漫谈Python语言

Hoxide

March 20, 2005

内容提要 I

内容提要 II

什么是Python

Note:

Python语言可能是第一种即简单又功能强大的编程语言。它不仅适合于初学者，也适合于专业人员使用，更加重要的是，用Python编程是一种愉快的事。本身将帮助你学习这个奇妙的语言，并且向你展示如何即快捷又方便地完成任务——真正意义上“为编程问题提供的完美解决方案！”

– 《简明Python 教程》Swaroop, C. H. 著 沈洁元译

Python的特点

- Python 是荷兰人Guido van Rossum 1989年写的一个脚本语言。
- Python 是基于字节码的动态解释语言
- Python 支持OOP
- Python 是可以扩展和嵌入的
- 支持函数式程序设计
- Python 是开放的Python 本身源代码开放很多python 模块提供源代码
- 平台无关的多线程支持

Python的影响力

Note: (2005-3-19)

Python 2.4 获语言和开发环境类生产力奖。每个奖项产生一个Joit大奖和三个生产力奖。在语言和开发环境类奖项中Joit奖为Eclipse 3.0。
详情见Software Develop Online 网站。

Note: (Limodou评论)

仔细看一下此奖项的得奖者：
Jolt Winner: – Eclipse 3.0 (Eclipse Foundation) Productivity Winners: – IntelliJ IDEA 4.5 (JetBrains) – Python 2.4 (python.org) – REALbasic 5.5 for Windows Professional Edition (REAL Software)
把Python 和别的集成工具放在一起真是不合适，但从此也说明Python的实力。

The Zen of Python I

The Zen of Python
蟒禅
Beautiful is better than ugly.
美丽好过丑陋;
Explicit is better than implicit.
明显好过隐晦;
Simple is better than complex.
简单好过复合;
Complex is better than complicated.
复合好过复杂;
Flat is better than nested. 扁平好过嵌套;
Sparse is better than dense.
稀疏好过密集;
Readability counts.
可读性最重要;

The Zen of Python II

Special cases aren't special enough to break the rules.
即便实用性比纯度重要,
Although practicality beats purity.
但是!特殊案例不可特殊到打破规则;
Errors should never pass silently.
错误从来不会默默消失,
Unless explicitly silenced.
直到明确的让它闭嘴!
In the face of ambiguity,
面对模糊,
refuse the temptation to guess.
拒绝猜测的诱惑;
There should be one– and preferably only one –obvious way to do it.
应该有一个(宁愿只有一个)显而易见的解决方法;
Although that way may not be obvious at first unless you're Dutch.

The Zen of Python III

尽管刚开始方法不会是很明显,除非你是(Dutch)

Now is better than never. Although never is often better than *right* now.

即使永远不做比"立刻"做要"聪明", 但是！现在就做永远比不做要好;

If the implementation is hard to explain, it's a bad idea.

只要实现很难解释,那么它就不是一个好主意;

If the implementation is easy to explain, it may be a good idea.

只要实现很容易解释,那么这就是一个好主意;

Namespaces are one honking great idea

名称空间是一个正在召唤的绝妙想法

– let's do more of those!

–大家一起来实践这些规则吧!

– by Tim Peters

Code:

```
import os
from os.path import join, getsize
import sys

print sys.argv[1]
for root, dirs, files in os.walk (sys.argv[1]):
    if 'CVS' in dirs:
        fn = join (root + '\CVS', 'ROOT')
        print root + ' :', fn
        f = open (fn, 'r')
        r = f.read ()
        if r.startswith ('e:\cvsroot'):
            open (fn, 'w').write ('g:\cvsroot')
            f = open (fn, 'r')
            r = f.read ()
            print r
```

Code:

```
import os
from os.path import join, getsize
import sys

print sys.argv[1]
for root, dirs, files ∈ os.walk (sys.argv[1]):
    if 'CVS' ∈ dirs:
        fn ← join (root + '\CVS', 'ROOT')
        print root + '␣:', fn
        f ← open (fn, 'r')
        r ← f.read ()
        if r.startswith ('e:\cvsroot'):
            open (fn, 'w').write ('g:\cvsroot')
            f ← open (fn, 'r')
            r ← f.read ()
            print r
```

