

BeautifulSoup

Web Crawler Related Packages



Python Web Crawler

爬蟲之前...

- ◆ 撰寫爬蟲程式，是為了取得網路上的資料並且進行後續分析，也就是說，獲取資料是我們的目的，寫程式只是手段。
- ◆ 因此，如果只是想要完成工作，而不是研究或學習爬蟲程式，在動手寫程式前，應該先搜尋是否有人已經把你做的事情做好了。
- ◆ 此外，在非寫程式不可的情況下，也應該考慮是否有更方便的資料取得方式。
- ◆ 搜尋現有服務、資料來源或程式碼。使用適當的關鍵字，如「爬蟲」、「下載」、「crawler」或「downloader」等，去搜尋是否已經有現成的服務或程式碼。例如搜尋關鍵字如「comic crawler」或是「python youtube download」等都能找到許多別人寫好的函式庫。

搜尋現有服務、資料來源或程式碼

- ◆ 使用「爬蟲」、「下載」、「crawler」或「downloader」等關鍵字搜尋是否已有現成的服務或程式碼。
- ◆ 以「YouTube」爬蟲為例，搜尋一下就能找到許多網路服務如OnlineVideoConverter等，可以直接拖拉點選完成YouTube影片的轉檔及下載，足以應付因常偶發的需求。
- ◆ 若想用程式大量或常態性的下載影片，也不需要自己土法煉鋼，例如搜尋關鍵字如「comic crawler」或是「python youtube download」等都能找到許多別人寫好的函式庫。
- ◆ 有些大型或常見的網站因為常應付外界的資料需求，可能會提供打包好的資料集供人下載，像維基百科或是政府開放資料平台都是典型例子。直接從這些資料集下手，就可以免去寫程式爬網頁的時間。

網路爬蟲基本流程

1. 導入爬蟲所需要的函式庫（如：urllib、requests、BeautifulSoup等）

2. 設定連接的網頁，例如：url = 'https://data.gov.tw/'

3. 連接請求，產生一個回傳物件，例如 res = requests.get(url)

4. 讀取HTML原始碼，例如 html=res.read()

5. 解析HTML，一般常用的方法有兩種：

➤ 正則表達式（根據正則規則擷取目標內容）

➤ BeautifulSoup（根據html標籤進行擷取，如：`<a>.....`）

BeautifulSoup 簡介

- ◆ BeautifulSoup 是一個Python的函式庫模組，讓開發者僅須撰寫少量的程式碼，就可以快速解析網頁HTML程式碼，從中擷取出使用者有興趣的資料，降低網路爬蟲程式的開發門檻，加快程式撰寫速度。
- ◆ BeautifulSoup 的運作方式就是讀取HTML原始碼，自動進行解析並產生一個BeautifulSoup物件，此物件中包含了整個HTML文件的結構樹，有了這個結構樹之後，就可以找出有興趣的資料了。



網頁文件解構

- ◆ 網頁是由標籤所組成的階層式文件，可以概分為三部份：HTML、CSS、JavaScript。
 - HTML：表示了網頁的骨架結構，也是與爬蟲程式最相關的部份。
 - CSS：處理網頁的樣式，如配色與排版。
 - JavaScript：負責瀏覽器端與使用者互動的程式功能。
- ◆ 一個現代化的網站網頁其實就是由這三大元素組成。如果你的目標只是編寫爬蟲程式與獲取資料，只需大致了解HTML的階層架構即可，並不需要深入地了解CSS與JavaScript。

網頁文件解構

- ◆ 假設我們想取得網頁的標題文字，首先要知道這段文字在網頁的哪個地方。
- ◆ 透過瀏覽器的**檢視原始碼功能**，觀察網頁文件的原始碼，我們可以發現網頁的確是由標籤(tag)所組成的階層式文件，且分為兩大區塊：
 - 首先是head區塊，記載網頁的描述性資訊，例如網頁的編碼(meta charset)，標題(title)，或需要的CSS及JavaScript檔案的連結(link)等。
 - 第二個區塊是body區塊，為網頁文件本體的內選，區塊裡面也有許多標籤，而我們需要的標題文字就夾在h1 tag之中。
- ◆ 各家瀏覽器都有提供檢視原始碼功能。若是Chrome瀏覽器的話，按**滑鼠右鍵**可以檢視原始碼；右上角選單或是**F12**也可以開啟開發者工具，檢視原始碼。

