



Branch: master opensource / libs / python.wiki

Find fileCopy path

 **programthink** 修改链接文字 3d58249 Feb 25, 2017

2 contributors 

2061 lines (1246 sloc) 52.7 KB

# Python 开源库及示例代码

## Table of Contents

- 说明
- 1 算法
  - 1.1 字符串
    - 1.1.1 正则表达式
    - 1.1.2 字符集
    - 1.1.3 （其它）
  - 1.2 编码 & 解码
    - 1.2.1 base64
    - 1.2.2 UUencode
    - 1.2.3 BinHex
  - 1.3 数学类
  - 1.4 容器
- 2 跨语言编程
  - 2.1 整合 C & C++
  - 2.2 整合 JVM 平台
  - 2.3 整合 dotNet 平台
  - 2.4 整合 Go
  - 2.5 整合 Objective-C
- 3 操作系统
  - 3.1 文件和目录操作
  - 3.2 线程
  - 3.3 进程
  - 3.4 本地进程间通信（IPC）
  - 3.5 Linux & Unix 系统相关
  - 3.6 Windows 系统相关
  - 3.7 程序打包
- 4 Web
  - 4.1 HTTP Client
  - 4.2 HTTP Server
  - 4.3 Web 开发框架
  - 4.4 Web前端 & JS整合
  - 4.5 浏览器整合
  - 4.6 （其它）
- 5 网络
  - 5.1 链路层 & 网络层

- └ 5.2 传输层
- └ 5.3 标准的应用层
  - └ 5.3.1 综合性的库
  - └ 5.3.2 HTTP
  - └ 5.3.3 文件传输
  - └ 5.3.4 电子邮件
  - └ 5.3.5 即时通讯
  - └ 5.3.6 远程控制
  - └ 5.3.7 （其它）
- └ 5.4 自定义的应用层
- └ 5.5 网络库、框架、中间件
- └ 5.6 云计算
- 6 数据库**
  - └ 6.1 数据库中间件
    - └ 6.1.1 ODBC
    - └ 6.1.2 JDBC
    - └ 6.1.3 ADO & ADO.NET
  - └ 6.2 特定数据库
    - └ 6.2.1 MySQL
    - └ 6.2.2 PostgreSQL
    - └ 6.2.3 Oracle
    - └ 6.2.4 MS SQL Server
    - └ 6.2.5 IBM DB2
    - └ 6.2.6 SQLite
    - └ 6.2.7 MongoDB
    - └ 6.2.8 Apache HBase
    - └ 6.2.9 Redis
    - └ 6.2.10 LevelDB
    - └ 6.2.11 Berkeley DB
  - └ 6.3 ORM（Object-Relational Mapping）
- 7 GUI**
  - └ 7.1 GUI 框架
    - └ 7.1.1 基于 Tk
    - └ 7.1.2 基于 wxWidgets
    - └ 7.1.3 基于 GTK+
    - └ 7.1.4 基于 Qt
    - └ 7.1.5 基于 FLTK
    - └ 7.1.6 基于 Windows 平台
    - └ 7.1.7 基于 JVM 平台
    - └ 7.1.8 （其它）
  - └ 7.2 图表 & 报表
- 8 信息安全**
  - └ 8.1 密码学
  - └ 8.2 访问控制
- 9 处理文件格式**
  - └ 9.1 结构化数据格式
    - └ 9.1.1 CSV
    - └ 9.1.2 JSON
    - └ 9.1.3 YAML
  - └ 9.2 压缩文件 & 打包文件
    - └ 9.2.1 zip
    - └ 9.2.2 bzip2（bz2）
    - └ 9.2.3 gzip（gz）
    - └ 9.2.4 tar

- └ 9.2.5 7zip (7z)
- └ 9.2.6 rar
- └ 9.2.7 msi
- └ 9.3 标记语言
  - └ 9.3.1 XML
  - └ 9.3.2 HTML
- └ 9.4 PDF
- └ 9.5 MS Office 文档
  - └ 9.5.1 Word (doc、docx)
  - └ 9.5.2 Excel (xls、xlsx)
  - └ 9.5.3 Power Point (ppt、pptx)
- └ 9.6 RTF
- └ 9.7 CHM
- 10 图像**
  - └ 10.1 图像处理
  - └ 10.2 图像格式转换
  - └ 10.3 图像渲染
- 11 游戏**
  - └ 11.1 综合性的游戏引擎
  - └ 11.2 3D 渲染引擎
- 12 数值计算 & 科学计算**
- 13 (其它)**

---

## 说明

---

本页面汇总俺收集的各种 Python 代码库，不定期更新。

本页面列出的各种 Python 库/模块，如果注明了官网的网址，说明这个库是第三方的；否则就是 Python 语言内置的标准库。

如果你发现本页面的开源库有错漏之处，非常欢迎给俺提供反馈——有 GitHub 帐号的同学，可以[给俺发 issue](#)；没帐号的同学，可以去[俺博客](#)留言。

---

## 1 算法

---

### 1.1 字符串

---

#### 1.1.1 正则表达式

**re**

【标准库】

提供基于正则的匹配和替换。

#### 1.1.2 字符集

**chardet**

Home: <https://github.com/erikrose/chardet>

chardet 可以猜测任意一段文本的字符集编码。对于编码类型未知的文本，它会很有用。

chardet 既可以作为模块来使用，也可以作为命令行工具来使用。

代码示例

```
import chardet
print(chardet.detect(bytes))
```

### 1.1.3 （其它）

#### StringIO & cStringIO

##### 【标准库】

以读写文件的方式来操作字符串（有点类似于内存文件）。

cStringIO 是 C 语言实现的，性能更高；而 StringIO 是 Python 实现的，提供 Unicode 兼容性。

#### difflib

##### 【标准库】

可以对两个字符串进行“按行”比较，其功能类似于命令行的 diff。

另外还支持“最佳匹配”功能——对给定的字符串 s 和字符串列表 l，在 l 里面找到最接近 s 的字符串。

## 1.2 编码 & 解码

---

### 1.2.1 base64

Base64 是一组编码算法的总称。用于把二进制数据编码为文本。

#### base64

##### 【标准库】

提供 Base16、Base32、Base64 格式的编码和解码。

### 1.2.2 UUencode

UUencode 出现于早期的 Unix 系统。用于把二进制编码为文本，以便通过邮件系统发送。

#### uu

##### 【标准库】

提供 UUencode 格式的编码和解码。

### 1.2.3 BinHex

BinHex 起先用于 Mac OS 系统，类似于 UUencode。

#### binhex

##### 【标准库】

提供 BinHex 格式的编码和解码。

## 1.3 数学类

---

#### math

##### 【标准库】

顾名思义，这个标准库封装了常用的数学函数（开方、指数、对数、三角函数.....）。

#### random

##### 【标准库】

顾名思义，这个标准库是用来进行随机数生成滴。

代码示例——生成 0-100 的随机数

```
import random
random.seed()
random.randint(0, 100)
```

## fractions

【标准库】

封装了跟有理数（分数）相关的运算

## 1.4 容器

---

### pytrie

Home: <https://github.com/google/pytrie>

这是 Google 实现的 [trie](#)（前缀树/字典树）封装库。

## 2 跨语言编程

---

Python 可以很容易地跟其它编程语言整合。整合之后，就可以在 Python 代码中使用其它编程语言的函数、模块、库，非常爽！

### 2.1 整合 C & C++

---

#### ctypes

ctypes 在 Python 2.5 版本加入到标准库中。

通过它，你可以很方便地调用 C/C++ 动态库导出的函数，可以在 Python 中使用各种 C/C++ 的数据类型（包括“指针”和“引用”）。

代码示例——调用 Linux/Unix 系统的标准 C 函数，获取当前时间

```
from ctypes import *

libc = CDLL("libc.so.6")
time = libc.time(None)
```