Python的发行版本

虽然通常所说的Python是指Guido van Rossum 用C语言实现的CPython, 但是其实还有一些其他的很有前途的Python实现. 大致分有以下一些版本:

- CPython — Guido van Rossum
- Jython — Jim Hugunin
- IronPython — Jim Hugunin
- Python.net — Jim Hugunin — MS

Note:

下面的代码都在CPython上运行通过

CPython

CPython也就是通常说的Python, 他最先由Guido van Rossum在90年代早期开发的. 目前Guido van Rossum仍然是主要的开发者. CPython的当前版本是2.4, 但是很多库仍然没有windows上的2.4发布版本, 因此如果在windows上我推荐2.3.4.

目前CPython在语法方面的改变已经非常少, 属于非常成熟的语言了. 它可以在各种Unix, Windows, Mac上运行

可从`http://www.python.org/`得到更多信息

Jython

Jython是Python的Java实现，可以让用户把Python源代码编译为Java的字节代码（byte code），而在任何Java虚拟机上运行。它能和Java无缝集成，通过Python可以完全存取所有的Java库，建立applets，并且和Java beans集成，并把Java类细分（subclass）。和Python一样，Jython可以交互使用，但是Java没有这个功能。

可以从Jython网站得到更多资料：`http://www.jython.org/`

IronPython

Note:

号称比CPython还快

IronPython的作者为Jim Hugunin，他同时也是Jython的作者，Jython是一个在Java平台上Python的实现。曾有开源社区的开发者认为.NET不是一个很好的动态语言实现平台。但是事实证明.NET可以很有效率的运行动态语言所生成的代码。IronPython这还仅仅是在.NET 1.1上就有很不错的成绩，据悉.NET 2.0将会强化对动态语言的支持，我们有理由期待Python在.NET平台上更好的表现。

Python.net

由Microsoft主导开发的.NET环境下的Python语言实现, 仍然由Jim Hugunin主持. 可以方便得访问CLR, 市IronPython的后继者. 目前版本1.0 – *beta*4, 他其实是Python2.4的一组扩展库, 简单的在Python2.4的安装目录中加入这些库就可以方便得使用Python.net.

Note:

Python2.4的windows版本由MSVC7.0编译.

Note:

例子

类型系统

- 基于对象模型, 一切皆对象, 强类型
- 包和库, 名字空间
- 多重继承, Mix-In(混入), 自省
- 丰富的基本类型, 包括数字, 序列, 映射, 文件和Python内部的一些类型
- 基于虚拟机技术, python字节码
- 垃圾收集
- 与$c/c++$的完美配合

基本类型— 数字

- 整型
  - Plain integers, -2147483648 到2147483647
  - Long integers, 无位数限制
  - Booleans, 逻辑型True False
- 浮点
- 复数

复合类型

- 简单序列
  - String
  - Unicode
  - Tuples (元组)
- 复合序列— List (列表)
- Mapping(映射) — Dictionary (字典)

例子 I

Note:

如何运行这些例子, 你可以简单的在命令行中输入'python' 来打开Python解释器. 如果是还有图形界面的标准Python IDE — IDLE, windows安装用户可以从开始→程序→Python→IDLE 打开.

例子 II

Code:

```
>>> t = (1, 2, 3)
>>> type(t)
< type 'tuple' >
>>> t[1]
2
>>> a = "hello"
>>> a
"hello"
>>> a[-1]
'o'
```

例子 III

Code:

```
>>> b = list(a)
>>> b
['h', 'e', 'l', 'l', 'o']
>>> b[4:]
['o']
>>> b[3:-1]
['l']
>>> b[1:2] = ['a','b','c']
>>> b
>>> ['h','a','b','c','l','l','o']
```

例子 IV

Code:

```
>>> d = dict(zip(b, range(len(b))))
>>> d
{'h': 0,' e': 1,' l': 3,' o': 4}
>>> d.keys()
['h',' e',' l',' o']
>>> d.items()
[('h', 0), ('e', 1), ('l', 3), ('o', 4)]
```

类 I

- class
- 多重继承
- Mix-in(混入)

类 II

Code:

```
>>> class c1 :
        def __init__(self) :
            self.message = 'init'
>>> i1 = c1()
>>> i1.message
'init'
```

