**Python**

|  |  |
| --- | --- |
| **Python** | |
| **[Official Python Logo](http://zh.wikipedia.org/wiki/File:Python_logo_and_wordmark.svg)** | |
| [**编程范型**](http://zh.wikipedia.org/wiki/%E7%BC%96%E7%A8%8B%E8%8C%83%E5%9E%8B) | [多泛型](http://zh.wikipedia.org/wiki/%E5%A4%9A%E6%B3%9B%E5%9E%8B)、[面向对象](http://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1)、[函数式编程](http://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B8%E5%BC%8F%E7%B7%A8%E7%A8%8B)、[面向切面编程](http://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%BC%96%E7%A8%8B) |
| **发行时间** | 1991年 |
| **设计者** | [吉多·范罗苏姆](http://zh.wikipedia.org/wiki/%E5%90%89%E5%A4%9A%C2%B7%E8%8C%83%E7%BD%97%E8%8B%8F%E5%A7%86)（Guido van Rossum） |
| **实作者** | [Python Software Foundation](http://en.wikipedia.org/wiki/Python_Software_Foundation) |
| **最新发行时间** | 3.3.5/ 2014年3月9日；6天前 (2014-03-09) 3.2.5/ 2013年5月15日；9个月前 (2013-05-15) 2.7.6/ 2013年11月10日；4个月前 (2013-11-10) |
| [**型态系统**](http://zh.wikipedia.org/wiki/%E9%A1%9E%E5%9E%8B%E7%B3%BB%E7%B5%B1) | [强类型](http://zh.wikipedia.org/wiki/%E5%BC%B7%E9%A1%9E%E5%9E%8B)、[动态类型](http://zh.wikipedia.org/wiki/%E5%8A%A8%E6%80%81%E7%B1%BB%E5%9E%8B)、["duck"](http://zh.wikipedia.org/wiki/%E9%B8%AD%E5%AD%90%E7%B1%BB%E5%9E%8B) |
| **主要实作产品** | [**CPython**](http://zh.wikipedia.org/wiki/CPython)、[IronPython](http://zh.wikipedia.org/wiki/IronPython)、[Jython](http://zh.wikipedia.org/wiki/Jython)、[Python for S60](http://zh.wikipedia.org/w/index.php?title=Python_for_S60&action=edit&redlink=1)、[PyPy](http://zh.wikipedia.org/wiki/PyPy)、[Unladen Swallow](http://zh.wikipedia.org/w/index.php?title=Unladen_Swallow&action=edit&redlink=1) |
| **衍生副语言** | Stackless Python |
| **启发语言** | [ABC](http://zh.wikipedia.org/wiki/ABC_(%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80))、[C语言](http://zh.wikipedia.org/wiki/C%E8%AA%9E%E8%A8%80)、[Haskell](http://zh.wikipedia.org/wiki/Haskell)、[Icon](http://zh.wikipedia.org/w/index.php?title=Icon%E8%AA%9E%E8%A8%80&action=edit&redlink=1)、[Java](http://zh.wikipedia.org/wiki/Java)、[Lisp](http://zh.wikipedia.org/wiki/LISP)、[Modula-3](http://zh.wikipedia.org/w/index.php?title=Modula-3&action=edit&redlink=1)、[Perl](http://zh.wikipedia.org/wiki/Perl)、[Smalltalk](http://zh.wikipedia.org/wiki/Smalltalk)、[Tcl](http://zh.wikipedia.org/wiki/Tcl) |
| **影响语言** | [Boo](http://zh.wikipedia.org/w/index.php?title=Boo&action=edit&redlink=1)、[Groovy](http://zh.wikipedia.org/wiki/Groovy)、[Ruby](http://zh.wikipedia.org/wiki/Ruby) |
| [**作业系统**](http://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1) | [跨平台](http://zh.wikipedia.org/wiki/%E8%B7%A8%E5%B9%B3%E5%8F%B0) |
| [**许可证**](http://zh.wikipedia.org/wiki/%E8%A8%B1%E5%8F%AF%E8%AD%89) | [Python Software Foundation License](http://en.wikipedia.org/wiki/Python_Software_Foundation_License) |
| [**网站**](http://zh.wikipedia.org/wiki/%E7%B6%B2%E7%AB%99) | [www.Python.org](http://www.python.org/) |

**Python**（英国发音：[/ˈpaɪθən/](http://zh.wikipedia.org/wiki/Wikipedia:%E5%9C%8B%E9%9A%9B%E9%9F%B3%E6%A8%99%E8%8B%B1%E8%AA%9E%E7%99%BC%E9%9F%B3) 美国发音：[/ˈpaɪθɑːn/](http://zh.wikipedia.org/wiki/Wikipedia:%E5%9C%8B%E9%9A%9B%E9%9F%B3%E6%A8%99%E8%8B%B1%E8%AA%9E%E7%99%BC%E9%9F%B3)），是一种[面向对象](http://zh.wikipedia.org/wiki/%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88)、[直译式](http://zh.wikipedia.org/wiki/%E7%9B%B4%E8%AD%AF%E8%AA%9E%E8%A8%80)[电脑程式语言](http://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80)，具有近二十年的发展历史，成熟且稳定。它包含了一组完善而且容易理解的标准库，能够轻松完成很多常见的任务。它的[语法](http://zh.wikipedia.org/wiki/%E8%AA%9E%E6%B3%95)简捷和清晰，尽量使用无异义的英语单词，与其它大多数程式设计语言使用大括号不一样，它使用**缩进**来定义语句块。

与[Scheme](http://zh.wikipedia.org/wiki/Scheme)、[Ruby](http://zh.wikipedia.org/wiki/Ruby)、[Perl](http://zh.wikipedia.org/wiki/Perl)、[Tcl](http://zh.wikipedia.org/wiki/Tcl)等动态语言一样，Python具备[垃圾回收](http://zh.wikipedia.org/wiki/%E5%9E%83%E5%9C%BE%E5%9B%9E%E6%94%B6_(%E8%A8%88%E7%AE%97%E6%A9%9F%E7%A7%91%E5%AD%B8))功能，能够自动管理内存使用。它经常被当作脚本语言用于处理系统管理任务和网路程式编写，然而它也非常适合完成各种高阶任务。Python虚拟机本身几乎可以在所有的[作业系统](http://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)中运行。使用一些诸如[py2exe](http://zh.wikipedia.org/w/index.php?title=Py2exe&action=edit&redlink=1)、[PyPy](http://zh.wikipedia.org/wiki/PyPy)、[PyInstaller](http://zh.wikipedia.org/w/index.php?title=PyInstaller&action=edit&redlink=1)之类的工具可以将Python原始码转换成可以脱离Python解释器执行的程式。

Python的官方解释器是[CPython](http://zh.wikipedia.org/wiki/CPython)，该解释器用[C语言](http://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)编写，是一个由社群驱动的自由软体，目前由[Python软体基金会](http://zh.wikipedia.org/w/index.php?title=Python%E8%BB%9F%E9%AB%94%E5%9F%BA%E9%87%91%E6%9C%83&action=edit&redlink=1)管理。

