

데이터 크롤링과 정제

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

War and Peace

Chapter 1

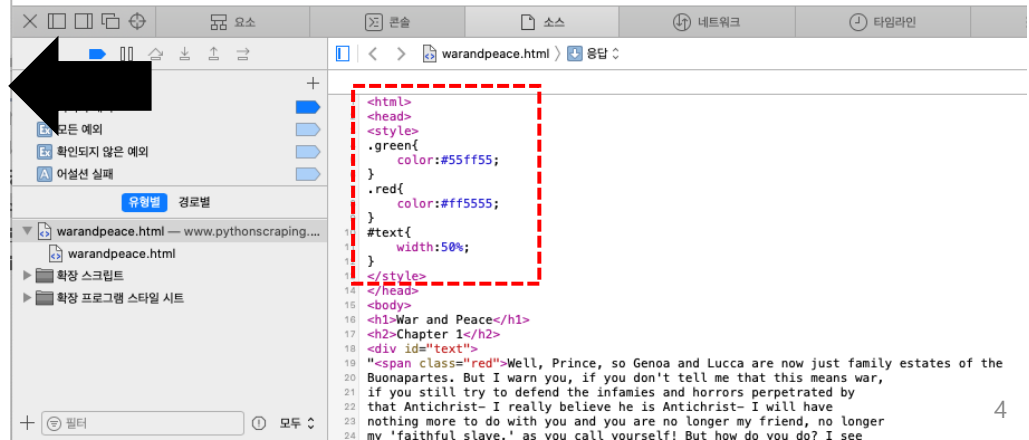
"Well, Prince, so Genoa and Lucca are now just family estates of the Buonapartes. But I warn you, if you don't tell me that this means war, if you still try to defend the infamies and horrors perpetrated by that Antichrist- I really believe he is Antichrist- I will have nothing more to do with you and you are no longer my friend, no longer my 'faithful slave,' as you call yourself! But how do you do? I see I have frightened you- sit down and tell me all the news."

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.

All her invitations without exception, written in French, and delivered by a scarlet-liveried footman that morning, ran as follows:

"If you have nothing better to do, Count [or Prince], and if the prospect of spending an evening with a poor invalid is not too terrible, I shall be very charmed to see you tonight between 7 and 10- Annette Scherer."

"Heavens! what a virulent attack!" replied the prince, not in the least disconcerted by this reception. He had just entered, wearing an embroidered court uniform, lace breeches, and shoes, and had stars on his breast and a serene expression on his flat face. He spoke in that refined French in which our grandfathers not only spoke but thought, and with the gentle, patronizing intonation natural to a man of importance who had grown old in society and at court. He went up to Anna Pavlovna, kissed her hand, presenting to her his bald, scented, and shining head, and complacently seated himself on the sofa.



CSS 속성을 이용한 검색

■ 속성(attrs) 사용

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

```
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

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

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

```
nameList = bs.find_all('span', {'class': 'green'})
for name in nameList:
    print(name.get_text())
```

등장인물의 이름은 모두 녹색
...
모두 검색

- `get_text()`: tag를 제외한 텍스트만 반환

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

줄 바꿈 문자 포함 (\n)

```
> bs = {BeautifulSoup} <html>\n<head>\n<style>\n.green{\n\tcolor:#55ff55;\r
> html = {HTTPResponse} <http.client.HTTPResponse object at 0x7f8c0d72b32
> name = {Tag} <span class="green">Anna\nPavlovna Scherer</span>
> nameList = {ResultSet: 41} [<span class="green">Anna\nPavlovna Scherer</s
> Special Variables
```

특정 단어 찾기

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

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

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

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

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

트리 이동

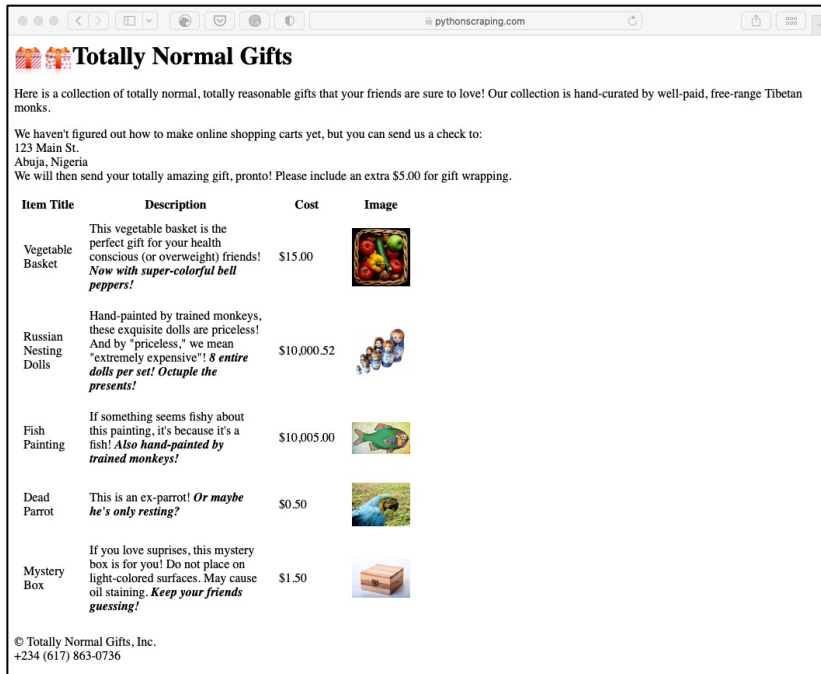
■ 트리 이동

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

