Python学习计划

**更改历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **状态** | **日期** | **责任人** | **更改原因** |
| 1.0 | 草稿 | 2024-04-25 | liuyongqi (刘永琪) | 创建 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[Python学习计划 1](#_Toc165208759)

[目录 2](#_Toc165208760)

[1 介绍 9](#_Toc165208761)

[1.1 文档目的 9](#_Toc165208762)

[1.2 术语和缩写 9](#_Toc165208763)

[1.2.1 术语 9](#_Toc165208764)

[1.3 参考文档 9](#_Toc165208765)

[2 编码规范 10](#_Toc165208766)

[2.1 PEP8 10](#_Toc165208767)

[2.1.1 概述 10](#_Toc165208768)

[2.1.2 Pylint工具 10](#_Toc165208769)

[2.2 一致性考虑 10](#_Toc165208770)

[2.3 代码布局 10](#_Toc165208771)

[2.3.1 缩进 10](#_Toc165208772)

[2.3.2 Tabs或空格 12](#_Toc165208773)

[2.3.3 最大行宽 12](#_Toc165208774)

[2.3.4 换行在二元运算符前还是后（推荐在换行符之后） 12](#_Toc165208775)

[2.3.5 空行 12](#_Toc165208776)

[2.3.6 源文件编码 12](#_Toc165208777)

[2.3.7 导入（Imports） 12](#_Toc165208778)

[3 解释器 13](#_Toc165208779)

[3.1 课前甜点 13](#_Toc165208780)

[3.2 Python解释器 13](#_Toc165208781)

[3.2.1 调用解释器 13](#_Toc165208782)

[3.2.2 传入参数 13](#_Toc165208783)

[3.2.3 交互模式 13](#_Toc165208784)

[3.2.4 源文件的字符编码 13](#_Toc165208785)

[3.3 Python速览 13](#_Toc165208786)

[4 语法 13](#_Toc165208787)

[4.1 词法分析 13](#_Toc165208788)

[4.1.1 行结构 13](#_Toc165208789)

[4.1.2 其他形符 15](#_Toc165208790)

[4.1.3 标识符和关键字 15](#_Toc165208791)

[4.1.4 字面值 15](#_Toc165208792)

[4.1.5 运算符 15](#_Toc165208793)

[4.1.6 分隔符 15](#_Toc165208794)

[4.2 数据模型 15](#_Toc165208795)

[4.2.1 对象、值与类型 15](#_Toc165208796)

[4.2.2 标准类型层级结构 15](#_Toc165208797)

[4.2.3 特殊方法名称 15](#_Toc165208798)

[4.2.4 协程 15](#_Toc165208799)

[4.3 执行模型 15](#_Toc165208800)

[4.3.1 程序的结构 15](#_Toc165208801)

[4.3.2 命名与绑定 15](#_Toc165208802)

[4.3.3 异常 15](#_Toc165208803)

[4.4 导入系统 15](#_Toc165208804)

[4.4.1 importlib 15](#_Toc165208805)

[4.4.2 包 15](#_Toc165208806)

[4.4.3 搜索 15](#_Toc165208807)

[4.4.4 加载 15](#_Toc165208808)

[4.4.5 基于路径的查找器 15](#_Toc165208809)

[4.4.6 替换标准导入系统 15](#_Toc165208810)

[4.4.7 包相对导入 15](#_Toc165208811)

[4.4.8 有关 \_\_main\_\_ 的特殊事项 15](#_Toc165208812)

[4.4.9 开放问题项 15](#_Toc165208813)

[4.4.10 参考文献 15](#_Toc165208814)

[4.5 表达式 15](#_Toc165208815)

[4.5.1 算术转换 15](#_Toc165208816)

[4.5.2 原子 15](#_Toc165208817)

[4.5.3 原型 15](#_Toc165208818)

[4.5.4 await 表达式 15](#_Toc165208819)

[4.5.5 幂运算符 15](#_Toc165208820)

[4.5.6 一元算术和位运算 16](#_Toc165208821)

[4.5.7 二元算术运算符 16](#_Toc165208822)

[4.5.8 移位运算 16](#_Toc165208823)

[4.5.9 二元位运算 16](#_Toc165208824)

[4.5.10 比较运算 16](#_Toc165208825)

[4.5.11 布尔运算 16](#_Toc165208826)

[4.5.12 赋值表达式 16](#_Toc165208827)

[4.5.13 条件表达式 16](#_Toc165208828)

[4.5.14 lambda 表达式 16](#_Toc165208829)

[4.5.15 表达式列表 16](#_Toc165208830)

[4.5.16 求值顺序 16](#_Toc165208831)

[4.5.17 运算符优先级 16](#_Toc165208832)

[4.6 简单语句 16](#_Toc165208833)

[4.6.1 表达式语句 16](#_Toc165208834)

[4.6.2 赋值语句 16](#_Toc165208835)

[4.6.3 assert 语句 16](#_Toc165208836)

[4.6.4 pass 语句 16](#_Toc165208837)

[4.6.5 del 语句 16](#_Toc165208838)

[4.6.6 return 语句 16](#_Toc165208839)

[4.6.7 yield 语句 16](#_Toc165208840)

[4.6.8 raise 语句 16](#_Toc165208841)

[4.6.9 break 语句 16](#_Toc165208842)

[4.6.10 continue 语句 16](#_Toc165208843)

[4.6.11 import 语句 16](#_Toc165208844)

[4.6.12 global 语句 16](#_Toc165208845)

[4.6.13 nonlocal 语句 16](#_Toc165208846)

[4.7 复合语句 16](#_Toc165208847)

[4.7.1 if 语句 16](#_Toc165208848)

[4.7.2 while 语句 16](#_Toc165208849)

[4.7.3 for 语句 16](#_Toc165208850)

[4.7.4 try 语句 16](#_Toc165208851)

[4.7.5 with 语句 16](#_Toc165208852)

[4.7.6 函数定义 16](#_Toc165208853)

[4.7.7 类定义 16](#_Toc165208854)

[4.7.8 协程 16](#_Toc165208855)

[4.8 最高层级组件 16](#_Toc165208856)

[4.8.1 完整的 Python 程序 16](#_Toc165208857)

[4.8.2 文件输入 16](#_Toc165208858)

[4.8.3 交互式输入 16](#_Toc165208859)

[4.8.4 表达式输入 16](#_Toc165208860)

[4.9 完整的语法规范 16](#_Toc165208861)

[5 标准库 16](#_Toc165208862)

[5.1 概述 18](#_Toc165208863)

[5.1.1 可用性注释 18](#_Toc165208864)

[5.2 内置函数 18](#_Toc165208865)

[5.3 内置常量 18](#_Toc165208866)

[5.3.1 由 site 模块添加的常量 18](#_Toc165208867)

[5.4 内置类型 18](#_Toc165208868)

[5.4.1 逻辑值检测 18](#_Toc165208869)

[5.4.2 布尔运算 --- and, or, not 18](#_Toc165208870)

[5.4.3 比较运算 18](#_Toc165208871)

[5.4.4 数字类型 --- int, float, complex 18](#_Toc165208872)

[5.4.5 迭代器类型 18](#_Toc165208873)

[5.4.6 序列类型 --- list, tuple, range 18](#_Toc165208874)

[5.4.7 文本序列类型 --- str 18](#_Toc165208875)

[5.4.8 二进制序列类型 --- bytes, bytearray, memoryview 18](#_Toc165208876)

[5.4.9 集合类型 --- set, frozenset 18](#_Toc165208877)

[5.4.10 映射类型 --- dict 18](#_Toc165208878)

[5.4.11 上下文管理器类型 18](#_Toc165208879)

[5.4.12 其他内置类型 18](#_Toc165208880)

[5.4.13 特殊属性 18](#_Toc165208881)

[5.4.14 整数字符串转换长度限制 18](#_Toc165208882)

[5.5 内置异常 18](#_Toc165208883)

