網際網路大數據 & 爬蟲應用

Dr. Cheng-Yu Ma

Postdoctoral Research Fellow

CAIM, Chang Gung Memorial Hospital

長庚醫療人工智能核心實驗室

Center for Artificial Intelligence in Medicine,
Chang Gung Memorial Hospital

馬誠佑 博士

博士：國立清華大學資訊工程研究所

碩士：國立陽明大學生物醫學資訊所

Outline

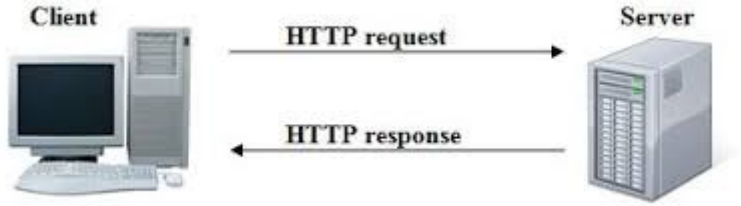
- HTTP & HTML
- urllib package
- Requests package (GET/POST)
- Regular Expression

HyperText Transfer Protocol (HTTP) protocol

HTTP是一種用於分佈式、協作式和超媒體資訊系統的應用層協定。HTTP是全球資訊網的資料通訊的基礎。

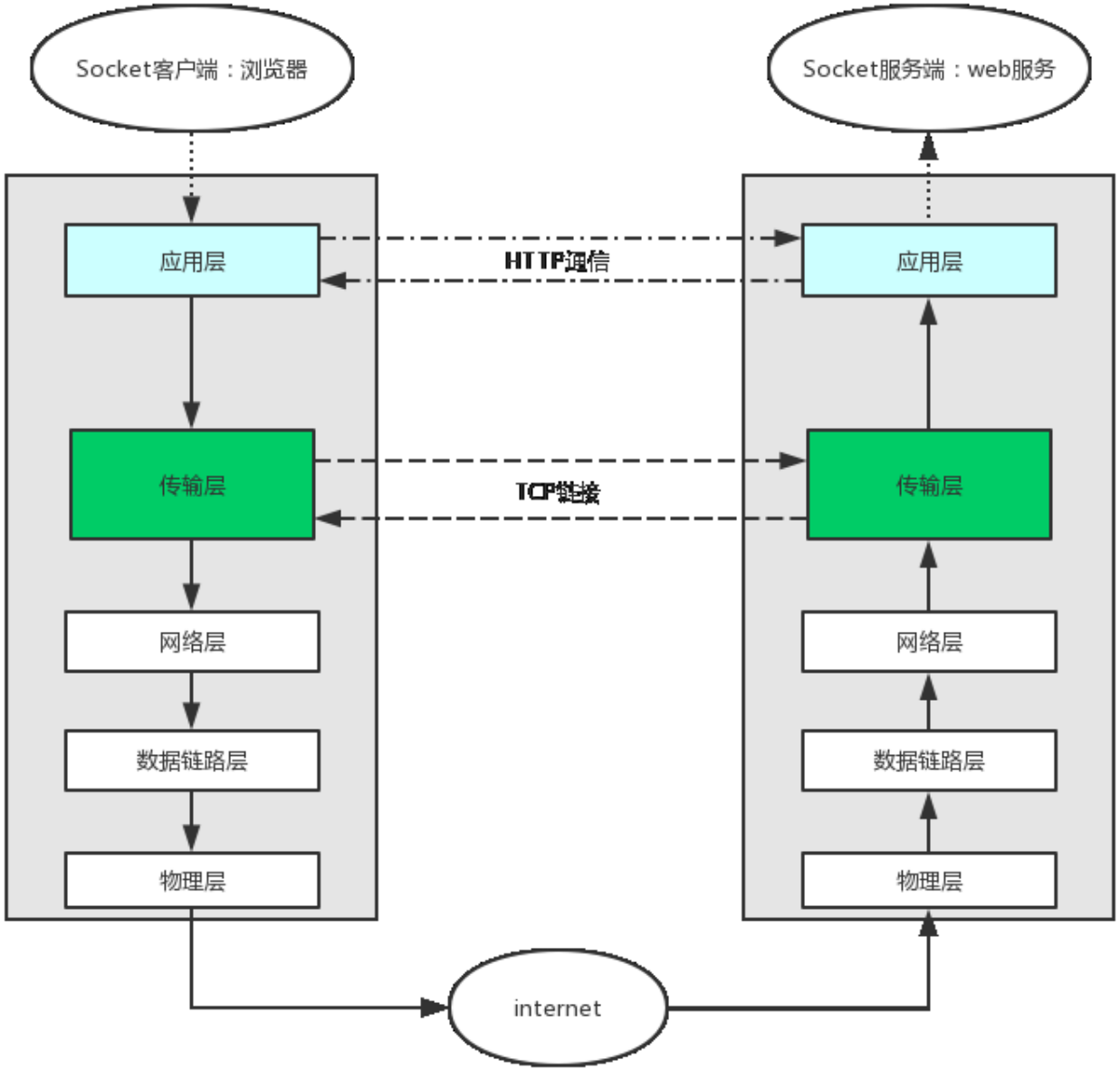
設計HTTP最初的目的是為了提供一種發布和接收HTML頁面的方法。通過HTTP或者HTTPS協定請求的資源由統一資源識別元（Uniform Resource Identifiers，URI）來標識。