Python支援[命令式程式设计](http://zh.wikipedia.org/wiki/%E5%91%BD%E4%BB%A4%E5%BC%8F%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88)、[面向对象程序设计](http://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)、[函数式编程](http://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B8%E5%BC%8F%E7%B7%A8%E7%A8%8B)、[面向侧面程序设计](http://zh.wikipedia.org/w/index.php?title=%E9%9D%A2%E5%90%91%E4%BE%A7%E9%9D%A2%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1&action=edit&redlink=1)、[泛型编程](http://zh.wikipedia.org/wiki/%E6%B3%9B%E5%9E%8B%E7%BC%96%E7%A8%8B)多种编程范式。

**目录**

 [[隐藏](http://zh.wikipedia.org/wiki/Python)]

* [1 历史](http://zh.wikipedia.org/wiki/Python#.E6.AD.B7.E5.8F.B2)
* [2 设计哲学与定位](http://zh.wikipedia.org/wiki/Python#.E8.A8.AD.E8.A8.88.E5.93.B2.E5.AD.B8.E8.88.87.E5.AE.9A.E4.BD.8D)
* [3 应用范围](http://zh.wikipedia.org/wiki/Python#.E6.87.89.E7.94.A8.E8.8C.83.E5.9B.B4)
  + [3.1 Web程式](http://zh.wikipedia.org/wiki/Python#Web.E7.A8.8B.E5.BC.8F)
  + [3.2 GUI开发](http://zh.wikipedia.org/wiki/Python#GUI.E5.BC.80.E5.8F.91)
  + [3.3 操作系统](http://zh.wikipedia.org/wiki/Python#.E6.93.8D.E4.BD.9C.E7.B3.BB.E7.BB.9F)
  + [3.4 其他](http://zh.wikipedia.org/wiki/Python#.E5.85.B6.E4.BB.96)
* [4 Hello World](http://zh.wikipedia.org/wiki/Python#Hello_World)
  + [4.1 单步执行](http://zh.wikipedia.org/wiki/Python#.E5.96.AE.E6.AD.A5.E5.9F.B7.E8.A1.8C)
* [5 语法](http://zh.wikipedia.org/wiki/Python#.E8.AA.9E.E6.B3.95)
  + [5.1 缩排](http://zh.wikipedia.org/wiki/Python#.E7.B8.AE.E6.8E.92)
  + [5.2 语句和控制流](http://zh.wikipedia.org/wiki/Python#.E8.AA.9E.E5.8F.A5.E5.92.8C.E6.8E.A7.E5.88.B6.E6.B5.81)
  + [5.3 表达式](http://zh.wikipedia.org/wiki/Python#.E8.A1.A8.E8.BE.BE.E5.BC.8F)
  + [5.4 函数](http://zh.wikipedia.org/wiki/Python#.E5.87.BD.E6.95.B0)
  + [5.5 面向对象开发方法](http://zh.wikipedia.org/wiki/Python#.E9.9D.A2.E5.90.91.E5.AF.B9.E8.B1.A1.E5.BC.80.E5.8F.91.E6.96.B9.E6.B3.95)
  + [5.6 数据类型&动态类型](http://zh.wikipedia.org/wiki/Python#.E6.95.B0.E6.8D.AE.E7.B1.BB.E5.9E.8B.26.E5.8A.A8.E6.80.81.E7.B1.BB.E5.9E.8B)
  + [5.7 数学运算](http://zh.wikipedia.org/wiki/Python#.E6.95.B0.E5.AD.A6.E8.BF.90.E7.AE.97)
* [6 标准库](http://zh.wikipedia.org/wiki/Python#.E6.A0.87.E5.87.86.E5.BA.93)
* [7 著名第三方库](http://zh.wikipedia.org/wiki/Python#.E8.91.97.E5.90.8D.E7.AC.AC.E4.B8.89.E6.96.B9.E5.BA.93)
  + [7.1 Web框架](http://zh.wikipedia.org/wiki/Python#Web.E6.A1.86.E6.9E.B6)
  + [7.2 科学计算](http://zh.wikipedia.org/wiki/Python#.E7.A7.91.E5.AD.A6.E8.AE.A1.E7.AE.97)
  + [7.3 GUI](http://zh.wikipedia.org/wiki/Python#GUI)
  + [7.4 其它](http://zh.wikipedia.org/wiki/Python#.E5.85.B6.E5.AE.83)
* [8 Python 3.0](http://zh.wikipedia.org/wiki/Python#Python_3.0)
  + [8.1 主要变化](http://zh.wikipedia.org/wiki/Python#.E4.B8.BB.E8.A6.81.E5.8F.98.E5.8C.96)
* [9 实现](http://zh.wikipedia.org/wiki/Python#.E5.AE.9E.E7.8E.B0)
* [10 开发环境](http://zh.wikipedia.org/wiki/Python#.E5.BC.80.E5.8F.91.E7.8E.AF.E5.A2.83)
  + [10.1 通用IDE / 文本编辑器](http://zh.wikipedia.org/wiki/Python#.E9.80.9A.E7.94.A8IDE_.2F_.E6.96.87.E6.9C.AC.E7.BC.96.E8.BE.91.E5.99.A8)
  + [10.2 专门为Python设计的IDE软件](http://zh.wikipedia.org/wiki/Python#.E4.B8.93.E9.97.A8.E4.B8.BAPython.E8.AE.BE.E8.AE.A1.E7.9A.84IDE.E8.BD.AF.E4.BB.B6)
* [11 使用Python编写的著名应用](http://zh.wikipedia.org/wiki/Python#.E4.BD.BF.E7.94.A8Python.E7.BC.96.E5.86.99.E7.9A.84.E8.91.97.E5.90.8D.E5.BA.94.E7.94.A8)
* [12 影响](http://zh.wikipedia.org/wiki/Python#.E5.BD.B1.E9.9F.BF)
* [13 参考文献](http://zh.wikipedia.org/wiki/Python#.E5.8F.82.E8.80.83.E6.96.87.E7.8C.AE)
* [14 关联条目](http://zh.wikipedia.org/wiki/Python#.E9.97.9C.E8.81.AF.E6.A2.9D.E7.9B.AE)
* [15 外部链接](http://zh.wikipedia.org/wiki/Python#.E5.A4.96.E9.83.A8.E9.93.BE.E6.8E.A5)

**历史[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=1)**]**

Python的创始人为[吉多·范罗苏姆](http://zh.wikipedia.org/wiki/%E5%90%89%E5%A4%9A%C2%B7%E8%8C%83%E7%BD%97%E8%8B%8F%E5%A7%86)（Guido van Rossum）。1989年的圣诞节期间，[吉多·范罗苏姆](http://zh.wikipedia.org/wiki/%E5%90%89%E5%A4%9A%C2%B7%E8%8C%83%E7%BD%97%E8%8B%8F%E5%A7%86)为了在[阿姆斯特丹](http://zh.wikipedia.org/wiki/%E9%98%BF%E5%A7%86%E6%96%AF%E7%89%B9%E4%B8%B9)打发时间，决心开发一个新的脚本解释程式，作为[ABC语言](http://zh.wikipedia.org/wiki/ABC%E8%AA%9E%E8%A8%80)的一种继承。之所以选中Python作为程式的名字，是因为他是一个[蒙提·派森的飞行马戏团](http://zh.wikipedia.org/wiki/%E8%92%99%E6%8F%90%C2%B7%E6%B4%BE%E6%A3%AE%E7%9A%84%E9%A3%9B%E8%A1%8C%E9%A6%AC%E6%88%B2%E5%9C%98)的爱好者。ABC是由吉多参加设计的一种教学语言。就吉多本人看来，ABC这种语言非常优美和强大，是专门为非专业程式设计师设计的。但是ABC语言并没有成功，究其原因，吉多认为是非开放造成的。吉多决心在Python中避免这一错误，并取得了非常好的效果，完美结合了[C](http://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)和其他一些语言。[[1]](http://zh.wikipedia.org/wiki/Python#cite_note-python_history-1)

