

正規表達式模組簡介

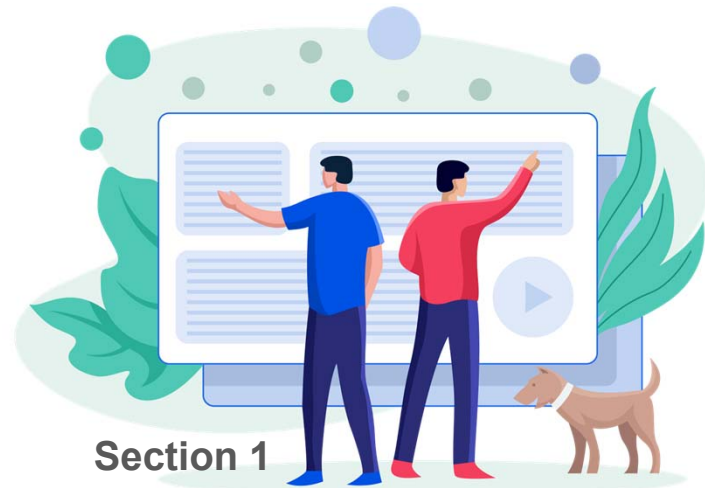
學習目標

- 正規表達式語法
- 使用 re 模組
- re 模組的 flag



正規表達式(Regular Expressions)

■瞭解正規表達式(Regular Expressions)



Section 1

正規表達式

- 正規表達式(Regular expression, regex) 是一特殊順序的字元組成，用來搜尋符合條件的字串
- 使用一些特殊字元組成表達式(Expression)
- 在 UNIX 的領域廣泛使用
- Python 的 “re” 模組提供支援 Perl-like 語法的正規表達式
- 在編譯或使用正規表達式時如果有發生錯誤，會觸發 re.error 的例外(Exception)

使用正規表達式的步驟

- 使用 “import re” 匯入正規表達式模組
- 使用 re.compile() 方法建立 Regex 物件
 - 記得使用 raw string
- 將字串傳入Regex 物件的 search() 方法搜尋符合條件的字串
 - 該方法會傳回 Match 物件
 - 也可用另一個 match() 方法搜尋文字的開頭是否符合條件
- 選擇性的，呼叫 Match 物件的 group() 方法取得實際符合條件的字串

特殊字元 -1

'.'	在預設模式，代表除了換行的任意一個字元
'^'	以匹配字串起始
'\$'	以匹配字串結尾或是換行之前以匹配字串結尾結束
'*'	匹配前面的元素零次或多次，例如 'ab*' 會匹配 'a'、'ab'、'abb' 等
'+'	匹配前面的元素一次或多次，例如 'ab+' 會匹配 'ab'、'abb'、'abbb' 等
'?'	匹配前面的元素零或一次，例如 'ab?' 會匹配 'a' 或 'ab'
'*?', '+?', '??'	將樣式(Pattern) 以非貪婪(non-greedy) 方式或是最少方式(minimal fashion) 進行匹配，取最少符合條件的字串。例如以 '<.*?>' 匹配 '<a> b <c>' 會得到 '<a>'

特殊字元 -2

{m}	匹配前面的元素精確地 m 次，例如 a{5} 會匹配 'aaaaa'
{m,n}	匹配前面的元件至少 m 次並且不超過 n 次，盡可能取最多匹配元件。例如 a{3,5} 會匹配 3 到 5 個 'a'。忽略 m 代表小於等於 n，忽略 n 代表大於等於 m
{m,n}?	匹配前面的元件至少 m 次並且不超過 n 次，盡可能取最少匹配元件。例如字串 'aaaaaa'， a{3,5} 會匹配到 'aaaaa'，而 a{3,5}? 則會匹配到 'aaa'
\	代表跳脫(escape) 某一特殊字元，允許你匹配 '*'、'?'、'.' 等。或者表示一特殊序列(special sequence)

特殊字元 -3

	選擇運算 $A B$ ，A 與 B 可以是任意的 Res，匹配 A 或 B
[]	<ul style="list-style-type: none">●代表一個集合，用來匹配任何包含在括號內的單一字元●例如 [amk] 會匹配到 'a'、'm' 或 'k'● '-' 代表字元範圍，例如 [a-z] 會匹配小寫英文字●假如括號內的第一個字為 '^'，代表反向(not)，所有不在集合內的字元會被匹配●在集合內的特殊字元會被視為一般字元
(...)	匹配括號內的任意正規表達式，也代表是一個群組(group)

正規表達式範例 -1

- '.' = {"a", "b", "c", ..., "A", "B", ... "@", ...}
- '^a' = {"a", "apple", "age", "amount" , "a happy day", ...}
- '^up' = {"up", "up and away", "upper", "upon", ...}
- 'ear\$' = {"ear", "clear", "top gear", ...}
- 'b*' = {"", "b", "bb", "bbb", ...}
- 'ab*' = {"a", "ab", "abb", "abbb", ...}
- '(ab)*' = {"", "ab", "abab", "ababab", ...}
- '(ab)+' = {"ab", "abab", "ababab", ...}
- '(ab)?' = {"", "ab"}