網頁文件解構

◆ 以上階層架構清楚地顯示HTML文件有head跟body兩大區塊，head裡面有一些描述性的資訊標籤(tag)，body裡面有網頁本體文件的標籤(tag)。假設我們需要的標題文字是

tag 夾住的文字。那我們要的爬蟲程式要進行的流程如下：

1. 與網站溝通，取得網頁文件
2. 解析(parse)取得的網頁文件，找到或定位出h1 tag
3. 印出h1 tag所包含的文字

BeautifulSoup 網頁爬蟲初探

爬蟲程式的重點：

1. 用requests連線
2. 用Beautiful解析取得的網頁文件
3. 用find()找標籤
4. 用text取得內容

BeautifulSoup 網頁爬蟲初探

- ◆ 首先我們引入(import) requests套件用以跟網站溝通，並引入BeautifulSoup套件用以解析網頁文件。
- ◆ 接著使用requests.get()方法，將網址放入其引數，若網站有回應，回傳的物件是一個response物件，response物件有很多屬性可以取用，我們最關心的是其文字屬性，亦即**resp.text**屬性的內容，因為它就是網頁內容。
- ◆ 我們將網頁內容做為BeautifulSoup物件的初始化參數之一，並指定解析器(parser)為BeautifulSoup內建的html.parser。
- ◆ 接著我們使用find()函式找出這個網頁裡面的h1 tag，找到之後使用 text 屬性將 h1 tag包含的文字輸出。

BeautifulSoup 解析器(parser)

解析器	使用方法	優點	缺點
Python's html.parser	<code>BeautifulSoup(markup, "html.parser")</code>	python自身帶有，速度比較快且能較好相容Python 2.7.3與 3.2.2以後版本	不能很好地相容於Python 2.7.3 或 3.2.2 以前版本
lxml's HTML parser	<code>BeautifulSoup(markup, "lxml")</code>	速度很快 相容性好	與外部C語言相關
lxml's XML parser	<code>BeautifulSoup(markup, "lxml-xml")</code> <code>BeautifulSoup(markup, "xml")</code>	速度很快的XML解析器	與外部C語言相關
html5lib	<code>BeautifulSoup(markup, "html5lib")</code>	相容性很好；可以像web 瀏覽器一樣解析html頁面； 可以產生正確的 HTML5	速度很慢；與外部Python 語言相關

BeautifulSoup 網頁爬蟲初探

1. 在這邊我們先定義一個html類型的檔案：

```
html_doc = """<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> """
```

2. 汇入BeautifulSoup，並使用lxml這個python的解析器。

```
from bs4 import BeautifulSoup  
  
soup = BeautifulSoup(html_doc, 'lxml')
```

BeautifulSoup 解析器

- ◆ 除了 BeautifulSoup 套件以外，我們還需要搭配使用 requests 與 lxml 套件來下載與解析網頁。
- ◆ 由於我們的開發環境是安裝 Anaconda，所以這些套件都不需要再另外下載與安裝，只要進行一貫的 import 就好。
- ◆ BeautifulSoup 可以支援的解析器 html.parser (Python 內建) 、html5lib與 lxml ，根據官方文件說明，lxml 套件是解析速度最快的。



BeautifulSoup 網頁爬蟲初探

3. soup就是的html_doc解析的結果，接著用prettyify()函式將soup這個物件以網頁的規格(格式化輸出)

呈現，並用print()將它顯示出來：print(soup.prettify())

```
<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 class="sister" href="http://example.com/elsie" id="link1">
        Elsie
      </a>
      ,
      <a class="sister" href="http://example.com/lacie" id="link2">
        Lacie
      </a>
      and
      <a class="sister" href="http://example.com/tillie" id="link3">
        Tillie
      </a>
      ;
    and they lived at the bottom of a well.
    </p>
    <p class="story">
      ...
    </p>
  </body>
</html>
```