[5.5.1 基类 18](#_Toc165208884)

[5.5.2 具体异常 18](#_Toc165208885)

[5.5.3 警告 18](#_Toc165208886)

[5.5.4 异常层次结构 18](#_Toc165208887)

[5.6 文本处理服务 18](#_Toc165208888)

[5.6.1 string --- 常见的字符串操作 18](#_Toc165208889)

[5.6.2 re --- 正则表达式操作 18](#_Toc165208890)

[5.6.3 difflib --- 计算差异的辅助工具 18](#_Toc165208891)

[5.6.4 textwrap --- 文本自动换行与填充 18](#_Toc165208892)

[5.6.5 unicodedata --- Unicode 数据库 18](#_Toc165208893)

[5.6.6 stringprep --- 因特网字符串预备 18](#_Toc165208894)

[5.6.7 readline --- GNU readline 接口 18](#_Toc165208895)

[5.6.8 rlcompleter --- GNU readline 的补全函数 18](#_Toc165208896)

[5.7 二进制数据服务 18](#_Toc165208897)

[5.7.1 struct --- 将字节串解读为打包的二进制数据 18](#_Toc165208898)

[5.7.2 codecs --- 编解码器注册和相关基类 18](#_Toc165208899)

[5.8 数据类型 18](#_Toc165208900)

[5.8.1 datetime --- 基本日期和时间类型 18](#_Toc165208901)

[5.8.2 calendar --- 日历相关函数 18](#_Toc165208902)

[5.8.3 collections --- 容器数据类型 18](#_Toc165208903)

[5.8.4 collections.abc --- 容器的抽象基类 18](#_Toc165208904)

[5.8.5 heapq --- 堆队列算法 18](#_Toc165208905)

[5.8.6 bisect --- 数组二分查找算法 18](#_Toc165208906)

[5.8.7 array --- 高效的数字数组 18](#_Toc165208907)

[5.8.8 weakref --- 弱引用 19](#_Toc165208908)

[5.8.9 types --- 动态类型创建和内置类型名称 19](#_Toc165208909)

[5.8.10 copy --- 浅层 (shallow) 和深层 (deep) 复制操作 19](#_Toc165208910)

[5.8.11 pprint --- 数据美化输出 19](#_Toc165208911)

[5.8.12 reprlib --- 另一种 repr() 实现 19](#_Toc165208912)

[5.8.13 enum --- 对枚举的支持 19](#_Toc165208913)

[5.9 数字和数学模块 19](#_Toc165208914)

[5.9.1 numbers --- 数字的抽象基类 19](#_Toc165208915)

[5.9.2 math --- 数学函数 19](#_Toc165208916)

[5.9.3 cmath --- 关于复数的数学函数 19](#_Toc165208917)

[5.9.4 decimal --- 十进制定点和浮点运算 19](#_Toc165208918)

[5.9.5 fractions --- 分数 19](#_Toc165208919)

[5.9.6 random --- 生成伪随机数 19](#_Toc165208920)

[5.9.7 statistics --- 数学统计函数 19](#_Toc165208921)

[5.10 函数式编程模块 19](#_Toc165208922)

[5.10.1 itertools --- 为高效循环而创建迭代器的函数 19](#_Toc165208923)

[5.10.2 functools --- 高阶函数和可调用对象上的操作 19](#_Toc165208924)

[5.10.3 operator --- 标准运算符替代函数 19](#_Toc165208925)

[5.11 文件和目录访问 19](#_Toc165208926)

[5.11.1 pathlib --- 面向对象的文件系统路径 19](#_Toc165208927)

[5.11.2 os.path --- 常用路径操作 19](#_Toc165208928)

[5.11.3 fileinput --- 迭代来自多个输入流的行 19](#_Toc165208929)

[5.11.4 stat --- 解析 stat() 结果 19](#_Toc165208930)

[5.11.5 filecmp --- 文件及目录的比较 19](#_Toc165208931)

[5.11.6 tempfile --- 生成临时文件和目录 19](#_Toc165208932)

[5.11.7 glob --- Unix 风格路径名模式扩展 19](#_Toc165208933)

[5.11.8 fnmatch --- Unix 文件名模式匹配 19](#_Toc165208934)

[5.11.9 linecache --- 随机读写文本行 19](#_Toc165208935)

[5.11.10 shutil --- 高阶文件操作 19](#_Toc165208936)

[5.12 数据持久化 19](#_Toc165208937)

[5.12.1 pickle --- Python 对象序列化 19](#_Toc165208938)

[5.12.2 copyreg --- 注册配合 pickle 模块使用的函数 19](#_Toc165208939)

[5.12.3 shelve --- Python 对象持久化 19](#_Toc165208940)

[5.12.4 marshal --- 内部 Python 对象序列化 19](#_Toc165208941)

[5.12.5 dbm --- Unix "数据库" 接口 19](#_Toc165208942)

[5.12.6 sqlite3 --- SQLite 数据库 DB-API 2.0 接口模块 19](#_Toc165208943)

[5.13 数据压缩和存档 19](#_Toc165208944)

[5.13.1 zlib --- 与 gzip 兼容的压缩 19](#_Toc165208945)

[5.13.2 gzip --- 对 gzip 格式的支持 19](#_Toc165208946)

[5.13.3 bz2 --- 对 bzip2 压缩算法的支持 19](#_Toc165208947)

[5.13.4 lzma --- 用 LZMA 算法压缩 19](#_Toc165208948)

[5.13.5 zipfile --- 使用ZIP存档 19](#_Toc165208949)

[5.13.6 tarfile --- 读写tar归档文件 19](#_Toc165208950)

[5.14 文件格式 19](#_Toc165208951)

[5.14.1 csv --- CSV 文件读写 19](#_Toc165208952)

[5.14.2 configparser --- 配置文件解析器 20](#_Toc165208953)

[5.14.3 netrc --- netrc 文件处理 20](#_Toc165208954)

[5.14.4 xdrlib --- 编码与解码 XDR 数据 20](#_Toc165208955)

[5.14.5 plistlib --- 生成与解析 Mac OS X .plist 文件 20](#_Toc165208956)

[5.15 加密服务 20](#_Toc165208957)

[5.15.1 hashlib --- 安全哈希与消息摘要 20](#_Toc165208958)

[5.15.2 hmac --- 基于密钥的消息验证 20](#_Toc165208959)

[5.15.3 secrets --- 生成安全随机数字用于管理密码 20](#_Toc165208960)

[5.16 通用操作系统服务 20](#_Toc165208961)

[5.16.1 os --- 多种操作系统接口 20](#_Toc165208962)

[5.16.2 io --- 处理流的核心工具 20](#_Toc165208963)

[5.16.3 time --- 时间的访问和转换 20](#_Toc165208964)

[5.16.4 argparse --- 命令行选项、参数和子命令解析器 20](#_Toc165208965)

[5.16.5 getopt --- C 风格的命令行选项解析器 20](#_Toc165208966)

[5.16.6 logging --- Python 的日志记录工具 20](#_Toc165208967)

[5.16.7 logging.config --- 日志记录配置 20](#_Toc165208968)

[5.16.8 logging.handlers --- 日志处理程序 20](#_Toc165208969)

[5.16.9 getpass --- 便携式密码输入工具 20](#_Toc165208970)

[5.16.10 curses --- 终端字符单元显示的处理 20](#_Toc165208971)

[5.16.11 curses.textpad --- 用于 curses 程序的文本输入控件 20](#_Toc165208972)

[5.16.12 curses.ascii --- 用于 ASCII 字符的工具 20](#_Toc165208973)

[5.16.13 curses.panel --- curses 的 panel 栈扩展 20](#_Toc165208974)

[5.16.14 platform --- 获取底层平台的标识数据 20](#_Toc165208975)

[5.16.15 errno --- 标准 errno 系统符号 20](#_Toc165208976)

[5.16.16 ctypes --- Python 的外部函数库 20](#_Toc165208977)

