

Python3

Insatallation of Requests

To install Requests, simply run this simple command in your terminal of choice:

```
$ pip install requests
```

Make a Request

Making a request with Requests is very simple. Begin by importing the Requests module:

```
import requests
```

let's try to get a webpage. For this example

```
r = requests.get('https://google.com')
```

Now, we have a **Response** object called `r`. We can get all the information we need from this object.

```
print("status: ", r.status_code)
```

```
print(r.url)
```

```
print(r.encoding)
```

Save Context to File

If you want to save something text to a file

```
import requests  
text = "I love Python"  
f = open("./Save_to_a_file.html", "w+")  
f.write(text)
```

If you use a context manager, the file is **closed automatically** for you

```
with open("Output.txt", "w") as text_file:  
    text_file.write(text)
```

Passing Parameters in URLs

You often want to send some sort of data in the URL's query string. If you were constructing the URL by hand, this data would be given as key/value pairs in the URL after a question mark.

Requests allows you to provide these arguments as a dictionary of strings, using the `params` keyword argument

```
payload = {"q": "python"}  
r = requests.get("https://www.google.com.tw/search?", params = payload)
```

Response Content (1/2)

```
import requests
payload = {"q": "python"}
r = requests.get("http://www.google.com/search", params=payload)
f = open("./page.html", "w+")
f.write(r.text)
```

Then you will see a **page.html** file in your directory :

```
├── page.html
└── requests_html_saving.py
```



page.html



requests_html_saving.py

Response Content (2/2)

Open [page.html](#)

搜尋 圖片 地圖 Play YouTube 新聞 Gmail 雲端硬碟 更多 >

python

全部 圖片 影片 新聞 地圖 書籍

約有 217,000,000 項結果

不限語言

搜尋所有中文網頁
搜尋繁體中文網頁

程式一定很難？你學過Python嗎？| 聯成電腦 程式資料分析應用班
[廣告] www.lccnet.com.tw/聯成電腦/Python資料分析 ◀
從基礎學起，第一次接觸程式設計也能無痛上手，完整培養數據分析開發的基本功！

不限時間

過去 1 小時
過去 24 小時
過去 1 週
過去 1 個月
過去 1 年

[Welcome to Python.org](https://www.python.org/)
<https://www.python.org/> ▶
The official home of the Python Programming Language.
[Python For Beginners](#) - [Python Docs](#) - [Alternative Python](#) - [Python Essays](#)

[Python 入門| Django Girls Taipei](http://djangogirlstaipai.herokuapp.com/tutorials/python/)
djangogirlstaipai.herokuapp.com/tutorials/python/ ▶
開始跟Python 培養感情之前，首先我們必須先稍微瞭解一下它。當我們說「Python」時，其實可能代表一個「語言」，或者一個「使用Python 這個語言的平台」。這是什麼 ...

所有結果
一字不差

[一小時Python入門-part 1 - 寫點科普](https://koppu.chat/2017/01/18/一小時python入門-part-1/)
<https://koppu.chat/2017/01/18/一小時python入門-part-1/> ▶
2017年1月18日 ... Python 是一個簡潔易讀的語言，學習者幾乎可以立刻上手，也適用於大量的商業 應用上。目前已超越C/C++、Java，成為各大學課程中的主流入門 ...

[Python - 維基百科，自由的百科全書 - Wikipedia](https://zh.wikipedia.org/zh-tw/Python)
<https://zh.wikipedia.org/zh-tw/Python> ▶
Python（英國發音：/ˈpaɪθən/ 美國發音：/ˈpaɪθoːn/），是一種廣泛使用的高階 程式語言，屬於通用型程式語言，由吉多·范羅蘇姆創造，第一版釋出於1991年。可以 視 ...
歷史 - 語法 - 著名第三方庫 - Python 3.0

Python

程式語言



Python，是一種廣泛使用的高階程式語言，屬於通用型程式語言，由吉多·范羅蘇姆創造，第一版釋出於1991年。可以視之為一種改良的LISP。作為一種直譯語言，Python的設計哲學強調代碼的可讀性和簡潔的語法。相比於C++或Java，Python讓開發者能夠用更少的代碼表達想法。 [維基百科](#)

面市時間：1991年，27年前
設計者：吉多·范羅蘇姆
型態系統："duck"、動態型別、強型別
作業系統：跨平台
最近測試版釋出日：3.7.0rc1 / 2018年6月11日; 3.6.6rc1 / 2018年6月12日
母公司：Python軟體基金會

其他人也搜尋了：

Binary Response Content(1/3)

You can also access the response body as bytes, for non-text requests:

The `gzip` and `deflate` transfer-encodings are automatically decoded for you.