正規表達式範例 -2

- `'b{4}'` = `{"bbbb"}`
- `'ab{4}'` = `{"abbbb"}`
- `'(ab){4}'` = `{"abababab"}`
- `'b{,4}'` = `{"", "b", "bb", "bbb", "bbbb"}`
- `'ab{4,}'` = `{"abbbb", "abbbbb", "abbbbbbb", ...}`
- `'[abc]'` = `{"a", "b", "c"}`
- `'[a-zA-Z]'` = `{"a", "A", "b", "B", ..., "z", "Z"}`
- `'[0-9]'` = `{"0", "1", "2", ..., "9"}`
- `'[^aeiou]'` = `{"b", "c", "d", "f", "g", ...}`
- `'[a\ -z]'` = `{"a", "-", "z"}`

編譯標誌(Compilation Flags)

re.A re.ASCII	讓 \w, \W, \b, \B, \d, \D, \s 和 \S 只匹配ASCII，而不是Unicode
re.I re.IGNORECASE	忽略大小寫匹配，大小寫視為相同
re.L re.LOCALE	由目前語言來決定 \w, \W, \b, \B 和大小寫的匹配
re.M re.MULTILINE	多列匹配，影響到 '^' 與 '\$'
re.S re.DOTALL	讓 '.' 匹配任何字元，包括換行字元
re.X re.VERBOSE	允許編寫更具可讀性的正規表達式，可以分段與加註解

常用方法(Methods) -1

■ re.compile(pattern, flags=0)

- 將正規表達式的樣式編譯成物件(Regex object)，可以透過該物件的方法來匹配符合條件的資料，例如 match()、search()、findall() 等
- ```
prog = re.compile(pattern)
result = prog.match(string)
```

  
相當於  

```
result = re.match(pattern, string)
```
- 如果需要多次使用相同的正規表達式，使用 re.compile() 將其事先編譯以便重複使用，可以提高效能

# 常用方法(Methods) -2

## ■ `pattern.search(string[, pos[, endpos]])`

- ▣ 掃描整個 string，搜尋第一個匹配的位置，傳回相對應的物件。如果沒有符合條件，就傳回 None。可以接收 pos 和 endpos 參數，限制搜索範圍

## ■ `pattern.match(string[, pos[, endpos]])`

- ▣ 掃描 string 開頭位置，如果符合匹配條件，就傳回相對應的物件，否則就傳回 None。pos 和 endpos 參數，可以限制搜索範圍

## ■ `pattern.fullmatch(string[, pos[, endpos]])`

- ▣ 如果整個 string 符合匹配條件，就傳回相對應的物件，否則就傳回 None。pos 和 endpos 參數，可以限制搜索範圍

# 正規表達式(Demo)

## ■ 如何使用正規表達式(Regular Expressions)



# 本章重點精華回顧

## ■ 正規表達式(Regular Expressions)

- 語法簡介
- 模組使用



# Lab:正規表達式(Regular Expressions)使用

## ■ Lab01: 使用正規表達式(Regular Expressions)



# Lab01: 使用正規表達式(Regular Expressions) -1

1. 啟動Python IDLE環境，做以下練習
2. 使用 “File/Open...” 開啟 “regex.py” 程式，了解正規表達式的用法
3. 開啟命令列(cmd)，切換到labs相關目錄，做以下測試
4. 顯示有包含 ‘the’ 的文字列  
`python regex.py regex_data.txt`
5. 將`pat=r'the'` 改為`pat=r'[Tt]he'`，顯示有包含 ‘the’ 或 ‘The’ 的文字列  
`python regex.py regex_data.txt`
6. 將`pat=r'[Tt]he'` 改為`pat=r'\d'`，顯示有包含數字的文字列  
`python regex.py regex_data.txt`

# Lab01: 使用正規表達式(Regular Expressions) -2

7. 將`pat=r'\d'` 改為`pat=r'\d{3}'` , 顯示有包含3位數字的文字列  
`python regex.py regex_data.txt`
8. 將`pat=r'\d{3}'` 改為`pat=r'^[A-Z]'` , 顯示有大寫字元開頭的文字  
`python regex.py regex_data.txt`
9. 將`pat=r'^[A-Z]'` 改為`pat=r'^[^A-Z]'` , 顯示非大寫字元開頭的文字列  
`python regex.py regex_data.txt`
10. 將`pat=r'^[^A-Z]'` 改為`pat=r'b.t'` , 顯示b與t之間有一個任意字元的文字列  
`python regex.py regex_data.txt`
11. 將`pat=r'b.t'` 改為`pat=r'\.$'` , 顯示句點結尾的文字列  
`python regex.py regex_data.txt`
12. 關閉`regex.py`程式視窗