代码示例——调用 Windows 系统的 API，弹出消息提示框

```
from ctypes import c_int, WINFUNCTYPE, windll
from ctypes.wintypes import HWND, LPCSTR, UINT

prototype = WINFUNCTYPE(c_int, HWND, LPCSTR, LPCSTR, UINT)
paramflags = (1, "hwnd", 0), (1, "text", "Hi"), (1, "caption", None), (1, "flags", 0)
MessageBox = prototype(("MessageBoxA", windll.user32), paramflags)
MessageBox(text="Hello, world", flags=2)
```

#### SWIG（Simplified Wrapper and Interface Generator）

Home: <http://swig.org/>

Links: [Wikipedia](#)

这是一个很老牌的、有名气的工具，它可以把多种语言（Java、Python、C#、Ruby、PHP、Perl、Lua、Go ...）整合到 C/C++ 中。

#### Cython

Home: <http://cython.org/>

这个工具可以让你用 Python 的语法写扩展模块的代码，然后它帮你把 Python 代码编译为本地动态库（机器码）。

用它编译出来的扩展模块，其性能跟 C/C++ 编写的扩展模块相当。

## 2.2 整合 JVM 平台

---

### Jython

Home: <http://www.jython.org/>

Links: [Wikipedia 维基百科](#)

通过 Jython 可以让 Python 代码运行在 JVM 上，并且可以调用其它的 JVM 语言的代码（比如 Java、Scala）

## 2.3 整合 dotNet 平台

---

### IronPython

Home: <http://ironpython.net/>

Links: [Wikipedia 维基百科](#)

通过 IronPython 可以让 Python 代码运行在 dotNET 平台上，并且可以调用其它的 dotNET 语言的代码（C#、F#、VB.Net ...）

## 2.4 整合 Go

---

### gopy

Home: <https://github.com/go-python/gopy>

gopy 可以把 Go 源代码编译为 Python 的一个 module。

它提供了两种方式（命令行、Python 库）来实现：Go 源码编译为 Python 模块。

## 2.5 整合 Objective-C

---

### PyObjC

Home: <http://pyobjc.sourceforge.net/>

这是用 Python 封装 Mac OS X 上的 Objective-C 库。

# 3 操作系统

---

## 3.1 文件和目录操作

---

### os

#### 【标准库】

这是非常基本的标准库，提供了常见的操作系统相关功能，很多功能是针对文件系统。

### shutil

#### 【标准库】

相对于 os 而言，shutil 提供了一些比较高级的文件和目录操作（目录递归复制、目录递归删除、目录压缩打包...）

代码示例——递归删除某个目录

```
import shutil
```

```
shutil.rmtree(yyyy)
```

## glob

### 【标准库】

用于查找文件，【支持通配符】（\* 和 ?）

代码示例——获取当前目录所有 txt 文件

```
import glob
for file in glob.glob("./*.txt") :
    print(file)
```

## fnmatch

### 【标准库】

用于匹配文件名（支持通配符，类似上面的 glob）

代码示例——列出当前目录所有 txt 文件

```
import os, fnmatch

for file in os.listdir("."):
    if fnmatch.fnmatch(file, "*.txt"):
        print(file)
```

## tempfile

### 【标准库】

使用它可以安全地生成临时文件或临时目录。

## 3.2 线程

### threading

#### 【标准库】

提供了比较高层的线程封装 API。它本身包含了线程同步/互斥的机制。

代码示例——基于“函数”的线程

```
import threading
import time

def my_thread() :
    print("Thread started!")
    time.sleep(3)
    print("Thread finished!")

threading.Thread(target=my_thread).start()
```

代码示例——基于“类”的线程

```
import threading
import time
from __future__ import print_function

class MyThread(threading.Thread) :
    def run(self) :
        print("{} started!".format(self.getName()))
        time.sleep(3)
```

```
print("{} finished!".format(self.getName()))

if __name__ == "__main__" :
    for n in range(10) :
        mythread = MyThread(name = "Thread-{}".format(n + 1))
        mythread.start()
        time.sleep(1)
```

## 3.3 进程

### subprocess

#### 【标准库】

用于进程管理，可以启动子进程，通过标准输入输出跟子进程交互。

代码示例——启动命令行进程，并获取该进程的标准输出

```
import subprocess
output = subprocess.check_output(["dir"]) # 获取当前目录的内容
output = subprocess.check_output(["netstat", "-an"]) # 获取当前网络链接
```

### multiprocessing

#### 【标准库】

它是 2.6 版本加入到标准库的，其 API 接口的风格类似于 threading 模块。

它本身包含了进程同步/互斥的机制。

代码示例——利用其 Lock 机制，确保多个子进程的标准输出不会混杂（每次只有一个进程调用 print）。

```
from multiprocessing import Process, Lock

def f(lock, n) :
    lock.acquire()
    print("hello world %d" % n)
    lock.release()

if __name__ == "__main__" :
    lock = Lock()
    for num in range(10):
        Process(target=f, args=(lock, num)).start()
```

## 3.4 本地进程间通信（IPC）

### mmap

#### 【标准库】

提供了内存映射文件的支持。

代码示例——利用 mmap 在父子进程间交换数据

```
import os
import mmap

map = mmap.mmap(-1, 13)
map.write("Hello, world")

pid = os.fork()
if pid == 0 : # 子进程
    map.seek(0)
    print(map.readline())
    map.close()
```



## signal

### 【标准库】

用于进程信号处理的标准库（主要用于 Linux & UNIX 系统）。

## 3.5 Linux & Unix 系统相关

---

## syslog

### 【标准库】

通过它可以很方便地跟 POSIX 的 syslog 服务进行交互。

## 3.6 Windows 系统相关

---

## PyWin32

Home: <http://python.net/crew/mhammond/win32/>

这个第三方库封装了 Windows API 及 COM API。通过它可以方便地用 Python 进行 Windows 编程（调用 COM 组件、编写 Windows 服务、等）。