BeautifulSoup 網頁爬蟲初探

4. 接下來我們要開始解析結構了，看一下在html中的標籤內容：

- `soup.title` 會得到結果 `title`資訊

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

- `soup.head` 會得到 `head` 資訊

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

- 我們也可以這樣取出`title`：

```
soup.head.title
```

- 若只想取字串內容：

```
soup.title.string
```

BeautifulSoup 網頁爬蟲初探

- ◆ `find_all()`：找出所有的標籤：

```
soup.find_all('p')
```

結果：`[<p class="title">The Dormouse's story</p>, <p class="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.</p>, <p class="story">...</p>]`

- ◆ `get()`：找出所有超連結的標籤，可以看到標籤中有一`href`屬性，用`get()`就可以取到它的連結位置：

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

結果：`http://example.com/elsie http://example.com/lacie http://example.com/tillie`

- ◆ 找出所有標籤內的`class`名稱：

```
for className in soup.find_all('p'):
    print(className.get('class'))
```

結果：`['title'] ['story'] ['story']`

BeautifulSoup 網頁爬蟲初探

- ◆ `find(id)`：依照id去取資料：

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

結果：

```
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

- ◆ `get_text()`：取出文字內容：

```
print(soup.get_text())
```

結果：

```
The Dormouse's story 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. ...
```

BeautifulSoup 網頁爬蟲 - 2

- ◆ 請解析具有超連結的html網頁，該網頁具有標題、超連結、class與id等。
- ◆ 該網頁原始碼以瀏覽器檢視時，其內容如下圖所示，具有一段文字與一段超連結文字。



三校聯盟 NTU SYSTEM

[臺灣大學](#) [臺灣師範大學](#) [臺灣科技大學](#)

BeautifulSoup 網頁爬蟲 - 2

◆ 透過BeautifulSoup套件的屬性、`find()`、`find_all()` 或`select()`等函式，將網頁原始碼擷取出如下圖的內容。

```
1 : <title>國立臺灣大學系統</title>
2 : <a class="union" href="http://www.ntu.edu.tw" id="link1">臺灣大學</a>
3 : <b>三校聯盟 NTU SYSTEM</b>
4 : [<a class="union" href="http://www.ntu.edu.tw" id="link1">臺灣大學</a>, <a
   class="union" href="http://www.ntnu.edu.tw" id="link2">臺灣師範大學</a>, <a
   class="union" href="http://www.ntust.edu.tw" id="link3">臺灣科技大學</a>]
5 : <a class="union" href="http://www.ntnu.edu.tw" id="link2">臺灣師範大學</a>
6 : <a class="union" href="http://www.ntust.edu.tw" id="link3">臺灣科技大學</a>
7 : http://www.ntu.edu.tw
8 : 臺灣大學
9 : 臺灣師範大學
10 : [<a class="union" href="http://www.ntust.edu.tw" id="link3">臺灣科技大學</
      a>]
```

BeautifulSoup 網頁爬蟲 - 2

1. 區入BeautifulSoup套件。
2. 將網頁原始檔之內容指定給變數html_text。
3. 在BeautifulSoup套件中運用html.parser解析原始碼，自訂bs為BeautifulSoup型別物件名稱。
4. 使用BeautifulSoup套件的title屬性取得網頁的標題。
5. 使用find()函式找出第一個超連結標籤及**標籤，並印出其內容。**

```
1 #以BeautifulSoup套件進行網頁解析
2 from bs4 import BeautifulSoup
3 html_text = """
4 <html><head><title>國立臺灣大學系統</title></head>
5 <body>
6 <p class="title"><b>三校聯盟 NTU SYSTEM</b></p>
7 <p class="ntu_system">
8 <a href="http://www.ntu.edu.tw" class="union" id="Link1">臺灣大學</a>
9 <a href="http://www.ntnu.edu.tw" class="union" id="Link2">臺灣師範大學</a>
10 <a href="http://www.ntust.edu.tw" class="union" id="Link3">臺灣科技大學</a>
11 </p></body></html>
12 """
13 bs = BeautifulSoup(html_text, 'html.parser')
14 print('1:', bs.title) #網頁標題屬性
15 print('2:', bs.find('a')) #<a>標籤
16 print('3:', bs.find('b')) #<b>標籤
```

BeautifulSoup 網頁爬蟲 - 2

6. 使用find_all()函式找出所有的超連結標籤且class名稱為「union」項目並輸出。
7. 使用find()函式找出文件的標籤且id名稱為「link2」項目並輸出。
8. 使用find()函式找出文件的標籤且href的內容為「<http://www.ntust.edu.tw>」項目並輸出。
9. 取出標籤且id名稱為「link1」的內容後，使用get取出其網址並輸出。
10. 使用select()函式取出class名稱為「union」的串列，data[0].text為串列的第一項，會印出「臺灣大學」字串，data[1].text為串列的第二項，會印出「臺灣師範大學」字串。
11. 使用select() 函式取出id名稱為「link3」的串列，並印出其內容。

