Python常见问题

目录

一般信息

什么是 Python?

Python 是一种解释型、交互式、面向对象的编程语言。 它包含了模块、异常、动态类型、高层级动态数据类型以及类等特性。 在面向对象编程以外它还支持多种编程范式，例如过程式和函数式编程等。 Python 结合了超强的功能和极清晰的语法。 它带有许多系统调用和库以及多种窗口系统的接口，并且能用 C 或 C++ 来进行扩展。 它还可用作需要可编程接口的应用程序的扩展语言。 最后，

Python 非常易于移植：它可以在包括 Linux 和 macOS 在内的许多 Unix 变种以及 Windows 上运行。

要了解更多详情，请先查看 Python 教程。 Python 新手指南 提供了学习 Python 的其他入门教程及资源的链接。

什么是 Python 软件基金会?

Python 软件基金会（Python Software Foundation，简称 PSF）是一个独立的非盈利组织，它拥有 Python 2.1 及以上各版本的版权。 PSF 的使命是推进与 Python 编程语言相关的开源技术，并推广 Python 的使用。 PSF 的主页是 https://www.python.org/psf/。

向 PSF 提供捐助在美国是免税的。 如果你在使用 Python 并且感觉它对你很有帮助，可以通过 PSF 捐助页 进行捐助。

使用 Python 是否存在版权限制?

你可以任意使用源码，只要你保留版权信息并在你基于 Python 的产品文档中显示该版权信息。 如果你遵守此版权规则，就可以将 Python 用于商业领域，以源码或二进制码的形式（不论是否经过修改）销售 Python 的副本，或是以某种形式包含了 Python 的产品。 当然，我们仍然希望获知所有对 Python 的商业使用。

请参阅 许可页 以查看进一步的说明以及 PSF 许可的完整文本。

Python 的徽标是注册商标，在某些情况下需要获得允许方可使用。 请参阅 商标使用政策 了解详情。

创造 Python 的最初理由是什么?

以下是有关最初缘起的一份 *非常* 简短的摘要，由 Guido van Rossum 本人撰写:

> 我在 CWI 的 ABC 部门时在实现解释型语言方面积累了丰富经验，通过与这个部门成员的协同工作，我学到了大量有关语言设计的知识。 这是许多 Python 特性的最初来源，包括使用缩进来组织语句以及包含非常高层级的数据结构（虽然在 Python 中具体的实现细节完全不同）。
>
> 我对 ABC 语言有过许多抱怨，但同时也很喜欢它的许多特性。 没有可能通过扩展 ABC 语言（或它的实现）来弥补我的不满 —— 实际上缺乏可扩展性就是它最大的问题之一。 我也有一些使用 Modula-2+ 的经验，并曾与 Modula-3 的设计者进行交流，还阅读了 Modula-3 的报告。 Modula-3 是 Python 中异常机制所用语法和语义，以及其他一些语言特性的最初来源。
>
> 我还曾在 CWI 的 Amoeba 分布式操作系统部门工作。 当时我们需要有一种比编写 C 程序或 Bash 脚本更好的方式来进行系统管理，因为 Amoeba 有它自己的系统调用接口，并且无法方便地通过 Bash 来访问。 我在 Amoeba 中处理错误的经验令我深刻地意识到异常处理在编程语言特性当中的重要地位。
>
> 我发现，某种具有 ABC 式的语法而又能访问 Amoeba 系统调用的脚本语言将可满足需求。 我意识到编写一种 Amoeba 专属的语言是愚蠢的，所以我决定编写一种具有全面可扩展性的语言。

在 1989 年的圣诞假期中，我手头的时间非常充裕，因此我决定开始尝试一下。 在接下来的一年里，虽然我仍然主要用我的业余时间来做这件事，但 Python 在 Amoeba 项目中的使用获得了很大的成功，来自同事的反馈让我得以增加了许多早期的改进。

到 1991 年 2 月，经过一年多的开发，我决定将其发布到 USENET。 之后的事情就都可以在 `Misc/HISTORY` 文件里面看了。

Python 适合做什么?