## 3.7 程序打包

---

## PyInstaller

Home: <http://www.pyinstaller.org/>

PyInstaller 可以把你的 Python 代码制作成独立运行的程序（不依赖 Python 环境就可以运行）。

该工具支持多种操作系统，包括：Windows、Linux、Mac OS X、Solaris、AIX、等。

## py2exe

Home: <http://www.py2exe.org/>

Links: [Wikipedia](#)

py2exe 的功能类似 PyInstaller，但只支持 Windows 平台。

## py2app

Home: <https://bitbucket.org/ronaldoussoren/py2app>

它很类似于 py2exe，差别在于 py2exe 支持 Windows 平台，而 py2app 支持 Mac OS X 平台。

## EasyInstall & Setuptools

Home: <https://pypi.python.org/pypi/setuptools>

这套工具可以帮助你进行第三方库的管理（下载、编译、安装、升级、卸载）

# 4 Web

---

## 4.1 HTTP Client

---

## httplib & httplib2 & http.request & urllib.parse

### 【标准库】

这几个库可以进行各种 HTTP 客户端请求（GET、POST、等）。

Python2 的模块名是 httplib 和 httplib2；到 Python3，模块名改为 http.request 和 urllib.parse

代码示例——读取指定 URL 的网页内容

```
import urllib
handle = urllib.urlopen("http://www.google.com")
page = handle.read()
handle.close()
```

## Requests

Home: <http://www.python-requests.org/>

这是一个用起来很优雅的库，如其名，封装了 HTTP 请求的功能。

代码示例

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...}'
>>> r.json()
{'u'private_gists': 419, u'total_private_repos': 77, ...}
```

## 4.2 HTTP Server

### SimpleHTTPServer & http.server

【标准库】

提供轻量级 HTTP Server 的标准库。

Python2 的模块名叫 SimpleHTTPServer；到 Python3 模块名改为 http.server

代码示例——一个极简单的 HTTP 服务

```
import SocketServer
import SimpleHTTPServer

PORT = 8080
Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
httpd = SocketServer.TCPServer(("", PORT), Handler)
print("serving at port %d" % PORT)
httpd.serve_forever()
```

## 4.3 Web 开发框架

（Python 的 Web 框架数不胜数，俺只挑选几个代表性的）

### Django

Home: <https://www.djangoproject.com/>

Links: [Wikipedia 维基百科](#)

在 Python 社区，Django 是目前最有影响力的 Web 开发框架。该框架很重型，内置了 Web 服务端开发常用的组件（比如：ORM、用户管理）。

Django 应用范围很广，比如 Google 的 Web 开发平台 GAE 就支持它。

而且它完全支持前面提到的 Jython 运行环境，可以运行在任何 J2EE 服务器上。

### TurboGears

Home: <http://www.turbogears.org/>

Links: [Wikipedia](#) 维基百科

又一个重型的 Web 开发框架，名气仅次于 Django。

它跟 Django 一样，都是“Full-Stack Frameworks”。

### CherryPy

Home: <http://www.cherrypy.org/>

Links: [Wikipedia](#)

轻量级的 Web 框架。某些 Web 框架（比如前面提到的 TurboGears）使用它作为底层。

代码示例——Hello world

```
import cherrypy

class HelloWorld(object):
    def index(self):
        return "Hello World!"
    index.exposed = True

cherrypy.quickstart(HelloWorld())
```

### web.py

Home: <http://webpy.org/>

与前两个（Django、TurboGears）不同，这是一个轻量级的框架。甚至被称为“It's the anti-framework framework.”

其作者是大名鼎鼎的黑客 [Aaron Swartz](#)。（俺在[某篇博文](#)中悼念过他）。

当年 Aaron Swartz 用 web.py 来搭建同样大名鼎鼎的网站 [reddit](#)（该网站是 Web 2.0 的标杆）。

代码示例——Hello world

```
import web

urls = (
    "/", "index"
)

class index:
    def GET(self):
        return "Hello, world!"

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

## 4.4 Web 前端 & JS 整合

### Pyjamas & pyjs

Home: <http://pyjs.org/>

这是从 GWT（Google Web Toolkit）移植的第三方库。提供了 Python 到 JS 的编译，AJAX 框架等功能。

Pyjamas 甚至能用来开发桌面 GUI 应用。

### pyjaco

Home: <https://github.com/chrivers/pyjaco>

这也是一个 Python 到 JavaScript 的编译工具。

## 4.5 浏览器整合

---

### webbrowser

#### 【标准库】

操纵当前系统的默认浏览器，访问指定 URL 的页面。

代码示例——用默认浏览器打开 Google 主页

```
import webbrowser
webbrowser.open("http://www.google.com")
```

### pyv8

Home: <https://pypi.python.org/pypi/PyV8>

v8 是 Google 开发的 JavaScript 解释引擎。这是对 v8 引擎的 Python 封装。

代码示例

```
import PyV8

ctxt1 = PyV8.JSContext()
ctxt1.enter()
ctxt1.eval("1+2") # 对 JS 表达式求值

class Global(PyV8.JSClass): # 定义一个兼容 JS 的类
    def hello(self):
        print("Hello, world")

ctxt2 = PyV8.JSContext(Global()) # 创建一个 JS 上下文，传入 Global 类的对象
ctxt2.enter()
ctxt2.eval("hello()") # 调用 hello() 函数
```

### PyWebKitGtk

Home: <https://github.com/jmalonzo/pywebkitgtk>

WebKitGtk 是一个基于 WebKit 的 Web 渲染引擎。

PyWebKitGtk 则提供了对 WebKitGtk 的 Python 封装。

## 4.6 （其它）

---

### pywebsocket

Home: <https://github.com/google/pywebsocket>

这是 Google 提供的 WebSocket 服务端。

该项目包含一个可独立运行的 server 以及一个 Apache 扩展模块（mod\_pywebsocket）。

## 5 网络

---

### 5.1 链路层 & 网络层

---

#### Scapy

Home: <http://www.secdev.org/projects/scapy/>

Links: [Wikipedia](#)

这是一个底层的网络库，可以在不同协议层次构造网络数据包（包括链路层、网络层、传输层），还支持 Sniffer 抓包。

搞网络安全的网友应该会喜欢这个库。

代码示例

```
# 传统的 ping 扫描（网络层）
ans,unans = sr(IP(dst="192.168.1.1-254")/ICMP())

# 局域网内的 ARP 扫描（链路层）
ans,unans = srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.1.0/24"), timeout=2)
```

## 5.2 传输层

---

### socket

Python 标准库很早就提供了对 socket 编程的支持。

这个标准库是对伯克利套接字进行简单的封装，其 API 基本上跟 BSD SOCKET 一一对应。

### asyncore

这个标准库提供了异步 SOCKET 的支持。

### asynchat

这个标准库基于上述的 asyncore，提供更高层的 API，简化异步通讯编程。

## 5.3 标准的应用层

---

### 5.3.1 综合性的库

#### PycURL

Home: <http://pycurl.sourceforge.net/>

cURL 是一个功能很强的网络库/网络工具，支持 N 多应用层协议。俺在前几年写过一篇博文推荐它（在“[这里](#)”）。

看名称就能猜到——PycURL 是 cURL 的 Python 封装。

代码示例——发起 HTTP GET 请求

```
import pycurl
try :
    from io import BytesIO
except ImportError :
    from StringIO import StringIO as BytesIO

buffer = BytesIO()
curl = pycurl.Curl()
curl.setopt(curl.URL, "http://pycurl.sourceforge.net/")
curl.setopt(curl.WRITEDATA, buffer)
curl.perform()
curl.close()
body = buffer.getvalue()
```

