Python快速入门

Python（英国发音：/ˈpaɪθən/ 美国发音：/ˈpaɪθɑːn/），是一种广泛使用的高级编程语言，属于通用型编程语言，由**吉多·范罗苏姆创造**，第一版发布于1991年。可以视之为一种改良（加入一些其他编程语言的优点，如面向对象）的LISP。作为一种解释型语言，Python的设计哲学强调代码的可读性和简洁的语法（尤其是使用空格缩进划分代码块，而非使用大括号或者关键词）。**相比于C++或Java，Python让开发者能够用更少的代码表达想法。不管是小型还是大型程序，该语言都试图让程序的结构清晰明了。**

与Scheme、Ruby、Perl、Tcl等动态类型编程语言一样，**Python拥有动态类型系统和垃圾回收功能，能够自动管理内存使用，并且支持多种编程范式，包括面向对象、命令式、函数式和过程式编程。其本身拥有一个巨大而广泛的标准库。**

Python 解释器本身几乎可以在所有的操作系统中运行。Python的正式解释器CPython是用C语言编写的、是一个由社群驱动的自由软件，目前由Python软件基金会管理。

历史

Python的创始人为吉多·范罗苏姆

Python的创始人为吉多·范罗苏姆。1989年的圣诞节期间，吉多·范罗苏姆为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序，作为ABC语言的一种继承。之所以选中Python作为程序的名字，是因为他是BBC电视剧——蒙提·派森的飞行马戏团的爱好者。**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，是专门为非专业程序员设计的。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，并获取了非常好的效果，完美结合了C和其他一些语言。**

就这样，Python在吉多手中诞生了。实际上，第一个实现是在Mac机上。可以说，Python是从ABC发展起来，主要受到了Modula-3（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**终身仁慈独裁者**（BDFL）。

Python 2.0于2000年10月16日发布，增加了实现完整的垃圾回收，并且支持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源代码。不过，很多新特性后来也被移植到旧的Python 2.6/2.7版本。

特性与设计哲学

Python是完全面向对象的语言。函数、模块、数字、字符串都是对象。并且完全支持继承、重载、派生、多重继承，有益于增强源代码的复用性。Python支持重载运算符，因此Python也支持泛型设计。 相对于Lisp这种传统的函数式编程语言，Python对函数式设计只提供了有限的支持。有两个标准库（functools, itertools）提供了与Haskell和Standard ML中类似的函数式程序设计工具。

虽然Python可能被粗略地分类为“**脚本语言**”，但实际上一些大规模软件开发计划例如Zope、Mnet及BitTorrent，Google也广泛地使用它。Python的支持者较喜欢称它为**一种高端动态编程语言**，原因是“脚本语言”泛指仅作简单程序设计任务的语言，如shell script、VBScript等只能处理简单任务的编程语言，并不能与Python相提并论。

Python本身被设计为可扩展的。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，以便程序员能够轻松地使用C、C++、Cython来编写扩展模块。Python编译器本身也可以被集成到其它需要脚本语言的程序内。因此，有很多人把Python作为一种“胶水语言”使用。**使用Python将其他语言编写的程序进行集成和封装。在Google内部的很多项目，例如Google应用服务引擎使用C++编写性能要求极高的部分，然后用Python或Java/Go调用相应的模块。《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++；在操控硬件的场合使用C++，在快速开发时候使用Python。”

Python的设计哲学是**“优雅”、“明确”、“简单”。**

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”，也因此它和拥有明显个人风格的其他语言很不一样。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。这些准则被称为“**Python格言**”。在Python解释器内运行`import this`可以获得完整的列表。

>>> import this

> ***The Zen of Python***
>
> by ***Tim Peters***
>
> Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than *right* now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!

--From: Python.org

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。再加上因为Python属于动态类型语言，动态类型语言是在运行期间检查数据的类型，不得不保持描述变量值的实际类型标记，程序在每次操作变量时，需要执行数据依赖分支，而静态类型语言相对于动态类型语言，在声明变量时已经指定了数据类型和表示方法，根据这一原理导致Python相对于C、Visual Basic等静态类型语言来说运行速度较慢。不过，根据二八定律，大多数程序对速度要求不高。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，或者用使用C/C++语言改写这部分程序。目前可用的JIT技术是PyPy。