Python 是一种高层级的多用途编程语言，可用于解决许多不同门类的问题。

本语言自带一个庞大标准库，所涵盖的编程领域包括字符串处理（正则表达式、Unicode、文件间的差异比较等），互联网协议（HTTP, FTP, SMTP, XML-RPC, POP, IMAP），软件工程（单元测试、日志记录、性能分析、Python 代码解析），以及操作系统接口（系统调用、文件系统、TCP/IP 套接字）。 请查看 Python 标准库 目录页以获取所有可用内容的概览。 此外还有大量第三方扩展包可供使用。 请访问 Python 软件包索引 来查找你感兴趣的软件包。

Python 版本的编号形式是怎样的?

Python 版本的编号形式为 "A.B.C" 或 "A.B":

- *A* 是主版本号 -- 它仅会针对语言中非常重大的改变而递增。
- *B* 是次版本号 -- 它会针对不太重大的改变而递增。
- *C* 是微版本号 -- 它针对每次问题修正发布而递增。

并非所有发布版本都是问题修正版本。 在新特征发布版本的开发过程中，会制作一系列的开发版本，它们以 alpha, beta 或 release candidate 来标示。 其中 alpha 版本是早期发布版，它的接口尚未最终确定；在两个 alpha 发布版本间出现接口的改变并不意外。 而 beta 版本更为稳定，它会保留现有的接口，但也可能增加新的模块，而 release candidate 版则会保持冻结状态，不做任何改变，除非有需要修复的严重问题。

Alpha, beta 和候选发布版带有额外的后缀:

- 带有某个小数字 *N* 的 alpha 版后缀是 "aN"。
- 带有某个小数字 *N* 的 beta 版后缀是 "bN"。
- 带有某个小数字 *N* 的候选发布版后缀是 "rcN"。

换句话说，所有标记为 *2.0aN* 的版本都早于标记为 *2.0bN* 的版本，后者又都早于标记为 *2.0rcN* 的版本，而 *后者* 又都早于标记为 2.0 的版本。

你还可能看到带有“+”后缀的版本号，例如“2.2+”。 这表示未发布版本，直接基于 CPython 开发代码仓库构建。 在实际操作中，当一个小版本最终发布后，未发布版本号会递增到下一个小版本号，成为“a0”版本，例如“2.4a0”。

请参阅 Developer's Guide 获取更多有关开发流程的信息，并参阅 **PEP 387** 了解更多有关 Python 的向下兼容策略的信息。 另请参阅有关 `sys.version`, `sys.hexversion` 和 `sys.version_info` 的文档。

我应如何获取一份 Python 源代码的副本?

最新的 Python 发布版源代码总能从 python.org 获取，下载页链接为 https://www.python.org/downloads/。 最新的开发版源代码可以在 https://github.com/python/cpython/ 获取。

发布版源代码是一个以 gzip 压缩的 tar 文件，其中包含完整的 C 源代码、Sphinx 格式的文档、Python 库模块、示例程序以及一些有用的自由分发软件。 该源代码将可在大多数 UNIX 类平台上直接编译并运行。

请参阅 Python 开发者指南的初步上手部分 了解有关获取源代码并进行编译的更多信息。

我应如何获取 Python 的文档?

当前的 Python 稳定版本的标准文档可在 https://docs.python.org/3/ 查看。 也可在 https://docs.python.org/3/download.html 获取 EPUB、纯文本以及可下载的 HTML 版本。

文档以 reStructuredText 格式撰写并使用 Sphinx 文档工具 生成。 文档的 reStructuredText 源文件是 Python 源代码发布版的一部分。

我之前从未接触过编程。 哪里有 Python 的教程?

有许多可选择的教程和书籍。 标准文档中也包含有 Python 教程。

请参阅 新手指南 以获取针对 Python 编程初学者的信息，包括教程的清单。

是否有专门针对 Python 的新闻组或邮件列表?

