

데이터 크롤링과 정제

3장. HTML 분석 및 정규식

목차

- HTML 분석
 - find()와 find_all() 함수
 - select()함수
 - 다른 BeautifulSoup 객체
 - 트리 이동
- 정규 표현식

HTML 분석 개념

- 복잡한 웹페이지에서 필요한 정보 가져오기
 - 원하지 않는 콘텐츠 제거
 - 원하는 정보는 다양한 곳에 존재
 - 페이지 타이틀
 - 페이지 URL
 - 원하는 정보가 정형화 되어 있지 않는 경우, 문제 발생

HTML 분석

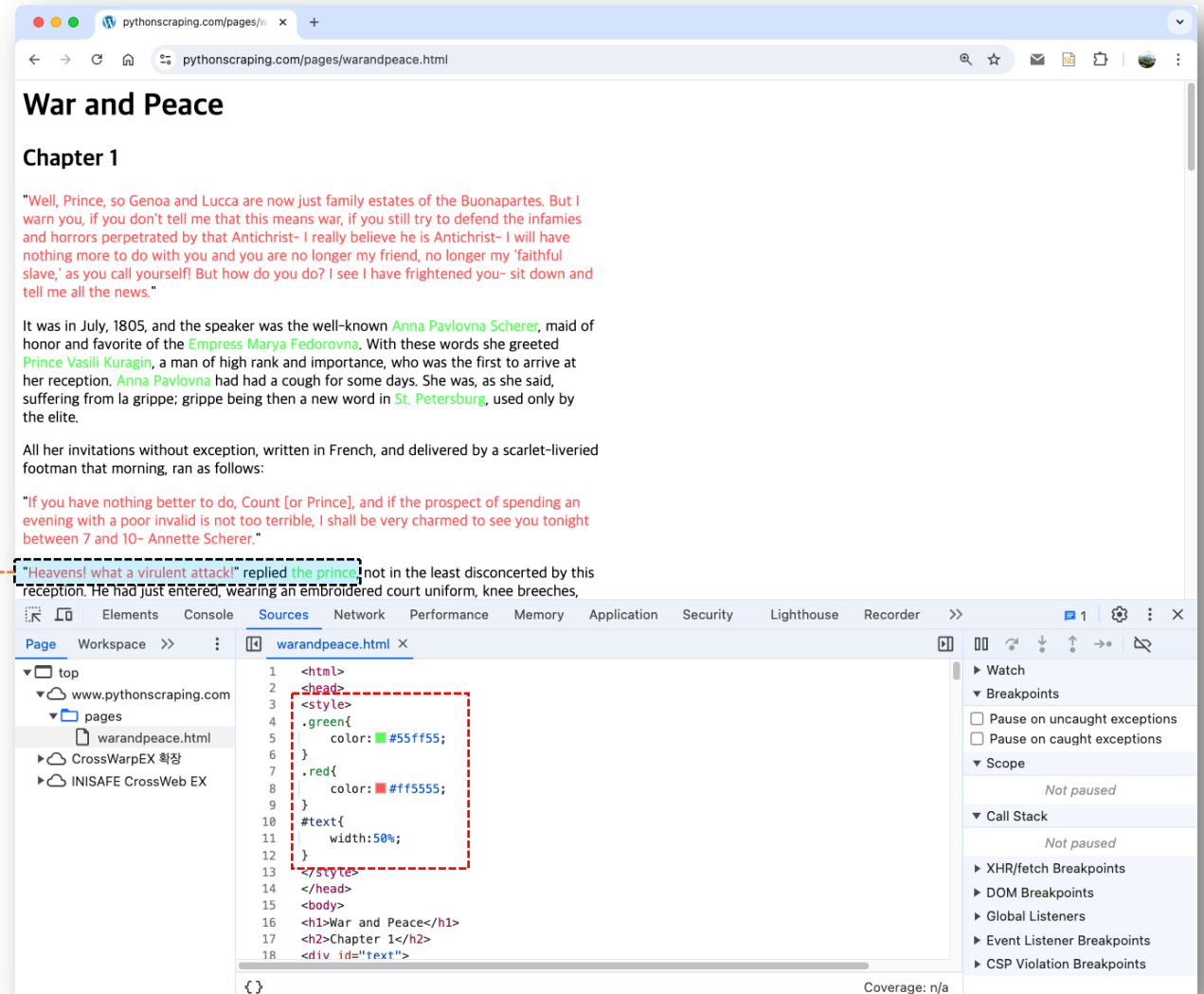
■ <http://www.pythonscraping.com/pages/warandpeace.html>

- 등장인물 대사: 빨간색
- 등장인물 이름: 녹색

```
<html>
<head>
<style>
.green{
  color:#55ff55;
}
.red{
  color:#ff5555;
}
#text{
  width:50%;
}
</style>
</head>
```

```
"<span class="red">Heavens! what a
virulent attack!</span>" replied
<span class="green">the prince</span>
```

: 인라인 요소들을 하나로 묶을 때 사용
- 은 줄 바꿈 안됨



CSS 속성을 이용한 검색

- 속성(attrs) 사용

- find() 함수에 이름, 속성, 속성값을 이용하여 원하는 태그를 검색
 - 맨 처음 검색 결과만 리턴

<chap03_html01.py>

```
from bs4 import BeautifulSoup

html_text='<span class="red">Heavens! what a virulent attack!</span>'
soup = BeautifulSoup(html_text, 'html.parser')

object_tag = soup.find('span')
print('object_tag:', object_tag)
print('attrs:', object_tag.attrs)
print('value:', object_tag.attrs['class'])
print('text:', object_tag.text)
```

attrs: 딕셔너리 형태로 리턴

```
object_tag: <span class="red">Heavens! what a virulent attack!</span>
attrs: {'class': ['red']}
value: ['red']
text: Heavens! what a virulent attack!
```

