Regular Expersions <正則表達式>

Regular Expersions

also reffered to as "regex" or "regexp", provides a concise and flexible means for matching strings of text, such as particular characters, words .....

A regular expression is written in a formal language that can be interpreted by a regular expression processor.

Understanding Regular Expressions

powerful and cryptic

A language of "marker characters"

It's kind of an "old school" language

a language for string matching

Regular Expression Quick Guide

+	One or more string
^	Matches the beginning of a line
$	Matches the end of the line
.	Matches any character
\s	Matches white space
\S	Matches any non-white space character
* +	Repeats a character zero or more times
*? +?	.... (non-greedy)
[aeiou]	
[^XYZ]	
[a-z0-9]	
(	
)	

The Regular Expression Module

must import library `import re`

re.search( ): see if a string matches a regular expression

re.findall( ): extract portions of a string that match your regular expression

Using re.search() Like find()

```
hand = open('mbox-short.txt')

for line in hand:
    line = line.rstrip()
    if re.search('From:', line):
        print(line)

    if line.find('From:') >= 0:
        print(line)
```

Using re.search() Like startwith():

```
hand = open('mbox-short.txt')

for line in hand:
    line = line.rstrip()
    if re.search('^From:', line):
        print(line)

    if line.startwith('From:'):
        print(line)
```

Wild-Card Characters

^X.*:

^ : 開頭是
. : 後面接任意字
* : 表該字是 "any number of times"

x-Sieve: CMU Sieve 2.3
x-DSPAM-Result: Innocent
x-DSPAM-Confidence: 0.8475
x-Content-Type-Message-Body: text/plain

Fine-Tuning Your Match

^X-\S+:

^ : 開頭是
\S : 接任何非 whitespace 的 character
+ : 表該字 "one or more times"

x-Sieve: CMU Sieve 2.3
~~x-Plane is behind schedule: two weeks~~

Extracting Data

Matching and Extracting Data
<提取>

re.search( ) : return True / False

re.findall( ) : want the matching strings to be extracted, use re.findall( )

```
import re
x = 'My 2 favorate numbers are 19 and 42'
y = re.findall('[0-9]+', x)
print(y)
```

→ `['2', '19', '42']`

`[0-9]+` → one or more digits

```
y = re.findall('[AEIOU]+', x)
print(y)
```

→ `[]`

Warning : Greedy Matching

The repeat characters (* and +) push outward in both directions to match the largest possible string.

```
import re
x = 'From: Using the : charater'
y = re.findall ('^F.+:', x)
print(y)
```

↖ Last character in the match is a: (greedy)

```
['From: Using the :']
```

```
y = re.findall ('^F.+?:', x)
```

↖ Non-greedy

```
print(y)
['From:']
```

Fine - Tuning String Extraction

Parentheses are not part of the match, but they tell where to start and stop what string to extract.

```
x = 'From stephen.marquard@uct.ac.za sat Jan
     5 09:14:16 2008'
```

non-whitespace character

```
y = re.findall ('\S+@\S+', x)
```

||

space

```
y = re.findall ('From (\S+@\S+)', x)
```

```
print (y) ⟶ ['stephen.marquard@uct.ac.za']
```

```
data = 'From stephen.marquard@uct.ac.za sat Jan
        5 09:14:16 2008'

atpos = data.find ('@')
sppos = data.find (' ', atpos)
host = data[atpos + 1 : sppos]

print (host) ⟶ 'uct.ac.za'
```

The Double Split Pattern

```
data = 'From stephen.marquard@uct.ac.za sat Jan
       5 09:14:16 2008'

word = line.split()
email = words[1]
pieces = email.split('@')
```

→ 將句子中空格去掉並把每個 word 做為元素丟入 list 中

→ ['stephen.marquard', 'uct.ac.za']

```
print(pieces[1]) ⟶ 'uct.ac.za'
```

The Regex Version

```
data = 'From stephen.marquard@uct.ac.za sat Jan
       5 09:14:16 2008'

y = re.findall('@([^ ]*)', data)
```

Not

Match many of them

space

Match non-blank character

```
print(y) ⟶ ['uct.ac.za']
```

Even Cooler Regex Version

```
data = 'From stephen.marquard@uct.ac.za sat Jan
       5 09:14:16 2008'

y = re.findall ('^From .*@([^ ]*)', data)
```

Starting at the beginning of the line, look for the string 'From'.

```
print(y) ⟶ ['uct.ac.za']
```

Spam Confidence

```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence
                       :([0-9.]+)', line)
    if len(stuff) != 1: continue
```

```
    num = float(stuff[0])
    numlist.append(num)

print('Maximum:', max(numlist))
```

↳ x-DSPAM-Confidence: 0.8475

Escape Character

If you want a special regular expression character to just behave normally, you prefix it with '\'.

```
import re
x = 'We just received $10.00 for cookies.'
y = re.findall('\$[0-9.]+', x)
```

print(y) → ['$10.00']