[5.17 并发执行 20](#_Toc165208978)

[5.17.1 threading --- 基于线程的并行 20](#_Toc165208979)

[5.17.2 multiprocessing --- 基于进程的并行 20](#_Toc165208980)

[5.17.3 multiprocessing.shared\_memory --- 可从进程直接访问的共享内存 20](#_Toc165208981)

[5.17.4 concurrent 包 20](#_Toc165208982)

[5.17.5 concurrent.futures --- 启动并行任务 20](#_Toc165208983)

[5.17.6 subprocess --- 子进程管理 20](#_Toc165208984)

[5.17.7 sched --- 事件调度器 20](#_Toc165208985)

[5.17.8 queue --- 一个同步的队列类 20](#_Toc165208986)

[5.17.9 contextvars 上下文变量 20](#_Toc165208987)

[5.17.10 \_thread --- 底层多线程 API 20](#_Toc165208988)

[5.17.11 \_dummy\_thread --- \_thread 的替代模块 20](#_Toc165208989)

[5.17.12 dummy\_threading --- 可直接替代 threading 模块。 20](#_Toc165208990)

[5.18 网络和进程间通信 20](#_Toc165208991)

[5.18.1 asyncio --- 异步 I/O 20](#_Toc165208992)

[5.18.2 socket --- 底层网络接口 20](#_Toc165208993)

[5.18.3 ssl --- 套接字对象的TLS/SSL封装 20](#_Toc165208994)

[5.18.4 select --- 等待 I/O 完成 20](#_Toc165208995)

[5.18.5 selectors --- 高级 I/O 复用库 20](#_Toc165208996)

[5.18.6 asyncore --- 异步socket处理器 20](#_Toc165208997)

[5.18.7 asynchat --- 异步 socket 指令/响应 处理器 21](#_Toc165208998)

[5.18.8 signal --- 设置异步事件处理程序 21](#_Toc165208999)

[5.18.9 mmap --- 内存映射文件支持 21](#_Toc165209000)

[5.19 互联网数据处理 21](#_Toc165209001)

[5.19.1 email --- 电子邮件与 MIME 处理包 21](#_Toc165209002)

[5.19.2 json --- JSON 编码和解码器 21](#_Toc165209003)

[5.19.3 mailcap --- Mailcap 文件处理 21](#_Toc165209004)

[5.19.4 mailbox --- 操作多种格式的邮箱 21](#_Toc165209005)

[5.19.5 mimetypes --- 映射文件夹到 MIME 类型 21](#_Toc165209006)

[5.19.6 base64 --- Base16, Base32, Base64, Base85 数据编码 21](#_Toc165209007)

[5.19.7 binhex --- 对binhex4文件进行编码和解码 21](#_Toc165209008)

[5.19.8 binascii --- 二进制和 ASCII 码互转 21](#_Toc165209009)

[5.19.9 quopri --- 编码与解码经过 MIME 转码的可打印数据 21](#_Toc165209010)

[5.19.10 uu --- 对 uuencode 文件进行编码与解码 21](#_Toc165209011)

[5.20 结构化标记处理工具 21](#_Toc165209012)

[5.20.1 html --- 超文本标记语言支持 21](#_Toc165209013)

[5.20.2 html.parser --- 简单的 HTML 和 XHTML 解析器 21](#_Toc165209014)

[5.20.3 html.entities --- HTML 一般实体的定义 21](#_Toc165209015)

[5.20.4 XML处理模块 21](#_Toc165209016)

[5.20.5 xml.etree.ElementTree --- ElementTree XML API 21](#_Toc165209017)

[5.20.6 xml.dom --- 文档对象模型 API 21](#_Toc165209018)

[5.20.7 xml.dom.minidom --- 最小化的 DOM 实现 21](#_Toc165209019)

[5.20.8 xml.dom.pulldom --- 支持构建部分 DOM 树 21](#_Toc165209020)

[5.20.9 xml.sax --- 支持 SAX2 解析器 21](#_Toc165209021)

[5.20.10 xml.sax.handler --- SAX 处理程序的基类 21](#_Toc165209022)

[5.20.11 xml.sax.saxutils --- SAX 工具集 21](#_Toc165209023)

[5.20.12 xml.sax.xmlreader --- 用于 XML 解析器的接口 21](#_Toc165209024)

[5.20.13 xml.parsers.expat --- 使用 Expat 的快速 XML 解析 21](#_Toc165209025)

[5.21 互联网协议和支持 21](#_Toc165209026)

[5.21.1 webbrowser --- 方便的Web浏览器控制器 21](#_Toc165209027)

[5.21.2 cgi --- 通用网关接口支持 21](#_Toc165209028)

[5.21.3 cgitb --- 用于 CGI 脚本的回溯管理器 21](#_Toc165209029)

[5.21.4 wsgiref --- WSGI 工具和参考实现 21](#_Toc165209030)

[5.21.5 urllib --- URL 处理模块 21](#_Toc165209031)

[5.21.6 urllib.request --- 用于打开 URL 的可扩展库 21](#_Toc165209032)

[5.21.7 urllib.response --- urllib 使用的 Response 类 21](#_Toc165209033)

[5.21.8 urllib.parse 用于解析 URL 21](#_Toc165209034)

[5.21.9 urllib.error --- urllib.request 引发的异常类 21](#_Toc165209035)

[5.21.10 urllib.robotparser --- robots.txt 语法分析程序 21](#_Toc165209036)

[5.21.11 http --- HTTP 模块 21](#_Toc165209037)

[5.21.12 http.client --- HTTP 协议客户端 21](#_Toc165209038)

[5.21.13 ftplib --- FTP 协议客户端 21](#_Toc165209039)

[5.21.14 poplib --- POP3 协议客户端 21](#_Toc165209040)

[5.21.15 imaplib --- IMAP4 协议客户端 21](#_Toc165209041)

[5.21.16 nntplib --- NNTP 协议客户端 21](#_Toc165209042)

[5.21.17 smtplib ---SMTP协议客户端 22](#_Toc165209043)

[5.21.18 smtpd --- SMTP 服务器 22](#_Toc165209044)

[5.21.19 telnetlib -- Telnet 客户端 22](#_Toc165209045)

[5.21.20 uuid --- RFC 4122 定义的UUID对象 22](#_Toc165209046)

[5.21.21 socketserver --- 用于网络服务器的框架 22](#_Toc165209047)

[5.21.22 http.server --- HTTP 服务器 22](#_Toc165209048)

[5.21.23 http.cookies --- HTTP状态管理 22](#_Toc165209049)

[5.21.24 http.cookiejar —— HTTP 客户端的 Cookie 处理 22](#_Toc165209050)

[5.21.25 xmlrpc --- XMLRPC 服务端与客户端模块 22](#_Toc165209051)

[5.21.26 xmlrpc.client --- XML-RPC 客户端访问 22](#_Toc165209052)

[5.21.27 xmlrpc.server --- 基本 XML-RPC 服务器 22](#_Toc165209053)

[5.21.28 ipaddress --- IPv4/IPv6 操作库 22](#_Toc165209054)

[5.22 多媒体服务 22](#_Toc165209055)

[5.22.1 audioop --- 处理原始音频数据 22](#_Toc165209056)

[5.22.2 aifc --- 读写 AIFF 和 AIFC 文件 22](#_Toc165209057)

[5.22.3 sunau --- 读写 Sun AU 文件 22](#_Toc165209058)

[5.22.4 wave --- 读写WAV格式文件 22](#_Toc165209059)

[5.22.5 chunk --- 读取 IFF 分块数据 22](#_Toc165209060)

[5.22.6 colorsys --- 颜色系统间的转换 22](#_Toc165209061)

[5.22.7 imghdr --- 推测图像类型 22](#_Toc165209062)

[5.22.8 sndhdr --- 推测声音文件的类型 22](#_Toc165209063)

[5.22.9 ossaudiodev --- 访问兼容OSS的音频设备 22](#_Toc165209064)