CSS 속성을 이용한 검색

■ CSS 속성을 이용한 태그 검색 (등장 인물 검색)

<chap03_html02.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
```

```
html = urlopen('http://www.pythonscraping.com/pages/warandpeace.html')
soup = BeautifulSoup(html, "html.parser")
```

등장인물의 이름: 녹색

```
name_list = soup.find_all('span', {'class': 'green'})
for name in name_list:
    print(name.string)
```

등장인물의 이름은 모두 녹색
...
모두 검색

• string: tag를 제외한 텍스트만 반환 (get_text() 또는 text 사용 가능)

```
Anna
Pavlovna Scherer
Empress Marya
Fedorovna
Prince Vasili Kuragin
Anna Pavlovna
St. Petersburg
the prince
Anna Pavlovna
Anna Pavlovna
...
```

줄 바꿈 문자 포함 (\n)

It was in July, 1805, and the speaker was the well-known Anna Pavlovna Scherer, maid of honor and favorite of the Empress Marya Fedorovna. With these words she greeted Prince Vasili Kuragin, a man of high rank and importance, who was the first to arrive at her reception. Anna Pavlovna had had a cough for some days. She was, as she said, suffering from la grippe; grippe being then a new word in St. Petersburg, used only by the elite.

```
> html = {HTTPResponse} <http.client.HTTPResponse object at 0x106b658a0>
> name = {Tag} <span class="green">Anna\nPavlovna Scherer</span>
> name_list = {ResultSet: 41} [<span class="green">Anna\nPavlovna Scherer</span>, <span class="green">Anna\nPavlovna Scherer</span>, ...]
> soup = {BeautifulSoup} <html>\n<head>\n<style>\n.green{\ntcolor:#55ff55;\n}\n.red{\ntcolor:#ff5555;\n}\n
> Special Variables
```

특정 단어 찾기

- find_all(string="검색어")
 - 대소문자 구분
 - 검색어: 'the prince'

<chap03_html03.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://www.pythonscraping.com/pages/warandpeace.html')
soup = BeautifulSoup(html, 'html.parser')

prince_list = soup.find_all(string='the prince')
print(prince_list)
print('the prince count: ', len(prince_list))
```

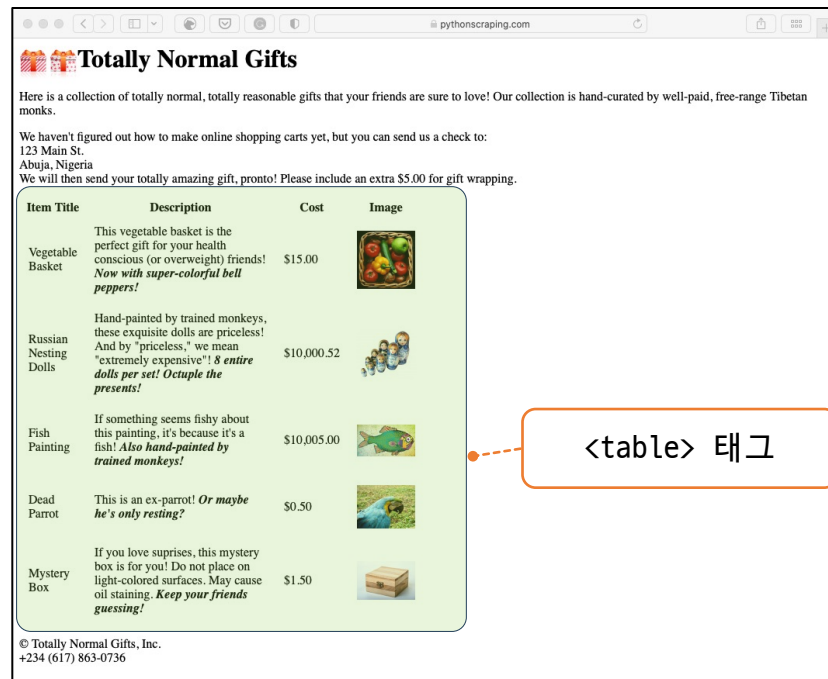
```
['the prince', 'the prince', 'the prince', 'the prince', 'the prince', 'the prince', 'the prince']
the prince count: 7
```

트리 이동






- 트리 이동
 - 문서 내부에서 특정 위치를 기준으로 태그를 찾을 때
 - 단방향으로 트리 이동

```
soup.tag.subTag.anotherSubTag
```

- 온라인 쇼핑 사이트 구성 및 트리 이동
 - <https://www.pythonscraping.com/pages/page3.html>



The screenshot shows a web browser displaying the 'Totally Normal Gifts' website. The page has a header with the site name and a paragraph of text. Below the text is a table with four columns: 'Item Title', 'Description', 'Cost', and 'Image'. The table contains five rows of gift items. A red dashed line points from the table to a callout box.

Item Title	Description	Cost	Image
Vegetable Basket	This vegetable basket is the perfect gift for your health conscious (or overweight) friends! <i>Now with super-colorful bell peppers!</i>	\$15.00	
Russian Nesting Dolls	Hand-painted by trained monkeys, these exquisite dolls are priceless! And by 'priceless,' we mean 'extremely expensive'! <i>8 entire dolls per set! Octuple the presents!</i>	\$10,000.52	
Fish Painting	If something seems fishy about this painting, it's because it's a fish! <i>Also hand-painted by trained monkeys!</i>	\$10,005.00	
Dead Parrot	This is an ex-parrot! <i>Or maybe he's only resting?</i>	\$0.50	
Mystery Box	If you love surprises, this mystery box is for you! Do not place on light-colored surfaces. May cause oil staining. <i>Keep your friends guessing!</i>	\$1.50	