有一个新闻组 *comp.lang.python* 和一个邮件列表 python-list。 新闻组和邮件列表是彼此互通的 —— 如果你可以阅读新闻就不必再订阅邮件列表。 *comp.lang.python* 的流量很大，每天会收到数以百计的发帖，Usenet 使用者通常更擅长处理这样大的流量。

有关新软件发布和活动的公告可以在 comp.lang.python.announce 中找到，这是个严格管理的低流量列表，每天发帖五个左右。 可在 python-announce 邮件列表 订阅。

有关其他邮件列表和新闻组的更多信息可以在 https://www.python.org/community/lists/ 找到。

我应如何获取 Python 的公开测试版本?

可以从 https://www.python.org/downloads/ 下载 alpha 和 beta 发布版。 所有发布版都会在 comp.lang.python 和 comp.lang.python.announce 新闻组以及 Python 主页 https://www.python.org/ 上进行公告；并会推送到 RSS 新闻源。

你还可以通过 Git 访问 Python 的开发版。 请参阅 Python 开发者指南 了解详情。

我应如何为 Python 提交错误报告和补丁?

要报告问题或提交补丁，请使用位于 https://github.com/python/cpython/issues 的问题追踪器。

有关 Python 开发流程的更多信息，请参阅 Python 开发者指南。

是否有任何公开发表的 Python 相关文章可以供我参考引用?

可能作为参考文献的最好方式还是引用你喜欢的 Python 相关书籍。

有关 Python 的 最早的文章 撰写于 1991 年因而现在已相当过时。

> Guido van Rossum 与 Jelke de Boer, "使用 Python 编程语言交互式地测试远程服务器", CWI 季刊, 第 4 卷, 第 4 期 (1991 年 12 月), 阿姆斯特丹, 第 283--303 页。

是否有任何 Python 相关的书籍?

是的，相关的书籍很多，还有更多即将发行。 请访问 python.org 的 wiki 页面 https://wiki.python.org/moin/PythonBooks 获取一份清单。

你也可以到各大在线书店搜索 "Python" 并过滤掉对 Monty Python 的引用；或者也可以搜索 "Python" 加 "language"。

www.python.org 具体位于世界上的哪个地点?

Python 项目的基础设施分布于世界各地并由 Python 基础设施团队负责管理。 相关细节请访问 这里。

为何命名为 Python?

在着手编写 Python 实现的时候，Guido van Rossum 同时还阅读了刚出版的 "Monty Python 的飞行马戏团" 剧本，这是一部自 1970 年代开始播出的 BBC 系列喜剧。 Van Rossum 觉得他需要选择一个简短、独特而又略显神秘的名字，于是他决定将这个新语言命名为 Python。

我必须喜欢 "Monty Python 的飞行马戏团" 吗?

不必，但这对学习会有帮助。 :)

现实世界中的 Python

Python 有多稳定?

非常稳定。 自 1991 年起大约每隔 6 至 18 个月就会推出新的稳定发布版，这种状态看来还会持续下去。 从 3.9 版开始，Python 将会每隔 12 个月推出一个新增特征版本 (**PEP 602**)。

开发者也会推出较旧版本的问题修正发布版，因此现有发布版的稳定性还会逐步提升。 问题修正发布版会以版本号第三部分的数字来标示（例如 3.5.3, 3.6.2），用于稳定性管理；只有对已知问题的修正会包含在问题修正发布版中，而同一系列的问题修正发布版中的接口将会始终保持一致。

最新的稳定发布版总是可以在 Python 下载页 中找到。 Python 3.x 是推荐的版本并被大多数广泛使用的库所支持。 Python 2.x **已不再维护**。

有多少人在使用 Python?

使用者应该数以百万计，但很难获得一个精确的数字。

Python 可以免费下载，因此并不存在销量数据，此外它也可以从许多不同网站获取，并且包含于许多 Linux 发行版之中，因此下载量统计同样无法完全说明问题。

comp.lang.python 新闻组非常活跃，但不是所有 Python 用户都会在新闻组发帖，许多人甚至不会阅读新闻组。

有哪些重要的项目是用 Python 开发的?