### 5.3.2 HTTP

（关于“HTTP 协议”，请参见另一个大类：“Web”）

### 5.3.3 文件传输

#### ftplib

【标准库】

封装 FTP（File Transfer Protocol）协议

代码示例——列出 FTP 服务器上某目录的内容

```
from ftplib import FTP

ftp = FTP("ftp.debian.org") # 连接服务器（如果不指定端口号，则用默认端口号 21）
ftp.login()                 # 登录（如果不指定用户名和密码，则用匿名登录）
ftp.cwd("debian")           # 切换到 "debian" 目录
ftp.retrlines("LIST")       # 列出当前目录的内容
ftp.quit()
```

### pysftp

Home: <https://bitbucket.org/dundeemt/pysftp>

封装 SFTP 协议，依赖于 ssh.py

代码示例——简单的上传/下载

```
import pysftp

with pysftp.Connection("hostxxx", username="userxxx", password="xxxxxx") as sftp :
    with sftp.cd("public")           # 服务端当前目录切换到 public
        sftp.put("/my/local/filename") # 上传某个本地文件到服务端的 public 目录
    sftp.get_r("myfiles", "/local")   # 递归复制某个服务端的目录到本地
```

## 5.3.4 电子邮件

### smtplib

【标准库】

封装 SMTP（Simple Mail Transfer Protocol）协议

### imaplib

【标准库】

封装 IMAP（Internet Message Access Protocol）协议

### poplib

【标准库】

封装 POP3（Post Office Protocol v3）协议

## 5.3.5 即时通讯

### jabber.py

Home: <http://jabberpy.sourceforge.net/>

Jabber（又称 XMPP）是IM（即时通信）协议的标准。这是用 Python 封装的第三方库。

### ircilib

Home: <https://bitbucket.org/jaraco/irc>

IRC 是 Internet Relay Chat 的缩写。这是用 Python 封装的第三方库。

## 5.3.6 远程控制

### telnetlib

【标准库】

封装 telnet 协议

代码示例——使用 telnet 登录到某个主机并执行简单命令

```
import telnetlib
import getpass

host = raw_input("Enter remote host: ")
user = raw_input("Enter your remote account: ")
password = getpass.getpass()

tn = telnetlib.Telnet(host)

tn.read_until("login: ")
tn.write(user + "\n")

if password:
    tn.read_until("Password: ")
    tn.write(password + "\n")

tn.write("ls\n")
tn.write("exit\n")

print tn.read_all()
```

## rdpy

Home: <https://github.com/citronneur/rdpy>

纯 Python 实现的 RDP（微软远程桌面协议）和 VNC（Virtual Network Computing）客户端，依赖于 Twisted 库

## 5.3.7 （其它）

### urlparse

【标准库】

用于解析 URL，提取各个部分的内容。从 Python 2.5 版本开始加入到标准库中，从 Python 2.7 开始支持包含 IPv6 的 URL

## 5.4 自定义的应用层

---

### Protocol Buffers

Home: <https://developers.google.com/protocol-buffers/>

Links: [Wikipedia](#)

这是 Google 开发的一个跨语言的库，用于网络传输业务数据时的“编码/解码”。

其优点是：跨多种语言、高性能、向前兼容、向后兼容。俺前几年写过一篇博文推荐 protobuf（在[这里](#)）。

作为 Protocol Buffers 的发明者，Google 默认实现了三种编程语言（C++、Java、Python）对它的支持。

### Apache Thrift

Home: <https://thrift.apache.org/>

Links: [Wikipedia](#)

来自于 Apache 社区，提供了一种跨语言的通讯机制。

程序员通过 Thrift 的“接口定义语言”定义通讯协议格式，然后 Thrift 根据协议格式自动帮你生成服务端和客户端代码。

（在这个方面，它有点类似于 Google 的 Protocol Buffers）

## 5.5 网络库、框架、中间件

---

### Twisted

Home: <http://twistedmatrix.com/>

Links: [Wikipedia](#)

这是一个基于 Python 网络通讯开发框架，诞生于2002年，名气很大。

它的某些设计类似于 C++ 的 [ACE](#) 框架。除了能用来进行传输层（TCP UDP）的开发，还提供了若干应用层协议（HTTP、XMPP、SSH、IRC ...）的支持。

代码示例——实现一个简单的 Echo 服务，监听在 12345 端口

```
from twisted.internet import protocol, reactor

class Echo(protocol.Protocol) :
    def dataReceived(self, data) :
        self.transport.write(data)

class EchoFactory(protocol.Factory) :
    def buildProtocol(self, addr) :
        return Echo()

reactor.listenTCP(12345, EchoFactory())
reactor.run()
```

## gevent

Home: <http://www.gevent.org/>

这是一个基于[协程](#)的网络库，原先其底层依赖于 libevent，后来改为 libev。

很多开源项目用到了 gevent，具体参见 [gevent 官方的 wiki](#)。

代码示例——并发执行网络请求

```
from gevent import socket
import gevent

hosts = ["google.com", "github.com", "program-think.blogspot.com"]
jobs = [gevent.spawn(socket.gethostbyname, host) for host in hosts]
gevent.joinall(jobs, timeout=2)
print([job.value for job in jobs])
```

## PyZMQ

Home: <https://github.com/zeromq/pyzmq>

这是 ZMQ（ZeroMQ）的 Python 封装库。同时支持 Python2 和 Python3。

PyZMQ 2.2 之后的版本同时支持 ZMQ 的 3.x 和 4.x 版本。

## nanomsg-python

Home: <https://github.com/tonysimpson/nanomsg-python>

这是 nanomsg 的 Python 封装库。同时支持 Python2 和 Python3。

代码示例——Hello world

```
from __future__ import print_function
from nanomsg import Socket, PAIR, PUB

s1 = Socket(PAIR)
s2 = Socket(PAIR)
s1.bind("inproc://test")
s2.connect("inproc://test")
s1.send(b"hello world")
print(s2.recv())
s1.close()
s2.close()
```



## 5.6 云计算

### Apache Libcloud

Home: <https://libcloud.apache.org/>

如今云提供商越来越多。这个库提供了统一的 API 让你访问各大知名云提供商提供的各种服务。

代码示例——创建 DNS 记录

```
from libcloud.dns.types import Provider, RecordType
from libcloud.dns.providers import get_driver

cls = get_driver(Provider.ZERIGO)
driver = cls("email", "api key")

zones = driver.list_zones()
zone = [zone for zone in zones if zone.domain == "mydomain.com"][0]

record = zone.create_record(name="www", type=RecordType.A, data="127.0.0.1")
print(record)
```