```
bs.tag.subTag.anotherSubTag
```

■ 온라인 쇼핑 사이트 구성 및 트리 이동

- <https://www.pythonscraping.com/pages/page3.html>



```
html
- body
- div.wrapper
- h1
- div.content
- table#giftList
- tr
- th
- th
- th
- th
- tr.gift#gift1
- td
- td
- span.excitingNote
- td
- td
- img
...
- div.footer
```


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

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

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

table_tag = bs.find('table', {'id': 'giftList'})
for child in table_tag.children:
    print(child)
```

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

```
<tr class="gift" id="gift1"><td>
Vegetable Basket
</td><td>

</td></tr>
```

tr 단위
(테이블
행 목록)

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

</td></tr>
```

트리 이동: 자손

■ 자손: descendants

```
desc = bs.find('table', {'id': 'giftList'}).descendants
print('descendants 개수: ', len(list(desc)))
```

descendants 개수: 86

```
for child in bs.find('table', {'id': 'giftList'}).descendants:
    print(child)
```

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

```
<th>
Item Title
</th>
```

```
Item Title
```

```
<th>
Description
</th>
```

```
Description
```

```
<th>
Cost
</th>
```

```
Cost
```

```
<th>
Image
</th>
```

```
Image
```

<tr> 분리
<th> 분리
<th> 내부 분리

트리 이동: 형제 다루기 #1

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

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

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

for sibling in bs.find('table', {'id': 'giftList'}).tr.next_siblings:
    print(sibling)
```






giftList의 첫 번째 tr선택

```
<tr class="gift" id="gift1">
<td>Vegetable Basket</td>
...

<tr class="gift" id="gift2">
<td>Russian Nesting Dolls</td>
...

<tr class="gift" id="gift5">
<td>Mystery Box</td>
</tr>
```

next_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	

트리 이동: 형제 다루기 #2

■ previous_siblings 속성






- 선택된 행 이전의 항목들을 선택

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

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

previous_siblings

선택

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 #3

▪ `next_sibling`, `previous_sibling`

- 태그 하나만 반환
- 문자열 마지막에 `whitespace('\n', '\r' 등)`가 있는 경우
 - 해당 `whitespace`를 `next_sibling`으로 반환

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

```
10
'\n'
```

• `next_sibling.next_sibling` 이용

```
sibling2 = bs.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 = bs.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)