HTTP的發展是由提姆·柏內茲-李於1989年在歐洲核子研究組織（CERN）所發起。HTTP的標準制定由全球資訊網協會（World Wide Web Consortium，W3C）和網際網路工程任務組（Internet Engineering Task Force，IETF）進行協調，最終發布了一系列的RFC，其中最著名的是1999年6月公布的 RFC 2616，定義了HTTP協定中現今廣泛使用的一個版本——HTTP 1.1。



https://zh.wikipedia.org

HyperText Transfer Protocol (HTTP)



HTTP Version

- ~~HTTP/0.9~~
  - GET
- HTTP/1.0
  - 代理伺服器 (ex: 防火牆，路由器)
- HTTP/1.1
  1. 快取處理
  2. 頻寬最佳化及網路連接的使用
  3. 錯誤通知的管理
  4. 訊息在網路中的傳送
  5. 網際網路位址的維護
  6. 安全性及完整性
- HTTP/2.0

HTTP Version

- HTTP/2.0
  - 不會破壞現有程式的工作，但是新的程式可以藉由新特性得到更好的速度
  - 保留了 HTTP/1.1 的大部分語意，例如請求方法、狀態碼乃至URI和絕大多數HTTP頭部欄位一致。而 HTTP/2 採用了新的方法來編碼、傳輸用戶端——伺服器間的資料
- SPDY (發音為"speedy") 是一個由 Google 主導的研究專案發明的HTTP替代協定。
- 一開始主要關注降低延遲，採用了TCP通道，但是使用了不同的協定來達到此目的。
  1. 實現無需先入先出的多路復用
  2. 為簡化用戶端和伺服器開發的訊息—影格機制
  3. 強制性壓縮（包括HTTP頭部）
  4. 優先級排序
  5. 雙向通訊

HTTP Request Method

- **GET**
  - 向指定的資源發出「顯示」請求
- HEAD
  - 取得其中「關於該資源的資訊」
- **POST**
  - 請求伺服器進行處理（例如提交表單或者上傳檔案）
- PUT
  - 上傳其最新內容
- DELETE
  - 刪除Request-URI所標識的資源
- TRACE
  - 回顯伺服器收到的請求，主要用於測試或診斷
- OPTIONS
  - 使伺服器傳回該資源所支援的所有HTTP請求方法
- CONNECT
  - 預留給能夠將連接改為管道方式的代理伺服器 (SSL)

HTTP Status code

- 1xx訊息
  - 請求已被伺服器接收，繼續處理
- 2xx成功
  - 請求已成功被伺服器接收、理解、並接受
- 3xx重新導向
  - 需要後續操作才能完成這一請求
- 4xx請求錯誤
  - 請求含有詞法錯誤或者無法被執行
- 5xx伺服器錯誤
  - 伺服器在處理某個正確請求時發生錯誤