就这样，Python在吉多手中诞生了。实际上，第一个实现是在[Mac](http://zh.wikipedia.org/wiki/Mac)机上。可以说，Python是从ABC发展起来，主要受到了[Modula-3](http://zh.wikipedia.org/w/index.php?title=Modula-3&action=edit&redlink=1)（另一种相当优美且强大的语言，为小型团体所设计的）的影响。并且结合了[Unix shell](http://zh.wikipedia.org/wiki/Unix_shell)和C的习惯。

目前吉多仍然是Python的主要开发者，决定整个Python语言的发展方向。Python社群经常称呼他是[仁慈的独裁者](http://zh.wikipedia.org/w/index.php?title=%E4%BB%81%E6%85%88%E7%9A%84%E7%8D%A8%E8%A3%81%E8%80%85&action=edit&redlink=1)。

Python 2.0于2000年10月16日发布，增加了实现完整的[垃圾回收](http://zh.wikipedia.org/wiki/%E5%9E%83%E5%9C%BE%E5%9B%9E%E6%94%B6_(%E8%A8%88%E7%AE%97%E6%A9%9F%E7%A7%91%E5%AD%B8))，并且支持[Unicode](http://zh.wikipedia.org/wiki/Unicode)。同时，整个开发过程更加透明，社群对开发进度的影响逐渐扩大。Python 3.0于2008年12月3日发布，此版不完全兼容之前的Python原始码。不过，很多新特性后来也被移植到旧的Python 2.6/2.7版本。

Python是完全物件导向的语言。函数、模组、数字、字串都是物件。并且完全支持继承、重载、派生、多继承，有益于增强原始码的复用性。Python支持重载运算符，因此Python也支持泛型设计。相对于Lisp这种传统的函数式程式语言，Python对[函数式设计](http://zh.wikipedia.org/w/index.php?title=%E5%87%BD%E6%95%B8%E5%BC%8F%E8%A8%AD%E8%A8%88&action=edit&redlink=1)只提供了有限的支持。有两个标准库（functools, itertools）提供了[Haskell](http://zh.wikipedia.org/wiki/Haskell)和[Standard ML](http://zh.wikipedia.org/wiki/Standard_ML)中久经考验的函数式程式设计工具。

虽然Python可能被粗略地分类为“[脚本语言](http://zh.wikipedia.org/wiki/%E8%85%B3%E6%9C%AC%E8%AA%9E%E8%A8%80)”（script language），但实际上一些大规模软体开发计划例如[Zope](http://zh.wikipedia.org/wiki/Zope)、Mnet及[BitTorrent](http://zh.wikipedia.org/wiki/BitTorrent)，[Google](http://zh.wikipedia.org/wiki/Google)也广泛地使用它。Python的支持者较喜欢称它为一种高阶动态程式语言，原因是“脚本语言”泛指仅作简单程式设计任务的语言，如shell script、[VBScript](http://zh.wikipedia.org/wiki/VBScript)等只能处理简单任务的程式语言，并不能与Python相提并论。

Python本身被设计为可扩充的。并非所有的特性和功能都集成到语言核心。Python提供了丰富的API和工具，以便程式设计师能够轻松地使用[C语言](http://zh.wikipedia.org/wiki/C)、[C++](http://zh.wikipedia.org/wiki/C%2B%2B)、[Cython](http://zh.wikipedia.org/w/index.php?title=Cython&action=edit&redlink=1)来编写扩充模组。Python编译器本身也可以被集成到其它需要脚本语言的程式内。因此，很多人还把Python作为一种“[胶水语言](http://zh.wikipedia.org/wiki/%E8%83%B6%E6%B0%B4%E8%AF%AD%E8%A8%80)”（glue language）使用。使用Python将其他语言编写的程式进行集成和封装。在[Google](http://zh.wikipedia.org/wiki/Google)内部的很多项目，例如Google Engine使用[C++](http://zh.wikipedia.org/wiki/C%2B%2B)编写性能要求极高的部分，然后用Python或[Java](http://zh.wikipedia.org/wiki/Java)/[Go](http://zh.wikipedia.org/wiki/Go)调用相应的模组。[[2]](http://zh.wikipedia.org/wiki/Python#cite_note-2)《Python技术手册》的作者马特利（Alex Martelli）说：“这很难讲，不过，2004年，Python已在Google内部使用，Google召募许多Python高手，但在这之前就已决定使用Python。他们的目的是尽量使用Python，在不得已时改用C++；在操控硬体的场合使用C++，在快速开发时候使用Python。”[[3]](http://zh.wikipedia.org/wiki/Python#cite_note-3)

**设计哲学与定位[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=2)**]**

Python的设计哲学是“优雅”、“明确”、“简单”。因此其对手Perl语言中“总是有多种方法来做同一件事”的理念在Python开发者中通常是难以忍受的。Python开发者的哲学是“用一种方法，最好是只有一种方法来做一件事”。在设计Python语言时，如果面临多种选择，Python开发者一般会拒绝花俏的语法，而选择明确没有或者很少有歧义的语法。由于这种设计观念的差异，Python原始码通常被认为比Perl具备更好的可读性，并且能够支撑大规模的软体开发。这些准则被称为[Python格言](http://zh.wikipedia.org/w/index.php?title=Zen_of_Python&action=edit&redlink=1)。在Python解释器内运行import this可以获得完整的列表。

Python开发人员尽量避开不成熟或者不重要的优化。一些针对非重要部位的加快运行速度的补丁通常不会被合并到Python内。所以很多人[[谁？]](http://zh.wikipedia.org/wiki/Wikipedia:%E4%B8%8D%E8%A6%81%E6%A8%A1%E6%A3%B1%E4%B8%A4%E5%8F%AF)认为Python很慢。不过，根据[二八定律](http://zh.wikipedia.org/wiki/%E5%B8%95%E9%9B%B7%E6%89%98%E6%B3%95%E5%88%99)，大多数程式对速度要求不高。在某些对运行速度要求很高的情况，Python设计师倾向于使用[JIT技术](http://zh.wikipedia.org/wiki/%E5%8D%B3%E6%99%82%E7%B7%A8%E8%AD%AF)，或者用使用C/C++语言改写这部分程式。[[来源请求]](http://zh.wikipedia.org/wiki/Wikipedia:%E5%88%97%E6%98%8E%E6%9D%A5%E6%BA%90)目前可用的JIT技术是[PyPy](http://zh.wikipedia.org/wiki/PyPy)。

**应用范围[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=3)**]**

**Web程式[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=4)**]**

Python经常被用于Web开发。比如，通过[mod\_wsgi](http://zh.wikipedia.org/w/index.php?title=Mod_wsgi&action=edit&redlink=1)模组，[Apache](http://zh.wikipedia.org/wiki/Apache)可以运行用Python编写的Web程式。使用Python语言编写的[Gunicorn](http://zh.wikipedia.org/w/index.php?title=Gunicorn&action=edit&redlink=1)作为Web服务器，也能够运行Python语言编写的Web程序。Python定义了[WSGI](http://zh.wikipedia.org/wiki/WSGI)标准应用接口来协调Http伺服器与基于Python的Web程式之间的沟通。一些Web框架，如[Django](http://zh.wikipedia.org/wiki/Django)、[Pyramid](http://zh.wikipedia.org/wiki/Pyramid)、[TurboGears](http://zh.wikipedia.org/wiki/TurboGears)、[web2py](http://zh.wikipedia.org/w/index.php?title=Web2py&action=edit&redlink=1)、[Zope](http://zh.wikipedia.org/wiki/Zope)、[Flask](http://zh.wikipedia.org/wiki/Flask)等，可以让程式设计师轻松地开发和管理复杂的Web程式。