© Totally Normal Gifts, Inc.
+234 (617) 863-0736

<table> 태그

```
<html>
  <head>
    <style>...</style>
  </head>
  <body> == $0
    <div id="wrapper">
      
      <h1>Totally Normal Gifts</h1>
      <div id="content">
        "Here is a collection of totally normal, totally reasonable gifts
        that your friends are sure to love! Our collection is hand-curated
        by well-paid, free-range Tibetan monks."
      <p>...</p>
    </div>
    <table id="giftList">
      <tbody>
        <tr>
          <th> Item Title </th>
          <th> Description </th>
          <th> Cost </th>
          <th> Image </th>
        </tr>
        <tr id="gift1" class="gift">...</tr>
        <tr id="gift2" class="gift">...</tr>
        <tr id="gift3" class="gift">...</tr>
        <tr id="gift4" class="gift">...</tr>
        <tr id="gift5" class="gift">...</tr>
      </tbody>
    </table>
    <p></p>
    <div id="footer">...</div>
  </div>
</body>
</html>
```


온라인 쇼핑몰 테이블 구성 현황

```
<table id="giftList">
```

```
<tr>  <th>..</th>
```






```
<tr id="gift1",  
  class="gift">
```

```
<tr id="gift2",  
  class="gift">
```

```
<tr id="gift3",  
  class="gift">
```

```
<tr id="gift4",  
  class="gift">
```

```
<tr id="gift5",  
  class="gift">
```

Item Title	Description	Cost	Image
Vegetable Basket	This vegetable basket is the perfect gift for your health conscious (or overweight) friends! <i>Now with super-colorful bell peppers!</i>	\$15.00	
Russian Nesting Dolls	Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless," we mean "extremely expensive"! <i>8 entire dolls per set! Octuple the presents!</i>	\$10,000.52	
Fish Painting	If something seems fishy about this painting, it's because it's a fish! <i>Also hand-painted by trained monkeys!</i>	\$10,005.00	
Dead Parrot	This is an ex-parrot! <i>Or maybe he's only resting?</i>	\$0.50	
Mystery Box	If you love surprises, this mystery box is for you! Do not place on light-colored surfaces. May cause oil staining. <i>Keep your friends guessing!</i>	\$1.50	

트리 이동: 자식과 자손

■ 자식: children

<chap03_html04_01.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://www.pythonscraping.com/pages/page3.html')
soup = BeautifulSoup(html, 'html.parser')

table_tag = soup.find('table', {'id': 'giftList'})
print('children 개수: ', len(list(table_tag.children)))

index=0
for child in table_tag.children:
    index += 1
    print(f"[{index}]: {child}")
    print('-' * 30)
```

table_tag의 모든
자식 tag를 가짐

children 개수: 13
[1]:

[2]: <tr><th>
Item Title
</th><th>
Description
</th><th>
Cost
</th><th>
Image
</th></tr>

<tr> 단위: 행 목록

[3]:

[4]: <tr class="gift" id="gift1"><td>
Vegetable Basket
</td><td>
This vegetable basket is the perfect
gift for your health conscious (or
overweight) friends!
Now with
super-colorful bell peppers!
</td><td>
\$15.00
</td><td>
. . . (중간 생략)

트리 이동: 자식과 자손

■ 자식: children

The screenshot shows a Python IDE with a file named `chap03_html03_01.py`. The code is using BeautifulSoup to parse an HTML document. The `WATCH` panel on the left shows the state of variables, including `table_tag` and its `contents` attribute. A callout box highlights that the newline character `'\n'` is included in the `children` list. The code iterates over `table_tag.children` and prints the index and child for each.

```
1 '''
2     Chap02. BeautifulSoup 활용
3     트리 이동 (table)
4 '''
5 from urllib.request import urlopen
6 from bs4 import BeautifulSoup
7
8 html = urlopen('http://www.pythonscraping.com/pages/page3.html')
9 soup = BeautifulSoup(html, 'html.parser')
10
11 # 1. children
12 table_tag = soup.find('table', {'id': 'giftList'})
13 print('children 개수: ', len(list(table_tag.children)))
14 index=0
15 for child in table_tag.children:
16     index +=1
17     print(f'[{index}]: {child}')
18     print('-' * 30)
19
```

Variables panel (WATCH):

- `table_tag` = `<table id="giftList">`
- `contents` = `['\n', <tr><th>`
- `len()` = 13

Terminal output:

```
children 개수: 13
```

트리 이동: 자손

■ 자손: descendants

<chap03_html04_02.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://www.pythonscraping.com/pages/page3.html')
soup = BeautifulSoup(html, 'html.parser')
# 자손: descendants
desc = soup.find('table', {'id': 'giftList'}).descendants
list_desc = list(desc)
print('descendants 개수: ', len(list_desc))

for tag in list_desc:
    print(tag)
```

<tr> 분리
<th> 분리
<th> 내부 분리

descendants 개수: 86

```
<tr><th>
Item Title
</th><th>
Description
</th><th>
Cost
</th><th>
Image
</th></tr>
```

```
<td>
$1.50
</td>

$1.50

<td>

</td>


```

트리 이동: 형제 다루기 #1

- 형제: `next_siblings` 속성
 - 임의의 행을 선택하고 `next_siblings`을 선택하면,
 - 테이블의 다음 행들을 모두 선택 (모든 형제를 선택)

<chap03_html04_03.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
```

```
html = urlopen('http://www.pythonscraping.com/pages/page3.html')
soup = BeautifulSoup(html, 'html.parser')
```

```
# next_siblings 속성
```

```
for sibling in soup.find('table', {'id': 'giftList'}).tr.next_siblings:
    print(sibling)
    print('-' * 30)
```

giftList의 첫 번째 tr선택

```
-----
<tr class="gift" id="gift1"><td>
Vegetable Basket
</td><td>
This vegetable basket is the perfect gift for your health conscious
<span class="excitingNote">Now with super-colorful bell peppers!</span>
</td><td>
$15.00
</td><td>