HTML

```
<HTML>
    <HEAD>
        <TITLE>Web Title </TITLE>
        <Meta>
    </HEAD>
    <BODY>
        web page content
    </BODY>
</HTML>
```

https://www.w3schools.com/html/

一般語法

屬性名稱	說明
<! - -與 - ->	註解
<a href target>	指定超連結的分割視窗
<a href>	指定超連結
<a name=名稱>	被連結點的名稱
<b>	粗體字
<base target>	指定超連結的分割視窗
<basefont size>	更改預設字形大小
<bgsound src>	加入背景音樂
<big>	顯示大字體
<blink>	閃爍的文字
<body text link vlink>	設定文字顏色
 	換行
<caption align>	設定表格標題位置
<caption>...</caption>	為表格加上標題
	向中對齊
<cite>...<cite>	用於引經據典的文字
<code>...</code>	用於列出一段程式碼
<comment>...</comment>	加上註解
<dd>	設定定義列表的項目解說
<dfn>...</dfn>	顯示"定義"文字
<dir>...</dir>	列表文字標籤
<dl>...</dl>	設定定義列表的標籤

⋮

urllib — URL handling modules

urllib is a package that collects several modules for working with URLs:

- urllib.request for opening and reading URLs 開啟後讀取url內容
- urllib.error containing the exceptions raised by urllib.request
  - 包含由urllib.request丟擲的異常類
- urllib.parse for parsing URLs 解析URL
- urllib.robotparser for parsing robots.txt files 解析robots.txt files

ParseResult 物件屬性如下表：

屬性	索引值	傳回值	不存在的傳回值
scheme	0	傳回 scheme 通訊協定	空字串
netloc	1	傳回網站名稱	空字串
path	2	傳回 path 路徑	空字串
params	3	傳回 url 查詢參數 params 字串	空字串
query	4	傳回query 查詢字串，即 GET 的參數。	空字串
fragment	5	傳回框架名稱	空字串
port	無	傳回通訊埠	None

Exercise

```
from urllib.parse import urlparse
output=
urlparse('https://movies.yahoo.com.tw/')
print(output)
print(output.scheme)
print(output.geturl())
```

quote_plus/unquote

```
#!pip install urllib3

import urllib.parse

encodedString = urllib.parse.quote_plus('大雨')
decodedString =
urllib.parse.unquote('%E5%A4%A7%E9%9B%A8')

print(encodedString)
print(decodedString)
```

Request Package - Requests: HTTP for Humans

Beloved Features
Requests is ready for today's web.
- Keep-Alive & Connection Pooling
- International Domains and URLs
- Sessions with Cookie Persistence
- Browser-style SSL Verification
- Automatic Content Decoding
- Basic/Digest Authentication
- Elegant Key/Value Cookies
- Automatic Decompression
- Unicode Response Bodies
- HTTP(S) Proxy Support
- Multipart File Uploads
- Streaming Downloads
- Connection Timeouts
- Chunked Requests
- .netrc Support


```
1 !pip install requests
```

```
Requirement already satisfied: requests in /usr/loca
Requirement already satisfied: chardet<3.1.0,>=3.0.2
Requirement already satisfied: urllib3<1.25,>=1.21.1
Requirement already satisfied: idna<2.9,>=2.5 in /us
Requirement already satisfied: certifi>=2017.4.17 in
```

HTTP Requests

#建立適當的 HTTP 請求

#匯入requests library
import requests

#將訪問該網址的結果回傳給 r
r = requests.get('http://www.google.com')
r

Response 基本操作

```
# 伺服器回應的狀態碼
print(r.status_code)

# 檢查狀態碼是否 OK
if r.status_code ==requests.codes.ok:
print("OK")

#伺服器網址
r.url

# 輸出網頁 HTML 原始碼
print(r.text)
dir(r)
```

