python 模块介绍- google 谷歌搜索引擎 python 接口

目录

项目简介

Python 中文库 https://bitbucket.org/xurongzhong/python-chinese-library 主要基于个人的使用经验，收集一些重要的外部和内部模块的中文教程和实例。发起人是 ouyangchongwu#gmail.com，xurongzhong#gmail.com 。

欢迎大家加入分享经验。联系方法：xurongzhong#gmail.com，微博：http://weibo.com/cizhenshi，python 及测试开发 qq 群 1：113938272，群 2:6089740。

文件下载：

1， https://bitbucket.org/xurongzhong/python-chinese-library/downloads。 推荐

2， hg clone 克隆所有文件 hg clone https://bitbucket.org/xurongzhong/python-chinese-library。

3， https://bitbucket.org/xurongzhong/python-chinese-library/src 浏览文件，右键点击文件，选另存为下载。

Bug 提交：https://bitbucket.org/xurongzhong/python-chinese-library/issuest。

版本管理

版本号	修订发布时间	修订人	备注
V1.0	2013-12-11	ouyangchongwu#gmail.com	初始版本，由 http://python-catalin.blogspot.com/2013/12/start-searching-with-python-google.html 生成。

参考资料：

官方网址：https://pypi.python.org/pypi/google

文档地址：http://pythonhosted.org/google/

作者博客地址：http://breakingcode.wordpress.com/（需翻墙）

Google 模块介绍：

http://python-catalin.blogspot.com/2013/12/start-searching-with-python-google.html（需翻墙）

简介：

功能：谷歌搜索引擎 python 接口月下载量：

Python 版本：*Python 2 和 Python 3*。

当前版本：1.0.5

下载地址： https://pypi.python.org/pypi/google

月下载量：1 万左右

平台：跨平台

相关模块：

BeautifulSoup 一个 html 解释器 月下载 20 万左右 强烈推荐

google 模块是谷歌搜索引擎 python 接口，系用 urllib2 在 google 上进行搜索，使用 BeautifulSoup 进行解释外部封装，非 google 官方接口。

安装

从 https://pypi.python.org/pypi/setuptools/下载最新版本的 setuptools。google 模块依赖 setuptools2.0。setuptools 是安装其他模块的基础，建议通过手动安装保持最新版本。

安装方法：

```
# tar xzvf setuptools-2.0.tar.gz

# cd setuptools-2.0

# python setup.py install

# easy_install pip

# pip install BeautifulSoup

# pip install google
```

使用

```
>>> from google import search

>>> for url in search('python', tld='com.hk', lang='cn', stop=2):

...         print(url)

...

http://www.python.org/

http://www.python.org/getit/

http://www.python.org/doc/

http://www.python.org/getit/releases/3.3.2/

http://www.python.org/getit/releases/2.7.5/

http://zh.wikipedia.org/zh-hk/Python

http://zh.wikipedia.org/wiki/PyPy

http://zh.wikipedia.org/wiki/%E5%8D%B3%E6%99%82%E7%B7%A8%E8%AD%AF

http://zh.wikipedia.org/wiki/%E5%90%89%E5%A4%9A%C2%B7%E8%8C%83%E7%BD%97%E8%8B%8F%E5%A7%8
6

http://zh.wikipedia.org/wiki/CPython

http://en.wikipedia.org/wiki/Python_(programming_language)

http://python.org.tw/

http://en.wikipedia.org/wiki/Burmese_python

http://en.wikipedia.org/wiki/Python_regius
```

http://news.nationalgeographic.com/news/2005/10/1006_051006_pythoneatsgator.html

http://www.animalpeoplenews.org/anp/2013/10/26/alligators-pigs-pythons-the-reptilian-ploy-to-resuscitate-sport-hunting/

http://kidfocused.com/mom-wakes-to-python-wrapped-around-toddler/

http://www.codecademy.com/tracks/python

http://openhome.cc/Gossip/Python/

http://pythonline.com/

http://learnpythonthehardway.org/book/

http://www.learnpython.org/

search(query, tld='com', lang='en', num=10, start=0, stop=None, pause=2.0): 使用谷歌搜索字符串 query。返回结果为生成器。

