CHAPTER 2: STRINGS AND TEXT

2.4. Matching and Searching for Text Patterns

Problem:

Bạn muốn match hay tìm kiếm văn bản cho một mẫu cụ thể.

Solution:

Nếu văn bản bạn đang cố gắng match là đơn giản, bạn có thể sử dụng các phương thức chuỗi cơ bản, chẳng hạn như str.find(), str.endswith(), str.startswith(), hoặc tương tự…
Ví dụ:

```
>>> text = 'yeah, but no, but yeah, but no, but yeah'

>>> # Exact match
>>> text == 'yeah'
False

>>> # Match at start or end
>>> text.startswith('yeah')
True
>>> text.endswith('no')
False

>>> # Search for the location of the first occurrence
>>> text.find('no')
10
>>>
```

Đối với nhiều matching phức tạp hơn, sử dụng biểu thức chính quy và module re. Để minh hoạ cơ chế cơ bản bằng cách sử dụng biểu thức chính quy, giả sử bạn muốn match các ngày được xác định như các chữ số, chẳng hạn như “11/22/2018”. Đây là đoạn code về cách bạn có thể thực hiện điều này:

```
>>> text1 = '11/27/2012'
>>> text2 = 'Nov 27, 2012'
>>>
>>> import re
>>> # Simple matching: \d+ means match one or more digits
>>> if re.match(r'\d+/\d+/\d+', text1):
...     print('yes')
... else:
...     print('no')
```

```
...
yes
>>> if re.match(r'\d+/\d+/\d+', text2):
...     print('yes')
... else:
...     print('no')
...
no
>>>
```

Nếu bạn đang thực thi nhiều matches bằng cách sử dụng mẫu giống nhau, nó thường biên dịch trước mẫu biểu thức chính quy thành một đối tượng mẫu đầu tiên.
Ví dụ:

```
>>> datepat = re.compile(r'\d+/\d+/\d+')
>>> if datepat.match(text1):
...     print('yes')
... else:
...     print('no')
...
yes
>>> if datepat.match(text2):
...     print('yes')
... else:
...     print('no')
...
no
>>>
```

Phương thức match() luôn tìm ra cái phù hợp tại điểm bắt đầu của một chuỗi. Nếu bạn muốn tìm văn bản cho tất cả các lần xuất hiện của một mẫu, thay vào đó hãy sử dụng phương thức findall().
Ví dụ:

```
>>> text = 'Today is 11/27/2012. PyCon starts 3/13/2013.'
>>> datepat.findall(text)
['11/27/2012', '3/13/2013']
>>>
```

Khi định nghĩa biểu thức chính quy, nó thường cho vào các nhóm bằng cách bao quanh các phần của mẫu trong dấu ngoặc đơn.
Ví dụ:

```
>>> datepat = re.compile(r'(\d+)/(\d+)/(\d+)')
>>>
```

Các nhóm thường đơn giản hoá việc xử lý tiếp theo của văn bản được matched bởi vì các nội dung của mỗi nhóm có thể được trích xuất riêng lẻ.
Ví dụ:

```
>>> m = datepat.match('11/27/2012')
>>> m
<_sre.SRE_Match object at 0x1005d2750>

>>> # Extract the contents of each group
>>> m.group(0)
'11/27/2012'
>>> m.group(1)
'11'
>>> m.group(2)
'27'
>>> m.group(3)
'2012'
>>> m.groups()
('11', '27', '2012')
>>> month, day, year = m.groups()
>>>

>>> # Find all matches (notice splitting into tuples)
>>> text
'Today is 11/27/2012. PyCon starts 3/13/2013.'
>>> datepat.findall(text)
[('11', '27', '2012'), ('3', '13', '2013')]
>>> for month, day, year in datepat.findall(text):
...     print('{}-{}-{}'.format(year, month, day))
...
2012-11-27
2013-3-13
>>>
```