</td></tr>
```

next_siblings

over

Item Title	Description	Cost	Image
Vegetable Basket	This vegetable basket is the perfect gift for your health conscious (or overweight) friends! <i>Now with super-colorful bell peppers!</i>	\$15.00	
Russian Nesting Dolls	Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless," we mean "extremely expensive"! <i>8 entire dolls per set! Octuple the presents!</i>	\$10,000.52	
Fish Painting	If something seems fishy about this painting, it's because it's a fish! <i>Also hand-painted by trained monkeys!</i>	\$10,005.00	
Dead Parrot	This is an ex-parrot! <i>Or maybe he's only resting?</i>	\$0.50	
Mystery Box	If you love surprises, this mystery box is for you! Do not place on light-colored surfaces. May cause oil staining. <i>Keep your friends guessing!</i>	\$1.50	

트리 이동: 형제 다루기 #2

■ previous_siblings 속성

- 선택된 행 이전의 항목들을 선택

<chap03_html04_03.py>

```
print('previous_siblings')
for sibling in soup.find('tr', {'id': 'gift2'}).previous_siblings:
    print(sibling)
```






previous_siblings

```
<tr class="gift" id="gift1"><td>
Vegetable Basket
</td><td>
This vegetable basket is the perfect gift for your health conscious (or overwe
<span class="excitingNote">Now with super-colorful bell peppers!</span>
</td><td>
$15.00
</td><td>

</td></tr>
```

```
<tr><th>
Item Title
</th><th>
Description
</th><th>
Cost
</th><th>
Image
</th></tr>
```

previous_siblings

Item Title	Description	Cost	Image
Vegetable Basket	This vegetable basket is the perfect gift for your health conscious (or overweight) friends! <i>Now with super-colorful bell peppers!</i>	\$15.00	
Russian Nesting Dolls	Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless," we mean "extremely expensive"! <i>8 entire dolls per set! Octuple the presents!</i>	\$10,000.52	
Fish Painting	If something seems fishy about this painting, it's because it's a fish! <i>Also hand-painted by trained monkeys!</i>	\$10,005.00	
Dead Parrot	This is an ex-parrot! <i>Or maybe he's only resting?</i>	\$0.50	
Mystery Box	If you love surprises, this mystery box is for you! Do not place on light-colored surfaces. May cause oil staining. <i>Keep your friends guessing!</i>	\$1.50	

선택
(gift2)

트리 이동: 형제 다루기 #3

■ next_sibling, previous_sibling

- 태그 하나만 반환
- 문자열 마지막에 whitespace('\n', '\r' 등)가 있는 경우
 - 해당 whitespace를 next_sibling으로 반환

<chap03_html04_03.py>

```
sibling1 = soup.find('tr', {'id': 'gift3'}).next_sibling
print('sibling1:', sibling1)
print(ord(sibling1)) # ord(문자): 문자의 Unicode 정수를 리턴
```

'\n'
10

출력되지 않음

- next_sibling.next_sibling 이용

```
sibling2 = soup.find('tr', {'id': 'gift3'}).next_sibling.next_sibling
print(sibling2)
```

```
<tr class="gift" id="gift4"><td>
Dead Parrot
</td><td>
This is an ex-parrot! <span class="excitingNote">Or maybe he's only resting?</span>
</td><td>
$0.50
</td><td>

</td></tr>
```

트리 이동: 부모 다루기 #1

■ parent 사용

```
style_tag = soup.style
print(style_tag.parent)
```

```
<head>
<style>
img{
  width:75px;
}
table{
  width:50%;
}
td{
  margin:10px;
  padding:10px;
}
.wrapper{
  width:800px;
}
.excitingNote{
  font-style:italic;
  font-weight:bold;
}
</style>
</head>
```

style의 parent
(head)

<chap03_html04_03.py>

```
<html>
<head>
<style>
img{
  width:75px;
}
table{
  width:50%;
}
td{
  margin:10px;
  padding:10px;
}
.wrapper{
  width:800px;
}
.excitingNote{
  font-style:italic;
  font-weight:bold;
}
</style>
</head>
```

style

트리 이동: 부모 다루기 #2

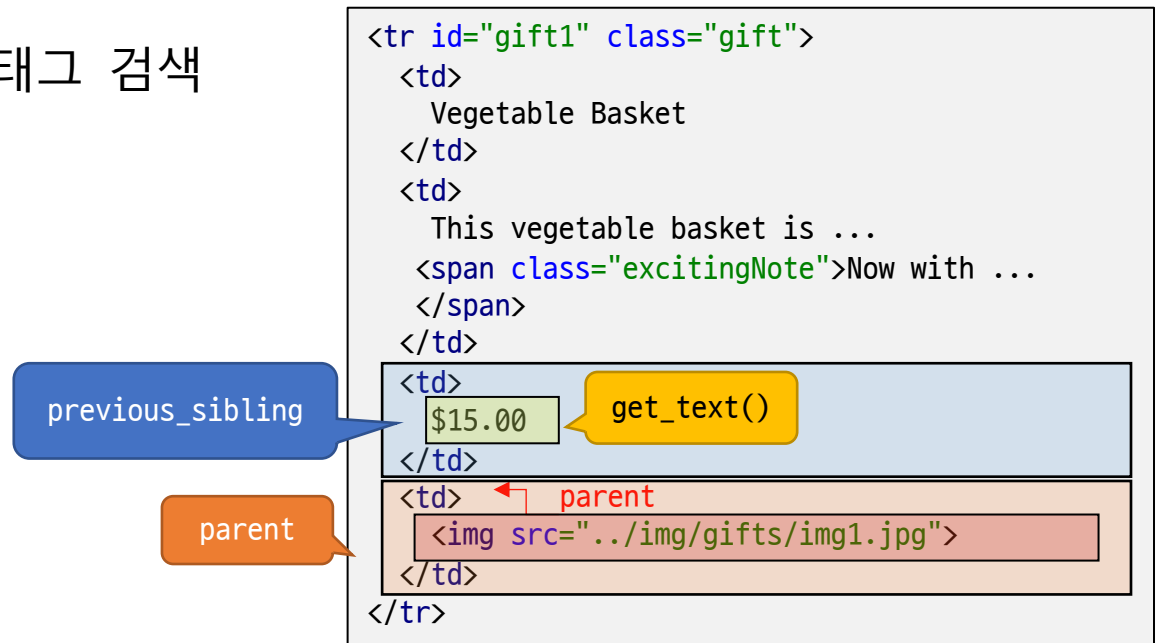
■ .parent 사용

<chap03_html04_03.py>