> CPython：是用C语言实现Pyhon，是目前应用最广泛的解释器。最新的语言特性都是在这个上面先实现，基本包含了所有第三方库支持，但是CPython有几个缺陷，一是全局锁使Python在多线程效能上表现不佳，二是CPython无法支持JIT（即时编译），导致其执行速度不及Java和Javascipt等语言。于是出现了Pypy。
>
> Pypy：是用Python自身实现的解释器。针对CPython的缺点进行了各方面的改良，性能得到很大的提升。最重要的一点就是Pypy集成了JIT。但是，Pypy无法支持官方的C/Python API，导致无法使用例如Numpy，Scipy等重要的第三方库。这也是现在Pypy没有被广泛使用的原因吧。
>
> 而PyPy与CPython的不同在于，别的一些python实现如CPython是使用解释执行的方式，这样的实现方式在性能上是很凄惨的。而PyPy使用了JIT(即时编译)技术，在性能上得到了提升。
>
> **Python的解释器：**
>
> 1、由于Python是动态编译的语言，和C/C++、Java或者Kotlin等静态语言不同，它是在运行时一句一句代码地边编译边执行的，而Java是提前将高级语言编译成了JVM字节码，运行时直接通过JVM和机器打交道，所以进行密集计算时运行速度远高于动态编译语言。
>
> 2、PyPy，它使用了JIT（即时编译）技术，混合了动态编译和静态编译的特性，仍然是一句一句编译源代码，但是会将翻译过的代码缓存起来以降低性能损耗。相对于静态编译代码，即时编译的代码可以处理延迟绑定并增强安全性。绝大部分 Python代码都可以在PyPy下运行，但是PyPy和CPython有一些是不同的。

应用范围

Web程序

Python经常被用于Web开发。比如，通过mod_wsgi模块，Apache可以运行用Python编写的Web程序。使用Python语言编写的Gunicorn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标准应用接口来协调Http服务器与基于Python的Web程序之间的沟通。一些Web框架，如Django、Pyramid、TurboGears、Tornado、web2py、Zope、Flask等，可以让程序员轻松地开发和管理复杂的Web程序。

Python对于各种网络协议的支持很完善，因此经常被用于编写服务器软件、网络爬虫。第三方库Twisted支持异步在线编写程序和多数标准的网络协议（包含客户端和服务器），并且提供了多种工具，被广泛用于编写高性能的服务器软件。另有gevent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

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或者PyQt等GUI包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还能将程序发布为独立的安装程序包。

操作系统

在很多操作系统里，Python是标准的系统组件。大多数Linux发行版和Mac OS X都集成了Python，可以在终端机下直接运行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。在RPM系列Linux发行版中，有一些系统组件就是用Python编写的。Gentoo Linux使用Python来编写它的Portage软件包管理系统。**Python标准库包含了多个调用作业系统功能的库。**通过pywin32这个第三方软件包，Python能够访问Windows的COM服务及其它Windows API。使用IronPython，Python程序能够直接调用.Net Framework。

其他

NumPy、SciPy、Matplotlib可以让Python程序员编写科学计算程序。有些公司会使用**Scons**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块，而使用Python或者Lua编写游戏的逻辑、服务器。相较于Python，Lua的功能更简单、体积更小；而Python则支持更多的特性和数据类型。很多游戏，如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的作业系统Sugar项目的大多数软件都是使用Python编写。

Hello World

一个在标准输出设备上输出Hello World的简单程序，这种程序通常作为开始学习编程语言时的第一个程序：

- 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!"
```

单步运行

Python也可以单步解释运行。运行Python解释器进入交互式命令行的环境，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，在3.0及以上版本中，**需要在"Hello,world"周围加上圆括号。其原因是在3.0及以上版本中，print命令不再是一个关键字，而是一个函数。**

语法

Python的设计目标之一是让代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，让代码看起来整洁美观。**因为Python是动态语言，它不像其他的静态语言如C、Pascal那样需要书写声明语句。**

缩进

Python开发者有意让违反了**缩进规则**的程序不能通过解释，以此来强迫程序员养成良好的编程习惯，也方便所有人查找和阅读。并且Python语言利用缩进表示语句块的开始和结束（Off-side规则），而非使用花括号或者某种关键字。**增加缩进表示语句块的开始，而减少缩进则表示语句块的结束。**缩进成为了语法的一部分。例如`if`语句：