[5.23 国际化 22](#_Toc165209065)

[5.23.1 gettext --- 多语种国际化服务 22](#_Toc165209066)

[5.23.2 locale --- 国际化服务 22](#_Toc165209067)

[5.24 程序框架 22](#_Toc165209068)

[5.24.1 turtle --- 海龟绘图 22](#_Toc165209069)

[5.24.2 cmd --- 支持面向行的命令解释器 22](#_Toc165209070)

[5.24.3 shlex —— 简单的词义分析 22](#_Toc165209071)

[5.25 Tk图形用户界面(GUI) 22](#_Toc165209072)

[5.25.1 tkinter --- Tcl/Tk的Python接口 22](#_Toc165209073)

[5.25.2 tkinter.ttk --- Tk主题小部件 22](#_Toc165209074)

[5.25.3 tkinter.tix --- TK扩展包 22](#_Toc165209075)

[5.25.4 tkinter.scrolledtext --- 滚动文字控件 22](#_Toc165209076)

[5.25.5 IDLE 22](#_Toc165209077)

[5.25.6 其他图形用户界面（GUI）包 22](#_Toc165209078)

[5.26 开发工具 22](#_Toc165209079)

[5.26.1 typing —— 对类型提示的支持 22](#_Toc165209080)

[5.26.2 pydoc --- 文档生成器和在线帮助系统 22](#_Toc165209081)

[5.26.3 doctest --- 测试交互式的 Python 示例 22](#_Toc165209082)

[5.26.4 unittest --- 单元测试框架 22](#_Toc165209083)

[5.26.5 unittest.mock --- 模拟对象库 22](#_Toc165209084)

[5.26.6 unittest.mock --- 上手指南 22](#_Toc165209085)

[5.26.7 2to3 - 自动将 Python 2 代码转为 Python 3 代码 22](#_Toc165209086)

[5.26.8 test --- Python回归测试包 22](#_Toc165209087)

[5.26.9 test.support --- 针对 Python 测试套件的工具 23](#_Toc165209088)

[5.26.10 test.support.script\_helper --- 用于 Python 执行测试工具 23](#_Toc165209089)

[5.27 调试和分析 23](#_Toc165209090)

[5.27.1 审计事件表 23](#_Toc165209091)

[5.27.2 bdb --- 调试器框架 23](#_Toc165209092)

[5.27.3 faulthandler —— 转储 Python 的跟踪信息 23](#_Toc165209093)

[5.27.4 pdb --- Python 的调试器 23](#_Toc165209094)

[5.27.5 Python Profilers 分析器 23](#_Toc165209095)

[5.27.6 timeit --- 测量小代码片段的执行时间 23](#_Toc165209096)

[5.27.7 trace --- 跟踪Python语句的执行 23](#_Toc165209097)

[5.27.8 tracemalloc --- 跟踪内存分配 23](#_Toc165209098)

[5.28 软件打包和分发 23](#_Toc165209099)

[5.28.1 distutils --- 构建和安装 Python 模块 23](#_Toc165209100)

[5.28.2 ensurepip --- 引导 pip 安装器 23](#_Toc165209101)

[5.28.3 venv --- 创建虚拟环境 23](#_Toc165209102)

[5.28.4 zipapp —— 管理可执行的 Python zip 打包文件 23](#_Toc165209103)

[5.29 Python运行时服务 23](#_Toc165209104)

[5.29.1 sys --- 系统相关的形参和函数 23](#_Toc165209105)

[5.29.2 sysconfig —— 提供对 Python 配置信息的访问支持 23](#_Toc165209106)

[5.29.3 builtins --- 内建对象 23](#_Toc165209107)

[5.29.4 \_\_main\_\_ --- 顶层脚本环境 23](#_Toc165209108)

[5.29.5 warnings —— 警告信息的控制 23](#_Toc165209109)

[5.29.6 dataclasses --- 数据类 23](#_Toc165209110)

[5.29.7 contextlib --- 为 with语句上下文提供的工具 23](#_Toc165209111)

[5.29.8 abc --- 抽象基类 23](#_Toc165209112)

[5.29.9 atexit --- 退出处理器 23](#_Toc165209113)

[5.29.10 traceback —— 打印或读取栈回溯信息 23](#_Toc165209114)

[5.29.11 \_\_future\_\_ --- Future 语句定义 23](#_Toc165209115)

[5.29.12 gc --- 垃圾回收器接口 23](#_Toc165209116)

[5.29.13 inspect --- 检查对象 23](#_Toc165209117)

[5.29.14 site —— 站点专属的配置钩子 23](#_Toc165209118)

[5.30 自定义 Python 解释器 23](#_Toc165209119)

[5.30.1 code --- 解释器基类 23](#_Toc165209120)

[5.30.2 codeop --- 编译Python代码 23](#_Toc165209121)

[5.31 导入模块 23](#_Toc165209122)

[5.31.1 zipimport --- 从 Zip 存档中导入模块 23](#_Toc165209123)

[5.31.2 pkgutil --- 包扩展工具 23](#_Toc165209124)

[5.31.3 modulefinder --- 查找脚本使用的模块 23](#_Toc165209125)

[5.31.4 runpy ——查找并执行 Python 模块 23](#_Toc165209126)

[5.31.5 importlib --- import 的实现 23](#_Toc165209127)

[5.31.6 使用 importlib.metadata 23](#_Toc165209128)

[5.32 Python 语言服务 23](#_Toc165209129)

[5.32.1 parser --- 访问 Python 解析树 23](#_Toc165209130)

[5.32.2 ast --- 抽象语法树 23](#_Toc165209131)

[5.32.3 symtable ——访问编译器的符号表 23](#_Toc165209132)

[5.32.4 symbol --- 与 Python 解析树一起使用的常量 24](#_Toc165209133)

[5.32.5 token --- 与Python解析树一起使用的常量 24](#_Toc165209134)

[5.32.6 keyword --- 检验Python关键字 24](#_Toc165209135)

[5.32.7 tokenize --- 对 Python 代码使用的标记解析器 24](#_Toc165209136)

[5.32.8 tabnanny --- 模糊缩进检测 24](#_Toc165209137)

[5.32.9 pyclbr --- Python 模块浏览器支持 24](#_Toc165209138)

[5.32.10 py\_compile --- 编译 Python 源文件 24](#_Toc165209139)

[5.32.11 compileall --- 字节编译 Python 库 24](#_Toc165209140)

[5.32.12 dis --- Python 字节码反汇编器 24](#_Toc165209141)

[5.32.13 pickletools --- pickle 开发者工具集 24](#_Toc165209142)

[5.33 杂项服务 24](#_Toc165209143)

[5.33.1 formatter --- 通用格式化输出 24](#_Toc165209144)

[5.34 Windows系统相关模块 24](#_Toc165209145)

[5.34.1 msilib --- 读写 Microsoft Installer 文件 24](#_Toc165209146)

[5.34.2 msvcrt --- 来自 MS VC++ 运行时的有用例程 24](#_Toc165209147)

[5.34.3 winreg --- Windows 注册表访问 24](#_Toc165209148)

[5.34.4 winsound —— Windows 系统的声音播放接口 24](#_Toc165209149)

[5.35 Unix 专有服务 24](#_Toc165209150)

[5.35.1 posix --- 最常见的 POSIX 系统调用 24](#_Toc165209151)

[5.35.2 pwd --- 用户密码数据库 24](#_Toc165209152)

[5.35.3 spwd —— shadow 密码库 24](#_Toc165209153)

[5.35.4 grp --- 组数据库 24](#_Toc165209154)

[5.35.5 crypt —— 检查 Unix 口令的函数 24](#_Toc165209155)

[5.35.6 termios --- POSIX 风格的 tty 控制 24](#_Toc165209156)

[5.35.7 tty --- 终端控制功能 24](#_Toc165209157)

[5.35.8 pty --- 伪终端工具 24](#_Toc165209158)