```
17 print('4 : ',bs.find_all('a',{"class":"union"}))#印出標籤且class為union
18 print('5 : ',bs.find("a", {"id": "Link2"}))#印出標籤且id為link2
19 print('6 : ',bs.find("a", {"href": "http://www.ntust.edu.tw"}))
20 web=bs.find("a", {"id": "Link1"})
21 print('7 : ',web.get("href")) #使用get取出網址
22 data = bs.select(".union") #select會傳回串列
23 print('8 : ',data[0].text) #串列的第一項
24 print('9 : ',data[1].text)
25 print('10 : ',bs.select("#Link3"))
```

正則表示式 (Regular Expression)

- ◆ 正規表示式(regular expression) · 又稱正則表達式 · 正規表示法 · 正規式算 · 規則運算式 · 常規表示法等 · 是用在字串搜尋及處理上的一些自定義規則 · 你可以依照既定的格式去規定字串的樣式(pattern) · 接著讓程式去尋找目標文件裡面是否出現你所定義的pattern以及出現了幾次等。

```
Mac使用者  
資料科學  
給初學者的 Python 網頁爬蟲與資料分析  
static/python-for-beginners.png  
static/python_crawler.png  
static/python_crawler.png  
static/python_crawler.png  
static/python_crawler.png  
static/python_crawler.png  
static/python-for-beginners.png  
static/python_crawler.png  
static/python_crawler.png  
static/python_crawler.png  
static/python_crawler.png  
static/python-for-beginners.png  
static/python-for-beginners.png
```



正則表示式 (Regular Expression)

- ◆ 正則表示式 (Regular Expression) 最早是由數學家Stephen Kleene於1956年提出，主要使用在字元字串的格式比對，後來在資訊領域廣為應用，現在為ISO(國際標準組織)的標準之一。
- ◆ Python自1.5版本起增加了re模組，使Python語言擁有全部的正則表示式功能。
- ◆ Python正則表示式是一個特殊的字串，它能幫助我們方便的檢查一個字串是否與某種模式匹配。
- ◆ compile函式根據一個模式字串和可選的標誌引數生成一個正則表示式物件，該物件擁有一系列方法用於正則表示式做為比對(匹配)和替換。
- ◆ re模組也提供了與這些方法功能完全一致的函式，這些函式使用一個模式字串作為它們的第一個引數。



正則表示式 (Regular Expression)

符號	說明	例子	
\b	字的界限如空白	\bAB	" AB"、" ABCD" ; " AABC "
\B	「非」字的界限	\BAB	"XAB"、"AAB" ; " AB "
\d	數字 [0-9]	A\dB	"A0B"、"A3B" ; " A@B "
\D	「非」數字[^0-9]	A\D\D	"AXY"、"ABCD" ; " A36B "
\s	空白[\t\n\x0B\f\r]	A\sB	"A Bar" ; " ABar "
\S	「非」空白[^t\n\x0B\f\r]	A\S B	"A-Bar" ; " A Bar "、" ABar "
\w	字母、數字或底線 [a-zA-Z0-9_]	A\w	"A7"、"A_" ; " A# "、" A- "
\W	「非」字母、數字或底線 [^a-zA-Z0-9_]	A\W	"A%"、"A\$" ; " A7 "、" A_ "

符號	說明	例子	
^	字首	^pos	"pose" ; " apos "
\$	字尾	ring\$	"spring" ; " ringer "

正則表示式 (Regular Expression)

符號	說明	例子	
.	匹配一個任意字元(不包含換行)	A.B	"A1B"、"A+B"、"AZB"
*	匹配前一字元 0至多次	A*B	"AB"、"ACB"、"A123B"
+	匹配前一字元 1至多次	A+B	"AACB"、"AAA123B"
?	匹配前一字元 0至1次	A?B	"AB"、"ACB"
[]	一個在括號中的字元	A[XYZ]B	"AXB"、"AYB"、"AZB"
[^]	一個不在括號中的字元	A[^XYZ]B	"APB"、"A+B"、"A1B"
-	指定範圍	A[A-F]B	"ABB"、"ADB"、"AFB"
	分隔樣式符號	A[A 1234 XY]B	"AAB"、"A1234B"、"AAB"
{n}	前面字元重複n次	A[A-F]{3}	"ABCD"、"ADEF"、"AABB"
{n,}	前面字元重複n次或以上	A[CD]{2,}	"ACD"、"ADDC"、"ADCDCDC"
{n,m}	前面字元重複n次至m次	AB{2,4}C	"ABBC"、"ABBBC"、"ABBBBC"

正則表示式 (Regular Expression)

- ◆ `.findall()` 函式會在字串中找到正則表示式所匹配的**所有字串**，並返回一個列表；如果沒有找到的，則返回空列表。其語法格式為：

.findall(string, [, pos[, endpos]])

- `string`：待匹配的字串；
 - `pos`：可選引數，指定字串的起始位置，預設為0；
 - `endpos`：可選引數，指定字串的結束位置，預設為字串的長度；
-
- ◆ 相對於 `.findall()`函式，`match()` 和 `search()` 函式則是只匹配一次。

正則表示式 (Regular Expression)

- ◆ Greediness(貪婪模式)

- 正則表示式會儘可能找出長度最長的符合文字

- ◆ Reluctant quantifiers(勉強模式)

- 正則表示式pattern加上 '?'

- 找出長度最短的符合文字

正則表示式 (Regular Expression)

- ◆ 查詢字串中的所有數字：