```
img1 = soup.find('img', {'src': '../img/gifts/img1.jpg'})  
text = img1.parent.previous_sibling.get_text()  
print(text)
```

\$15.00

- parent: 부모 tag를 먼저 검색(<td>)
- previous_sibling: 부모 태그의 이전 형제 태그 검색
- get_text(): 태그 내부의 문자열 반환



정규 표현식 (Regular Expression)

- 정규 표현식

- 특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어

- 정규 표현식 사용

- 문자열과 관련된 문제 해결을 위해 사용
- 문자열 치환, 검색, 추출 등
- 문자열의 유효성 검사
 - Email 주소, 전화번호, 웹사이트 주소 등

- 장점

- 다양한 입력 문자열 처리가 간결
- 범용성: 다양한 프로그래밍 언어에서 지원
- 생산성 향상

- 단점

- 정규 표현식 자체의 어려움
- 소스 코드가 어려워짐

정규 표현식 기호: 문자 집합

■ 문자 집합

표현식	설명
<code>^</code>	• 문자열의 시작
<code>\$</code>	• 문자열의 종료
<code>.</code>	• 임의의 한 문자 (문자의 종류는 상관 없음)
<code>\b</code>	• 단어의 경계(공백) 검색
<code>\s</code>	• 공백 문자 (space, tab, carriage return, line feed, form feed)
<code>\S</code>	• 공백 문자가 아닌 나머지 문자
<code>\w</code>	• 알파벳, 숫자, <code>_</code> (underscore)를 검색, <code>[a-zA-Z_0-9]</code> 와 동일
<code>\W</code>	• 알파벳, 숫자, <code>_</code> 를 제외한 문자: <code>[^\w]</code> 와 동일
<code>\d</code>	• 숫자, <code>[0-9]</code> 와 동일 (digit)
<code>\D</code>	• 숫자를 제외한 모든 문자: <code>[^0-9]</code> 와 동일
<code>\</code>	• 확장 문자, 역 슬래시 다음에 일반 문자가 오면 특수 문자로 취급 • 역 슬래시 다음에 특수 문자가 따라오면 그 문자 자체를 의미 • <code>*</code> 는 <code>*</code> 자체를 의미
<code>r'패턴'</code>	• 컴파일 할 정규식이 순수 문자열임을 알림 (raw string) • 역슬래시(<code>\</code>)를 1번만 사용하여 <code>'\'</code> 문자 자체를 표현하기 위함

정규 표현식: 그룹과 범위

■ 그룹과 범위 지정

표현식	설명
[]	<ul style="list-style-type: none">문자의 집합이나 범위를 나타냄. 두 문자 사이는 '-' 기호로 범위를 나타냄[a-z]: 알파벳 소문자 모두
[^]	<ul style="list-style-type: none">[] 내부에 ^ (caret) 기호가 선행하면 not을 나타냄[^abc]: a,b,c 제외
()	<ul style="list-style-type: none">소괄호 안의 문자를 하나의 문자로 인식 (그룹)
	<ul style="list-style-type: none">패턴 안에서 OR 연산을 수행할 때 사용
(?i)	<ul style="list-style-type: none">대소문자를 구분하지 않음
(?:)	<ul style="list-style-type: none">뒤에 따라 나오는 문자들을 하나의 그룹으로 합치기 위함
?(regex)	<ul style="list-style-type: none">정규식(regex)과 일치하는 문자열을 만나면 그 정규식(regex) 앞의 값을 반환전방(?) 긍정(=) 탐색.*(?:)
?!(regex)	<ul style="list-style-type: none">전방(?) 부정(!) 탐색 (전반 긍정 탐색의 부정)
?<=(regex)	<ul style="list-style-type: none">정규식(regex)과 일치하는 문자열을 만나면 그 정규식 뒤의 값을 반환후방(?<) 긍정(=) 탐색
?<!(regex)	<ul style="list-style-type: none">후방(?<) 부정(!) 탐색 (후방 긍정 탐색의 부정)

정규 표현식: 수량 표시

- 수량자: 패턴 발생 수를 표현

수량자	설명
*	<ul style="list-style-type: none">• 앞 문자가 없을 수도 있고 무한정 많을 수도 있음: (zero or more)• x^*
+	<ul style="list-style-type: none">• 앞 문자가 하나 이상: (one or more)• x^+
?	<ul style="list-style-type: none">• 앞 문자가 없거나 하나가 있음: (zero or one)• $1?$
{n}	<ul style="list-style-type: none">• 정확히 n개• $x\{n\}$
{n,}	<ul style="list-style-type: none">• 최소 n개• $x\{n, \}$
{n, m}	<ul style="list-style-type: none">• n개에서 m개까지• $x\{n, m\}$
*?	<ul style="list-style-type: none">• 가장 적게 일치하는 패턴 검색

자주 사용하는 정규 표현식

설명	정규 표현식
• 숫자	<code>^[0-9]*\$</code>
• 영문자	<code>^[a-zA-Z]*\$</code>
• 한글	<code>^[가-힣]*\$</code>
• 영문자와 숫자	<code>^[a-zA-Z0-9]*\$</code>
• 이메일	<code>^[a-zA-Z0-9]+@[a-zA-Z0-9]+\$</code>
• 휴대전화번호	<code>^(01(?:0 1 [6-9]))-(\d{3,4})-(\d{4})\$</code>
• 일반전화	<code>^(\d{2,3})-(\d{3,4})-(\d{4})\$</code>
• 주민등록번호	<code>^(\d{6})[-][1-4](\d{6})\$</code>
• IP주소	<code>^([0-9]{1,3})\.([0-9]{1,3})\.([0-9]{1,3})\.([0-9]{1,3})\$</code>
• 비밀번호	<code>^[a-z0-9_-]{6,18}\$</code> 소문자, 숫자, _, - 포함하여 6글자 이상 18글자 이하

정규 표현식: re 모듈 함수

■ re (regular expression) 모듈 함수

함수	설명
<code>compile(pattern, flags=0)</code>	pattern을 이용하여 정규식 객체를 반환 - 패턴을 여러 번 사용할 경우, <code>compile()</code> 함수를 이용하여 객체 생성
<code>search(pattern, string)</code>	문자열 전체를 검색하여 pattern이 존재하는지 검색 - 처음 매칭되는 문자열 리턴
<code>match(pattern, string)</code>	string의 처음부터 pattern이 존재하는지 검사 - 공백이 있는 경우나 중간에 패턴이 존재하는 경우 검색하지 못함 - 문장 처음부터 패턴과 일치하는 문자열만 검색
<code>split(pattern, string)</code>	pattern을 구분자로 string을 분리해서 리스트로 반환
<code>findall(pattern, string)</code>	string에서 pattern과 매치되는 모든 경우를 찾아 리스트로 반환
<code>finditer(pattern, string)</code>	string에서 pattern과 매치되는 문자열을 반복 가능한 객체로 반환
<code>sub(pattern, repl, string)</code>	string에서 pattern과 일치하는 부분을 repl로 교체한 문자열 반환
<code>subn(pattern, repl, string, count)</code>	sub()와 동일, 결과로 (결과 문자열, 매칭 횟수)를 튜플로 반환
<code>escape(string)</code>	영문자, 숫자가 아닌 문자에 대해 백슬래시 문자를 추가

정규 표현식 예제 #1

- 정규 표현식 객체 사용:
 - 정규식 객체를 생성: `compile(pattern)`
 - 동일 패턴을 여러 번 검색하는 경우, 편리하게 사용
 - re모듈 함수들은 pattern 파라미터 없이 호출이 가능
 - `search(string, pos)`, `match(string, pos)` 등

<chap03_regex01.py>