Python对于各种网路协定的支援很完善，因此经常被用于编写伺服器软体、网路蠕虫。第三方函式库[Twisted](http://zh.wikipedia.org/w/index.php?title=Twisted&action=edit&redlink=1" \o "Twisted（页面不存在）)支援非同步线上编写程式和多数标准的网路协定（包含客户端和伺服器），并且提供了多种工具，被广泛用于编写高性能的伺服器软体。另有[gevent](http://zh.wikipedia.org/w/index.php?title=Gevent&action=edit&redlink=1)这个流行的第三方库，同样能够支持高性能高并发的网络开发。

**GUI开发[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=5)**]**

Python本身包含的Tkinter库能够支持简单的GUI开发。但是越来越多的Python程序员选择[wxPython](http://zh.wikipedia.org/wiki/WxPython)或者[PyQt](http://zh.wikipedia.org/wiki/PyQt)来开发跨平台的桌面软件。使用它们开发的桌面软件运行速度快，与用户的桌面环境相契合。通过[PyInstaller](http://zh.wikipedia.org/w/index.php?title=PyInstaller&action=edit&redlink=1)还能将程序发布为独立的安装程序包。与C++相比较，使用Python开发桌面软件通常更快更容易。

**操作系统[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=6)**]**

在很多作业系统里，Python是标准的系统元件。大多数Linux发行版以及NetBSD、OpenBSD和Mac OS X都集成了Python，可以在终端机下直接执行Python。有一些Linux发行版的安装器使用Python语言编写，比如Ubuntu的Ubiquity安装器、Red Hat Linux和Fedora的Anaconda安装器。Gentoo Linux使用Python来编写它的Portage包管理系统。Python标准库包含了多个调用作业系统功能的函式库。通过pywin32这个第三方软体包，Python能够访问Windows的COM服务及其它Windows API。使用IronPython，Python程式能够直接调用.Net Framework。一般说来，Python编写的系统管理脚本在可读性、性能、原始码重用度、扩展性几方面都优于普通的shell脚本。

**其他[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=7)**]**

[NumPy](http://zh.wikipedia.org/w/index.php?title=NumPy&action=edit&redlink=1)、[SciPy](http://zh.wikipedia.org/wiki/SciPy)、[Matplotlib](http://zh.wikipedia.org/w/index.php?title=Matplotlib&action=edit&redlink=1)可以让Python程式设计师编写科学计算程式。有些公司会使用[Scons](http://zh.wikipedia.org/w/index.php?title=Scons&action=edit&redlink=1)代替make构建C++程序。

很多游戏使用C++编写图形显示等高性能模组，而使用Python或者[Lua](http://zh.wikipedia.org/wiki/Lua)编写游戏的逻辑、伺服器。相较于Python，[Lua](http://zh.wikipedia.org/wiki/Lua)的功能更简单、体积更小；而Python则支援更多的特性和数据类型。很多游戏，如[EVE Online](http://zh.wikipedia.org/wiki/EVE_Online)使用Python来处理游戏中繁多的逻辑。

YouTube、Google、Yahoo!、NASA都在内部大量地使用Python。[OLPC](http://zh.wikipedia.org/w/index.php?title=%E2%80%9C%E6%AF%8F%E5%80%8B%E5%AD%A9%E5%AD%90%E4%B8%80%E5%8F%B0%E7%AC%94%E8%AE%B0%E6%9C%AC%E2%80%9D&action=edit&redlink=1)的作业系统Sugar项目的大多数软体都是使用Python编写。

[**Hello World**](http://zh.wikipedia.org/wiki/Hello_World)**[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=8)**]**

一个在标准输设备上输出[Hello World](http://zh.wikipedia.org/wiki/Hello_World)的简单程式，这种程式通常作为开始学习程式语言时的第一个程式：

* 适用于Python 3.0以上版本以及Python 2.6、Python 2.7

print("Hello, world!")

* 适用于Python 2.6以下版本以及Python 2.6、Python 2.7

print "Hello, world!"

将这行程式码保存为myhello.py。然后在Linux终端机下输入python myhello.py，或者在Windows命令编辑字元下输入myhello.py执行。

**单步执行[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=9)**]**

Python也可以单步直译执行。执行Python直译器进入互动式命令列的环境，你可以在提示符号>>>旁输入print（"Hello, world!"），按Enter键输出结果：

* 适用于Python 3.0以上版本以及Python 2.6、Python 2.7

>>> print("Hello, world!")

Hello, world!

* 适用于Python 2.6以下版本以及Python 2.6、Python 2.7

>>> print "Hello, world!"

Hello, world!

注意，低于3.0版本的Python，"Hello, world!"周围不需要括号。Python 3.x与Python 2.x的print语法是不一样的。

**语法[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=10)**]**

Python的设计目标之一是让程式码具备高度的可阅读性。它设计时尽量使用其它语言经常使用的标点符号和英文单字，让程式码看起来整洁美观。它不像其他的静态语言如C、Pascal那样需要书写**声明**语句，也不像它们的语法那样经常有特殊情况。

**缩排[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=11)**]**

Python开发者有意让违反了**缩排规则**的程序不能通过编译，以此来强迫程序员养成良好的编程习惯。并且Python语言利用缩排表示语句块的开始和结束（[Off-side规则](http://zh.wikipedia.org/wiki/Off-side%E8%A6%8F%E5%89%87)），而非使用花括号或者某种[关键字](http://zh.wikipedia.org/wiki/%E9%97%9C%E9%8D%B5%E5%AD%97)。增加缩排表示语句块的开始，而减少缩排则表示语句块的结束。缩排成为了语法的一部分。例如if语句：

if age < 21:

print("你不能買酒。")

print("不過你能買口香糖。")

print("這句話處於if語句塊的外面。")

* ***注：****上述例子为Python 3.0以上版本的程式码。*

根据PEP的规定，必须使用**4个空格**来表示每级缩排。使用Tab字符和其它数目的空格虽然都可以编译通过，但不符合编码规范。支持Tab字符和其它数目的空格仅仅是为兼容很旧的的Python程式和某些有问题的编辑程式。

**语句和控制流[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=12)**]**

* if语句，当条件成立时执行语句块。经常与else, elif（相当于else if）配合使用。
* for语句，遍列列表、字符串、字典、集合等[迭代器](http://zh.wikipedia.org/wiki/%E8%BF%AD%E4%BB%A3%E5%99%A8)，依次处理迭代器中的每个元素。
* while语句，当条件为真时，循环执行语句块。
* try语句。与except, finally, else配合使用处理在程式执行中出现的异常情况。
* class语句。用于定义类型。
* def语句。用于定义函数和类型的方法。
* pass语句。表示此行为空，不执行任何操作。
* assert语句。用于程式调适阶段时测试执行条件是否满足。
* with语句。Python2.6以后定义的语法，在一个场景中执行语句块。比如，执行语句块前加锁，然后在语句块执行结束后释放锁。
* yield语句。在迭代器函数内使用，用于返回一个元素。自从Python 2.5版本以后。这个语句变成一个运算符。
* raise语句。抛出一个异常。
* import语句。导入一个模块或包。常用写法：from module import name, import module as name, from module import name as anothername