[5.35.9 fcntl —— 系统调用 fcntl 和 ioctl 24](#_Toc165209159)

[5.35.10 pipes --- 终端管道接口 24](#_Toc165209160)

[5.35.11 resource --- 资源使用信息 24](#_Toc165209161)

[5.35.12 nis --- Sun 的 NIS (黄页) 接口 24](#_Toc165209162)

[5.35.13 Unix syslog 库例程 24](#_Toc165209163)

[5.36 被取代的模块 24](#_Toc165209164)

[5.36.1 optparse --- 命令行选项的解析器 24](#_Toc165209165)

[5.36.2 imp —— 由代码内部访问 import 。 24](#_Toc165209166)

[5.37 未创建文档的模块 24](#_Toc165209167)

[5.37.1 平台特定模块 24](#_Toc165209168)

# 介绍

## 文档目的

本文档用于记录学习、提高python编码技术的过程，包括设置学习计划、记录学习过程中的关键内容，以及一些心得和感悟，用于指导后续工作中的python编码，提供解决问题的效率。

学习范围，python官网的python3.8版本的中文教程着手学习，不回避任何冷门知识点，力求全面和系统，为后续的python开发打下坚实的基础。

学习目标：理解、掌握、运用、尝试弄清楚Why的问题。

## 术语和缩写

### 术语

1. PEPs：Python Enhancement Proposals，Python增强建议，被熟知为PEPs
2. L1C：Layer 1 Control，层1控制
3. RRC：Radio Resource Control，无线资源控制

## 参考文档

1. [Python语言参考 — Python 3.8.18 文档](https://docs.python.org/zh-cn/3.8/reference/index.html)
2. 《L1C与DSP接口设计.docx》
3. 《L1C与协议栈接口设计.docx》

# 编码规范

## PEP8

### 概述

参考链接：[PEP 0 – Index of Python Enhancement Proposals (PEPs) | peps.python.org](https://peps.python.org/)

PEP8：Python Enhancement Proposal 8

PEPs是一系列的Python增强提案，PEP的号码由PEP的编辑分配后将不再变化。

表格 1中列出了常用的PEP，其中PEP8是关于Python编码的风格指导。

表格 1 关于PEPs和处理的PEP列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **PEP** | **Title** | **Authors** |  |
| PA | [1](https://peps.python.org/pep-0001/) | [PEP Purpose and Guidelines](https://peps.python.org/pep-0001/) | Barry Warsaw, Jeremy Hylton, David Goodger, Alyssa Coghlan |  |
| PA | [2](https://peps.python.org/pep-0002/) | [Procedure for Adding New Modules](https://peps.python.org/pep-0002/) | Brett Cannon, Martijn Faassen |  |
| PA | [4](https://peps.python.org/pep-0004/) | [Deprecation of Standard Modules](https://peps.python.org/pep-0004/) | Brett Cannon, Martin von Löwis |  |
| PA | [7](https://peps.python.org/pep-0007/) | [Style Guide for C Code](https://peps.python.org/pep-0007/) | Guido van Rossum, Barry Warsaw |  |
| PA | [8](https://peps.python.org/pep-0008/) | [Style Guide for Python Code](https://peps.python.org/pep-0008/) | Guido van Rossum, Barry Warsaw, Alyssa Coghlan |  |
| PA | [10](https://peps.python.org/pep-0010/) | [Voting Guidelines](https://peps.python.org/pep-0010/) | Barry Warsaw |  |
| PA | [11](https://peps.python.org/pep-0011/) | [CPython platform support](https://peps.python.org/pep-0011/) | Martin von Löwis, Brett Cannon |  |
| PA | [12](https://peps.python.org/pep-0012/) | [Sample reStructuredText PEP Template](https://peps.python.org/pep-0012/) | David Goodger, Barry Warsaw, Brett Cannon |  |
| PA | [13](https://peps.python.org/pep-0013/) | [Python Language Governance](https://peps.python.org/pep-0013/) | The Python core team and community |  |
| PA | [387](https://peps.python.org/pep-0387/) | [Backwards Compatibility Policy](https://peps.python.org/pep-0387/) | Benjamin Peterson |  |
| PA | [581](https://peps.python.org/pep-0581/) | [Using GitHub Issues for CPython](https://peps.python.org/pep-0581/) | Mariatta |  |
| PA | [602](https://peps.python.org/pep-0602/) | [Annual Release Cycle for Python](https://peps.python.org/pep-0602/) | Łukasz Langa | 3.9 |
| PA | [609](https://peps.python.org/pep-0609/) | [Python Packaging Authority (PyPA) Governance](https://peps.python.org/pep-0609/) | Dustin Ingram, Pradyun Gedam, Sumana Harihareswara |  |
| PA | [676](https://peps.python.org/pep-0676/) | [PEP Infrastructure Process](https://peps.python.org/pep-0676/) | Adam Turner |  |
| PA | [729](https://peps.python.org/pep-0729/) | [Typing governance process](https://peps.python.org/pep-0729/) | Jelle Zijlstra, Shantanu Jain |  |
| PA | [731](https://peps.python.org/pep-0731/) | [C API Working Group Charter](https://peps.python.org/pep-0731/) | Guido van Rossum, Petr Viktorin, Victor Stinner, Steve Dower, Irit Katriel |  |
| PA | [732](https://peps.python.org/pep-0732/) | [The Python Documentation Editorial Board](https://peps.python.org/pep-0732/) | Joanna Jablonski |  |

PEP8 Python代码规范的参考链接：[PEP 8 – Style Guide for Python Code | peps.python.org](https://peps.python.org/pep-0008/)

中文参考链接：[python代码风格指南(PEP8中文版) (china-testing.github.io)](https://china-testing.github.io/python_pep8.html)

PEP8中给出了Python主发行版本中包含标准库的Python代码规范。

### Pylint工具

Autopep8，yapf

## 一致性考虑

1. Guido的关键点之一是：**代码更多是用来读而不是写**。本指南旨在改善Python代码的可读性，即[PEP 20](https://www.python.org/dev/peps/pep-0020/)所说的“可读性计数”(Readability counts)。
2. 风格指南强调**一致性**。项目、模块或函数保持一致都很重要。
3. 最重要的是**知道何时不一致**, 有时风格指南并不适用。当有疑惑时运用你的最佳判断，参考其他例子并多问！
4. 特别注意：不要因为遵守本PEP而破坏向后兼容性！

部分可以违背指南情况：

1. 遵循指南会降低可读性。
2. 与周围其他代码不一致。
3. 代码在引入指南完成，暂时没有理由修改。
4. 旧版本兼容。

## 代码布局

### 缩进

1. 每级缩进用4个空格；
2. 括号中使用**垂直隐式缩进**或使用悬挂缩进。后者应该注意第一行要没有参数，后续行要有缩进；
3. 4个空格对续行是可选的；

代码实例

|  |
| --- |
| *# Correct:*  *# Aligned with opening delimiter.*  foo **=** long\_function\_name**(**var\_one**,** var\_two**,**  var\_three**,** var\_four**) // 垂直对齐**  *# Add 4 spaces (an extra level of indentation) to distinguish arguments from the rest.*  **def** long\_function\_name**(**  var\_one**,** var\_two**,** var\_three**,**  var\_four**): // 使用悬挂缩进，则参数与函数名存在4个空格的缩进**  print**(**var\_one**)**  *# Hanging indents should add a level.*  foo **=** long\_function\_name**(**  var\_one**,** var\_two**,**  var\_three**,** var\_four**)** |

不规范的缩进

|  |
| --- |
| *# Wrong:*  *# Arguments on first line forbidden when not using vertical alignment.*  foo **=** long\_function\_name**(**var\_one**,** var\_two**,**  var\_three**,** var\_four**) // 不使用垂直对齐时，第一行不应由参数**  *# Further indentation required as indentation is not distinguishable.*  **def** long\_function\_name**(**  var\_one**,** var\_two**,** var\_three**,**  var\_four**): // 参数的缩进和后续的函数内容无法区分**  print**(**var\_one**)** |