Phương thức findall() tìm kiếm văn bản và tìm tất cả các kết quả phù hợp, trả về chúng như một list. Nếu bạn muốn tìm các kết quả trùng lặp lặp lại, sử dụng phương thức finditer().
Ví dụ:

```
>>> for m in datepat.finditer(text):
...     print(m.groups())
...
('11', '27', '2012')
('3', '13', '2013')
>>>
```

Discussion:

Một hướng dẫn cơ bản trên lý thuyết của biểu thức chính quy nằm ngoài phạm vi của cuốn sách này. Tuy nhiên, recipe này minh hoạ các khái niệm cơ bản tuyệt đối bằng cách sử dụng module re để match và tìm kiếm văn bản.

Chức năng cần thiết đầu tiên là biên dịch một mẫu bằng cách sử dụng re.compile() và sau đó sử dụng các phương thức như match(), findall(), hoặc finditer().

Khi chỉ định các mẫu, nó thường liên quan đến sử dụng các chuỗi như r'(\d+)/(\d+)/(\d+)'. Các chuỗi như vậy để lại dấu gạch chéo ngược không được biên dịch, có thể hữu ích trong ngữ cảnh của các biểu thức chính quy. Nếu không thì bạn cần sử dụng 2 dấu gạch chéo ngược như thế này '(\\d+)/(\\d+)/(\\d+)'.

Lưu ý rằng phương thức match() chỉ kiểm tra sự bắt đầu của một chuỗi. Nó có thể sẽ match với những thứ bạn không mong đợi.

Ví dụ:

```
>>> m = datepat.match('11/27/2012abcdef')
>>> m
<_sre.SRE_Match object at 0x1005d27e8>
>>> m.group()
'11/27/2012'
>>>
```

Nếu bạn muốn match chính xác, đảm bảo mẫu của bạn bao gồm điểm đánh dấu ở cuối ($), như dưới đây:

```
>>> datepat = re.compile(r'(\d+)/(\d+)/(\d+)$')
>>> datepat.match('11/27/2012abcdef')
>>> datepat.match('11/27/2012')
<_sre.SRE_Match object at 0x1005d2750>
>>>
```

Cuối dùng nếu bạn chỉ thao tác matching/searching một văn bản đơn giản, bạn có thể thường bỏ qua bước biên dịch và sử dụng các hàm mức mô đuntrong module re.

Ví dụ:

```
>>> re.findall(r'(\d+)/(\d+)/(\d+)', text)
[('11', '27', '2012'), ('3', '13', '2013')]
>>>
```

Tuy nhiên, hãy lưu ý rằng nếu bạn thực hiện nhiều matching hoặc searching, nó thường biên dịch trước và sử dụng nó hơn và hơn nữa. các hàm mức module lưu giữ bộ nhớ cache các mẫu được biên dịch gần đây, nên không có hiệu suất lớn, nhưng bạn sẽ lưu một vài tra cứu và xử lý thêm bằng cách sử dụng mẫu được biên dịch của riêng bạn.

2.5. Searching and Replacing Text

Problem:

Bạn muốn tìm kiếm và thay thế một mẫu văn bản trong một chuỗi.

Solution:

Đối với các mẫu đơn giản, sử dụng phương thức str.replace().

Ví dụ:

```
>>> text = 'yeah, but no, but yeah, but no, but yeah'

>>> text.replace('yeah', 'yep')
'yep, but no, but yep, but no, but yep'
>>>
```

Đối với các mẫu phức tạp hơn, sử dụng hàm sub() trong module re. Để minh hoạ, giả sử bạn muốn viết lại định dạng ngày “11/27/2012” thành “2012-11-27”.

Đây là một ví dụ về cách thực hiện điều đó:

```
>>> text = 'Today is 11/27/2012. PyCon starts 3/13/2013.'
>>> import re
>>> re.sub(r'(\d+)/(\d+)/(\d+)', r'\3-\1-\2', text)
'Today is 2012-11-27. PyCon starts 2013-3-13.'
>>>
```

Tham số đầu tiên của sub() là mẫu để match và tham số thứ hai là mẫu thay thế. Các kí tự gạch chéo ngược như \3 tham chiếu đến số nhóm trong mẫu.