**表达式[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=13)**]**

Python的表达式写法与C/C++类似。只是在某些写法有所差别。

* 主要的算术运算符与C/C++类似。+, -, \*, /, //, \*\*, ~, %分别表示加法或者取正、减法或者取负、乘法、除法、整除、乘方、取补、取模。>>, <<表示右移和左移。&, |, ^表示二进制的AND, OR, XOR运算。>, <, ==, !=, <=, >=用于比较两个表达式的值，分别表示大于、小于、等于、不等于、小于等于、大于等于。在这些运算符里面，~, |, ^, &, <<, >>必须应用于整数。
* Python使用and, or, not表示逻辑运算。
* is, is not用于比较两个变量是否是同一个对象。in, not in用于判断一个对象是否属于另外一个对象。
* Python支持字典、集合、列表的推导式（dict comprehension, set comprehension, list comprehension）。比如：

>>> [x + 3 for x in range(4)]

[3, 4, 5, 6]

>>> {x + 3 for x in range(4)}

{3, 4, 5, 6}

>>> {x: x + 3 for x in range(4)}

{0: 3, 1: 4, 2: 5, 3: 6}

* Python支持“迭代表达式”（generator comprehension），比如计算0-9的平方和：

>>> sum(x \* x for x in range(10))

285

* Python使用lambda表示匿名函数。匿名函数体只能是表达式。比如：

>>> add = lambda x, y : x + y

>>> add(3, 2)

5

* Python使用y if cond else x表示条件表达式。意思是当cond为真时，表达式的值为y，否则表达式的值为x。相当于C++和Java里的cond?y:x。
* Python区分列表（list）和元组（tuple）两种类型。list的写法是[1,2,3]，而tuple的写法是(1,2,3)。可以改变list中的元素，而不能改变tuple。在某些情况下，tuple的括号可以省略。tuple对于赋值语句有特殊的处理。因此，可以同时赋值给多个变量，比如：

>>> x, y=1, 2 #同时给x,y赋值，最终结果：x=1, y=2

特别地，可以使用以下这种形式来交换两个变量的值：

>>> x, y = y, x #最终结果：y=1, x=2

* Python使用'（单引号）和"（双引号）来表示字符串。与Perl、Unix Shell语言或者Ruby、Groovy等语言不一样，两种符号作用相同。一般地，如果字符串中出现了双引号，就使用单引号来表示字符串;反之则使用双引号。如果都没有出现，就依个人喜好选择。出现在字符串中的\（反斜杠）被解释为特殊字符，比如\n表示换行符。表达式前加r指示Python不解释字符串中出现的\。这种写法通常用于编写正则表达式或者Windows文件路径。
* Python支持列表切割（list slices），可以取得完整列表的一部分。支持切割操作的类型有str, bytes, list, tuple等。它的语法是...[left:right]或者...[left:right:stride]。假定nums变量的值是[1, 3, 5, 7, 8, 13, 20]，那么下面几个语句为真：
* nums[2:5] == [5, 7, 8]从下标为2的元素切割到下标为5的元素，但不包含下标为5的元素。
* nums[1:] == [3, 5, 7, 8, 13, 20]切割到最后一个元素。
* nums[:-3] == [1, 3, 5, 7]从最开始的元素一直切割到倒数第3个元素。
* nums[:] == [1, 3, 5, 7, 8, 13, 20]返回所有元素。改变新的列表不会影响到nums。
* nums[1:5:2] == [3, 7]从下标为1的元素切割到下标为5的元素但不包含下标为5的元素，且步长为2

**函数[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=14)**]**

Python的函数支持[递归](http://zh.wikipedia.org/wiki/%E9%80%92%E5%BD%92)、默认参数值、可变参数、闭包，但不支持函数[重载](http://zh.wikipedia.org/wiki/%E9%87%8D%E8%BC%89)。为了增强代码的可读性，可以在函数后书写“文档字符串”（Documentation Strings，或者简称docstrings），用于解释函数的作用、参数的类型与意义、返回值类型与取值范围等。可以使用内置函数help()打印出函数的使用帮助。比如：

>>> def randint(a, b):

... "Return random integer in range [a, b], including both end points."

...

>>> help(randint)

Help on function randint in module \_\_main\_\_:

randint(a, b)

Return random integer in range [a, b], including both end points.

函数调用时，实参可以如同C语言那样按照位置与形参匹配；也可以按照keyword argument形式调用，即param\_name=value形式的实参。在一个函数调用的实参表中，keyword argument必须出现在位置参数之后。

可变参数用\*args或\*\*dictargs表示，即在形式参数名字前加一个\*号，表示这是由多个实参组成的可变参数，该形参视作tuple数据类型；在形式参数名字前加\*\*号，表示这是由多个实参组成的可变参数，该形参视作dict数据类型。实际上，在一个collection类型（包括set、list、tuple甚至bytes、str等）的变量前加一个\*号，获得了其中所有元素作为多个对象。

Python的函数作为first-class citizen，具有和普通变量平等的地位。函数一旦定义，即可视作为普通对象，其形参会保留上次调用时的值，但在函数新的一次调用时会被实参值覆盖。因此函数的**缺省参数值**在连续多次调用该函数时，如果不被实参值覆盖，就会一直保留。例如：

def f(a, L=[]):

L.append(a)

return L

print(f(1))

print(f(2))

print(f(3))

结果为：

[1]

[1, 2]

[1, 2, 3]

函数的**缺省参数值**在函数被定义时被一次性计算其初值。

Python的函数实参与形参之间的结合是传递*对象的引用*[[4]](http://zh.wikipedia.org/wiki/Python#cite_note-4)。这是因为Python的赋值操作是把（变量）名字绑定到对象上。形实结合也是这种方式。如果形参绑定到一个mutable对象，则通过形参对此对象内容的修改，在函数外也是可见的。如果形参绑定到一个immutable对象，则通过形参是不能修改此对象内容，但可以把形参重新绑定到其它对象上，这并不影响函数外的对象的值。例如：

def foo(a):

a.append('haha')

def bar(b):

b=101 #实际上是重新绑定了另一个整形对象101

a=[]

b=100

foo(a)

bar(b)

print(a) #结果为['haha']

print(b) #结果为100

**面向对象开发方法[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=15)**]**

面向对象开发方法是指绑定到对象的函数。调用对象方法的语法是instance.method(arguments)。它等价于调用Class.method(instance, arguments)。当定义对象方法时，必须显式地定义第一个参数，一般该参数名都使用self，用于访问对象的内部数据。这里的self相当于C++, Java里面的this变量，但是我们还可以使用任何其它合法的参数名，比如this和mine等，self与C++,Java里面的this不完全一样，它可以被看作是一个习惯性的用法，我们传入任何其它的合法名称都行，比如：

class Fish:

def eat(self, food):

if food is not None:

self.hungry=False

class User:

def \_\_init\_\_(myself, name):

myself.name = name

#构造Fish的实例：

f=Fish()