Code:

```
>>> def __init__(self, message) :
        self.message = message
>>> c1.__init__ = __init__
>>> l2 = c1('hello')
>>> l2.message
'hello'
```

类 III

Code:

```
>>> class c2(c1):
        def __del__(self):
            print 'del c2 instance', self.message
>>> def a():
        ls = c2('hello')
>>> a()
del c2 instance hello
```

类 IV

Code:

```
>>> class c3 :
        def __call__(self) :
            print self.message
>>> l4 = c1('hello')
>>> c1.__bases__ += (c3,)
>>> l4()
hello
```

类 V

Code:

```
>>> class c4 :
        def __call__(self) :
            self.__class__ = c5
            print 'c4'
>>> class c5 :
        def __call__(self) :
            self.__class__ = c4
            print 'c5'
>>> l4 = c4()
>>> l4()
c4
>>> l4()
c5
```

关于类的魔法元素 I

Python中一般以“__”包围的变量都有特殊含义.

- *__doc__*
- *__class__*
- *__bases__*
- *__module__*

Note:

关于“__doc__”即Document String, 可以用来生成文档, 工具有python标准模块pydoc和非常流行的epydoc.

其他魔法元素 I

- *__main__*
- *__builtins__*

Code:

```
if __name__ == '__main__':
    print __name__
```

Code:

```
>>> __builtins__
<module '__builtin__' (built-in)>
```

还有很多带有“魔法”的元素, 可以参见Python Manual

基本结构

- while
- for
- break

例子 I

Code:

```
>>> l = range(10)
>>> while l :
        print l.pop(),
9 8 7 6 5 4 3 2 1 0
>>> for i in range(9, -1, -1) :
        print i,
9 8 7 6 5 4 3 2 1 0
```

生成器

Python支持生成器, 你可以把他简单得看成一个模拟iter(叠代器)的函数, 使用'yield'关键字.

Code:

```
>>> def g():
        l = range(10)
        while l:
            yield l.pop()
>>> l = g()
>>> for i in l:
        print i,
9 8 7 6 5 4 3 2 1 0
```

异常 I

- try
- except
- finally
- raise

Code:

```
>>> l = g()
>>> while True :
        try :
            print l.next(),
        except StopIteration :
            break
9 8 7 6 5 4 3 2 1 0
```

匿名函数 I

- lambda
- map
- reduce
- filter
- eval

匿名函数 II

Python部分支持函数编程的功能, 提供关键字'lambda'创建匿名函数. 结合map, reduce, filter和谓词(and, or) 等可以创建及其复杂的函数式的程序. 你可以混用一般的过程式风格和函数式风格, 选择更简洁的方式表达算法.

Code:

```
>>> map(str, range(10))
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
>>> from random import random as _random
>>> [i for i in range(10) if i > 5]
[6, 7, 8, 9]
>>> filter(lambda x: x > 0.5 and x, [_random() for i in range(10)])
[0.9938822073011242, 0.9420340085030887,
0.9951802578009955, 0.7993882930938341]
```

匿名函数 III

Code:

```
>>> def e():
        r = raw_input('pleaseinput : ')
        print eval(r)
    e()
pleaseinput : 1 + 2
3
```

Note:

http://wiki.woodpecker.org.cn/moin.cgi/PyPorgramGames 上的游戏收集中有hoxide写的解24点问题和爱因司坦难题的程序, 使用了很多函数式的方法, 是不错的例子.

了解类型 I

- *type* — 获取类型
- *dir* — 列出对象所含的属性和方法
- *traceback* — 访问调用栈

了解类型 II

Code:

```
>>> dir()
['__builtins__', '__doc__', '__name__']
```

了解类型 III

Code:

```
>>> from traceback import extract_stack
>>> def a() :
        b()
>>> def b() :
        fname = extract_stack()[-2][2]
        print 'caller :', fname
        print eval(fname)
>>> a()
caller : a
< function at 0x00A56370 >
```

网络应用 I

Note:

Python中最完整最强大的网络应用库是Twisted, 它是一个网络应用框架, 在这个框架不仅包含了众多已有协议的服务端和客户端实现. 而且你也可以在Twisted框架下快速的构建各种网络应用服务端和客户端, Twisted还提供完整的支持工具集, 例如deamon管理,日志系统等.