Nếu bạn định thực hiện các thay thế được lặp lại của cùng một mẫu, hãy xem xét việc biên dịch nó trước cho việc thực thi tốt hơn.

Ví dụ:

```
>>> import re
>>> datepat = re.compile(r'(\d+)/(\d+)/(\d+)')
>>> datepat.sub(r'\3-\1-\2', text)
```

```
'Today is 2012-11-27. PyCon starts 2013-3-13.'
>>>
```

Đối với các thay thế phức tạp hơn, nó có thể sử dụng hàm callback thay thế.
Ví dụ:

```
>>> from calendar import month_abbr
>>> def change_date(m):
...     mon_name = month_abbr[int(m.group(1))]
...     return '{} {} {}'.format(m.group(2), mon_name, m.group(3))
...
>>> datepat.sub(change_date, text)
'Today is 27 Nov 2012. PyCon starts 13 Mar 2013.'
>>>
```

Như đầu vào, tham số của hàm callback thay thế là một đối tượng match, được trả về bởi match(), hoặc find(). Sử dụng phương thức .group() để chỉ định các phần cụ thể của match. Hàm nên trả về văn bản thay thế.
Nếu bạn muốn biết có bao nhiêu thay thế được thực hiện ngoài việc nhận được văn bản thay thế, sử dụng re.subn().
Ví dụ:

```
>>> newtext, n = datepat.subn(r'\3-\1-\2', text)
>>> newtext
'Today is 2012-11-27. PyCon starts 2013-3-13.'
>>> n
2
>>>
```

Discussion:
Không có nhiều biểu thức chính quy tìm kiếm và thay thế hơn phương thức sub() như đã thể hiện. Phần khó nhất là xác định mẫu biểu thức chính quy - một cái gì đó tốt nhất còn lại như một bài tập cho người đọc.

2.6. Matching Text at the Start or End of a String.
Problem:
Bạn muốn tìm kiếm và có thể thay thế văn bản theo cách không phân biệt chữ hoa chữ thường.

Solution:
Để thực thi các thao tác văn bản không phân biệt chữ hoa chữ thường, bạn cần sử dụng module re và cung cấp cờ re.IGNORECASE cho các thao tác.

Ví dụ:

```
>>> text = 'UPPER PYTHON, lower python, Mixed Python'
>>> re.findall('python', text, flags=re.IGNORECASE)
['PYTHON', 'python', 'Python']
>>> re.sub('python', 'snake', text, flags=re.IGNORECASE)
'UPPER snake, lower snake, Mixed snake'
>>>
```

Ví dụ cuối cùng cho thấy giới hạn mà việc thay thế văn bản sẽ không match trường hợp văn bản đã được matched. Nếu bạn cần sửa điều này, bạn có thể phải sử dụng một hàm hỗ trợ, như dưới đây:

```
def matchcase(word):
    def replace(m):
        text = m.group()
        if text.isupper():
            return word.upper()
        elif text.islower():
            return word.lower()
        elif text[0].isupper():
            return word.capitalize()
        else:
            return word
    return replace
```

Đây là một ví dụ sử dụng hàm trên:

```
>>> re.sub('python', matchcase('snake'), text, flags=re.IGNORECASE)
'UPPER SNAKE, lower snake, Mixed Snake'
>>>
```

Discussion:
Đối với các trường hợp đơn giản, chỉ cần cung cấp cờ re.IGNORECASE là đủ để thực thi việc matching không phân biệt chữ hoa chữ thường. Tuy nhiên, lưu ý rằng điều này có thể không đủ đối với một số loại matching nhất định của Unicode liên quan đến trường hợp gấp. Có thể xem recipe 2.10 để chi tiết hơn.