測試 status code

- www.google.com
- tw.yahoo.com
- www.goo.com
- www.pchome.com.tw

Response 基本操作

 # 輸出 r 內容

dir(r)

```
In [42]: dir(r)
Out[42]:
['__attrs__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__enter__',
 '__eq__',
 '__exit__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__nonzero__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 'headers',
 'history',
 'is_permanent_redirect',
 'is_redirect',
 'iter_content',
 'iter_lines',
 'json',
 'links',
 'next',
 'ok',
 'raise_for_status',
 'raw',
 'reason',
 'request',
 'status_code',
 'text',
 'url']
```

增加 URL 查詢參數

```
# 查詢參數
my_params = {'key1': 'value1', 'key2': 'value2’  }

# 將查詢參數加入 GET 請求中
r = requests.get('http://www.google.com', params = my_params)

# 觀察 URL
print(r.url)
```

自訂請求表頭

```
# 自訂表頭
my_headers = {'user-agent': 'my-app/0.0.1’  }

# 將自訂表頭加入 GET 請求中
r = requests.get('http://www.google.com', headers =
my_headers)
```

[37]
```
# 自訂表頭
my_headers = {'user-agent': 'my-app/0.0.1'}
```

[38]
```
# 將自訂表頭加入 GET 請求中
r = requests.get('http://www.google.com', headers = my_headers)
```

帳號密碼登入

```
# 需要帳號登入的網頁
r = requests.get('https://api.github.com/user', auth=('user',
'pass'))
```

Exercise

- 做出可以查詢 google 的程式

- Ex: 奧運、戴資穎...

POST Request

- *# 資料*
- my_data = {'key1': 'value1', 'key2': 'value2’ }

- *# 將資料加入 POST 請求中*
- r = requests.post('http://www.google.com', data = my_data)

- *# 具有重複鍵值的資料*
- my_data = (('key1', 'value1'), ('key1', 'value2’))

- *# 將資料加入 POST 請求中*
- r = requests.post('http:// www.google.com', data = my_data)

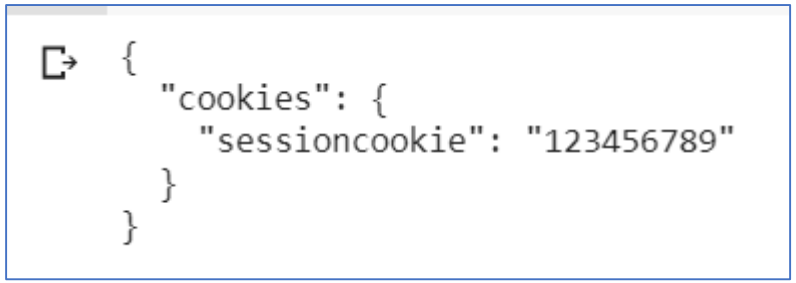
Upload File

- *# 要上傳的檔案*
- my_files = {'my_filename': open('my_file.docx', 'rb’)}

- *# 將檔案加入 POST 請求中*
- r = requests.post('http://httpbin.org/post', files = my_files)

Cookie

- # 含有 cookie 的內容
- r = requests.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
- # 取出 cookie
- print(r.text)



- # 設定 cookie
- my_cookies = dict(my_cookie_name='CGU_test')
- # 將 cookie 加入 GET 請求
- r = requests.get("http://httpbin.org/cookies", cookies = my_cookies)

網頁擷取常見問題

- *# 等待 3 秒無回應則放棄*
- requests.get('http://github.com/', timeout = 3)

- *# 關閉憑證檢查*
- r = requests.get(http://httpbin.org/cookies ', verify = False)

尋找指定字串

```
import requests
url = 'http://www.cgu.edu.tw/bin/home.php'
html = requests.get(url)
html.encoding="utf-8"


htmllist = html.text.splitlines()
n=0
for row in htmllist:
    if "長庚" in row: n+=1
print("找到 {} 次!".format(n))
```

Exercise

- 請以urlparse & requests.get 連接
- 長庚大學網頁網址
  https://goodnews.cgu.edu.tw/

1. 將ParseResult屬性全部輸出
2. 以requests.get尋找「長庚」出現次數

```
html = requests.get(url)
html.encoding="utf-8"
lines = html.text.splitlines()
```