For example, to create an image from binary data returned by a request, you can use the following code:



Binary Response Content(2/3)

```
>>> r = requests.get('https://www.google.com.tw/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png')
```

```
>>> r.content
```

```
>>> r.content
b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02 \x00\x00\x08\b\x08\x06\x00\x00\xda#\w\x1b\x08\x004\xB7IDAT\x01\xcexdd\k03\%Z\XcE\
xc0\xff1\x9cm\xd0\xb6}\xaf\x93\kc1\xd9(\xdY:\x9b\xafn\x92\xf4z\kf\xdbm\xdb\xbb\xad\xfb\xdbw\x83$=\xeb\xd7\x97\x9c\xd7\xbb\xf3^\xbf\x99\xe9\x
feN\xd5g\xbd\x9e\xd2|\xab\x93_\xf8X\xfd\xa3\xd5.\xf1\xfd\x9v\xcf9\xcd4\kc1\xb9\xa30\x1bz\xdc0y\xdc2\xbbv*\xf1#\xca\xfc2\x11\x91\x08\x00\x1el\xe1\|
\xd0\x9404\x85\x84\x1c\xffloR\xd5)Y\xe6\xe1\x97\xca\x86\xed\x94\xcf5\xcc7f\xd6_\x94if\x7f[\xa6\xcc3fxdd*eC\xb9nnIt\x9f\xab4\xei\xea\xe8\xbc\xa9\rc
3\x01\xca\xcf89\xca\x16?\x94y\x18n\xcd-\^l\xcd\xe8P\x01\x00\x00\x00i\xb7\xcbG\xa6\xdc3\x906]_Z\xbf\x872\xee\xca\xcc\xbae)5*\xa5\kctht\xa62a\xbb1\k
b4\xee\x8b\xca\xfa\xe7Ebu\x08\x00\xa06\x9f\wc8\xe0\xbc\xe2\x852\x0f\xdfV6\x1c\x9cB \w05\kc1L\x90\xe9ps\xfc~\xab\xcf4\x96\xde4\xe3[u\x1e\x17\x89
\xa6\x03\x00?V? \xfc\xe0\xdc\x9\x97*\xeblx7f\x91\x96Q\xba\x8f\x85\xae\xda\x87\xef8\xfd\xfel\xca\x14\x9f\x9J\xbbj|S\x9a\x08\x00\x00Y\xef7\xdc0\
\x83\xa84\xei\x1b\x99\x0egW\x1c\x0c\xd5/\xd7\xd8\xbb\x95\xb2\xee\x8d\x91h\x12\x08\x00\xcd\x83\xbb\xef2\xef1H+\x13\xbb6N\x9bEl\x95\rel\xad\xe8p\xa6
\xcc\x8b\xcf\xa5\xbd+\x91\xd88\x00\x00\xbb\x0f\xfcR\xd9#\x02\xfd7\xe3\x1fP\xea\x89\x8fJ\xe3\x8fP\xdaux94\rel\x9de:\\\x97\xde\xee\xa4D\xe0H\xa
c\x07\x00wa0\xaa\x00\x81\x012\x980--\wd3\xede1deC\xd98;\xdc\x98\x19\xef7\xa5\xef5\x8d\xef5\x02\x00z\x1c @\xae6\xdc\x1b\xdc2\xef9\x1c\xca\x86\xbb2\k
f1\x0c\xbf f/xde\x13\x89\x05\x00\x00\x08\x90\xdc@_@\xae6\xef2\x89gl\x13\xbb6\xcd\xbb4[J]\xfcl\x1f\xad:\xd2\xef8][\xef7\xecH\xfc\x0b\x00\x88\x00\xe9
\n\x18\x92\x96\x19\xfe=\\xd5\xef27\x82c\x03tH\x19 @\xf95H\xcc0\x03\x00\x812u\x64\xcc8\xcc0\x9cK\xafa\x946\x9cB!/:!\xc2!\xe9mQ[S\x00\x00\x0c9\x008\x1
6\x02d\xaac=\xbe\xa31\xef0Q\xbb9y\x90Ym\xccF\x80\xef6Y$Y\x00\xa0\xa1\x08\x10\x02d*= \xfc\x11\x84D\x97\xfe5\x96\xfc\xce7\x86\x9e\x1d\x82\x08\x84\xcc8
\xa6#eZ\xa6\xbb7\xbb2\xee\x0e\x02\xa2w\xa4\xef1\kc7\xbb4\xdacO\x8f\x04\x004\x08\x01\xbb2i\x08\x10e\kc2\xef7*\xbbb\x91\x167J=\xfelxaaH\x00C\x10 \x1