## 6 数据库

为了便于数据库开发，Python 社区制定了数据库的 API 规范（[PEP 249](#)）。

只要是涉及到数据库操作，标准库和大部分第三方库都会遵循该规范（请看如下几个模块的示例代码）。

### 6.1 数据库中间件

#### 6.1.1 ODBC

##### pyODBC

Home: <https://github.com/mkleehammer/pyodbc>

pyODBC 封装了 ODBC API，通过它可以访问各种数据库（只要有 ODBC 驱动即可）。

代码示例——查询某个 ODBC 数据源的某个表

```
import pyodbc

conn = pyodbc.connect("DSN=xxx;PWD=password")
cursor = conn.cursor()
cursor.execute("SELECT field1 FROM table1")

while True :
    row = cursor.fetchone()
    if not row :
        break
    print(row)

cursor.close()
conn.close()
```

##### ceODBC

Home: <http://ceodbc.sourceforge.net/>

又一个封装 ODBC API 的第三方库

#### 6.1.2 JDBC

##### Jython

Jython 前面已经介绍过。有了它，你可以基于 [JDBC](#) 操作数据库。

### 6.1.3 ADO & ADO.NET

#### PyWin32

PyWin32 前面已经介绍过。有了它，你可以基于 [ADO](#) 操作数据库。

#### IronPython

IronPython 前面已经介绍过。有了它，你可以基于 [ADO.NET](#) 操作数据库。

## 6.2 特定数据库

---

### 6.2.1 MySQL

#### MySQL for Python

Home: <http://mysql-python.sourceforge.net/>

操作 MySQL 的第三方库。

代码示例——查询某个 MySQL 数据库的某个表

```
import MySQLdb

conn = MySQLdb.connect(db="test", passwd="password")
cursor = conn.cursor()
cursor.execute("SELECT field1 FROM table1")

while True :
    row = cursor.fetchone()
    if not row :
        break
    print(row)

cursor.close()
conn.close()
```

### 6.2.2 PostgreSQL

#### psycopg

Home: <http://initd.org/psycopg/>

操作 PostgreSQL 的第三方库。

#### PyGreSQL

Home: <http://www.pygresql.org/>

操作 PostgreSQL 的第三方库。

### 6.2.3 Oracle

#### cx\_Oracle

Home: <http://cx-oracle.sourceforge.net/>

操作 Oracle 的第三方库。

### 6.2.4 MS SQL Server

#### pymssql

Home: <http://pymssql.org/>

操作微软 SQL Server 的第三方库。

## 6.2.5 IBM DB2

### ibm-db

Home: [https://pypi.python.org/pypi/ibm\\_db](https://pypi.python.org/pypi/ibm_db)

操作 DB2 的第三方库。

## 6.2.6 SQLite

### sqlite3

【标准库】

sqlite3 从 Python 2.5 版本开始加入到标准库中。通过它，你可以很方便地操作 SQLite 数据库。

SQLite 是一个很优秀的轻量级数据库，俺前几年写过一篇博文推荐它（在“[这里](#)”）。

代码示例——创建一个内存数据库，建表并插入记录

```
import sqlite3
conn = sqlite3.connect(":memory:") # ":memory:" 表示这是一个内存数据库
cursor = conn.cursor()
cursor.execute("CREATE TABLE person (name text, age int)")
cursor.execute("INSERT INTO stocks VALUES ('TOM',20)")
conn.commit()
conn.close()
```

## 6.2.7 MongoDB

### PyMongo

Docs: <https://docs.mongodb.com/ecosystem/drivers/python/>

这是 MongoDB 官方提供的 Python 驱动。

## 6.2.8 Apache HBase

### HappyBase

Home: <https://github.com/wbolster/happybase>

操作 HBase 的 Python 库，基于 Thrift 连接到 HBase。

代码示例——简单的存取操作

```
import happybase
connection = happybase.Connection("hostname")
table = connection.table("table-name")
table.put(b"row-key", {b"test1": b"data1", b"test2": b"data2"})
row = table.row(b"row-key")
print(row[b"test1"])
```

## 6.2.9 Redis

### redis-py

Home: <https://github.com/andymccurdy/redis-py>

操作 Redis 的第三方 Python 客户端。

代码示例——简单的存取操作

```
import redis
r = redis.StrictRedis(host="localhost", port=6379, db=0)
```

```
r.set("foo", "bar")
print(r.get("foo"))
```

### 6.2.10 LevelDB

#### Plyvel

Home: <https://github.com/wbolster/plyvel>

操作 LevelDB 的 Python 库，速度快，同时兼容 Python2 和 Python3。

代码示例——简单的存取操作

```
import plyvel
db = plyvel.DB("/tmp/testdb/", create_if_missing=True)
db.put(b"key", b"value")
print(db.get(b"key"))
db.close()
```

### 6.2.11 Berkeley DB

#### PyBSddb

Home: <http://www.jcea.es/programacion/pybsddb.htm>

操作 Berkeley DB 的第三方库。

## 6.3 ORM (Object-Relational Mapping)

### SQLAlchemy

Home: <http://www.sqlalchemy.org/>

Links: [Wikipedia](#) 维基百科

SQLAlchemy 支持的数据库有：MySQL、PostgreSQL、Sqlite、Oracle、MS SQL Server、Firebird、Sybase SQL Server、Informix、等。

代码示例——通过对象的方式创建两张依赖关系的表

```
from sqlalchemy import *
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relation, sessionmaker

Base = declarative_base()

class Movie(Base):
    __tablename__ = "movies"

    id = Column(Integer, primary_key=True)
    title = Column(String(255), nullable=False)
    year = Column(Integer)
    directed_by = Column(Integer, ForeignKey("directors.id"))
    director = relation("Director", backref="movies", lazy=False)

    def __init__(self, title=None, year=None):
        self.title = title
        self.year = year

    def __repr__(self):
        return "Movie(%r, %r, %r)" % (self.title, self.year, self.director)

class Director(Base):
    __tablename__ = "directors"

    id = Column(Integer, primary_key=True)
    name = Column(String(50), nullable=False, unique=True)
```

```
def __init__(self, name=None) :
    self.name = name

def __repr__(self) :
    return "Director(%r)" % (self.name)
```

```
Base.metadata.create_all(create_engine("dbms://user:pwd@host/dbname"))
```

## SQLObject

Home: <http://sqlobject.org/>

Links: [Wikipedia](#)

SQLObject 支持的数据库有: MySQL、PostgreSQL、Sqlite、MS SQL Server、Firebird、Sybase SQL Server、SAP DB、等。

代码示例——通过对象的方式创建表

```
from sqlobject import *
sqlhub.processConnection = connectionForURI("sqlite://:memory:")

class Person(SQLObject) :
    first_name = StringCol()
    last_name = StringCol()

Person.createTable()
```