#以下两种调用形式是等价的：

Fish.eat(f, "earthworm")

f.eat("earthworm")

u = User('username')

u.name

Python认识一些以“\_\_”开始并以“\_\_”结束的特殊方法名，它们用于实现运算符重载和实现多种特殊功能。

**数据类型&动态类型[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=16)**]**

Python采用动态类型系统。在编译的时候，Python不会检查对象是否拥有被调用的方法或者属性，而是直至运行时，才做出检查。所以操作对象时可能会抛出异常。不过，虽然Python采用动态类型系统，它同时也是强类型的。Python禁止没有明确定义的操作，比如数字加字符串。

与其它面向对象语言一样，Python允许程序员定义类型。构造一个对象只需要像函数一样调用类型即可，比如，对于前面定义的Fish类型，使用Fish()。类型本身也是特殊类型type的对象（type类型本身也是type对象），这种特殊的设计允许对类型进行[反射](http://zh.wikipedia.org/wiki/%E5%8F%8D%E5%B0%84_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6))编程。

Python内置丰富的数据类型。与Java、C++相比，这些数据类型有效地减少代码的长度。下面这个列表简要地描述了Python内置数据类型（适用于Python 3.x）：

|  |  |  |
| --- | --- | --- |
| **类型** | **描述** | **例子** |
| [str](http://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2) | 一个由字符组成的不可更改的有序列。在Python 3.x里，字符串由Unicode字符组成。 | 'Wikipedia' "Wikipedia" """Spanning multiple lines""" |
| bytes | 一个由字节组成的不可更改的有序列。 | b'Some ASCII' b"Some ASCII" |
| list | 可以包含多种类型的可改变的有序列 | [4.0, 'string', True] |
| tuple | 可以包含多种类型的不可改变的有序列 | (4.0, 'string', True) |
| set, frozenset | 与数学中集合的概念类似。无序的、每个元素唯一。 | {4.0, 'string', True} frozenset([4.0, 'string', True]) |
| dict或map | 一个可改变的由键值对组成的无序列？。 | {'key1': 1.0, 3: False} |
| int | 精度不限的整数 | 42 |
| float | 浮点数。精度与系统相关。 | 3.1415927 |
| complex | 复数 | 3+2.7j |
| bool | 逻辑值。只有两个值：真、假 | True False |

除了各种数据类型，Python语言还用类型来表示函数、模块、类型本身、对象的方法、编译后的Python代码、运行时信息等等。因此，Python具备很强的动态性。

**数学运算[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=17)**]**

Python使用与C、Java类似的运算符，支持整数与浮点数的数学运算。同时还支持复数运算与无穷位数（实际受限于计算机的能力）的整数运算。除了求绝对值函数abs()外，大多数数学函数处于math和cmath模块内。前者用于实数运算，而后者用于复数运算。使用时需要先导入它们，比如：

>>> import math

>>> print(math.sin(math.pi/2))

1.0

fractions模块用于支持分数运算；decimal模块用于支持高精度的浮点数运算。

Python定义求余运行a % b的值处于开区间[0, b)内，如果b是负数，开区间变为(b, 0]。这是一个很常见的定义方式。不过其实它依赖于整除的定义。为了让方程式：b \* (a // b) + a % b = a恒真，整除运行需要向负无穷小方向取值。比如7 // 3的结果是2，而(-7) // 3的结果却是-3。这个算法与其它很多编程语言不一样，需要注意，它们的整除运算会向0的方向取值。

Python允许像数学的常用写法那样连着写两个比较运行符。比如a < b < c与a < b and b < c等价。C++的结果与Python不一样，首先它会先计算a < b，根据两者的大小获得0或者1两个值之一，然后再与c进行比较。

**标准库[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=18)**]**

Python拥有一个强大的标准库[[5]](http://zh.wikipedia.org/wiki/Python#cite_note-5)。Python语言的核心只包含数字、字符串、列表、字典、文件等常见类型和函数，而由Python标准库提供了系统管理、网络通信、文本处理、数据库接口、图形系统、XML处理等额外的功能。Python标准库命名接口清晰、文档良好，很容易学习和使用。

Python标准库的主要功能有：

* 文本处理，包含文本格式化、正则表达式匹配、文本差异计算与合并、Unicode支持，二进制数据处理等功能
* 文件处理，包含文件操作、创建临时文件、文件压缩与归档、操作配置文件等功能
* 操作系统功能，包含线程与进程支持、IO复用、日期与时间处理、调用系统函数、写日记（logging）等功能
* 网络通信，包含网络套接字，SSL加密通信、异步网络通信等功能
* 网络协议，支持HTTP，FTP，SMTP，POP，IMAP，NNTP，XMLRPC等多种网络协议，并提供了编写网络服务器的框架
* W3C格式支持，包含HTML，SGML，XML的处理。
* 其它功能，包括国际化支持、数学运算、HASH、Tkinter等

Python社区提供了大量的第三方模块，使用方式与标准库类似。它们的功能无所不包，覆盖科学计算、Web开发、数据库接口、图形系统多个领域，并且大多成熟而稳定。第三方模块可以使用Python或者[C语言](http://zh.wikipedia.org/wiki/C)编写。[SWIG](http://zh.wikipedia.org/w/index.php?title=SWIG&action=edit&redlink=1),[SIP](http://zh.wikipedia.org/wiki/SIP)常用于将C语言编写的程序库转化为Python模块。[Boost C++ Libraries](http://zh.wikipedia.org/wiki/Boost_C%2B%2B_Libraries)包含了一组函式库，Boost.Python，使得以Python或C++ 编写的程式能互相呼叫。借助于拥有基于标准库的大量工具、能够使用低级语言如C和可以作为其他库接口的C++，Python已成为一种强大的应用于其他语言与工具之间的“胶水”语言。

**著名第三方库[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=19)**]**

**Web框架[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=20)**]**

[Django](http://zh.wikipedia.org/wiki/Django)

开源web开发框架，它鼓励快速开发,并遵循MVC设计，开发周期短。

ActiveGrid

企业级的Web2.0解决方案。

Karrigell

简单的Web框架，自身包含了Web服务，py脚本引擎和纯python的数据库PyDBLite。

webpy

一个小巧灵活的Web框架，虽然简单但是功能强大。

CherryPy

基于Python的Web应用程序开发框架。

Pylons

基于Python的一个极其高效和可靠的Web开发框架。

Zope

开源的Web应用服务器。

TurboGears

基于Python的MVC风格的Web应用程序框架。

Twisted

流行的网络编程库，大型Web框架。

Quixote

Web开发框架。

**科学计算[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=21)**]**

Matplotlib

用Python实现的类matlab的第三方库，用以绘制一些高质量的数学二维图形。

SciPy

基于Python的matlab实现，旨在实现matlab的所有功能。

NumPy

基于Python的科学计算第三方库，提供了矩阵，线性代数，傅立叶变换等等的解决方案。

**GUI[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=22)**]**

PyGtk

基于Python的GUI程序开发GTK+库。

PyQt

用于Python的QT开发库。

WxPython

Python下的GUI编程框架，与MFC的架构相似。

**其它[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=23)**]**

BeautifulSoup

基于Python的HTML/XML解析器，简单易用。

PIL

基于Python的图像处理库，功能强大，对图形文件的格式支持广泛。

PyGame

基于Python的多媒体开发和游戏软件开发模块。

Py2exe

将python脚本转换为windows上可以独立运行的可执行程序。

**Python 3.0[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=24)**]**

Python的3.0版本，常被称为**Python 3000**，或简称**Py3k**[[6]](http://zh.wikipedia.org/wiki/Python#cite_note-py3k_release-6)。相对于Python的早期版本，这是一个较大的[升级](http://zh.wikipedia.org/wiki/%E5%8D%87%E7%BA%A7)。为了不带入过多的累赘，Python 3.0在设计的时候没有考虑[向下兼容](http://zh.wikipedia.org/wiki/%E5%90%91%E4%B8%8B%E5%85%BC%E5%AE%B9)。许多针对早期Python版本设计的程序都无法在Python 3.0上正常运行。为了照顾现有程序，Python 2.6作为一个过渡版本，基本使用了Python 2.x的语法和[库](http://zh.wikipedia.org/wiki/%E5%BA%93_(%E8%AE%A1%E7%AE%97%E6%9C%BA))，同时考虑了向Python 3.0的迁移，允许使用部分Python 3.0的语法与函数。基于早期Python版本而能正常运行于Python 2.6并无警告的程序可以通过一个2 to 3的转换工具无缝迁移到Python 3.0[[7]](http://zh.wikipedia.org/wiki/Python#cite_note-py3k_2to3-7)。