- Twisted — 网络应用框架
- libgmail — Binding for Google's Gmail service, includes FTP, POP, SMTP & archive functions. Gmail的软件库

网络应用— WEB应用

- MoinMoin 简单的, 功能全面的可扩展wiki 软件包
- PyBlosxom 可扩展的Blog 服务器软件包
- mod_python — Apache的Python接口.
- SOAPpy SOAP(Simple Object Access Protocol)的接口, 服务器和客户端库.
- pyGoogle 使用SOAP, Google的Python接口.
- CherryPy — 面向对象的Web开发框架, 美味樱桃.

应用服务器 I

- ZOPE — Python编写的应用服务器, 可以和domino竞争.
- Plone — ZOPE上的内容管理框架.

什么是应用服务器? 就是将应用都集中在一起的服务器, ZOPE是软件部分他不仅仅是一个Web服务器, 还提供其上的各种服务, 服务以插件的方式存在.

例如最有名的插件就是Plone, (所以你也许会听到Zope/Plone的叫法).

PEAK

PEAK is the “Python Enterprise Application Kit”. If you develop “enterprise” applications with Python, or indeed almost any sort of application with Python, PEAK may help you do it faster, easier, on a larger scale, and with fewer defects than ever before. The key is component-based development, on a reliable infrastructure.

PEAK tools can be used with other “Python Enterprise” frameworks such as Zope, Twisted, and the Python DBAPI to construct web-based, GUI, or command-line applications, interacting with any kind of storage, or with no storage at all. Whatever the application type, PEAK can help you put it together. Package Features

目前PEAK的版本是0.5α.

文本处理 I

Note:

Python的标准库中已经含有了非常丰富的文本处理工具, 常规的re(正则表达式)就不必说了, 还有一些html xml解析和字符转换模块等模块.

- elementtree — ElementTree 轻量级XML对象模型接口, 有一个c写的加速版本cElementtree.
- 4suite — 强大完整的XML工具包
- jotweb, SimpleTAL — web模版框架

图形用户界面(GUI)

- TK Python的标准GUI包, 非常简洁
- wxPython 夸平台的图形开发框架, Newedit, Feednow 都使用这个框架
- pyQT 非常有前途的, 不过Windows上的许可证有点问题
- py-GTK 很多linux上的图形化配置工具就是用它写的.

数据库

Note:

自从Python的数据库接口(DB API2.0)标准, 发布以来, 众多Python数据库接口库都做了调整, 以符合这个标准, 因此在Python中调用各种数据库接口的方法都是几乎相同的.

- pySqlite — 模块化的Sqlite接口, 轻量级数据库
- pymssql — MS-SQL的接口
- pyPgSQL — PostgreSQL的接口
- pyDB2 — DB2接口
- cx_Oracle, adodbapi, 等等……
- PDO — Python Database Objects通用数据库接口, 可以支持OBDC
- PLPython — Python的PostgreSQL过程语言接口
- ZODB3 — Zope Object Database, 面向对象数据库, ZOPE的核心

多媒体

- PySonic 和fmod 音频播放模块, mp3, wma, wav — 天成制作的NewEdit插件使用了这两个模块
- PIL — Python Imaging Library 图形库
- MayaVi — The MayaVi Data Visualizer
- PyMedia — PyMedia is a module/library for manipulating mp3, avi, ogg, divx, mpeg, dvd media files
- ZOE — A trivial OpenGL rendering engine written entirely in Python

开发工具

Note:

Python发行包中本身就含有很多有用的开发工具, distutile组件专门用来管理python软件包.

- epydoc — API Documentation Generation Tool, 用于生成Python库的文档.很多库使用这种方法.
- pyUnit — 单元测试工具
- cx_Freeze — cross platform method of freezing Python scripts into executables
- py2exe — 将Python程序打包成exe文件的工具

其他

- pyWin — Windows上专用的工具包合集, 可以方便得访问windows上的各种组建, 如调用dll, 访问com等
- pyGAME — SDL的简单封装, 游戏开发库
- pycrypto — Cryptographic modules for Python.
- TwistedSNMP, yapsnmp — SNMP组件

国外Python成功应用 I

- BitTorrent — BT下载系统
- Google — 数据库与文本处理
- Disney — 网络游戏Disney Online's Toontown
- SciPy — 科学计算
- ZOPE — Python编写的应用服务器, 可以和domino竞争.