bF\x88|\xe5\xe0\xef2Q\xe9zz"\a1Z1\x8ahF\xbb\x0fG\x02\x00\x1a\x89\x00Y?\x02dhA\xe7/I\xca\xfb\x8a3\x98\x84\xfe\kc8\xbb4[\x9a\xee\x94\x89\x04\x00\k
d4\x1c\x01\x0b2n\x04\kc8\x87\xe7\x8c?CY\x7f\kc1\xec\xfb9\xeb2\xed\x832\xee~\xc3\xcd\xdc2\xbb8\xcb\xce3\xff7\xe7(\xeb\xcf5\xdc6]\x92\xeeH I\xbf\xa7\xbb
4[>\x8b"d\xae5\xcc\kc3\xdc7#y\x81\x085F\x80\xac\x8d\x08Q\xdc6?/\xd3\xei\xaaal\x989\xbb2<\xd3\xfe\xfc\t\xbb2\xa8\xcc\kc3/\x95)>\xd3\xa7r^\xfbxael\x9d\
xca\xcc7Dbc\xdc2\xfbp:\x914m\xa6\x95\xdc6\x7fu\x1a\xff\x97\xe8\x004\x16;\x83Ft\x0bFt\x014\x08\x01\xbb2:\x02\xa45?<m\x8a\x9c!_ \x8cWe\xdc6_\xa4\x8c
\x9f#\xf30\x9c\x96\x84"\xd1k)L\xa4\x0e|Yz\xfb3\xa0t\xdc3/\xed\kc7 >\x00\x00\x00A\x9f\x82d\xa8\xed\x9f\x9b\x96+\xa6\x7f4\xdc5|\xae\x8c\xffiz\x9eH
\xef4[\xdax+,\xb5W\xdc2\xffa] Z\x1b&\x88\x8f\xdc5\x01\x00\x91w82\x9e\xee\xfb9\x98\xcee\x97/L\x87q\xa9\xcc3\kc2\x81w11\xff\x96H\xcc\x14\xe9\xad\x8b\k
cc\xcc3\xbb75\x14\x11\x1f\x00\x00\xef4\x10\x01\xdc2j\x97\x8fw5\x9c5J\xe3\xa7\xff1\xfb\xdf\x0f\xce\x1d)Z$f\xaaxb4L\x93Y7\x90\xe9p:\xf1\xdc1%\xe0 @
\x00\x90t\x14\xbb8\xdc2\xei\xa8\xe9\xbb8\x1dv0\xfb\x9fw\xbb5\xaf\xa3JR(\x95\xa6k\x08\x0f\x00 @\x00\x90)R\kc6\xeb~\xdf\x01\x92Y\xbfCZ\xfb2\x89\xcc4\
\x95\kc2--\xcd\xa4\x90">6\x03\x00\x10 \x04Hf\xdc\x97\xfa\xfb\xe5\kc3]\x91\xe5\kc5\xfb"Q\x17\xe9r>e\xdc\xde\kc4\kc7&\x00\x08\x02\x84\x00\x91#\
xferM\x99\xfb6\xa1 _\xe3\xbb4\x9fG\xabb]>:\x12u\xa4L\xfb1\x85r\x8c\xef\x12\x1f\x00@\x00\x19 \xef_ \xd4y\x82\xbb2\xei\xda\xa8\xac\x9cq\xfb7g\xdc6\xbb5"
Qw\x83\xfb3\x8a\x17\xa6\x13X\x89\x8fua\x00\x00\x02\x84\x00Q&l\xdfa7e\x97\xdc3\xdc2\xa9\xaa\x91h\x8at*\xab4ak\xdc2\xeb3\xff\x00\x00\x01B\x00\xa4\k
e3\xcb\xfb\x11\x1f)\rd\xdc2\x81^\x91h"i\x8a\x1f\xa4(j\xea\x18\x1f\k000\x80^n\lx17\xcc\xbb9\xfb\xaa\x8f\x00\xfa\xfb\x13\xbb9\x19\x91h2\x99\x87D\xa6)\x9
9F\xcc7\x07\x00\x10 \x042\x00\xbb4~\x8f=\kc4\kc70#\xf1/\xf0t\xbbjb5V\x03\x80HJ\x00\x10 \xcaczYuJH\x1b\xbe\x1f\x89\x8d\x00\x00\xa0t\x01B\x08\xa4\k
ff1\xdc7\xaa1ka7^\xa4r\x7f\x8c\x04\x00 \x80\x00!@\x92\xcc\xfb8\x1fW\xcc\x87\xdcfb\x81=\x1f\x9b\x1b\x08P\xfb7\x00!@Z\xed\xbb1\xa7+\x1dF\xabb\x0b\x1
6\x7f\xae4we6N\xbb\x90\x00P\xfb\x80!@wada\xfb5\xa6\kc2\x9bh\ot\xbb5\xfbfr$6\x83\x08\x00\x00e\x10\x02f\x1d\x92Y\xdc5q\xeb\x99\xcb\x86\xe6\x84\ub3
```