Regular Expression

正規表示式	功能說明
.	代表一個除了換列字元 (\n) 以外的所有字元
^	代表輸入列的開始
$	代表輸入列的結束
*	代表前一個項目可以出現 0 次或無限多次

正規表示式	功能說明
+	代表前一個項目可以出現 1 次或無限多次
?	代表前一個項目可以出現 0 次或 1 次
[abc]	代表一個符合 a 或 b 或 c 的任何字元
[a-z]	代表一個符合 a、b、c ~z 的任何字元
\	代表後面的字元以一般字元處理
{m}	代表前一個項目必須正好出現 m 數
{m,}	代表前一個項目出現次數最少 m 次，最多無限次。
{m,n}	代表前一個項目出現次數最少 m 次，最多 n 次。
\d	一個數字字元，相當於 [0123456789] 或 [0-9]。
^	反運算，例如：[^a-d] 代表除了 a、b、c 、d 外的所有字元。
\D	一個非數字字元，相當於 [^0-9]。
\n	換列字元
\r	回列首字元 (carriage return)
\t	tab 定位字元
\s	空白字元，相當於 [\r\t\n\f]。
\S	非空白字元，相當於 [^ \r\t\n\f]。
\w	一個數字、字母或底線字元，相當於 [0-9a-zA-Z_]。
\W	一個非數字、字母或底線字元，相當於 [^\w]，即 [^0-9a-zA-Z_]。

Regular Expression- Examples

語法	正規表示式	範例
整數	[0-9]+	33025
有小數點的實數	[0-9]+\.[0-9]+	75.93
英文詞彙	[A-Za-z]+	Python

語法	正規表示式	範例
變數名稱	[A-Za-z_][A-Za-z0-9_]*	_pointer
Email	[a-zA-Z0-9_]+@[a-zA-Z0-9\._]+	guest@kimo.com.tw
URL	http://[a-zA-Z0-9\./_]+	http://e-happy.com.tw/

re package

```
import re
pat = re.compile('[a-z]+')
```

方法	說明
match(string)	傳回指定的字串中符合正規表示式的字串，直到不符合字元為止，並把結果存入 match 物件 (object) 之中；若無符合字元，傳回 None。
search(string)	傳回指定的字串中第一組符合正規表示式的字串，並把結果存入 match 物件 (object) 之中；若無符合字元，傳回 None。
findall()	傳回指定的字串中所有符合正規表示式的字串，並傳回一個串列。

match()

方法	說明
group()	傳回符合正規表示式的字串，直到不符合字元為止，並把結果存入 match 物件 (object) 之中；若無合法字元，傳回 None。
start()	傳回 match 的開始位置
end()	傳回 match 結束位置
span()	傳回 (開始位置 , 結束位置) 的元組物件

```
import re
pat = re.compile('[a-z]+')

m = pat.match('tem12po')
print(m)
```

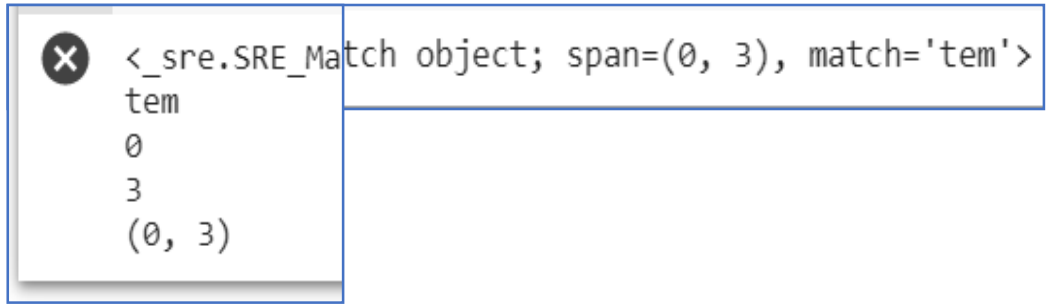
```
<_sre.SRE_Match object; span=(0, 3), match='tem'>
```

```
if not m==None:
    print(m.group())
    print(m.start())
    print(m.end())
    print(m.span())
```

```
<_sre.SRE_Ma
tem
0
3
(0, 3)
```

re.match()

```
import re
m = re.match(r'[a-z]+', 'tem12po')

print(m)

if not m==None:
    print(m.group())
    print(m.start())
    print(m.end())
    print(m.span())
```



search()

```
import re
pat = re.compile('[a-z]+')
m = pat.search('3tem12po')
print(m)

if not m==None:
print(m.group())  # tem
print(m.start())  # 1
print(m.end())    # 4    print(m.span())
# (1,4)
```