```
if age < 21:
    print("你不能买酒")                    #美國法律規定21歲以下的人不能購買酒
    print("不过你能买口香糖")
print("这句话处于if之外")
```

- ***注：**上述例子为Python 3.0以上版本的代码。*

根据PEP 8的规定，必须使用**4个空格**来表示每级缩进。**使用Tab字符和其它数目的空格虽然都可以被解释器识别，但不匹配编码规范。**支持Tab字符和其它数目的空格仅仅是为兼容很旧的Python程序和某些有问题的编辑程序。偏向使用Tab字符的程序员可以设置文本编辑器将Tab键转换为4个空格实现缩进而不致导致缩进错误。

标识符

- **_单下划线开头**：弱“内部使用”标识。对于“from M import *”，将不导入所有以下划线开头的对象，包括包、模块、成员。
- **单下划线结尾_**：为了避免与python关键字的命名冲突
- **__双下划线开头**：模块内的成员，表示私有成员，外部无法直接调用
- **双下划线开头双下划线结尾**：指那些包含在用户无法控制的名字空间中的“魔术”对象或属性，如类成员的name 、doc、init、import、file、等。**推荐永远不要将这样的命名方式应用于自己的变量或函数。**

语句和控制流

- `if`语句，当条件成立时运行语句块。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，依次处理迭代器中的每个元素。
- `while`语句，当条件为真时，循环运行语句块。
- `try`语句。与`except, finally, else`配合使用处理在程序运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，不运行任何操作。
- `assert` 语句。用于程序调适阶段时测试运行条件是否满足。
- `with` 语句。Python2.6以后定义的语法，在一个场景中运行语句块。比如，运行语句块前加锁，然后在语句块运行结束后释放锁。
- `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

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、取余数。`>>`, `<<` 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），可以获取完整列表的一部分。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

Python的函数支持递归、默认参数值、可变参数、闭包，但不支持函数重载。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；也可以按照命名参数形式调用，即param_name=value 形式的实参。在一个函数调用的实参表中，关键字引数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"集合(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的函数作为第一类对象，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***。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可变的对象，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不可变的对象，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型对象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

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(object):
    def eat(self, food):
        if food is not None:
            self.hungry=False
class User(object):
    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支持一些以“”**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。

数据类型与动态类型

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编程。

Python内置多种数据类型。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`	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"` `"""Spanningmultiplelines"""`
`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

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；第三方库`Sympy`用于支持数学符号运算。

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

Python拥有一个强大的标准库。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、日志（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。它们的功能覆盖科学计算、Web开发、数据库接口、图形系统多个领域。第三方模块可以使用Python或者C语言编写。SWIG, SIP常用于将C语言编写的程序库转化为Python模块。Boost C++ Libraries包含了一组库，Boost.Python，使得以Python或C++编写的程序能互相调用。Python常被用做其他语言与工具之间的"胶水"语言。

著名第三方库

Web框架

- **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。

- **Flask**

  轻量级的Web框架。

- Pyramid

  轻量，同时有可以规模化的Web框架，Pylon projects 的一部分。

- 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。

- Tornado

  一个轻量级的Web框架，内置非阻塞式服务器，而且速度相当快
- 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。
- aiohttp

  轻量级的Web框架，采用的是Python3的asyncio异步特性。
- **Pandas**

  非常高效的数据科学库

科学计算

- **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。
- **Pandas**

  用于数据分析、数据建模、数据可视化的第三方库。
- **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

- 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

- BeautifulSoup

  基于Python的HTML/XML解析器，简单易用。

- **gevent**

  python的一个高性能并发框架,使用了epoll事件监听、协程等机制将异步调用封装为同步调用。

- **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。目前已无维护，另一个第三方库Pillow实现了对PIL库的支持和维护。

- **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。

- Requests

  适合于人类使用的HTTP库，封装了许多繁琐的HTTP功能，极大地简化了HTTP请求所需要的代码量。

- scikit-learn

  机器学习第三方库，实现许多知名的机器学习算法。

- **TensorFlow**

  Google开发维护的开源机器学习库。

- **Keras**

  基于TensorFlow，Theano与CNTK)的高端神经网络API。

- SQLAlchemy

  关系型数据库的对象关系映射(ORM)工具

Python 3.0

Python的3.0版本，常被称为**Python 3000**，或简称**Py3k**。相对于Python的早期版本，这是一个较大的升级。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。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，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。

新的Python程序建议使用Python 3.0版本的语法。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, 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

Python 3.0的变化主要在以下几个方面：

- `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`"fish" + b"panda"`是错误的。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。例如给参数增加类型信息：

```
def sendMail(from_: str, to: str, title: str, body: str) -> bool:
    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。

- 其他变化详见相关文档。**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

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。

- **CPython，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**（原名JPython；Java语言实现的Python，现已正式发布）。Jython可以直接调用Java的各种函数库。

- **PyPy**（使用Python语言写的Python）
- **IronPython**（面向.NET和ECMA CLI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**（周蟒，支持使用繁/简中文语句编写程序的Python语言）