query (str) - 查询字符串。非 URL 编码格式。

tld (str) - 顶级域名 Top level domain。比如 com, com.hk。

lang (str) - 语言。

num(int) - 每页的结果数。

start(int) – 记录开始点。

stop(int) – 记录结束点。

pause(int) - HTTP 请求之间的 sleep 秒数。

get_page(url)：使用 cookiejar 返回页面。

相关变量：

url_home = "http://www.google.%(tld)s/"

url_search = "http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&btnG=Google+Search"

url_next_page = "http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&start=%(start)d"

url_search_num = "http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&num=%(num)d&btnG=Google+Search"

url_next_page_num =

"http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&num=%(num)d&start=%(start)d"

源码

Google 是一个相当小巧强悍的模块，代码只有 200 多行，是大家学习 urlib2，BeautifulSoup 的绝好资源，对如何发布 python 包也相当有借鉴意义。今后在 urlib2 等模块会深入 Google 模块，下面附上源码：

```
#!/usr/bin/env python

# Python bindings to the Google search engine
# Copyright (c) 2009-2013, Mario Vilas
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice,
#       this list of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright
#       notice,this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#     * Neither the name of the copyright holder nor the names of its
#       contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.

__all__ = ['search']

import os
import sys
import time

if sys.version_info[0] > 2:
```

```
    from http.cookiejar import LWPCookieJar
    from urllib.request import Request, urlopen
    from urllib.parse import quote_plus, urlparse, parse_qs
else:
    from cookielib import LWPCookieJar
    from urllib import quote_plus
    from urllib2 import Request, urlopen
    from urlparse import urlparse, parse_qs

# Lazy import of BeautifulSoup.
BeautifulSoup = None

# URL templates to make Google searches.
url_home          = "http://www.google.%(tld)s/"
url_search        =
"http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&btnG=Google+Search"
url_next_page     =
"http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&start=%(start)d"
url_search_num    =
"http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&num=%(num)d&btnG=Go
ogle+Search"
url_next_page_num =
"http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&num=%(num)d&start=%
(start)d"

# Cookie jar. Stored at the user's home folder.
home_folder = os.getenv('HOME')
if not home_folder:
    home_folder = os.getenv('USERHOME')
    if not home_folder:
        home_folder = '.'   # Use the current folder on error.
cookie_jar = LWPCookieJar(os.path.join(home_folder, '.google-cookie'))
try:
    cookie_jar.load()
except Exception:
    pass

# Request the given URL and return the response page, using the cookie jar.
def get_page(url):
    """
    Request the given URL and return the response page, using the cookie jar.

    @type  url: str
    @param url: URL to retrieve.
```

```
    @rtype:  str
    @return: Web page retrieved for the given URL.

    @raise IOError: An exception is raised on error.
    @raise urllib2.URLError: An exception is raised on error.
    @raise urllib2.HTTPError: An exception is raised on error.
    """
    request = Request(url)
    request.add_header('User-Agent',
                       'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0)')
    cookie_jar.add_cookie_header(request)
    response = urlopen(request)
    cookie_jar.extract_cookies(response, request)
    html = response.read()
    response.close()
    cookie_jar.save()
    return html

# Filter links found in the Google result pages HTML code.
# Returns None if the link doesn't yield a valid result.
def filter_result(link):
    try:

        # Valid results are absolute URLs not pointing to a Google domain
        # like images.google.com or googleusercontent.com
        o = urlparse(link, 'http')
        if o.netloc and 'google' not in o.netloc:
            return link

        # Decode hidden URLs.
        if link.startswith('/url?'):
            link = parse_qs(o.query)['q'][0]

            # Valid results are absolute URLs not pointing to a Google domain
            # like images.google.com or googleusercontent.com
            o = urlparse(link, 'http')
            if o.netloc and 'google' not in o.netloc:
                return link

    # Otherwise, or on error, return None.
    except Exception:
        pass
    return None
```

```
# Returns a generator that yields URLs.
def search(query, tld='com', lang='en', num=10, start=0, stop=None, pause=2.0):
    """
    Search the given query string using Google.

    @type  query: str
    @param query: Query string. Must NOT be url-encoded.

    @type  tld: str
    @param tld: Top level domain.

    @type  lang: str
    @param lang: Languaje.

    @type  num: int