```
import re

pattern = re.compile(r'\d+') # 查詢數字

result1 = pattern.findall('runoob 123 google 456')

result2 = pattern.findall('run88oob123google456', 0, 10)

print(result1)

print(result2)
```

- ◆ 以上例項執行輸出結果為：

['123', '456'] ['88', '12']

正則表示式 (Regular Expression)

```
1 import re
2 key='abcde@abc.edu.tw'
3 p1='@.+.'
4 pattern1=re.compile(p1)
5 print(pattern1.findall(key))
6 p1='@.+\.'
7 pattern1=re.compile(p1)
8 print(pattern1.findall(key))
9 p1='@.+?\.'
10 pattern1=re.compile(p1)
11 print(pattern1.findall(key))
```

['@abc.edu.tw']
['@abc.edu.']
['@abc.']

正則表示式範例_1

```
1 import requests
2 import re      # 正則表示式 (Regular expression operations)
3
4 url = 'https://web.mcu.edu.tw/'
5 htmlfile = requests.get(url)
6 if htmlfile.status_code == requests.codes.ok:
7     pattern = input("請輸入欲搜尋的字串 : ")    # pattern存放欲搜尋的字串
8 # 判斷是否搜尋到
9     if pattern in htmlfile.text:
10         print("搜尋 %s 成功" % pattern)
11     else:
12         print("搜尋 %s 失敗" % pattern)
13 # 計算出現次數
14     name = re.findall(pattern, htmlfile.text)
15     if name != None:
16         print("%s 出現 %d 次" % (pattern, len(name)))
17     else:
18         print("%s 出現 0 次" % pattern)
19 else:
20     print("網頁下載失敗")
```

請輸入欲搜尋的字串 : mcu
搜尋 mcu 成功
mcu 出現 194 次

請輸入欲搜尋的字串 : 銘傳
搜尋 銘傳 成功
銘傳 出現 40 次

請輸入欲搜尋的字串 : 123
搜尋 123 成功
123 出現 4 次

正則表示式範例_2