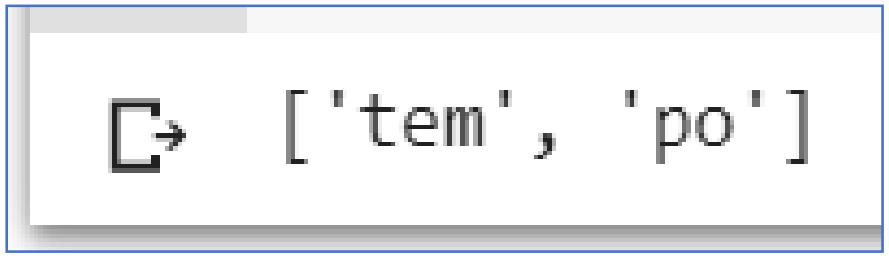
```
<_sre.SRE_Match object; span=(1, 4), match='tem'>
tem
1
4
(1, 4)
```

Exercise Match vs. Search

```
import re

s = '0www.weather.com'
pt = r'w'
rm = re.match(pt,s)
print(rm)

rm = re.search(pt,s)
print(rm.group())
```

findall()

```
import re
pat = re.compile('[a-z]+')
m = pat.findall('tem12po')

print(m)
```



Findall

```
import re

s = '0www.weather.com'
pt1 = r'w'
pt2 = r'c'
rf1 = re.findall(pt1,s)
rf2 = re.findall(pt2,s)
print(rf1,rf2)
```

```
import re

s = 'cat pat hat'
pt = r'[^p]a.'
rc = re.compile(pt)
print(rc.findall(s))
```

Findall

```
import re

s = '0www.weather.com'
pt1 = r'w'
pt2 = r'c'
rf1 = re.findall(pt1,s)
rf2 = re.findall(pt2,s)
print(rf1,rf2)

>>>['w', 'w', 'w', 'w'] ['c']
```

```
import re

s = 'cat pat hat'
pt = r'[^p]a.'
rc = re.compile(pt)
print(rc.findall(s))

>>>['cat', 'hat']
```

Exercise 使用正規表示式搜尋郵件帳號

```
import requests,re
regex = re.compile('...判斷式...')

url = 'https://auth.cht.com.tw/ldaps/'
html = requests.get(url)
emails = regex.findall(html.text)
for email in emails:
    print(email)
```

Exercise: 使用正規表示式搜尋郵件帳號

```
import requests,re
regex = re.compile(‘[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+’)

url = 'https://auth.cht.com.tw/ldaps/'
html = requests.get(url)
emails = regex.findall(html.text)
for email in emails:
    print(email)
```

邊界匹配

語法	說明	語法舉例
^	匹配字符串的起始點	re.search('^a','abcd')
$	匹配字符串的最後面	re.search('c$','abc')
\A	匹配字符串的起始點	re.search('\Aa','abcd')
\Z	匹配字符串的最後面	re.search('c\Z','abc')
\B	匹配除了字符串最後面的其他位置	re.search('co\B','colorful') re.search('fu\B','colorful') re.search('ful\B','colorful')##因為包含了最後的邊界符"l"，所以不能匹配

http://reference

邏輯與分組

語法	說明	語法舉例
|	匹配"|"符合左右邊的字符，等同於“或or”的概念，只要成功匹配任意一邊就算成功	re.search('a|b','a').group() re.search('abc|bca','bca').group()
(...)	分組的概念，會以整個括號內的字符為一個單位	re.search('(abc)*d','abcabcd').group() re.search('(a){2,}bc','aaaaaabc').group()
(?<name>)	分組的概念，但賦予這個組額外的別名	re.search('(?<name>abc){2}d','abcabcd').group()
(?P=name)	引用別名為<name>的分組的匹配字符	re.search('(?P<id1>\d)abc(?P=id1)','2abc8').group() re.search('(?P<id2>\d)abc(?P=id2)','6abc6').group()