    @param num: Number of results per page.

    @type  start: int
    @param start: First result to retrieve.

    @type  stop: int
    @param stop: Last result to retrieve.
        Use C{None} to keep searching forever.

    @type  pause: float
    @param pause: Lapse to wait between HTTP requests.
        A lapse too long will make the search slow, but a lapse too short may
        cause Google to block your IP. Your mileage may vary!

    @rtype:  generator
    @return: Generator (iterator) that yields found URLs. If the C{stop}
        parameter is C{None} the iterator will loop forever.
    """

    # Lazy import of BeautifulSoup.
    # Try to use BeautifulSoup 4 if available, fall back to 3 otherwise.
    global BeautifulSoup
    if BeautifulSoup is None:
        try:
            from bs4 import BeautifulSoup
        except ImportError:
            from BeautifulSoup import BeautifulSoup
```

```
# Set of hashes for the results found.
# This is used to avoid repeated results.
hashes = set()

# Prepare the search string.
query = quote_plus(query)

# Grab the cookie from the home page.
get_page(url_home % vars())

# Prepare the URL of the first request.
if start:
    if num == 10:
        url = url_next_page % vars()
    else:
        url = url_next_page_num % vars()
else:
    if num == 10:
        url = url_search % vars()
    else:
        url = url_search_num % vars()

# Loop until we reach the maximum result, if any (otherwise, loop forever).
while not stop or start < stop:

    # Sleep between requests.
    time.sleep(pause)

    # Request the Google Search results page.
    html = get_page(url)

    # Parse the response and process every anchored URL.
    soup = BeautifulSoup(html)
    anchors = soup.find(id='search').findAll('a')
    for a in anchors:

        # Get the URL from the anchor tag.
        try:
            link = a['href']
        except KeyError:
            continue

        # Filter invalid links and links pointing to Google itself.
        link = filter_result(link)
```

```
            if not link:
                continue

            # Discard repeated results.
            h = hash(link)
            if h in hashes:
                continue
            hashes.add(h)

            # Yield the result.
            yield link

        # End if there are no more results.
        if not soup.find(id='nav'):
            break

        # Prepare the URL for the next request.
        start += num
        if num == 10:
            url = url_next_page % vars()
        else:
            url = url_next_page_num % vars()

# When run as a script...
if __name__ == "__main__":

    from optparse import OptionParser, IndentedHelpFormatter

    class BannerHelpFormatter(IndentedHelpFormatter):
        "Just a small tweak to optparse to be able to print a banner."
        def __init__(self, banner, *argv, **argd):
            self.banner = banner
            IndentedHelpFormatter.__init__(self, *argv, **argd)
        def format_usage(self, usage):
            msg = IndentedHelpFormatter.format_usage(self, usage)
            return '%s\n%s' % (self.banner, msg)

    # Parse the command line arguments.
    formatter = BannerHelpFormatter(
        "Python script to use the Google search engine\n"
        "By Mario Vilas (mvilas at gmail dot com)\n"
        "https://github.com/MarioVilas/google\n"
    )
    parser = OptionParser(formatter=formatter)
```

```
    parser.set_usage("%prog [options] query")
    parser.add_option("--tld", metavar="TLD", type="string", default="com",
                      help="top level domain to use [default: com]")
    parser.add_option("--lang", metavar="LANGUAGE", type="string",
default="en",
                      help="produce results in the given language [default: en]")
    parser.add_option("--num", metavar="NUMBER", type="int", default=10,
                      help="number of results per page [default: 10]")
    parser.add_option("--start", metavar="NUMBER", type="int", default=0,
                      help="first result to retrieve [default: 0]")
    parser.add_option("--stop", metavar="NUMBER", type="int", default=0,
                      help="last result to retrieve [default: unlimited]")
    parser.add_option("--pause", metavar="SECONDS", type="float", default=2.0,
                      help="pause between HTTP requests [default: 2.0]")
    (options, args) = parser.parse_args()
    query = ' '.join(args)
    if not query:
        parser.print_help()
        sys.exit(2)
    params = [(k,v) for (k,v) in options.__dict__.items() if not
k.startswith('_')]
    params = dict(params)

    # Run the query.
    for url in search(query, **params):
        print(url)
```