```
import re
```

```
# compile() 사용 안함
```

```
m = re.match('[a-z]+', 'Python')  
print(m)  
print(re.search('apple', 'I like apple!'))
```

정규식 객체 생성 안함
- 매번 패턴 입력 (소문자)

```
# compile() 사용: 객체 생성
```

```
p = re.compile('[a-z]+') # 알파벳 소문자  
m = p.match('python')  
print(m)  
print(p.search('I like apple 123'))
```

정규식 객체(p) 생성
- 알파벳 소문자 패턴
- 여러 번 사용

```
None -> 'Python'은 대문자로 시작하기 때문에 match()함수의 리턴값이 None  
<re.Match object; span=(7, 12), match='apple'>
```

```
<re.Match object; span=(0, 6), match='python'>  
<re.Match object; span=(2, 6), match='like'>
```


정규 표현식 예제 #12

- match(): 문자열의 처음부터 검사

```
m = re.match('[a-z]+', 'pythoN ') # 소문자가 1개 이상  
print(m)
```

```
<re.Match object; span=(0, 5), match='pytho'>
```

```
m = re.match('[a-z]+', 'PYthon') # 소문자가 1개 이상  
print(m)
```

```
None
```

```
print(re.match('[a-z]+', 'regex python'))  
print(re.match('[a-z]+', ' regexpython'))
```

```
print(re.match('[a-z]+', 'regexpythoN'))  
print(re.match('[a-z]+$', 'regexpythoN'))
```

```
print(re.match('[a-z]+', 'regexPython'))  
print(re.match('[a-z]+$', 'regexpython'))
```



```
<re.Match object; span=(0, 5), match='regex'>  
None # 문자열 처음에 공백 포함
```

```
<re.Match object; span=(0, 10), match='regexpytho'>  
None # $: 문자열의 마지막에 소문자 1회 이상 검사
```

```
<re.Match object; span=(0, 5), match='regex'>  
<re.Match object; span=(0, 11), match='regexpython'>
```

정규 표현식 예제 #3

■ findall() 함수

- 일치하는 모든 문자열을 리스트로 리턴

```
p = re.compile('[a-z]+') # 알파벳 소문자
```

```
print(p.findall('life is too short! Regular expression test'))
```

```
['life', 'is', 'too', 'short', 'egular', 'expression', 'test']
```

■ search() 함수

- 일치하는 첫 번째 문자열만 리턴

```
result = p.search('I like apple 123')
```

```
print(result)
```

```
result = p.findall('I like apple 123')
```

```
print(result)
```

```
<re.Match object; span=(2, 6), match='like'>
```

```
like
```

```
['like', 'apple']
```

정규 표현식: Match 객체 메소드

■ Match 객체

함수	설명
<code>group([group1, ...])</code>	매치된 문자열 중 인덱스에 해당하는 문자열을 반환 <code>group(0)</code> 또는 <code>group()</code> : 전체 매칭 문자열 반환
<code>groups()</code>	매칭된 결과를 튜플 형태로 반환
<code>groupdict()</code>	매칭 결과를 사전(dict) 형태로 반환
<code>start([group])</code>	매칭된 결과 문자열의 시작 위치를 반환
<code>end([group])</code>	매칭된 문자열의 끝 위치를 반환
<code>span()</code>	매치된 문자열의 (시작, 끝)에 해당하는 튜플을 반환

Match 메소드 예제 #1

■ 전화번호 분석

- 전화번호: '지역번호-국번-전화번호'
 - 전화번호: (2, 3자리)-(3, 4자리)-(4자리)
 - 예: 02-123-4567, 053-123-1234
- groups(): 매칭되는 문자열의 전체 그룹을 리턴