1. if语句跨行时，两个字符关键字(比如if)加上一个空格，再加上左括号构成了很好的缩进。后续行暂时没有规定，至少有如下三种格式，建议使用第3种。

|  |
| --- |
| *# No extra indentation.*  **if** **(**this\_is\_one\_thing **and**  that\_is\_another\_thing**):**  do\_something**()**  *# Add a comment, which will provide some distinction in editors*  *# supporting syntax highlighting.*  **if** **(**this\_is\_one\_thing **and**  that\_is\_another\_thing**):**  *# Since both conditions are true, we can frobnicate.*  do\_something**()**  *# Add some extra indentation on the conditional continuation line. 更加清晰*  **if** **(**this\_is\_one\_thing  **and** that\_is\_another\_thing**):**  do\_something**()** |

1. 右边括号也可以另起一行。有两种格式，建议第2种；

|  |
| --- |
| 格式1：右括号与第一个非空白元素对齐,示例中]与1,4对齐  my\_list **=** **[**  **1,** **2,** **3,**  **4,** **5,** **6,**  **]**  result **=** some\_function\_that\_takes\_arguments**(**  'a'**,** 'b'**,** 'c'**,**  'd'**,** 'e'**,** 'f'**,**  **)**  格式2：右括号与多行结构的第一个字符对齐，如]与m对齐  my\_list **=** **[**  **1,** **2,** **3,**  **4,** **5,** **6,**  **]**  result **=** some\_function\_that\_takes\_arguments**(**  'a'**,** 'b'**,** 'c'**,**  'd'**,** 'e'**,** 'f'**,**  **)** |

### Tabs或空格

1. 空格是首选的缩进方法；
2. Tab仅仅在已经使用tab缩进的代码中为了保持一致性而使用；
3. Python 3中不允许混合使用Tab和空格缩进。

### 最大行宽

1. 限制所有行的最大行宽为**79字符**；
2. 文本长块，比如**文档字符串或注释**，行长度应限制为72个字符；
3. 多数工具默认的续行功能会破坏代码结构，使它更难理解，不推荐使用。但是超过80个字符加以提醒是必要的。一些工具可能根本不具备动态换行功能。
4. 一些团队强烈希望更长的行宽。如果能达成一致，可以从从80提高到100个字符(最多99个字符)增加了标称线的长度，不过依旧建议文档字符串和注释保持在72的长度。
5. Python标准库比较保守，限制行宽79个字符(文档字符串/注释72）。
6. 续行的首选方法是使用小括号、中括号和大括号，反斜线仍可能在适当的时候。其次是反斜杠。比如with语句中：

|  |
| --- |
| **with** open**(**'/path/to/some/file/you/want/to/read'**)** **as** file\_1**,** \  open**(**'/path/to/some/file/being/written'**,** 'w'**)** **as** file\_2**:**  file\_2**.**write**(**file\_1**.**read**())** |

### 换行在二元运算符前还是后（推荐在换行符之后）

几十年来，二元运算符后添加分隔符一直是推荐的风格，但是这样做会有两个影响：

1. 运算符分散在不同的列中；
2. 每个运算符会移动到上一行

总结：不利于判断哪些是加项，哪些是减项。

|  |
| --- |
| *# Wrong:*  *# operators sit far away from their operands*  income **=** **(**gross\_wages **+**  taxable\_interest **+**  **(**dividends **-** qualified\_dividends**)** **-**  ira\_deduction **-**  student\_loan\_interest**)** |

为了解决可读性的问题，将二元运算符放到换行符之后。

|  |
| --- |
| *# Correct:*  *# easy to match operators with operands*  income **=** **(**gross\_wages  **+** taxable\_interest  **+** **(**dividends **-** qualified\_dividends**)**  **-** ira\_deduction  **-** student\_loan\_interest**)** |

### 空行

1. **两行空行分割顶层函数和类的定义**；
2. 类的方法定义用**单个空行**分割；
3. 额外的空行可以必要的时候用于分割**不同的函数组**，但是要尽量节约使用；
4. 额外的空行可以必要的时候在函数中用于**分割不同的逻辑块**，但是要尽量节约使用；
5. Python接 contol-L作为空白符；许多工具视它为分页符，这些要因编辑器而异。

### 源文件编码

1. 在核心Python发布的代码应该总是使用**UTF-8**(ASCII在Python 2)；
2. ASCII文件(Python 2)或UTF-8(Python 3)**不应有编码声明**；
3. 标准库中非默认的编码应**仅用于测试或当注释或文档字符串**，比如包含非ASCII字符的作者姓名，**尽量使用\x , \u , \U , or \N**；
4. Python 3.0及以后版本，PEP 3131可供参考，部分内容如下：在Python标准库**必须使用ASCII标识符**，并尽量只使用英文字母。此外字符串和注释也必须用ASCII。唯一的例外是：（a）测试非ASCII的功能，和（b）作者的名字不是拉丁字母；

Note：不需要声明编码格式，默认UTF-8，只能出现ASCII字符，不能出现中文。

### 导入（Imports）

1. 多个导入通常在不同的行；
2. 单行中可以从一个模块中导入多个模块；
3. 导入始终在文件的顶部，在模块注释和文档字符串之后，在模块全局变量和常亮之前；
4. 导入的顺序：

# 解释器

## 课前甜点

Python能够满足的需求：

1. 完成部分自动化任务，如文本处理、照片重命名、小型数据库、图形界面、小游戏等；
2. Python语言的特点：易用、提供很多数据结构、支持大型程序、高级的内置类型（数组、字典）；
3. 提供了可重复使用的模块，丰富的内置模块，如文件输入输出、系统调用、套接字、图形界面接口Tk；
4. Python是解释型语言，在开发阶段节省大量的时间（不需要编译和链接）；
5. Python程序的书写是紧凑和易读的，使用缩减标识模块的划分，不需要预先定义变量或参数；
6. Python是可扩展的，通过C语言给解释器添加新的内置函数或模块（在Python中扩展或控制C语音编写的应用）；
7. Python（蟒蛇）得名与BBC节目“Monty Python的飞行马戏团”（英国电视喜剧片）而与爬行动物无关。

## Python解释器

### 调用解释器

1. Python解释器安装后，为了更方便的调用，应将其路径设置到环境变量中；
2. 调用Python解释器的两种方式：
   1. 在Windows的控制台窗口中执行命令：py（等同于python）
   2. Python –c command [arg] …，其中-c后面的内容为python的指令，示例：python -c print("Hello World ")，执行后将显示Hello World
3. 执行module：python –m module [arg] …

### 传入参数

解释器读取命令行参数，转化为字符串列表存入sys模块中的argv变量中，使用步骤：

1. 导入sys模块：import sys；
2. sys.argv[0]是第一个参数，若无参数列表，则sys.argv[0]是一个空字符串；

### 交互模式

在终端中输入并执行命令时，解释器运行在交互模式下，在这个模式下，窗口的主提示符为>>>，连续多个换行符为…

### 源文件的字符编码

默认情况下，Python源文件以UTF-8编码方式处理，若需要特殊指定，则使用如下方式在第一行指定。encoding为需要指定的编码类型。

*# -\*- coding: encoding -\*-*

## Python速览

# 语法

参考链接：[Python语言参考 — Python 3.8.18 文档](https://docs.python.org/zh-cn/3.8/reference/index.html)

## 词法分析

1. Python程序由解析器读取，输入到解析器的时一个由词法分析器所生成的形符流；
2. Python会将读取的程序文本转为Unicode码点，默认为UTF-8。

### 行结构

#### 逻辑行

逻辑行结束以NEWLINE形式表示，语句不能跨越逻辑行的边界，除非其语法允许包含 NEWLINE (例如复合语句可由多行子语句组成)。一个逻辑行可由一个或多个物理行按照明确或隐含的行拼接规则构成。