```
<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

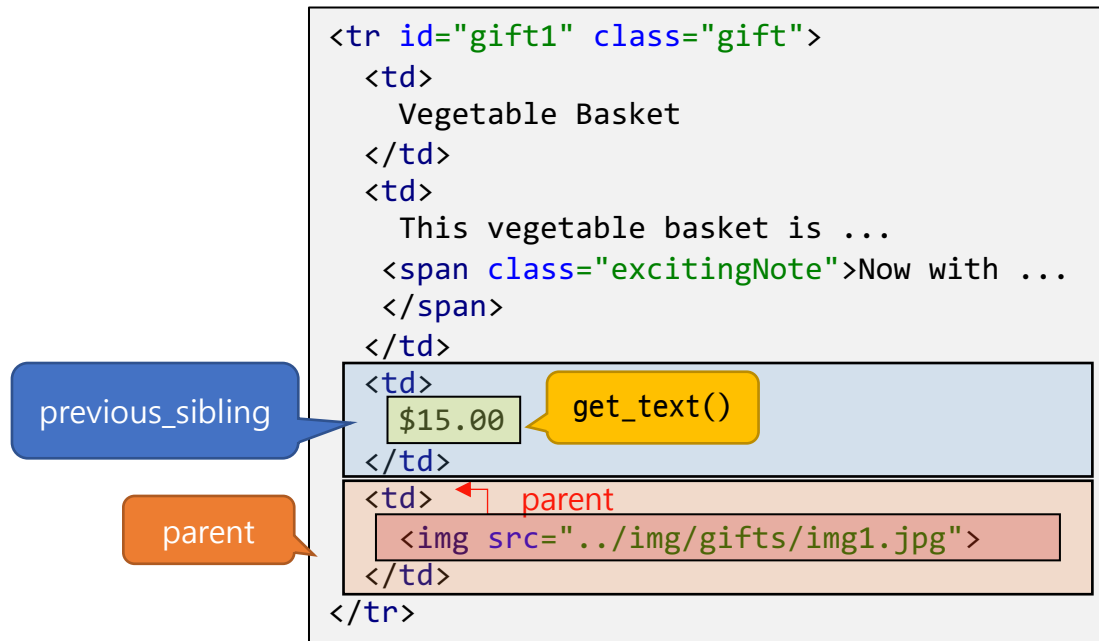
트리 이동: 부모 다루기 #2

■ .parent 사용

```
img1 = bs.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(<td>) 태그의 이전 형제 태그를 검색
- **get_text()**: 태그 내부의 문자열을 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, 역슬래시 다음에 일반 문자가 오면 특수 문자로 취급 • 역슬래시 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 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 flags 옵션 - <code>re.DOTALL</code> : dot문자가 줄 바꿈 문자를 포함하여 모든 문자와 매치 - <code>re.IGNORECASE</code> : 대소문자 구분없이 매치 - <code>re.MULTILINE</code> : 여러 줄과 매치 - <code>re.VERBOSE</code> : 정규식을 보기 편하게 만들고 주석등을 사용
<code>search(pattern, string)</code>	문자열 전체를 검색하여 pattern이 존재하는지 검색 - 처음 매칭되는 문자열 리턴
<code>match(pattern, string)</code>	string이 시작하는 부분부터 pattern이 존재하는지 검사 - 정확히 일치하는지 검사 - 공백이 있는 경우나 중간에 패턴이 존재하는 경우 검색하지 못함
<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

```
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
```

```
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
```

```
<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 #2

▪ findall() 함수

- 일치하는 모든 문자열을 리스트로 리턴

```
p = re.compile('[a-z]+') # 알파벳 소문자  
print(p.findall('life is too short'))
```

```
['life', 'is', 'too', 'short']
```

▪ 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', '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

■ 전화 번호 분석

- 전화번호: '지역번호-국번-전화번호'
 - 전화번호: (2, 3자리)-(3, 4자리)-(4자리)
 - 예: 02-123-4567, 053-123-1234

```
import re
```

```
# ^ .. $ 을 명시해야 정확한 자리수 검사가 이루어짐
```

```
tel_checker = re.compile("^(\d{2,3})-(\d{3,4})-(\d{4})$")
```

```
print(tel_checker.match('02-123-4567'))
```

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

```
print(tel_checker.match('053950-4567'))
```

마지막 숫자의 자리수가 맞지 않음

dash(-) 가 없음

```
<re.Match object; span=(0, 11), match='02-123-4567'>
```

```
None
```

```
None
```

- `groups()`: 매칭 결과를 튜플로 출력

```
m = tel_checker.match('02-123-4567')
```

```
print(m.groups())
```

```
('02', '123', '4567')
```

Match 메소드 예제 #2

■ 전화 번호 분석

- `group()`
 - 매칭된 전체 문자열 반환
- `group(index)`
 - 해당 인덱스에 매칭된 문자열 반환
 - `index=0`: 전체 리턴

```
m = tel_checker.match('02-123-4567')

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

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

전방 탐색(lookahead)

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

```
import re
# 전방 긍정 탐색
lookahead1 = re.search('.*(?=won)', '1000 won')
print(lookahead1)

lookahead2 = re.search('.*(?=log:)', '2022-07-01 00:00:01 ABC.log: 전방탐색')
print(lookahead2)

# 전방 부정 탐색
lookahead3 = re.search('\d{4}(?!-)', '010-1234-5678')
print(lookahead3)
```

```
<re.Match object; span=(0, 5), match='1000 '>
<re.Match object; span=(0, 24), match='2022-07-01 00:00:01 ABC.'>
<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 해당 문자열 반환
- **(?<!패턴)**

후방 긍정 탐색

```
lookbehind1 = re.search('(?!<=log:).+', '2022-07-01 00:00:01 ABC.log: this is python')  
print(lookbehind1)
```