```
import re
# ^ .. $ 을 명시해야 정확한 자리수 검사가 이루어짐
tel_checker = re.compile(r'^(\d{2,3})-(\d{3,4})-(\d{4})$')

print(tel_checker.match('02-123-4567'))
match_groups = tel_checker.match('02-123-4567').groups()
print(match_groups)

print(tel_checker.match('053-950-45678'))
print(tel_checker.match('053950-4567'))
```

Python 12버전부터 정규식 패턴을 raw string으로
표시해야 됨
(SyntaxWarning: invalid escape sequence 발생)

마지막 숫자의 자리수가 맞지 않음

dash(-)가 없음

```
<re.Match object; span=(0, 11), match='02-123-4567'>
('02', '123', '4567')
None
None
```

Match 메소드 예제 #2

- 전화번호에서 dash(-) 제거하고 검사하기

```
tel_number = '053-950-4567'
tel_number = tel_number.replace('-', '')
print(tel_number)

tel_checker1 = re.compile(r'^(\d{2,3})(\d{3,4})(\d{4})$')
print(tel_checker1.match(tel_number))
print(tel_checker1.match('0239501234')) # 02-3950-1234
```

```
0539504567
<re.Match object; span=(0, 10), match='0539504567'>
<re.Match object; span=(0, 10), match='0239501234'>
```

Match 메소드 예제 #3

- `groups()`
 - 매칭 결과를 튜플로 출력
- `group()`
 - 매칭된 전체 문자열 반환
- `group(index)`
 - 해당 인덱스에 매칭된 문자열 반환
 - `index=0`: 전체 리턴

```
tel_checker = re.compile('^(\d{2,3})-(\d{3,4})-(\d{4})$')
m = tel_checker.match('02-123-4567')

print(m.groups())
print('group(): ', m.group())
print('group(0): ', m.group(0))
print('group(1): ', m.group(1))
print('group(2,3): ', m.group(2,3))
print('start(): ', m.start()) # 매칭된 문자열의 시작 인덱스
print('end(): ', m.end()) # 매칭된 문자열의 마지막 인덱스+1
```

```
('02', '123', '4567')
group(): 02-123-4567
group(0): 02-123-4567
group(1): 02
group(2,3): ('123', '4567')
start(): 0
end(): 11
```

Match 메소드 예제 #3

■ 휴대전화번호

- 휴대전화번호 구성: '사업자번호(3자리)-국번(3,4자리)-전화번호(4자리)'
 - 사업자 번호: 010, 011, 016, 017, 018, 019
 - 예: 010-123-4567, 011-1234-5678, 019-111-2222
- (?:0|1|[6-9]) 의미
 - ?: 뒤에 따라 나오는 숫자(0|1|6|7|8|9)를 하나의 그룹으로 합침

```
cell_phone = re.compile('^01(?:0|1|[6-9]))-(\d{3,4})-(\d{4})$')
```

```
print(cell_phone.match('010-123-4567'))  
print(cell_phone.match('019-1234-5678'))  
print(cell_phone.match('001-123-4567'))  
print(cell_phone.match('010-1234567'))
```

```
<re.Match object; span=(0, 12), match='010-123-4567'>  
<re.Match object; span=(0, 13), match='019-1234-5678'>  
None  
None
```

전방 탐색(lookahead)

- 전방 긍정 탐색
 - 패턴과 일치하는 문자열을 만나면 패턴 앞의 문자열 반환: (?=패턴)
- 전방 부정 탐색(?!)
 - 패턴과 일치하지 않는 문자열을 만나면 패턴 앞의 문자열 반환: (?!패턴)

```
import re

# 전방 긍정 탐색: (문자열이 won을 포함하고 있으면 won 앞의 문자열 리턴)
lookahead1 = re.search('.*(?=won)', '1000 won')
if(lookahead1 != None):
    print(lookahead1.group())
else:
    print('None')

lookahead2 = re.search('.*(?=am)', '2023-01-26 am 10:00:01')
print(lookahead2.group())

# 전방 부정 탐색 (?!): 4자리 숫자 다음에 '-'를 포함하지 않으면 앞의 문자열 리턴
lookahead3 = re.search('\d{4}(?!-)', '010-1234-5678')
print(lookahead3)
```

```
1000
2023-01-26
<re.Match object; span=(9, 13), match='5678'>
```


후방 탐색(lookbehind)

- 후방(?<) 긍정(=) 탐색
 - 패턴과 일치하는 문자열을 만나면 패턴 뒤의 문자열 반환: (?<=패턴)
- 후방(?<) 부정(!) 탐색
 - 패턴과 일치하지 않는 문자열을 만나면 패턴 뒤의 문자열 반환: (?<!패턴)

```
# 후방 긍정 탐색 ('am' 다음에 문자가 1개 이상 있으면, 해당 문자열 리턴)
lookbehind1 = re.search('(?<=am).+', '2023-01-26 am 11:10:01')
print(lookbehind1)

lookbehind2 = re.search('(?<=:).+', 'USD: $51')
print(lookbehind2)

# 후방 부정 탐색 ('\b': 공백)
# 공백 다음에 $기호가 없고 숫자가 1개 이상이고 공백이 있는 경우
lookbehind4 = re.search(r'\b(?<!\$)\d+\b', 'I paid $30 for 100 apples.')
print(lookbehind4)
```