Note：一个逻辑行时一个语句组成的行，一个python语句可以跨越多行，多行之间通过显示的续行或隐含的续行，视为一个逻辑行。

#### 物理行

物理行是以一个**行终止序列结束**的字符序列。 Unix 所用的 ASCII 字符 LF (换行), Windows 所用的 ASCII 字符序列 **CR LF (回车加换行)**

#### 注释

1. 一条注释以**不包含在字符串**字面值内的井号 **(#)** 开头，并在**物理行的末尾结束**；
2. 一条注释标志着**逻辑行的结束，除非存在隐含的行拼接规则；**
3. 注释在语法分析中会被忽略。

#### 编码声明

1. 指定源码文件编码声明位于 Python 脚本的第一或第二行；
2. 匹配正则表达式 coding[=:]\s\*([-\w.]+)，这条注释会被作为**编码声明**来处理；
3. 编码声明必须独占一行，**如果它是在第二行，则第一行也必须是注释**；
4. 若没有编码声明，则默认编码为UTF-8；
5. 推荐的编码声明形式如下

*# -\*- coding: <encoding-name> -\*-*

#### 显式的行拼接

#### 隐式的行拼接

#### 空白行

#### 缩进

#### 形符之间的空白

### 其他形符

### 标识符和关键字

#### 关键字

#### 保留的标识符类

### 字面值

#### 字符串和字符串字面值

#### 字符串字面值拼接

#### 格式化字符串字面值

#### 数字字面值

#### 整型数字字面值

#### 浮点数字字面值

#### 虚数字面值

### 运算符

### 分隔符

## 数据模型

### 对象、值与类型

### 标准类型层级结构

### 特殊方法名称

### 协程

## 执行模型

### 程序的结构

### 命名与绑定

### 异常

## 导入系统

### importlib

### 包

### 搜索

### 加载

### 基于路径的查找器

### 替换标准导入系统

### 包相对导入

### 有关 \_\_main\_\_ 的特殊事项

### 开放问题项

### 参考文献

## 表达式

### 算术转换

### 原子

### 原型

### await 表达式

### 幂运算符

### 一元算术和位运算

### 二元算术运算符

### 移位运算

### 二元位运算

### 比较运算

### 布尔运算

### 赋值表达式

### 条件表达式

### lambda 表达式

### 表达式列表

### 求值顺序

### 运算符优先级

## 简单语句

### 表达式语句

### 赋值语句

### assert 语句

### pass 语句

### del 语句

### return 语句

### yield 语句

### raise 语句

### break 语句

### continue 语句

### import 语句

### global 语句

### nonlocal 语句

## 复合语句

### if 语句

### while 语句

### for 语句

### try 语句

### with 语句

### 函数定义

### 类定义

### 协程

## 最高层级组件

### 完整的 Python 程序

### 文件输入

### 交互式输入

### 表达式输入

## 完整的语法规范

# 标准库

参考链接：[Python 标准库 — Python 3.8.18 文档](https://docs.python.org/zh-cn/3.8/library/index.html)