新的Python程序建议使用Python 3.0版本的语法[[8]](http://zh.wikipedia.org/wiki/Python#cite_note-python2_or_python3-8)。除非运行环境无法安装Python 3.0或者程序本身使用了不支持Python 3.0的第三方库。目前不支持Python 3.0的第三方库有Twisted, py2exe, PIL等。大多数第三方库都正在努力地兼容Python 3.0版本。即使无法立即使用Python 3.0，也建议编写兼容Python 3.0版本的程序，然后使用Python 2.6, Python 2.7来运行。

**Python 2.7**被确定为**最后一个Python 2.x版本**，它除了支持Python 2.x语法外，还支持部分Python 3.1语法。

**主要变化[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=25)**]**

Python 3.0的变化主要在以下几个方面[[9]](http://zh.wikipedia.org/wiki/Python#cite_note-py3k_whatsnew-9)：

* print语句没有了，取而代之的是print()函数。可以使用2to3工具来自动转换。Python 2.6与Python 2.7部分地支持这种形式的print语法。在Python 2.6与Python 2.7里面，以下三种形式是等价的：

print "fish"

print ("fish") #注意print后面有个空格

print("fish") #print()不能带有任何其它参数

然而，Python 2.6实际已经支持新的print()语法：

from \_\_future\_\_ import print\_function

print("fish", "panda", sep=', ')

* 新的str类型表示一个[Unicode](http://zh.wikipedia.org/wiki/Unicode)字符串，相当于Python 2.x版本的unicode类型。而字节序列则用类似b"abc"的语法表示，用bytes类表示，相当于Python 2.x的str类型。现在两种类型不能再隐式地自动转换，因此在Python 3.x里面"fish"+b"panda"是错误。正确的写法是"fish"+b"panda".decode("utf-8")。Python 2.6可以自动地将字节序列识别为Unicode字符串，方法是：

from \_\_future\_\_ import unicode\_literals

print(repr("fish"))

* 除法运算符/在Python 3.x内总是返回浮点数。而在Python 2.6内会判断被除数与除数是否是整数。如果是整数会返回整数值，相当于整除;浮点数则返回浮点数值。为了让Python 2.6统一返回浮点数值，可以：

from \_\_future\_\_ import division

print(3/2)

* 捕获异常的语法由except exc, var改为except exc as var。使用语法except (exc1, exc2) as var可以同时捕获多种类型的异常。Python 2.6已经支持这两种语法。
* 集合（set）的新写法：{1,2,3,4}。注意{}仍然表示空的字典（dict）。
* 字典推导式（Dictionary comprehensions）{expr1: expr2 for k, v in d}，这个语法等价于：

result={}

for k, v in d.items():

result[expr1]=expr2

return result

* 集合推导式（Set Comprehensions）{expr1 for x in stuff}。这个语法等价于：

result = set()

for x in stuff:

result.add(expr1)

return result

* 八进制数必须写成0o777，原来的形式0777不能用了；二进制必须写成0b111。新增了一个bin()函数用于将一个整数转换成二进制字符串。Python 2.6已经支持这两种语法。
* dict.keys(), dict.values(), dict.items(), map(), filter(), range(), zip()不再返回列表，而是迭代器。
* 如果两个对象之间没有定义明确的有意义的顺序。使用<, >, <=, >=比较它们会抛出异常。比如1 < ""在Python 2.6里面会返回True，而在Python 3.0里面会抛出异常。现在cmp(), instance.\_\_cmp\_\_()函数已经被删除。
* 可以注释函数的参数与返回值。此特性可方便IDE对源代码进行更深入的分析。例如：

def sendMail(from\_:"nobody@example.com", \

to:"somebody@example.com", \

title:"hello", \

body:"Just say hello to you.") -> True:

pass

* 合并int与long类型。
* 多个模块被改名（根据PEP8）：

|  |  |
| --- | --- |
| **旧的名字** | **新的名字** |
| \_winreg | winreg |
| ConfigParser | configparser |
| copy\_reg | copyreg |
| Queue | queue |
| SocketServer | socketserver |
| repr | reprlib |

1. StringIO模块现在被合并到新的io模块内。new, md5, gopherlib等模块被删除。Python 2.6已经支持新的io模块。
2. httplib, BaseHTTPServer, CGIHTTPServer, SimpleHTTPServer, Cookie, cookielib被合并到http包内。
3. 取消了exec语句，只剩下exec()函数。Python 2.6已经支持exec()函数。

* 其他变化详见参考文献[[9]](http://zh.wikipedia.org/wiki/Python#cite_note-py3k_whatsnew-9)。基本上，可以编写出使用Python 3.0语法并运行于Python 2.6, Python 2.7的程序。

**实现[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=26)**]**

Python是一门跨平台的脚本语言，Python规定了一个Python语法规则，根据该规则可编写Python[解释器](http://zh.wikipedia.org/wiki/%E8%A7%A3%E9%87%8A%E5%99%A8)。

* [CPython](http://zh.wikipedia.org/wiki/CPython)[[10]](http://zh.wikipedia.org/wiki/Python#cite_note-10)，官方的解释器。需要区别于其他解释器的时候才以CPython称呼。这是最常用的Python版本。
* [Jython](http://zh.wikipedia.org/wiki/Jython)（原名**JPython**；**J**ava语言实现的P**ython**，现已正式发布）。Jython可以直接调用Java的各种函数库。
* [PyPy](http://zh.wikipedia.org/wiki/PyPy)（使用**Py**thon语言写的**Py**thon）
* [IronPython](http://zh.wikipedia.org/wiki/IronPython)（面向[.NET](http://zh.wikipedia.org/wiki/.NET)和[ECMA CLI](http://zh.wikipedia.org/w/index.php?title=ECMA_CLI&action=edit&redlink=1)的Python实现）。IronPython能够直接调用.net平台的各种函数库。可以将Python程序编译成.net程序。
* [ZhPy](http://zh.wikipedia.org/wiki/ZhPy)（周蟒，支持使用繁/简中文语句编写程序的**Py**thon语言）

**开发环境[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=27)**]**

**通用IDE / 文本编辑器[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=28)**]**

很多并非集成开发环境软件的[文本编辑器](http://zh.wikipedia.org/wiki/%E6%96%87%E6%9C%AC%E7%BC%96%E8%BE%91%E5%99%A8)，也对Python有不同程度的支持，并且加上专门为Python设计的编辑器插件也会有很高的可用性。