```
1 import requests
2 import re
3 from bs4 import BeautifulSoup
4 def main():
5     resp = requests.get('http://jwlin.github.io/py-scraping-analysis-book/ch2/bLog/bLog.html')
6     soup = BeautifulSoup(resp.text, 'html.parser')
7     # 找出所有 'h' 開頭的標題文字
8     titles = soup.find_all(['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])
9     for title in titles:
10         print(title.text.strip())
11     # 利用 regex 找出所有 'h' 開頭的標題文字
12     for title in soup.find_all(re.compile('h[1-6]')):
13         print(title.text.strip())
14     # 找出所有 .png 結尾的圖片
15     imgs = soup.find_all('img')
16     for img in imgs:
17         if 'src' in img.attrs:
18             if img['src'].endswith('.png'):
19                 print(img['src'])
20     # 利用 regex 找出所有 .png 結尾的圖片
21     for img in soup.find_all('img', {'src': re.compile('.png$')}):
22         print(img['src'])
23     # 找出所有 .png 結尾且含 'beginner' 的圖片
24     imgs = soup.find_all('img')
25     for img in imgs:
26         if 'src' in img.attrs:
27             if 'beginner' in img['src'] and img['src'].endswith('.png'):
28                 print(img['src'])
29     # 利用 regex 找出所有 .png 結尾且含 'beginner' 的圖片
30     for img in soup.find_all('img', {'src': re.compile('beginner.*\.png$')}):
31         print(img['src'])
32
33 main()
```



Python教學文章



Python爬蟲



Python爬蟲

資料科學
給初學者的 Python 網頁爬蟲與資料分析
(1) 前言
[Read More](#)

資料科學
給初學者的 Python 網頁爬蟲與資料分析
(2) 套件安裝與啟動網頁爬蟲
[Read More](#)

正則表示式範例_2

◆ 找出所有h開頭標籤包含的文字：

- 範例網頁中有許多h開頭的標籤(如

、等)，我們想找出所有此類標籤所包含的文字，可以使用find_all()，並傳入一個list，表示符合需求的標籤名稱。
 - Find_all()的目標是名稱屬於['h1','h2','h3','h4','h5','h6']其中之一的標籤。若改用正規表示式時，程式碼較為簡潔。
 - 在此我們傳入了一個由re.compile()產生的regular expression 物件給find_all()，並將字串規則放在re.compile()的引數內，規則h[1-6]表示字串為h開頭且後面接上數字1-6其中之一。

開發環境設定

Mac使用者

資料科學

給初學者的 Python 網頁爬蟲與資料分析

Python教學文章

開發環境設

Mac使用者

資料科學

給初學者的

資料科學

給初學者的

資料科學

給初學者的

資料科學

給初學者的

資料科學

給初學者的 Python 網頁爬蟲與資料分析

正則表示式範例_2

◆ 找出.png結尾的圖片網址：

- 範例網頁中有六張圖片，有些是png圖片，有些不是，若我們要找出png結尾的圖片，在不用正規表示式的狀況下，我們可以先把全部的標籤找出來，並檢查其src屬性是否以.png結尾。
- 若使用正規表示式搭配find_all()，第一個傳入的引數是標籤名稱(img)，接著在標籤屬性的部份使用dict，dict的key是「src」，value是一個regular expression物件，規則「\.\.png\$」中，因為「.」是表示規則用的特殊字元，所以要表示「.」時要多加一個反斜線；「\$」代表「以該字串結尾」，因此「\.\.png\$」就是「.png結尾的字串」。

所有的標籤，若其src屬性符合此規則，都會被find_all()回傳。

```
static/python-for-beginners.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python-for-beginners.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python-for-beginners.png
static/python-for-beginners.png
```

正則表示式範例_2

◆ 找出.png結尾的圖片網址：

- 若要更進一步找出所有.png結尾且含'beginner'的圖片，可以定義規則「beginner.*\.png\$」，其中「.」代表的是任意字元，「*」代表的是該字元出0或多次，所以此規則的意義是「檔名為.png且含有beginner的字串，beginner字串與檔名之間可以間隔0或多個字元」。

```
static/python-for-beginners.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python-for-beginners.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python_crawler.png
static/python-for-beginners.png
static/python-for-beginners.png
```

Q & A