## Peewee

Home: <http://www.peewee-orm.com/>

一个轻量级的 ORM，支持 SQLite、MySQL 和 PostgreSQL，通过插件机制支持更多数据库。

同时支持 Python2 和 Python3。

代码示例——通过对象的方式创建表

```
from peewee import *

db = SqliteDatabase("test.db")

class Person(Model) :
    name = CharField()
    birthday = DateField()
    is_relative = BooleanField()
    class Meta :
        database = db # This model uses the "test.db".

class Pet(Model) :
    owner = ForeignKeyField(Person, related_name="pets")
    name = CharField()
    animal_type = CharField()
    class Meta :
        database = db # This model uses the "test.db".

db.connect()
db.create_tables([Person, Pet])
```

# 7 GUI

## 7.1 GUI 框架

### 7.1.1 基于 Tk

[Tk](#) 是一个跨平台的界面组件库。

## Tkinter & tkinter

### 【标准库】

这是 Python 内置的标准库，封装了 Tcl/Tk 界面库。

Python2 的模块名叫 Tkinter，到 Python3 模块名改为 tkinter

代码示例——用 Tkinter 写 Hello world

```
from Tkinter import *

if __name__ == "__main__":
    root = Tk()
    label = Label(root, text="Hello, world")
    label.pack()
    root.mainloop()
```

## 7.1.2 基于 wxWidgets

wxWidgets 是 C++ 开发的跨平台框架（不仅包括 GUI，还有其它功能）。

### wxPython

Home: <http://www.wxpython.org/>

Links: [Wikipedia 维基百科](#)

在所有的 wxWidgets 的 Python 封装库中，这个是名气最大的。

Ulipad（知名的国产的 Python IDE）就是基于 wxPython 开发的。

代码示例——用 wxPython 写 Hello world

```
import wx

class Frame(wx.Frame):
    pass

class App(wx.App):
    def OnInit(self):
        self.frame = Frame(parent=None, title="Hello, world")
        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True

if __name__ == "__main__":
    app = App()
    app.MainLoop()
```

### PythonCard

Home: <http://pythoncard.sourceforge.net/>

又一个基于 wxWidgets 的 GUI 库。

## 7.1.3 基于 GTK+

GTK+ 全称是（GIMP Toolkit），由 C 开发的跨平台界面组件库。

### PyGTK

Home: <http://www.pygtk.org/>

Links: [Wikipedia](#)

它是 Python 对 GTK+2 的封装。

代码示例——用 PyGTK 写 Hello world

```

import pygtk
pygtk.require("2.0")
import gtk

class HelloWorld :
    def __init__(self) :
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("delete_event", self.delete_event)
        self.window.connect("destroy", self.destroy)
        self.window.set_border_width(10)

        self.button = gtk.Button("Hello, world")
        self.button.connect("clicked", self.hello, None)
        self.button.connect_object("clicked", gtk.Widget.destroy, self.window)
        self.window.add(self.button)

        self.button.show()
        self.window.show()

    def main(self) :
        gtk.main()

    def hello(self, widget, data=None) :
        print("Hello, world")

    def delete_event(self, widget, event, data=None) :
        print("delete event occurred")
        return False

    def destroy(self, widget, data=None) :
        gtk.main_quit()

if __name__ == "__main__" :
    hello = HelloWorld()
    hello.main()

```

## PyGObject (PyGI)

Home: <https://live.gnome.org/PyGObject>

它是 Python 对 GTK+3 的封装。PyGTK 的官网也推荐它。

代码示例——用 PyGObject 写 Hello world

```

from gi.repository import Gtk

class MyWindow(Gtk.Window):
    def __init__(self):
        Gtk.Window.__init__(self, title="Hello World")

        self.button = Gtk.Button(label="Click Here")
        self.button.connect("clicked", self.on_button_clicked)
        self.add(self.button)

    def on_button_clicked(self, widget):
        print("Hello, world!")

win = MyWindow()
win.connect("delete-event", Gtk.main_quit)
win.show_all()
Gtk.main()

```

## 7.1.4 基于 Qt

Qt 是 C++ 开发的跨平台框架（不仅包括 GUI，还有其它功能）。

### PyQt

Home: <http://www.riverbankcomputing.com/software/pyqt/>

Links: [Wikipedia 维基百科](#)

这是 Python 对 Qt 的封装。

代码示例——用 PyQt 写 Hello world

```
import sys
from PyQt4.QtGui import *

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = QWidget()

    window.resize(320, 240)
    window.setWindowTitle("Hello, world")
    window.show()
    sys.exit(app.exec_())
```

## PySide

Home: <http://www.pyside.org/>

这也是 Python 对 Qt 的封装。

## 7.1.5 基于 FLTK

FLTK 全称是（Fast Light Tool Kit），由 C++ 开发的跨平台、轻量级界面组件库。

### PyFLTK

Home: <http://pyfltk.sourceforge.net/>

这是 Python 对 FLTK 的封装。

## 7.1.6 基于 Windows 平台

### PyWin32

PyWin32 前面已经介绍过。它可以提供原生的 Windows GUI 界面。

### IronPython

IronPython 前面已经介绍过。它可以提供 dotNET 的 GUI 界面。

## 7.1.7 基于 JVM 平台

### Jython

Jython 前面已经介绍过。它可以提供基于 Java 的 [Swing](#) 界面。

## 7.1.8 （其它）

### EasyGUI

Home: <http://easygui.sourceforge.net/>

EasyGUI 这是一个很轻量级的库。跟其它 GUI 不同之处在于——它没有“事件驱动”。

### PyGUI

Home: [http://www.cosc.canterbury.ac.nz/greg.ewing/python\\_gui/](http://www.cosc.canterbury.ac.nz/greg.ewing/python_gui/)

PyGUI 是一个更高层的 GUI 库，底层分别封装了 [PyWin32](#)（Windows 平台）、[PyGTK](#)（Linux 平台）、[PyObjC](#)（Mac OS X 平台）。

### Kivy

Home: <http://kivy.org/>



跨平台的多媒体框架和界面库，用来开发比较炫的界面。

除了支持桌面操作系统，还支持 Android / iOS，支持多点触摸。

### OceampGUI

Home: <http://oceamp.sourceforge.net/gui.html>

基于 PyGame 的一个跨平台 GUI 库（PyGame 下面会提到）。

## 7.2 图表 & 报表

---

### matplotlib

Home: <http://matplotlib.org/>

Links: [Wikipedia](#)

这是一个有名的图形库，主要用来绘制数学相关的图形。

它跟后面提到的 SciPy 整合可以起到类似 MATLAB 的效果。效果图在“[这里](#)”。

### Gnuplot.py

Home: <http://gnuplot-py.sourceforge.net/>

这是 Python 对 [gnuplot](#) 的封装。gnuplot 的效果图在“[这里](#)”。

### PyQtGraph

Home: <http://www.pyqtgraph.org/>

这是一个纯 Python 的库，依赖于 PyQt4 / PySide。效果图在“[这里](#)”。

### PyX

Home: <http://pyx.sourceforge.net/>

这个库可以跟 TeX / LaTeX 无缝整合，支持导出为 PostScript / PDF 格式。适合用来制作报表。效果图在“[这里](#)”。

### Chaco

Home: <http://code.enthought.com/chaco/>

这是一个商业公司维护的库，主要提供2维图表。效果图在“[这里](#)”。

## 8 信息安全

---

### 8.1 密码学

---

#### hashlib

##### 【标准库】

在 Python 2.5 版本加入到标准库中。通过它，你可以很方便地计算各种散列值。

它支持的哈希算法有：MD5 SHA1 SHA224 SHA256 SHA384 SHA512

关于散列算法，俺写过一篇扫盲（在“[这里](#)”）。

代码示例——计算字符串的 SHA1 散列值

```
import hashlib
sha1 = hashlib.sha1("Hello, world").hexdigest()
```