国内Python应用项目

- OpenUSS, Compass, Otter — 基于Twisted的网络集群管理软件. Woodpecker的主要项目. 开发:river, 项目管理:HD
- Newedit — 基于wxPython的可扩展编辑器, 主要作为Python编辑器– Limodou
- Feednow — 基于wxPython的rss阅读器– Dream Y
- douban — Python编写的书评网站. 此为商业应用.
- ZqLib — Python代码收集项目– Zoomq
- WeKnow — 知识管理系统– Zoomq

个人项目

- babywork — 基于mod_python的文章管理系统
- PureProlog — Prolog运行环境

Python人物 I

- Guido van Rossum — Python的主要作者, 数学家
- David Mertz 是非常有名的Python 专家, 在IBM开发者园地里面有很多他的文章
- Tim Peters — Python gurus who hangs out on news:comp.lang.python .

Python国内人物 I

- HD — Sina邮件存储部门主管, Python先行者
- limodou — Python现行者, 国内最活跃的Pythoner, `www.donews.net/limodou`上有丰富的Python文档
- Zoomq — Pythoner, 思想活跃, 啄木鸟社区管理员

Note:

中国Python用户正在不断壮大,你可以从啄木鸟社区的Python行者堂结识更多的Pythoner. 另外可以通过QQ群:1073669和他们聊天.

集成开发环境IDE

- IDLE — Python的标准的IDE
- emacs — Guide推荐的IDE, 强大, 便捷
- vi — 同样强大, 选emacs还是vi, 完全是个习惯问题
- eclipse — Java开发的IDE, 一个完整强大的开发平台
- UltraEdit — 比较简单, 但功能完备的编辑器,可以自己做语法高亮

名家作品 I

- Python Cook Book — 代码片段的收集和解释
- Dive in Python — 深入Python
- Python Manuals — 万变不离其宗, Manuals永远是根本, Guido执笔
- Text Processing in Python — Python中的文本处理, 由David Mertz执笔

Note:

Python Cook Book, Dive in Python 和TPiP 在啄木鸟社区都有翻译计划, CookBook的进度最快, 而且比较稳定.

教程

[Python 手册]
Guido van Rossum, Fred L. Drake, Jr., editor
刘鑫译2005-02-21 更新http://www.woodpecker.org.cn/share/doc/abyteofpython_cn/chinese/index.html

[简明Python教程]
Swaroop, C. H. 2002-11-10
沈洁元译http://www.woodpecker.org.cn/share/doc/abyteofpython_cn/chinese/index.html

Python国外资源

- `http://www.python.org` — Python的官方站
- `http://www.activestate.com/Products/ActivePython/` — Active State 维护的Python版本
- Python-Dev — Python语言开发的官方列表
- comp.lang.python — Python新闻组和Python-user列表同步

Note:

Python的国外资源非常丰富, 邮件列表更是满天飞, 这里只列出官方的, 比较重要的站点. 你可以根据自己的情况订阅列表, 例如mod_python的twisted的等等.

Python网络资源

- Python中文社区— `http://python.cn/` 有一些资源下载, 最重要的是提供邮件列表服务.
- 中蟒— `http://www.chinesepython.org` 用中文进行python编程
- 啄木鸟社区(WoodPecker) — `www.woodpecker.org.cn` 由HD提供主机, 目前最活跃的Python社区, 网站提供wiki, trac, blog等多种服务.

Python邮件列表 I

- Python中文python-chinese@lists.python.cn — 最活跃的Python邮件列表
- 水木清华BBS Python信区— 目前好像无法在校外访问.
- Google上的Python中文信区(python-cn@googlegroups.com — 用户和Python Chinese基本重叠

Python的纸质书籍

Python与Tkinter编程 (美) John E.Grayson著国防工业出版社2002 TP312PY/2

Python编程金典 (美) H. M. Deitel ... [等] 著清华大学出版社2003 TP312PY/3

Python编程基础 肖建, 林海波等编著清华大学出版社2003 TP312PY/4

谢谢!