* [eclipse](http://zh.wikipedia.org/wiki/Eclipse) + pydev插件，目前对Python 3.X只支持到3.0
* [emacs](http://zh.wikipedia.org/wiki/Emacs) + 插件
* [NetBeans](http://zh.wikipedia.org/wiki/NetBeans) + 插件
* [SlickEdit](http://zh.wikipedia.org/wiki/SlickEdit)
* [TextMate](http://zh.wikipedia.org/wiki/TextMate)
* [Python Tools for Visual Studio](http://zh.wikipedia.org/w/index.php?title=Python_Tools_for_Visual_Studio&action=edit&redlink=1)
* [Vim](http://zh.wikipedia.org/wiki/Vim) + 插件
* [Sublime Text](http://zh.wikipedia.org/wiki/Sublime_Text) + 插件
* [EditPlus](http://zh.wikipedia.org/wiki/EditPlus)
* [UltraEdit](http://zh.wikipedia.org/wiki/UltraEdit)
* [PSPad](http://zh.wikipedia.org/wiki/PSPad)
* [Editra](http://zh.wikipedia.org/w/index.php?title=Editra&action=edit&redlink=1)[[11]](http://zh.wikipedia.org/wiki/Python#cite_note-11)由Python开发的程式编辑器。
* [PyCharm](http://zh.wikipedia.org/w/index.php?title=PyCharm&action=edit&redlink=1)
* [Notepad++](http://zh.wikipedia.org/wiki/Notepad%2B%2B)

**专门为Python设计的IDE软件[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=29)**]**

适用于Python的[集成开发环境](http://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83)（IDE）软件，除了标准二进制发布包所附的IDLE之外，还有许多其他选择。其中有些软件设计有语法着色、语法检查、运行[调试](http://zh.wikipedia.org/wiki/%E8%B0%83%E8%AF%95)、自动补全、智能感知等便利功能。由于Python的[跨平台](http://zh.wikipedia.org/wiki/%E8%B7%A8%E5%B9%B3%E5%8F%B0)出身，这些软件往往也具备各种操作系统的版本或一定的移植性。

* [Eric](http://zh.wikipedia.org/w/index.php?title=Eric_(Python)&action=edit&redlink=1)：基于[PyQt](http://zh.wikipedia.org/wiki/PyQt)的自由软件，功能强大。支持自动补全、智能感知、自动语法检查、工程管理、svn/cvs集成、自动单元测试等功能。调试功能与Visual Studio和Eclipse类似。目前同时有两个版本。Eric4支持Python2.x，Eric5支持Python3.x。使用前需要先安装相应的PyQt版本。
* [IDLE](http://zh.wikipedia.org/w/index.php?title=IDLE&action=edit&redlink=1)：Python“标准”IDE。一般随Python而安装，支持较少的编辑功能。调试功能也比较弱。
* [Komodo](http://zh.wikipedia.org/w/index.php?title=Komodo&action=edit&redlink=1)和[Komodo Edit](http://zh.wikipedia.org/w/index.php?title=Komodo_Edit&action=edit&redlink=1)：后者是前者的免费精简版。也可以用于PHP，Ruby，Javascript，Perl，Web和云开发。
* [PyCharm](http://zh.wikipedia.org/w/index.php?title=PyCharm&action=edit&redlink=1)：由[JetBrains](http://zh.wikipedia.org/wiki/JetBrains)打造，该公司的Java IDE软件IntelliJ拥有海量的使用者；PyCharm具备一般IDE的功能，比如，调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制等等，同时另外，PyCharm还提供了一些很好的功能用于Django开发，同时支持[Google App Engine](http://zh.wikipedia.org/wiki/Google_App_Engine)，更酷的是，PyCharm支持IronPython。PyCharm是商业软件，目前已经到3.1版本。
* [PythonWin](http://zh.wikipedia.org/w/index.php?title=PythonWin&action=edit&redlink=1)：包含在pywin32内的编辑器，仅适用于Windows。
* [SPE](http://zh.wikipedia.org/w/index.php?title=SPE&action=edit&redlink=1)（Stani's Python Editor）：功能较多的免费软件，依赖[wxPython](http://zh.wikipedia.org/wiki/WxPython)。
* [Ulipad](http://zh.wikipedia.org/w/index.php?title=Ulipad&action=edit&redlink=1)：功能较全的免费软件，依赖[wxPython](http://zh.wikipedia.org/wiki/WxPython)。
* [WingIDE](http://zh.wikipedia.org/w/index.php?title=WingIDE&action=edit&redlink=1)：可能是功能最全的IDE，但不是免费软件。
* [PyScripter](http://zh.wikipedia.org/w/index.php?title=PyScripter&action=edit&redlink=1)：功能较全的开源IDE，使用Delphi开发。

**使用Python编写的著名应用[**[**编辑**](http://zh.wikipedia.org/w/index.php?title=Python&action=edit&section=30)**]**

* [Reddit](http://zh.wikipedia.org/wiki/Reddit) - 社交分享网站
* [Dropbox](http://zh.wikipedia.org/wiki/Dropbox) - 文件分享服务
* [豆瓣网](http://zh.wikipedia.org/wiki/%E8%B1%86%E7%93%A3%E7%BD%91) - 图书、唱片、电影等文化产品的资料数据库网站
* [Django](http://zh.wikipedia.org/wiki/Django) - 鼓励快速开发的Web应用框架
* [Pylons](http://zh.wikipedia.org/wiki/Pylons) - [Web应用框架](http://zh.wikipedia.org/wiki/Web%E5%BA%94%E7%94%A8%E6%A1%86%E6%9E%B6)
* [Zope](http://zh.wikipedia.org/wiki/Zope) - 应用服务器
* [Plone](http://zh.wikipedia.org/wiki/Plone) - 内容管理系统
* [TurboGears](http://zh.wikipedia.org/wiki/TurboGears) - 另一个Web应用快速开发框架
* [Twisted](http://twistedmatrix.com/) - Python的网络应用程序框架
* [Fabric](http://fabfile.org/) - 用于管理成百上千台Linux主机的程序库
* [Python Wikipedia Robot Framework](http://sourceforge.net/projects/pywikipediabot) - MediaWiki的机器人程式
* [MoinMoinWiki](http://moinmo.in/) - Python写成的[Wiki](http://zh.wikipedia.org/wiki/Wiki)程序
* [Trac](http://trac.edgewall.org/) - 使用Python编写的BUG管理系统
* [Mailman](http://www.gnu.org/s/mailman) - 使用Python编写的邮件列表软件
* [Mezzanine](http://mezzanine.jupo.org/) - 基于Django编写的[内容管理系统](http://zh.wikipedia.org/wiki/CMS)系统
* [flask](http://zh.wikipedia.org/wiki/Flask) - Python微Web框架
* [Webpy](http://webpy.org/) - Python微Web框架
* [Bottle](http://zh.wikipedia.org/w/index.php?title=Bottle&action=edit&redlink=1) - Python微Web框架
* [EVE](http://zh.wikipedia.org/wiki/EVE) - 网络游戏EVE大量使用Python进行开发
* [Blender](http://zh.wikipedia.org/wiki/Blender) - 使用Python作为建模工具与GUI语言的开源3D绘图软体
* [Inkscape](http://zh.wikipedia.org/wiki/Inkscape) - 一个开源的SVG矢量图形编辑器。