## PyCrypto

Home: <http://www.dlitz.net/software/pycrypto/>

这个库包含了常见的对称加密算法（DES、AES、IDEA 等）、公钥加密算法（RSA、DSA 等）、散列算法（MD5、SHA1、RIPEMD 等）。

## pyOpenSSL

Home: <http://pyopenssl.sourceforge.net/>

[OpenSSL](#) 在加密领域可是大名鼎鼎。这个库使用 Python 对 OpenSSL 进行很薄的封装。

## Keyczar

Home: <https://github.com/google/keyczar>

这是 Google 提供的加密库，同时提供 C++、Java、Python 三种语言的实现。

它提供了比较高层的 API，使用者无需关心太多的细节。

## 8.2 访问控制

---

### oauth2client

Home: <https://github.com/google/oauth2client>

这是 Google 提供的 OAuth 客户端，支持 OAuth 2.0 规范。

## 9 处理文件格式

---

### 9.1 结构化数据格式

---

#### 9.1.1 CSV

[CSV](#) 是一种历史悠久的结构化数据存储格式。其效果类似于一张数据库二维表。

##### csv

###### 【标准库】

提供 CSV 格式文件的读写，可以手动指定行列分隔符。

#### 9.1.2 JSON

JSON 格式源自 JavaScript，如今在 Web 开发中广为应用。

##### json

###### 【标准库】

提供 JSON 格式的编码和解码。

代码示例——编码/解码 JSON 字符串

```
import json

json.dumps(["foo", {"bar": ("baz", None, 1.0, 2)}})
# JSON 编码
# 得到如下【字符串】
# """["foo", {"bar": ["baz", null, 1.0, 2]}]"""

json.loads("""["foo", {"bar": ["baz", null, 1.0, 2]}]""")
# JSON 解码
# 得到如下【对象】
# [u"foo", {u"bar": [u"baz", None, 1.0, 2]}]
```

### 9.1.3 YAML

**YAML** 是一种类似于 json 的结构化数据格式。它在确保可读性的基础上，提供了超越 json 的灵活性和扩展性。

#### PyYAML

Home: <http://pyyaml.org/>

pyyaml 提供了 Python 对 YAML 的封装。

## 9.2 压缩文件 & 打包文件

---

### 9.2.1 zip

#### zipfile

【标准库】

提供对 zip 格式的读写。

### 9.2.2 bzip2 (bz2)

#### bz2

【标准库】

提供对 bzip2 格式的读写。

### 9.2.3 gzip (gz)

#### gzip

【标准库】

提供对 gzip 格式的读写。

#### zlib

【标准库】

提供对 zlib 格式的读写。

### 9.2.4 tar

#### tarfile

【标准库】

提供对 tar 格式的读写。

### 9.2.5 7zip (7z)

#### PyLZMA

Home: <http://www.joachim-bauch.de/projects/pylzma/>

处理 7zip 格式的第三方库。

### 9.2.6 rar

#### rarfile

Home: <http://rarfile.berlios.de/>

处理 rar 格式的第三方库。

## 9.2.7 msi

### msilib

#### 【标准库】

提供对 msi 格式的读写，从 Python 2.5 版本开始加入标准库。

## 9.3 标记语言

---

### 9.3.1 XML

#### xml.dom & xml.miniDom & xml.etree.ElementTree

#### 【标准库】

用 DOM（Document Object Model）方式处理 XML 文件。

#### xml.sax & xml.parsers.expat

#### 【标准库】

用 SAX（Simple API for XML）方式处理 XML 文件。

### lxml

Home: <http://lxml.de/>

著名的 C 语言库 libxml 和 libxslt 的 Python 封装。

功能很强，支持 XPath 1.0、XSLT 1.0、扩展 EXSLT、等。还可以用来解析 HTML 格式。

### 9.3.2 HTML

#### HTMLParser

#### 【标准库】

以回调方式解析 HTML/XHTML 文件内容。

## 9.4 PDF

---

### pyfpdf

Home: <https://github.com/reingart/pyfpdf>

这是 FPDF 的 Python 移植库，用来生成 PDF 文档。

支持的功能比较全（嵌入字体、嵌入图片），文档也比较详细。

代码示例——简单的 Hello World 示例

```
from fpdf import FPDF

pdf = FPDF()
pdf.add_page()
pdf.set_font("Arial", "B", 16)
pdf.cell(40, 10, "Hello, World")
pdf.output("test.pdf", "F")
```

代码示例——支持写入 HTML 语法（目前支持几种常见的 HTML tag）

```
from pyfpdf import FPDF, HTMLMixin

class MyFPDF(FPDF, HTMLMixin):
    pass
```

```
pdf = MyFPDF()
pdf.add_page()
pdf.write_html(html)
pdf.output("test.pdf", "F")
```

## pyPdf & PyPDF2

Home: <http://knowah.github.com/PyPDF2/>

pyPdf 目前已经不继续升级维护了。PyPDF2 是从 pyPdf 派生出来的，并继续增加新功能。

它除了可以提取文件属性，还可以切分/合并文档，加密/解密文档。

## PDFMiner

Home: <http://www.unixuser.org/~euske/python/pdfminer/>

它可以提取 PDF 文件属性以及每页的文本，支持把内容输出为 HTML 格式。

## 9.5 MS Office 文档

---

### 9.5.1 Word（doc、docx）

#### python-docx

Home: <https://github.com/python-openxml/python-docx>

纯 python 实现的 docx 操作库，能够处理 docx 中的“文本、图片、样式”。

同时支持 Python2 和 Python3。

#### PyWin32

PyWin32 前面已经介绍过。它可以基于 COM 操作 Office 文档，包括 Word。

（本地需要安装 Office）

### 9.5.2 Excel（xls、xlsx）

#### pyExcellerator

Home: <http://sourceforge.net/projects/pyexcellerator/>

它可以支持 Office Excel（97/2000/XP/2003）以及 OpenOffice Calc 的文档。无需依赖外部软件。

#### PyWin32

PyWin32 前面已经介绍过。它可以基于 COM 操作 Office 文档，包括 Excel。

（本地需要安装 Office）

### 9.5.3 Power Point（ppt、pptx）

#### python-pptx

Home: <https://github.com/scanny/python-pptx>

它可以用来生成 pptx（Open XML PowerPoint）格式的文档。

#### PyWin32

PyWin32 前面已经介绍过。它可以基于 COM 操作 Office 文档，包括 Power Point。

（本地需要安装 Office）

## 9.6 RTF

---

### PyRTF

Home: <http://pyrtf.sourceforge.net/>

它可以用来处理 RTF（富文本格式）文档。

## 9.7 CHM

### PyCHM

Home: <http://gnochm.sourceforge.net/pychm.html>

这是基于 [chmlib](#) 的 Python 封装库。可以提取 CHM 文件的属性以及每个页面的内容。

## 10 图像

### 10.1 图像处理

#### Python Imaging Library (PIL)

Home: <http://www.pythonware.com/products/pil/>

Links: [Wikipedia](#)