Python和其他语言比较起来怎么样

- 比Tcl强大。Python支持"大规模编程"，使其适宜于开发大型系统。
- 有着比Perl更简洁的语法和更简单的设计，这使得Python更具可读性、更易于维护，有助于减少程序bug。
- 比Java更简单、更易于使用。Python是一种脚本语言，Java从C++这样的系统语言中继承了许多语法和复杂性。
- **比C++更简单、更易于使用，但通常也不与C++竞争。因为Python作为脚本语言，常常扮演多种不同的角色。**
- 比Visual Basic更强大也更具备跨平台特性。由于Python是开源的，也就意味着它不可能被某一个公司所掌控。
- **比PHP更易懂并且用途更广。Python有时候用来构建Web站点，但是，它也广泛地应用于几乎每个计算机领域，从机器人到电影动画。**
- 比Ruby更成熟、语法更具可读性。与Ruby和Java不同的是，OOP对于Python是可选的：这意味着Python不会强制用户或项目选择OOP进行开发。
- 具备SmallTalk和Lisp等动态类型的特性，但是对开发者及定制系统的终端用户来说更简单，也更接近传统编程语言的语法。

开发环境

通用IDE / 文本编辑器

很多并非集成开发环境软件的文本编辑器，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 + pydev插件，目前对Python 3.X只支持到3.0
- emacs +插件
- NetBeans +插件
- SlickEdit
- TextMate
- Python Tools for Visual Studio
- **Visual Studio Code + 插件**
- Vim +插件
- **Sublime Text + 插件**
- EditPlus
- UltraEdit
- PSPad
- Editra，由Python开发的程序编辑器。
- Notepad++
- **JetBrains PyCharm**

专门为Python设计的IDE软件

适用于Python的集成开发环境（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、自动补全、智能感知等便利功能。由于Python的跨平台出身，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。

- **Anaconda**：适用于windows和Linux等多个平台，采用conda对其包管理，随软件打包了许多科学计算的第三方Python库。
- **Eric**：基于PyQt的自由软件。支持自动补全、智能感知、自动语法检查、工程管理、svn/mercurial集成、自动单元测试等功能，具有可扩展的插件系统，通过可选插件支持Git集成。调试功能与Visual Studio和Eclipse类似。当前版本为Eric6，可同时支持Python2.x和Python3.x，以及PyQt4和PyQt5。使用前需要先安装相应的Python和PyQt版本。
- **IDLE**：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**和**Komodo Edit**：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**：由JetBrains打造，该公司的Java IDE软件IntelliJ（此软件也有Python开发插件）拥有海量的用户；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，PyCharm也支持IronPython。PyCharm是商业软件，但也具有社区版和教育版。
- **PyScripter**：功能较全的开源IDE，使用Delphi开发。
- **PythonWin**：包含在pywin32内的编辑器，仅适用于Windows。
- **SPE**（Stani's Python Editor）：功能较多的免费软件，依赖wxPython。
- **Spyder**：开源的跨平台科学计算IDE。
- **Ulipad**：功能较全的免费软件，依赖wxPython。
- **WingIDE**：商业软件，有免费的Wing IDE 101，功能有限，适用于入门者教学。

使用Python编写的著名应用

- **Reddit** - 社交分享网站
- **Dropbox** - 文件分享服务
- **豆瓣网** - 图书、唱片、电影等文化产品的资料数据库网站
- **Django** - 鼓励快速开发的Web应用框架
- **Pylons** - Web应用框架
- **Zope** - 应用服务器
- **Plone** - 内容管理系统
- **Instagram** - 是一款免费提供在线图片及视频分享的社交应用软件,使用Django作为后台
- **TurboGears** - 另一个Web应用快速开发框架
- **Twisted** - Python的网络应用程序框架
- **Fabric** - 用于管理成百上千台Linux主机的程序库
- **Python Wikipedia Robot Framework** - MediaWiki的机器人程序
- **MoinMoinWiki** - Python写成的Wiki程序
- **Trac** - 使用Python编写的BUG管理系统
- **Mailman** - 使用Python编写的邮件列表软件
- **Mezzanine** - 基于Django编写的内容管理系统
- **Flask**- Python微Web框架
- **Webpy** - Python微Web框架
- **Bottle** - Python微Web框架
- **EVE** - 网络游戏EVE大量使用Python进行开发
- **Blender** - 使用Python作为建模工具与GUI语言的开源3D绘图软件

- **Inkscape** - 一个开源的SVG矢量图形编辑器。
- **知乎** - 一个问答网站
- 果**壳** - 一个泛科技主题网站
- **Odoo** - 仍在持续发展壮大且最受欢迎的ERP软件