```
<re.Match object; span=(13, 22), match=' 11:10:01'>
<re.Match object; span=(4, 8), match=' $51'>
<re.Match object; span=(15, 18), match='100'>
```

정규 표현식과 BeautifulSoup #1

- BeautifulSoup의 문자열을 받는 함수들
 - 정규 표현식을 매개 변수로 받을 수 있음
- 제품 이미지 검색:
 - `` 태그의 속성['src'] 사용

<chap03_regex02.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

html = urlopen('http://www.pythonscraping.com/pages/page3.html')
soup = BeautifulSoup(html, 'html.parser')

# 정규식: ('img.*\.jpg'): img 다음에 임의의 한 문자가 0회 이상
# - img.jpg, img1.jpg, imga.jpg 등
img_tag = re.compile('/img/gifts/img.*.jpg')
# find_all()에서 img의 src 속성값에 정규식 사용
images = soup.find_all('img', {'src': img_tag})

for image in images:
    print(image, end=" -> ['src'] 속성값: ")
    print(image['src'])
```

```
 -> ['src'] 속성값: ../../../img/gifts/img1.jpg
 -> ['src'] 속성값: ../../../img/gifts/img2.jpg
 -> ['src'] 속성값: ../../../img/gifts/img3.jpg
 -> ['src'] 속성값: ../../../img/gifts/img4.jpg
 -> ['src'] 속성값: ../../../img/gifts/img6.jpg
```

정규 표현식과 BeautifulSoup #2

- 대소문자 구분없이 특정 단어 검색
 - '[T|t]{1}he prince'
 - T 또는 t가 1회

<chap03_regex03.py>

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

html = urlopen('http://www.pythonscraping.com/pages/warandpeace.html')
bs = BeautifulSoup(html, 'html.parser')

princeList = bs.find_all(string='the prince')
print('the prince count: ', len(princeList))

prince_list = bs.find_all(string=re.compile('[T|t]{1}he prince'))
print('T|the prince count:', len(prince_list))
```

```
the prince count: 7
T|the prince count: 11
```

참고 자료

Python 정규식 테스트 사이트

■ pythex

- <https://pythex.org>

pythex

Your regular expression:

/

/img/gifts/img.*.jpg

/

IGNORECASE

MULTILINE

DOTALL

VERBOSE

Your test string:

Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless," we mean "extremely expensive"! 8 entire dolls per set! Octuple the presents!
</td><td>
\$10,000.52
</td><td>

Match result:

```
<html>
<head>
<style>
img{
  width:75px;
}
table{
  width:50%;
}
td{
  margin:10px;
  padding:10px;
}
```

Match captures:

No groups.

No groups.

No groups.

No groups.

No groups.

Match result

```

</td></tr>

<tr id="gift2" class="gift"><td>
Russian Nesting Dolls
</td><td>
Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless,"
we mean "extremely expensive"! <span class="excitingNote">8 entire dolls per set!
Octuple the presents!</span>
</td><td>
$10,000.52
</td><td>

</td></tr>

<tr id="gift3" class="gift"><td>
Fish Painting
</td><td>
If something seems fishy about this painting, it's because it's a fish! <span
class="excitingNote">Also hand-painted by trained monkeys</span>
</td><td>
$10,005.00
</td><td>

</td></tr>

<tr id="gift4" class="gift"><td>
Dead Parrot
</td><td>
This is an ex-parrot! <span class="excitingNote">Or maybe he's only resting?</span>
</td><td>
$0.50
</td><td>

</td></tr>

<tr id="gift5" class="gift"><td>
Mystery Box
</td><td>
If you love surprises, this mystery box is for you! Do not place on light-colored surfaces.
May cause oil staining. <span class="excitingNote">Keep your friends guessing!</span>
</td><td>
$1.50
</td><td>

```

정규 표현식 테스트 사이트

- RegExr

- <https://regexr.com>

- regex101

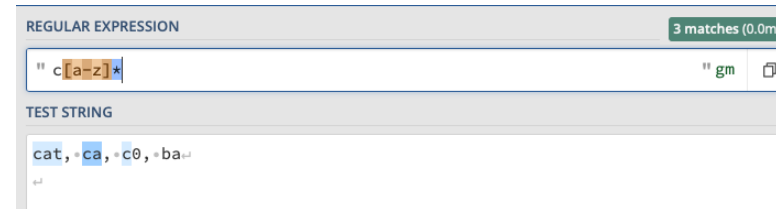
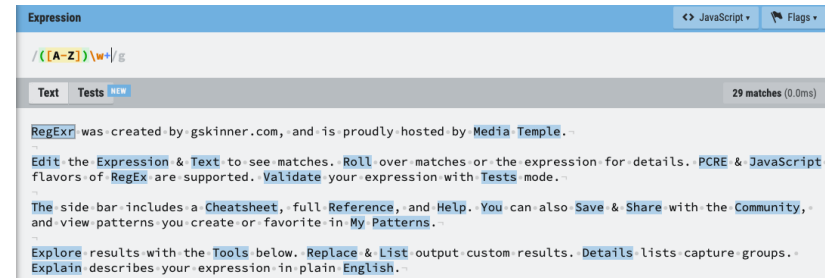
- <https://regex101.com>

- RegexPlanet

- <https://www.regexplanet.com>

- regexper

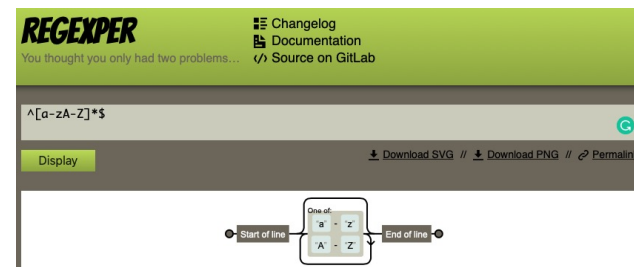
- <https://regexper.com>
- 그래픽으로 정규식을 표현



Online Regular Expression Testing

Regular expressions are an incredibly powerful tool, but can be rather tricky to get exactly right. This is a website that I wrote so I could quickly and easily test regular expressions during development.

Pick which programming language you are using:





Questions?