`.content` method will return binary response content

Binary Response Content(3/3)

For example, to create an image from binary data returned by a request, you can use the following code:

```
>>> from PIL import Image
>>> import requests
>>> from io import BytesIO
>>> r = requests.get('https://www.google.com.tw/images/branding/
googlelogo/2x/googlelogo_color_272x92dp.png')
>>> i = Image.open(BytesIO(r.content))
>>> i.save('img.png', 'png')
```

Beautiful Soup (1/9)

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

官方文档（中文）：http://beautifulsoup.readthedocs.io/zh_CN/v4.4.0/

官方文档（英文）：<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

To install BeautifulSoup4, simply run this simple command in your terminal:

```
$ pip install bs4
```

完成安裝後的問題：

Beautiful Soup發佈時打包成Python2版本的代碼，在Python3環境下安裝時，會自動轉換成Python3的代碼，如果沒有一個安裝的過程，那麼代碼就不會被轉換。

如果代碼拋出了 `ImportError` 的異常：“No module named HTMLParser”，這是因為你在Python3版本中執行Python2版本的代碼。

如果代碼拋出了 `ImportError` 的異常：“No module named html.parser”，這是因為你在Python2版本中執行Python3版本的代碼。

如果遇到上述2種情況，最好的解決方法是重新安裝BeautifulSoup4。

Beautiful Soup (2/9)

1. Create a simple html file name: AliceWonderland.html:

```
<html>
<head><title>The Dormouse's story</title></head>
<body>
  <p class="title">
    <b>The Dormouse's story</b>
  </p>
  <p class="story">Once upon a time there were three little sisters; and their names
were
  <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
  <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
  <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
  <p class="story">...</p>
</body>
</html>
```

Beautiful Soup (3/9)

2. Open the html file by your browser, you will see :

The Dormouse's story

Once upon a time there were three little sisters; and their names were [Elsie](#), [Lacie](#) and [Tillie](#); and they lived at the bottom of a well.

...

Beautiful Soup (4/9)

3. Create your python file then writing code :

```
from bs4 import BeautifulSoup
html_alice_wonder = open('./AliceWonderland.html', 'r')
soup = BeautifulSoup( html_alice_wonder, "html.parser")
print(soup.prettify())
```


Beautiful Soup (5/9)

Firt, import BeautifulSoup from bs4 library :

```
from bs4 import BeautifulSoup
```

Open your Html file assign to a variable `html_alice_wonder`

```
html_alice_wonder = open('./AliceWonderland.html', 'r')
```

Create a BeautifulSoup Object by :

```
soup = BeautifulSoup(html_alice_wonder, "html.parser")
```

Prettify your html

```
print(soup.prettify())
```


Beautiful Soup (6/9)

Beautiful Soup supports the HTML parser included in Python's standard library, but it also supports a number of third-party Python parsers. One is the [lxml parser](#). Depending on your setup, you might install lxml with one of these commands:

