**Python的功能包括：**

* **易于学习：**Python有几个关键字，结构简单，明确定义的语法。这让初学者可以快速上手语言。
* **易于阅读：** Python代码更加明确和看得见。
* **易于维护：**Python源代码是相当容易维护的。
* **广泛的标准库：**Python的库是非常便携的，在UNIX，Windows和Macintosh跨平台兼容。
* **交互式模式：**Python有一个交互动的模式，可支持互动测试和代码段的调试。
* **便携：**Python可以在多种硬件平台上运行，并且对所有平台相同的接口。
* **可扩展:** 可以添加低级别的模块到Python解释器。这些模块使程序员能够更有效地添加或定制自己的工具。
* **数据库:** Python提供了接口，支持所有主要的商业数据库。
* **GUI编程:** Python支持，可以创建并移植到许多系统调用GUI应用程序， 库和Windows系统，如：Windows MFC，Macintosh和Unix X窗口系统。
* **可扩展:** 对于大型程序，Python提供比shell脚本更加合理的结构和支持。

**除了上面提到的功能，Python还有很多的功能列表，这里列举一些如下：**

* 它支持函数和结构化以及OOP程序设计方法。
* 它可以被用来作为一种脚本语言或可以被编译成字节码用于构建大的应用程序。
* 它提供了非常高的水平动态数据类型和支持动态类型检查。
* 它支持自动垃圾收集。
* 它可与C，C ++，COM和ActiveX，CORBA和Java轻松集成。
* **\_\_future\_\_ 模块**
* Python 3.x引入一些Python2不兼容的关键字和函数，可以通过在 Python2 内置的模块 \_\_future\_\_ 导入。建议如果你想在代码中支持 Python3.x，使用\_\_future\_\_导入它。
* 例如，如果想在 Python2 中拥有 Python 3.x 整数的除法行为，添加下面的 import 语句
* from \_\_future\_\_ import division
* print()函数默认情况下在结束时会插入一个换行。在 Python2，它可以通过 '，' 在末行抑制输出换行。 在 Python3 则使用"end=' '" 附加空格，而不是换行
* print x, # Trailing comma suppresses newline in Python 2
* print(x, end=" ") # Appends a space instead of a newline in Python 3

range（类似python2的xrange），range（）对象支持切片

在 Python3，异常参数应以 'as' 关键字来声明。

except Myerror, err: # In Python2

except Myerror as err: #In Python 3

在Python 2，next() 作为生成器对象的一个方法是允许的。在 Python2，next()函数过度产生器对象遍历也是可以接受的。在Python3，但是，next()函数作为生成器方法来中止并引发AttributeError。

gen = (letter for letter in 'Hello World') # creates generator object

next(my\_generator) #allowed in Python 2 and Python 3

my\_generator.next() #allowed in Python 2. raises AttributeError in Python 3

随着 Python3 解释器，2to3.py 脚本将被通常安装在  tools/scripts 文件夹。 它读取 Python2.x 源代码，并应用了一系列的修复将它转变成有效的 Python3.x 代码。



GUI：图形用户界面

IDE：集成开发环境

下面是 Python 标识符命名的约定 -

* 类名称使用大写字母。所有其它标识符开始使用小写字母。
* 开头使用一个下划线的标识符表示该标识符是私有的。
* 开始以两个前导下划线的标识符表示强烈私有的标识符。
* 如果标识符使用两个下划线作为结束时，所述标识符是语言定义的特殊的名字。
* Python 的关键字如下列出。这些是保留字，不能把它们作为常量或变量或任何其他标识符名称。 所有Python的关键字仅包含小写字母。

|  |  |  |
| --- | --- | --- |
| and | exec | Not |
| as | finally | or |
| assert | for | pass |
| break | from | print |
| class | global | raise |
| continue | if | return |
| def | import | try |
| del | in | while |
| elif | is | with |
| else | lambda | yield |
| except |  |  |

代码块类和函数定义或流程控制

Python中所有连续不换行，同样数量的空格缩进将形成一个块。

在Python语句通常使用一个新行作为结束。但是，Python 允许使用续行字符(\)表示让行可以连续下去。例如-

total = item\_one + \

item\_two + \

item\_three

语句中包含 [], {}, 或() 括号内不需要使用续行字符。 例如−

days = ['Monday', 'Tuesday', 'Wednesday',

'Thursday', 'Friday']