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

후방 부정 탐색(\b: 공백(blank) 검색)

```
lookbehind3 = re.search('\\\\b(?!\\$)\\\\d+\\\\b', 'I paid $30 for 100 apples.')  
print(lookbehind3)
```

```
lookbehind4 = re.search(r'\\b(?!\\$)\\\\d+\\\\b', 'I paid $30 for 100 apples.')  
print(lookbehind4)
```

```
<re.Match object; span=(28, 43), match=' this is python'>  
<re.Match object; span=(4, 8), match=' $51'>  
<re.Match object; span=(15, 18), match='100'>  
<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

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

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

```
#img_tag = re.compile('.\.\./img/gifts/img.*\.jpg')
```

```
img_tag = re.compile('/img/gifts/img.*.jpg') # '.' 임의의 한 문자 0회 이상
images = bs.find_all('img', {'src': img_tag})
```

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

```
, ../img/gifts/img1.jpg
, ../img/gifts/img2.jpg
, ../img/gifts/img3.jpg
, ../img/gifts/img4.jpg
, ../img/gifts/img6.jpg
```

정규 표현식과 BeautifulSoup #2

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

```
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(text='the prince')
print('the prince count: ', len(princeList))

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

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

Tag 속성에 접근하기 #1

■ Tag 속성 사용

- 태그:
 - `src` 속성에 이미지 정보를 가지고 있음

```

```

- 태그의 `src` 속성값 가져오기
 - `attrs['src']`

```
from bs4 import BeautifulSoup

soup = BeautifulSoup('<img src=../img/gifts/img1.jpg>', 'html.parser')
img_tag = soup.img

print(img_tag)
print(img_tag.attrs) # dictionary 형태로 저장
print(img_tag.attrs['src'])
```

```

{'src': '../img/gifts/img1.jpg'}
../img/gifts/img1.jpg
```


태그 속성에 접근하기 #2

■ URL 정보 가져오기

- <a> 태그: href 속성에 URL 정보를 가짐

```
<a href="https://www.naver.com">네이버로 이동하기</a>
```

- <a>태그의 href 속성값 가져오기

- <a>태그 선택(직접 선택, find, find_all() 등)
- `attrs['href']`

```
html = '<a href="https://www.naver.com">Naver</a>'
soup = BeautifulSoup(html, 'html.parser')

link = soup.a # <a>태그 지정
print(link)
print(link.attrs['href']) # <a> 태그의 href 속성 접근

link1 = soup.find('a')
print('find:', link1)
print(link1.attrs['href'])
```

```
<a href="https://www.naver.com">Naver</a>
https://www.naver.com
find: <a href="https://www.naver.com">Naver</a>
https://www.naver.com
```

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"! <span
class="excitingNote">8 entire dolls per set! Octuple the presents!</span>
</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

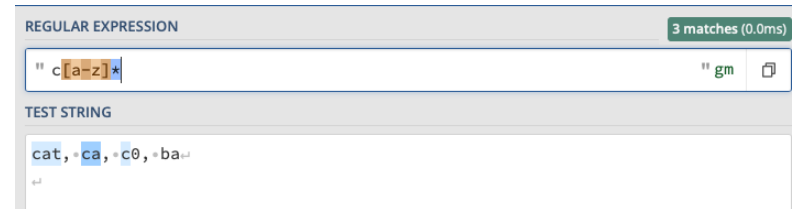
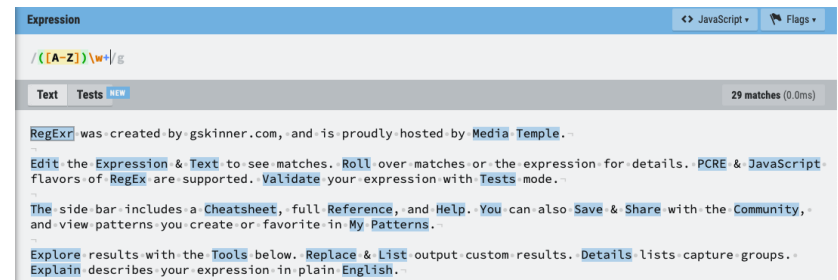
- <https://regex101.com>

■ RegexPlanet

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

■ regexper

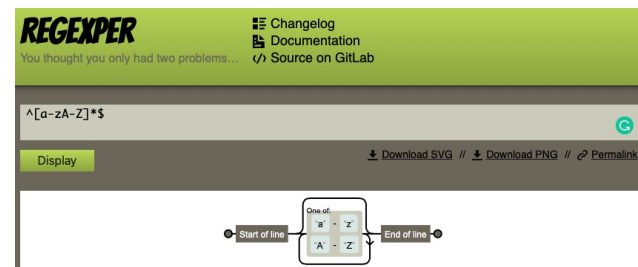
- <https://regexper.com>
- 그래픽으로 정규식을 표현



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?