1. Python标准库与Python一同发行，标准库非常庞大，所涉及的组件十分广泛；
2. 标准库内置多个模块，均已C编写，用以实现系统级功能。

## 概述

### 可用性注释

## 内置函数

## 内置常量

### 由 site 模块添加的常量

## 内置类型

### 逻辑值检测

### 布尔运算 --- and, or, not

### 比较运算

### 数字类型 --- int, float, complex

### 迭代器类型

### 序列类型 --- list, tuple, range

### 文本序列类型 --- str

### 二进制序列类型 --- bytes, bytearray, memoryview

### 集合类型 --- set, frozenset

### 映射类型 --- dict

### 上下文管理器类型

### 其他内置类型

### 特殊属性

### 整数字符串转换长度限制

## 内置异常

### 基类

### 具体异常

### 警告

### 异常层次结构

## 文本处理服务

### string --- 常见的字符串操作

### re --- 正则表达式操作

### difflib --- 计算差异的辅助工具

### textwrap --- 文本自动换行与填充

### unicodedata --- Unicode 数据库

### stringprep --- 因特网字符串预备

### readline --- GNU readline 接口

### rlcompleter --- GNU readline 的补全函数

## 二进制数据服务

### struct --- 将字节串解读为打包的二进制数据

### codecs --- 编解码器注册和相关基类

## 数据类型

### datetime --- 基本日期和时间类型

### calendar --- 日历相关函数

### collections --- 容器数据类型

### collections.abc --- 容器的抽象基类

### heapq --- 堆队列算法

### bisect --- 数组二分查找算法

### array --- 高效的数字数组

### weakref --- 弱引用

### types --- 动态类型创建和内置类型名称

### copy --- 浅层 (shallow) 和深层 (deep) 复制操作

### pprint --- 数据美化输出

### reprlib --- 另一种 repr() 实现

### enum --- 对枚举的支持

## 数字和数学模块

### numbers --- 数字的抽象基类

### math --- 数学函数

### cmath --- 关于复数的数学函数

### decimal --- 十进制定点和浮点运算

### fractions --- 分数

### random --- 生成伪随机数

### statistics --- 数学统计函数

## 函数式编程模块

### itertools --- 为高效循环而创建迭代器的函数

### functools --- 高阶函数和可调用对象上的操作

### operator --- 标准运算符替代函数

## 文件和目录访问

### pathlib --- 面向对象的文件系统路径

### os.path --- 常用路径操作

### fileinput --- 迭代来自多个输入流的行

### stat --- 解析 stat() 结果

### filecmp --- 文件及目录的比较

### tempfile --- 生成临时文件和目录

### glob --- Unix 风格路径名模式扩展

### fnmatch --- Unix 文件名模式匹配

### linecache --- 随机读写文本行

### shutil --- 高阶文件操作

## 数据持久化

### pickle --- Python 对象序列化

### copyreg --- 注册配合 pickle 模块使用的函数

### shelve --- Python 对象持久化

### marshal --- 内部 Python 对象序列化

### dbm --- Unix "数据库" 接口

### sqlite3 --- SQLite 数据库 DB-API 2.0 接口模块

## 数据压缩和存档

### zlib --- 与 gzip 兼容的压缩

### gzip --- 对 gzip 格式的支持

### bz2 --- 对 bzip2 压缩算法的支持

### lzma --- 用 LZMA 算法压缩

### zipfile --- 使用ZIP存档

### tarfile --- 读写tar归档文件

## 文件格式

### csv --- CSV 文件读写

### configparser --- 配置文件解析器

### netrc --- netrc 文件处理

### xdrlib --- 编码与解码 XDR 数据

### plistlib --- 生成与解析 Mac OS X .plist 文件

## 加密服务

### hashlib --- 安全哈希与消息摘要

### hmac --- 基于密钥的消息验证

### secrets --- 生成安全随机数字用于管理密码

## 通用操作系统服务

### os --- 多种操作系统接口

### io --- 处理流的核心工具

### time --- 时间的访问和转换

### argparse --- 命令行选项、参数和子命令解析器

### getopt --- C 风格的命令行选项解析器

### logging --- Python 的日志记录工具

### logging.config --- 日志记录配置

### logging.handlers --- 日志处理程序

### getpass --- 便携式密码输入工具

### curses --- 终端字符单元显示的处理

### curses.textpad --- 用于 curses 程序的文本输入控件

### curses.ascii --- 用于 ASCII 字符的工具

### curses.panel --- curses 的 panel 栈扩展

### platform --- 获取底层平台的标识数据

### errno --- 标准 errno 系统符号

### ctypes --- Python 的外部函数库

## 并发执行

### threading --- 基于线程的并行

### multiprocessing --- 基于进程的并行

### multiprocessing.shared\_memory --- 可从进程直接访问的共享内存

### concurrent 包

### concurrent.futures --- 启动并行任务

### subprocess --- 子进程管理

### sched --- 事件调度器

### queue --- 一个同步的队列类

### contextvars 上下文变量

### \_thread --- 底层多线程 API

### \_dummy\_thread --- \_thread 的替代模块

### dummy\_threading --- 可直接替代 threading 模块。

## 网络和进程间通信

### asyncio --- 异步 I/O

### socket --- 底层网络接口

### ssl --- 套接字对象的TLS/SSL封装

### select --- 等待 I/O 完成

### selectors --- 高级 I/O 复用库

### asyncore --- 异步socket处理器

### asynchat --- 异步 socket 指令/响应 处理器

### signal --- 设置异步事件处理程序

### mmap --- 内存映射文件支持

## 互联网数据处理

### email --- 电子邮件与 MIME 处理包

### json --- JSON 编码和解码器

### mailcap --- Mailcap 文件处理

### mailbox --- 操作多种格式的邮箱

### mimetypes --- 映射文件夹到 MIME 类型

### base64 --- Base16, Base32, Base64, Base85 数据编码

### binhex --- 对binhex4文件进行编码和解码

### binascii --- 二进制和 ASCII 码互转

### quopri --- 编码与解码经过 MIME 转码的可打印数据

### uu --- 对 uuencode 文件进行编码与解码

## 结构化标记处理工具

### html --- 超文本标记语言支持

### html.parser --- 简单的 HTML 和 XHTML 解析器

### html.entities --- HTML 一般实体的定义

### XML处理模块

### xml.etree.ElementTree --- ElementTree XML API

### xml.dom --- 文档对象模型 API

### xml.dom.minidom --- 最小化的 DOM 实现

### xml.dom.pulldom --- 支持构建部分 DOM 树

### xml.sax --- 支持 SAX2 解析器

### xml.sax.handler --- SAX 处理程序的基类

### xml.sax.saxutils --- SAX 工具集

### xml.sax.xmlreader --- 用于 XML 解析器的接口

### xml.parsers.expat --- 使用 Expat 的快速 XML 解析

## 互联网协议和支持

### webbrowser --- 方便的Web浏览器控制器

### cgi --- 通用网关接口支持

### cgitb --- 用于 CGI 脚本的回溯管理器

### wsgiref --- WSGI 工具和参考实现

### urllib --- URL 处理模块

### urllib.request --- 用于打开 URL 的可扩展库

### urllib.response --- urllib 使用的 Response 类

### urllib.parse 用于解析 URL

### urllib.error --- urllib.request 引发的异常类

### urllib.robotparser --- robots.txt 语法分析程序

### http --- HTTP 模块

### http.client --- HTTP 协议客户端

### ftplib --- FTP 协议客户端

### poplib --- POP3 协议客户端

### imaplib --- IMAP4 协议客户端

### nntplib --- NNTP 协议客户端

### smtplib ---SMTP协议客户端

### smtpd --- SMTP 服务器

### telnetlib -- Telnet 客户端

### uuid --- RFC 4122 定义的UUID对象

### socketserver --- 用于网络服务器的框架

### http.server --- HTTP 服务器

### http.cookies --- HTTP状态管理

### http.cookiejar —— HTTP 客户端的 Cookie 处理

### xmlrpc --- XMLRPC 服务端与客户端模块

### xmlrpc.client --- XML-RPC 客户端访问

### xmlrpc.server --- 基本 XML-RPC 服务器

### ipaddress --- IPv4/IPv6 操作库

## 多媒体服务

### audioop --- 处理原始音频数据

### aifc --- 读写 AIFF 和 AIFC 文件

### sunau --- 读写 Sun AU 文件

### wave --- 读写WAV格式文件

### chunk --- 读取 IFF 分块数据

### colorsys --- 颜色系统间的转换

### imghdr --- 推测图像类型

### sndhdr --- 推测声音文件的类型

### ossaudiodev --- 访问兼容OSS的音频设备

## 国际化

### gettext --- 多语种国际化服务

### locale --- 国际化服务

## 程序框架

### turtle --- 海龟绘图

### cmd --- 支持面向行的命令解释器

### shlex —— 简单的词义分析

## Tk图形用户界面(GUI)

### tkinter --- Tcl/Tk的Python接口

### tkinter.ttk --- Tk主题小部件

### tkinter.tix --- TK扩展包

### tkinter.scrolledtext --- 滚动文字控件

### IDLE

### 其他图形用户界面（GUI）包

## 开发工具

### typing —— 对类型提示的支持

### pydoc --- 文档生成器和在线帮助系统

### doctest --- 测试交互式的 Python 示例

### unittest --- 单元测试框架

### unittest.mock --- 模拟对象库

### unittest.mock --- 上手指南

### 2to3 - 自动将 Python 2 代码转为 Python 3 代码

### test --- Python回归测试包

### test.support --- 针对 Python 测试套件的工具

### test.support.script\_helper --- 用于 Python 执行测试工具

## 调试和分析

### 审计事件表

### bdb --- 调试器框架

### faulthandler —— 转储 Python 的跟踪信息

### pdb --- Python 的调试器

### Python Profilers 分析器

### timeit --- 测量小代码片段的执行时间

### trace --- 跟踪Python语句的执行

### tracemalloc --- 跟踪内存分配

## 软件打包和分发

### distutils --- 构建和安装 Python 模块

### ensurepip --- 引导 pip 安装器

### venv --- 创建虚拟环境

### zipapp —— 管理可执行的 Python zip 打包文件

## Python运行时服务

### sys --- 系统相关的形参和函数

### sysconfig —— 提供对 Python 配置信息的访问支持

### builtins --- 内建对象

### \_\_main\_\_ --- 顶层脚本环境

### warnings —— 警告信息的控制

### dataclasses --- 数据类

### contextlib --- 为 with语句上下文提供的工具

### abc --- 抽象基类

### atexit --- 退出处理器

### traceback —— 打印或读取栈回溯信息

### \_\_future\_\_ --- Future 语句定义

### gc --- 垃圾回收器接口

### inspect --- 检查对象

### site —— 站点专属的配置钩子

## 自定义 Python 解释器

### code --- 解释器基类

### codeop --- 编译Python代码

## 导入模块

### zipimport --- 从 Zip 存档中导入模块

### pkgutil --- 包扩展工具

### modulefinder --- 查找脚本使用的模块

### runpy ——查找并执行 Python 模块

### importlib --- import 的实现

### 使用 importlib.metadata

## Python 语言服务

### parser --- 访问 Python 解析树

### ast --- 抽象语法树

### symtable ——访问编译器的符号表

### symbol --- 与 Python 解析树一起使用的常量

### token --- 与Python解析树一起使用的常量

### keyword --- 检验Python关键字

### tokenize --- 对 Python 代码使用的标记解析器

### tabnanny --- 模糊缩进检测

### pyclbr --- Python 模块浏览器支持

### py\_compile --- 编译 Python 源文件

### compileall --- 字节编译 Python 库

### dis --- Python 字节码反汇编器

### pickletools --- pickle 开发者工具集

## 杂项服务

### formatter --- 通用格式化输出

## Windows系统相关模块

### msilib --- 读写 Microsoft Installer 文件

### msvcrt --- 来自 MS VC++ 运行时的有用例程

### winreg --- Windows 注册表访问

### winsound —— Windows 系统的声音播放接口

## Unix 专有服务

### posix --- 最常见的 POSIX 系统调用

### pwd --- 用户密码数据库

### spwd —— shadow 密码库

### grp --- 组数据库

### crypt —— 检查 Unix 口令的函数

### termios --- POSIX 风格的 tty 控制

### tty --- 终端控制功能

### pty --- 伪终端工具

### fcntl —— 系统调用 fcntl 和 ioctl

### pipes --- 终端管道接口

### resource --- 资源使用信息

### nis --- Sun 的 NIS (黄页) 接口

### Unix syslog 库例程

## 被取代的模块

### optparse --- 命令行选项的解析器

### imp —— 由代码内部访问 import 。

## 未创建文档的模块

### 平台特定模块