始使用Python编程的人来说，本书是我所推荐图书中的首选。

” ——Doug Hellmann Racemi公司，高级软件工程师

编辑推荐

Google和YouTube由于PythonF的高可适应性、易于维护以及适合于快速开发而采用它。

《Python学习手册(第4版)》将帮助你使用Python编写出高质量、高效的并且易于与其他语言和工具集成的代码。

《Python学习手册(第4版)》根据Python专家MarkLutz的著名培训课程编写而成，是易于掌握和自学的Python教程。

《Python学习手册(第4版)》每一章都对Python语言的关键内容做单独讲解，并且配有章首习题，便于你学习新的技能并巩固加深自己的理解。

书中配有大量注释的示例以及图表，它们都将帮助你轻松地学习Python3.0。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>