请访问 https://www.python.org/about/success 查看使用了 Python 的项目列表。 阅览 历次 Python 会议 的日程纪要可以看到许多不同公司和组织所做的贡献。

高水准的 Python 项目包括 Mailman 邮件列表管理器 和 Zope 应用服务器。 多个 Linux 发行版，其中最著名的是 Red Hat，都使用 Python 来编写其部分或全部的安装器和系统管理软件。 在内部使用 Python 的公司包括了 Google, Yahoo 和 Lucasfilm 等等。

在未来可以期待 Python 将有什么新进展?

请访问 https://peps.python.org/ 查看 Python 增强提议（PEP）。 PEP 是为 Python 加入某种新特性的提议进行描述的设计文档，其中会提供简明的技术规格说明与基本原理。 可查找标题为 "Python X.Y Release Schedule" 的 PEP，其中 X.Y 是某个尚未公开发布的版本。

新版本的开发会在 python-dev 邮件列表 中进行讨论。

提议对 Python 加入不兼容的更改是否合理?

通常来说是不合理的。 世界上已存在的 Python 代码数以亿计，因此，任何对该语言的更改即便仅会使得现有程序中极少的一部分失效也是难以令人接受的。 就算你可以提供一个转换程序，也仍然存在需要更新全部文档的问题；另外还有大量已出版的 Python 书籍，我们不希望让它们在一瞬间全部变成废纸。

如果必须更改某个特性，则应该提供渐进式的升级路径。 **PEP 5** 描述了引入向后不兼容的更改所需遵循的流程，以尽可能减少对用户的干扰。

Python 是一种对编程初学者友好的语言吗?

是的。

从过程式、静态类型的编程语言例如 Pascal, C 或者 C++ 以及 Java 的某一子集开始引导学生入门仍然是常见的做法。 但以 Python 作为第一种编程语言进行学习对学生可能更有利。Python 具有非常简单和一致的语法和庞大的标准库，而且最重要的是，在编程入门教学中使用 Python 可以让学生专注于更重要的编程技能，例如问题分解与数据类型设计。 使用 Python，可以快速向学生介绍基本概念例如循环与过程等。 他们甚至有可能在第一次课里就开始接触用户自定义对象。

对于之前从未接触过编程的学生来说，使用静态类型语言会感觉不够自然。 这会给学生带来必须掌握的额外复杂性，并减慢教学的进度。 学生需要尝试像计算机一样思考，分解问题，设计一致的接

口并封装数据。 虽然从长远来看，学习和使用一种静态类型语言是很重要的，但这并不是最适宜在学生的第一次编程课上就进行探讨的主题。

还有许多其他方面的特点使得 Python 成为很好的入门语言。 像 Java 一样，Python 拥有一个庞大的标准库，因此可以在课程非常早期的阶段就给学生布置一些 *实用* 的编程项目。 编程作业不必仅限于标准四则运算和账目检查程序。 通过使用标准库，学生可以在学习编程基础知识的同时开发真正的应用，从而获得更大的满足感。 使用标准库还能使学生了解代码重用的概念。 而像 PyGame 这样的第三方模块同样有助于扩大学生的接触领域。

Python 的解释器使学生能够在编程时测试语言特性。 他们可以在一个窗口中输入程序源代码的同时开启一个解释器运行窗口。 如果他们不记得列表有哪些方法，他们这以这样做:

```
>>> L = []
>>> dir(L)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__',
'__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__',
'__sizeof__', '__str__', '__subclasshook__', 'append', 'clear',
'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove',
'reverse', 'sort']
>>> [d for d in dir(L) if '__' not in d]
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove',

>>> help(L.append)
Help on built-in function append:

append(...)
    L.append(object) -> None -- append object to end

>>> L.append(1)
>>> L
[1]
```

通过使用解释器，学生编写程序时参考文档总是能伴随在他们身边。

Python 还拥有一些很好的 IDE。 IDLE 是一个以 Python 基于 Tkinter 编写的跨平台 Python IDE。 Emacs 用户将高兴地了解到 Emacs 具有非常好的 Python 模式。 所有这些编程环境都提供语法高亮、自动缩进以及在编写代码时使用交互式解释器等功能。 请访问 Python wiki 查看 Python 编程环境的完整列表。

如果你想要讨论 Python 在教育中的使用，你可能会有兴趣加入 edu-sig 邮件列表。