这是一个很有名气的 Python 图像处理库，支持常见图像文件格式（BMP、JPG、GIF、PNG ...）。

它可以对图像进行各种常见的处理（旋转、缩放、剪切 ...）。

代码示例——为某个目录下所有 JPEG 创建缩略图

```
import os, glob
from PIL import Image

size = 128, 128
for file in glob.glob("*.jpg"):
    name, ext = os.path.splitext(file)
    img = Image.open(file)
    img.thumbnail(size)
    img.save(name+".thumbnail", "JPEG")
```

代码示例——旋转某图片并显示

```
from PIL import Image

img = Image.open("xxx.jpg")
img = img.rotate(90)
img.show()
```

#### Wand

Home: <http://docs.wand-py.org/>

它通过前面提到 [ctypes](#) 实现了对 [ImageMagick](#) 的封装（ImageMagick 是最强大的开源图片处理工具集）。

代码示例——旋转并缩放某图片

```
from wand.image import Image
from wand.display import display

with Image(filename="mona-lisa.png") as img :
    print(img.size)
    for r in 1, 2, 3 :
        with img.clone() as new_img :
            new_img.resize(int(new_img.width/2), int(new_img.height/2))
            new_img.rotate(90 * r)
            new_img.save(filename="mona-lisa-{}.png".format(r))
```

```
display(new_img)
```

## Pillow

Home: <http://python-pillow.org/>

你可以把它视作“轻量级的 PIL”。

它的目标是比 PIL 更容易使用，并尽可能与 PIL 的 API 兼容。

## PyGraphviz

Home: <https://github.com/pygraphviz/pygraphviz>

**Graphviz** 是一个功能很强大的关系图【自动】生成工具，具体介绍可以参见俺的博文（在“[这里](#)”）

这个库如其名所示，提供了 Python 对 Graphviz 的封装（基于 SWIG）。

## Graphviz

Home: <https://github.com/xflr6/graphviz>

这个库与上一个类似，也提供了 Graphviz 的 Python 的封装。

这两个库都在 GitHub 上。（可能是因为出现较晚）这个库的 Star 和 Fork 数都不如上一个，不过俺感觉文档比较全。

代码示例——创建一个 DOT 图并加入若干节点和连线

```
from graphviz import Digraph

dot = Digraph(comment='The Round Table')
# 添加节点
dot.node('A', 'King Arthur')
dot.node('B', 'Sir Bedevere the Wise')
dot.node('L', 'Sir Lancelot the Brave')
# 添加连线
dot.edges(['AB', 'AL'])
dot.edge('B', 'L', constraint='false')
```

## 10.2 图像格式转换

### Python Imaging Library (PIL)

PIL 前面已经介绍过。它支持常见图像文件格式（BMP、JPG、GIF、PNG ...）之间的相互转换。

### Wand

Wand 前面已经介绍过。由于它是针对 **ImageMagick** 的封装。只要 ImageMagick 能转换的格式，它也可以转换。

## 10.3 图像渲染

### Pycairo

Home: <http://cairographics.org/pycairo/>

**Cairo** 是一个图像渲染引擎，提供了矢量图像的渲染功能。支持多种后端输出（包括：Win32 GDI、OpenGL、Xlib、XCB、PDF、PNG、SVG.....）。

Pycairo 是 Cairo 官方提供 Python 封装。

## 11 游戏

### 11.1 综合性的游戏引擎

#### PyGame

Home: <http://www.pygame.org/>

Links: [Wikipedia 维基百科](#)

这是名气很大的跨平台游戏引擎，构建于 [SDL](#)（Simple DirectMedia Layer）之上。

它起先是用来替代终止开发的 [pySDL](#)，包含了图像和音频的库。

### Cocos2d

Home: <http://cocos2d.org/>

它是一个开源的 2D 游戏框架，最初使用 [Python](#) 编写的。后来该框架已经被移植到了多种语言和平台上。

其功能包括了：GUI 组件、音效、物理引擎、脚本语言绑定、场景编辑器 ...

很多手机游戏是基于 Cocos2d 的衍生框架开发的。

### Blender Game Engine

Home: <http://www.blender.org/>

Links: [Wikipedia 维基百科](#)

它是 [Blender](#) 的组成部分，虽然是以 C++ 编写，但内置了 [Python](#) 脚本的扩展。

其功能包括：3D 渲染、碰撞检测、角色编辑器、音效、网络通讯、AI ...

## 11.2 3D 渲染引擎

---

### PyOpenGL

Home: <http://pyopengl.sourceforge.net/>

封装 [OpenGL](#) 的 [Python](#) 库。

### Python-Ogre

Home: <http://www.python-ogre.org/>

封装 [OGRE](#) 的 [Python](#) 库。

## 12 数值计算 & 科学计算

---

### NumPy

Home: <http://www.numpy.org/>

Links: [Wikipedia 维基百科](#)

它提供了功能强大、性能很高的数值数组，可以用来进行各种数值计算（包括矩阵运算）。

代码示例

```
# 以下是传统 Python 写法，冗长且速度较慢
a = range(1000000)
b = range(1000000)
c = []
for i in range(len(a)) :
    c.append(a[i] + b[i])

# 以下是 NumPy 的写法，简洁且速度飞快
import numpy as np
a = np.arange(1000000)
b = np.arange(1000000)
c = a + b
```



## SciPy

Home: <http://www.scipy.org/>

Links: [Wikipedia 维基百科](#)

它依赖 NumPy 提供的多维数组。相比 NumPy，它提供了更高层的数学运算模块（统计、线性代数、积分、常微分方程求解、傅立叶变换、信号处理 ...）。

它被广泛用于科研和工程领域。

## SymPy

Home: <http://sympy.org/>

Links: [Wikipedia 维基百科](#)

它是用来做符号计算的，其目标是成为一个全功能的“计算机代数系统”。

它支持的功能包括：符号计算、高精度计算、模式匹配、绘图、解方程、微积分、组合数学、离散数学、几何学、概率与统计 .....

# 13 （其它）

---

一些不方便归类的，暂时放到这里。

## PyPy

Home: <http://www.pypy.org/>

Links: [Wikipedia 维基百科](#)

它是一个用 Python 写的 Python 解释器（有点绕口令）。

PyPy 支持 JIT（Just-in-time compilation）和沙箱技术，可做到【比 CPython 更快的运行速度】。