P

http://reference

特殊

語法	說明	語法舉例
(?:re...)	類似於（...），但是它不分組，我自己實作時時常用到，舉例來說，我想爬取文件中格式為：（時間點|time|值，如66|time|5858）的所有符合格式數據的值，並只要前後的時間點與值，這樣我就需要把時間點與值分組，但是我不需要|time|，所以我就會使用(?:re...)	re.findall('(\d+)(?:\Dtime\D)(\d+)','66|time|5858') ## ['66','5858']
(?#)	此組內的字符為註解	re.search('ab(?#annotation)c','abc').group()
(?=...)	位於後面的字符串內容都必須符合裡面的規則，才會匹配	re.search('a(?=\d)','a66').group() ## a後面一定要為數字
(?!...)	位於後面的字符串內容都必須不符合裡面的規則，才會匹配	re.search('a(?!\d)','abc').group() ## a後面一定只能是非數字
(?<=...)	位於前面的字符串內容都必須符合裡面的規則，才會匹配	re.search('(?<=\d)a','66a').group() ## a前面一定只能有數字
(?<!...)	位於前面的字符串內容都必須不符合裡面的規則，才會匹配	re.search('(?<!\d)a','bca').group() ## a前面一定不能為數字

http://reference

轉義

轉義字符	說明
\f	匹配換頁
\n	匹配換行
\r	匹配“enter”鍵
\t	匹配“tab”鍵，水平制表
\v	重直制表
\\	代表一個反斜槓字符“\”

http://reference

Escape sequence

語法	需要匹配的字串	結果	說明
\d	\d	不能匹配	因為\d本身代表匹配任意數字
\\d	\d	可以匹配	多加了一個"\"，就能匹配"\d"了
\\\d	\\d	可以匹配	如果需要兩個的話（ex. \\d），就多加一個'\'，以此類推
r'\\d'	不論前面有多少個"\" ex. \\\\\\d	可以匹配	如果前面有太多個'\'時，可以直接使用r' \\d'

http://reference

Flag

可選標誌	說明
re.I	忽略大小寫，也就是不管大寫或小寫都能匹配
re.M	匹配多行，會影響'^'（開頭）與'$'（結尾）
re.L	做字符集本地化的識別，這是為了支援多種語言版本的字符集環境
re.S	使任意匹配符'.'，不受空白符限制，也就是沒有任何限制
re.U	\w\W\b\B\s\S\d\D 須依照UNICODE定義的屬屬性來使用
re.X	更靈活的應用，忽略了正則表達式中的空白與註釋（#），也可以是多行的

http://reference

Non-greedy

語法	說明	舉例	匹配結果
?	匹配0或更多次以上，但不能重複	re.search('go?','goooood').group()	'g'
+?	匹配1次或更多次以上，但不能重複	re.search('go+?','goooooood').group()	'go'
{n,m}?	匹配n到m次，但不要重複，但m不受到貪婪模式的限制	re.search('go{2,8}?','goooooood').group() re.search('go{2,8}?','gooooooood').group()	'goo ‘ 'gooooood'
{n,}?	匹配n次或n次以上，但不能重複	re.search('go{4,}?','goooooood').group()	'goooo'
??	匹配0次或1次或更多次以上，但不能重複	re.search('go??','goooooooood')	'g'

http://reference

Homework

- https://www.yelp.com.tw/search?cflt=restaurants&find_loc=%E5%8F%B0%E5%8C%97%E5%B8%82%2C%20%E5%8F%B0%E5%8C%97%E5%B8%82%2C%20TW&start=0
- 抓出前九頁台北餐廳的名稱、地址、電話
- 統計前九頁台北餐廳的分類(tag)並畫成長條圖