Parser	Typical usage	Advantages	Disadvantages
Python's <code>html.parser</code>	<code>BeautifulSoup(markup, "html.parser")</code>	<ul style="list-style-type: none">• Batteries included• Decent speed• Lenient (as of Python 2.7.3 and 3.2.)	<ul style="list-style-type: none">• Not very lenient (before Python 2.7.3 or 3.2.2)
lxml's HTML parser	<code>BeautifulSoup(markup, "lxml")</code>	<ul style="list-style-type: none">• Very fast• Lenient	<ul style="list-style-type: none">• External C dependency
lxml's XML parser	<code>BeautifulSoup(markup, "lxml-xml")</code> <code>BeautifulSoup(markup, "xml")</code>	<ul style="list-style-type: none">• Very fast• The only currently supported XML parser	<ul style="list-style-type: none">• External C dependency
html5lib	<code>BeautifulSoup(markup, "html5lib")</code>	<ul style="list-style-type: none">• Extremely lenient• Parses pages the same way a web browser does• Creates valid HTML5	<ul style="list-style-type: none">• Very slow• External dependency

Python

Beautiful Soup (7/9)

Html Content :

```
<head><title>The Dormouse's story</title></head>
```

Python Code :

```
print("soup.title: ", soup.title)
# <title>The Dormouse's story</title>
```

```
print("soup.title.string: ", soup.title.string)
# 'The Dormouse's story'
```

```
print("soup.title.parent: ", soup.title.parent)
# <head><title>The Dormouse's story</title></head>
```

```
print("soup.title.parent.name: ", soup.title.parent.name)
# head
```

Beautiful Soup (8/9)

You can use the `find()` method find first `<a>` tag:

Html Content :

```
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,  
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and  
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
```

Python Code :

```
soup.find('a')
```

The only difference is that `find_all()` returns a list containing the single result, and `find()` just returns the result.

```
soup.find_all('title', limit=1)  
# [<title>The Dormouse's story</title>]  
soup.find('title')  
# <title>The Dormouse's story</title>
```

The `find_all()` method scans the entire document looking for results, but sometimes you only want to find one result.

Html Content :

```
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,  
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and  
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
```

Python Code :

```
for link in soup.find_all('a'):  
    print(link.get('href'))  
# http://example.com/elsie  
# http://example.com/lacie  
# http://example.com/tillie
```

One common task is extracting all the URLs found within a page's `<a>` tags :

Beautiful Soup (9/9)

Html Content :

```
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,  
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and  
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
```

Python Code :

```
soup.find(id="link3")
```

Found element by id.

.contents & .children

A tag's children are available in a list called `.contents`:

Python Code :

```
head_tag = soup.head
print(head_tag)
# <head><title>The Dormouse's story</title></head>
```

```
head_tag.contents
[<title>The Dormouse's story</title>]
```

```
title_tag = head_tag.contents[0]
print(title_tag)
# <title>The Dormouse's story</title>
```

```
title_tag.contents
# [u'The Dormouse's story']
```

.parent

Html Content :

```
<head><title>The Dormouse's story</title></head>
```

Python Code :

You can access an element's parent with the `.parent` attribute

```
title_tag = soup.title
title_tag
# <title>The Dormouse's story</title>
```

```
title_tag.parent
# <head><title>The Dormouse's story</title></head>
```

next & previous sibling

The `` tag and the `<c>` tag are at the same level: they're both direct children of the same tag. When a document is pretty-printed, siblings show up at the same indentation level. You can also use this relationship in the code you write.

Python Code :

```
sibling_soup = BeautifulSoup("<a><b>text1</b><c>text2</c></b></a>")  
print(sibling_soup.prettify())
```

You can use `.next_sibling` and `.previous_sibling` to navigate between page elements that are on the same level of the parse tree:

Python Code :

```
sibling_soup.b.next_sibling  
# <c>text2</c>  
sibling_soup.c.previous_sibling  
# <b>text1</b>
```


CSS select (1/3)

Python Code :

```
soup.select("body a")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select("html head title")
# [<title>The Dormouse's story</title>]
```

CSS select (2/3)

Python Code :

```
soup.select("head > title")  
# [<title>The Dormouse's story</title>]
```

```
soup.select("p > a")  
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,  
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,  
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

```
soup.select("p > a:nth-of-type(2)")  
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

```
soup.select("p > #link1")  
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]
```

```
soup.select("body > a")  
# []
```

CSS select (2/3)

Python Code :

```
soup.select(".sister")  
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,  
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,  
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

```
soup.select("#link1")  
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]  
soup.select("a#link2")  
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

<http://docs.python-requests.org/en/master/user/quickstart/#json-response-content>

http://beautifulsoup.readthedocs.io/zh_CN/v4.4.0/

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>