Chapter. 02

데이터 크롤링

정규 표현식

FASTCAMPUS
ONLINE
금융공학/퀀트

강사. 서찬웅

이번 시간에 배울 내용에 대해서 알아보겠습니다.

1. 정규 표현식이란 무엇인가?
2. 정규식 사용법
3. 파이썬에서 정규식 사용법

I 정규 표현식에 대해서 알아보겠습니다.

정규 표현식이란?

- 정규 표현식(Regular Expression) 또는 정규식(Regex)은 특정한 규칙을 가진 문자열 집합을 표현하는 데 사용하는 언어입니다. 정규표현식은 많은 텍스트 편집기와 프로그래밍 언어에서 문자열의 검색과 치환을 위해 지원하고 있습니다. - Wikipedia
- 다시 정리하면 특수한 문자열 패턴으로 데이터를 추출하는 일종의 도구입니다. 그리고 다른 도구들처럼 특정한 문제를 해결할 목적으로 만들어 졌습니다.
- 예를 들어 큰 범위의 텍스트에서 특정 패턴과 일치하는 단어들을 찾아내거나, 해당 패턴과 일치하는 문자열을 다른 문자열로 치환하는 방법을 적용할 수 있습니다. 정규 표현식은 문자열을 찾고 조작하는데 사용되는 문자열입니다.

정규 표현식은 어디에 사용하나요?

- 원하는 정보가 어디에 있는지 검색하거나, 정보를 찾은 뒤에 해당 정보를 다른 정보로 치환할 때 사용

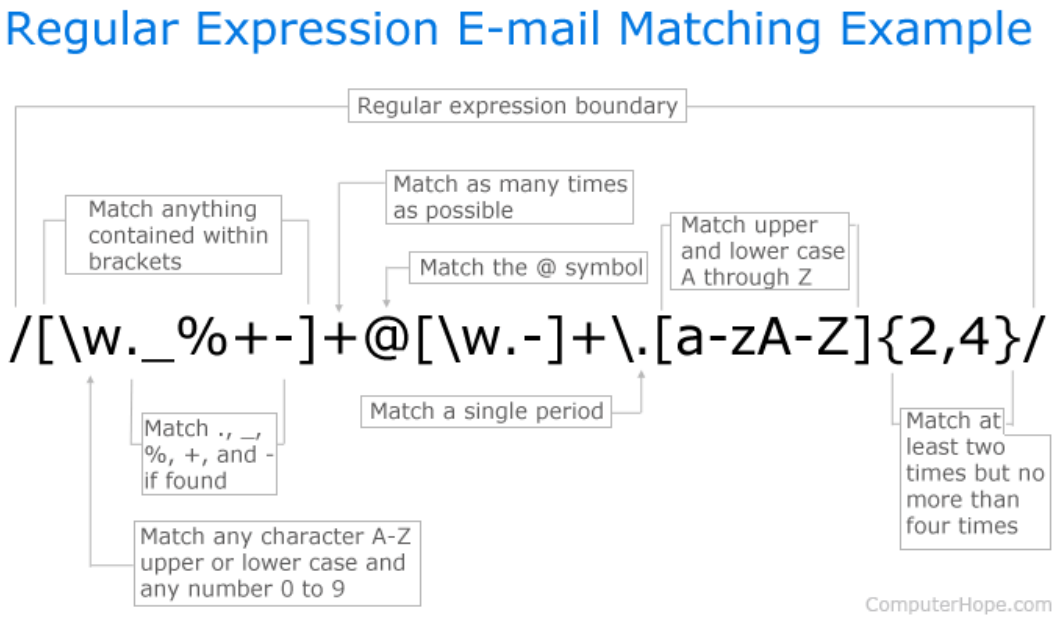
I 정규 표현식에 대해서 알아보겠습니다.

정규 표현식의 예를 들어주세요.

- 웹 페이지에서 숫자로 된 값들만 추출하고 싶다.
- 찾고 싶은 문자열이 소괄호안에 존재할 때 소괄호 안의 문자들만 추출하고 싶다
- 텍스트에서 대상 문자열 패턴을 특정한 문자로 치환하고 싶다.

정규 표현식을 연습할 사이트를 추천합니다.

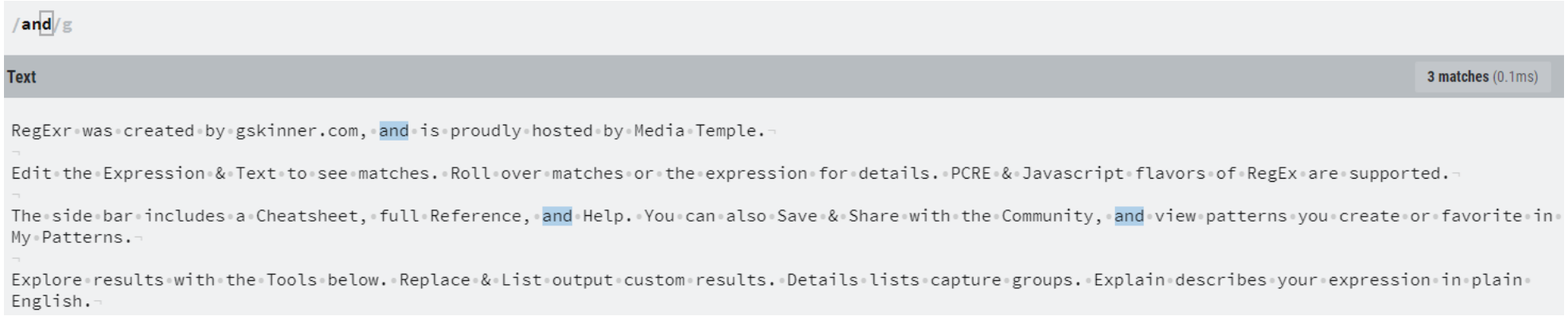
- https://regex101.com
- 정규 표현식을 연습을 통해 학습을 할 수 있습니다.



I 메타 문자에 대해서 알아보겠습니다.

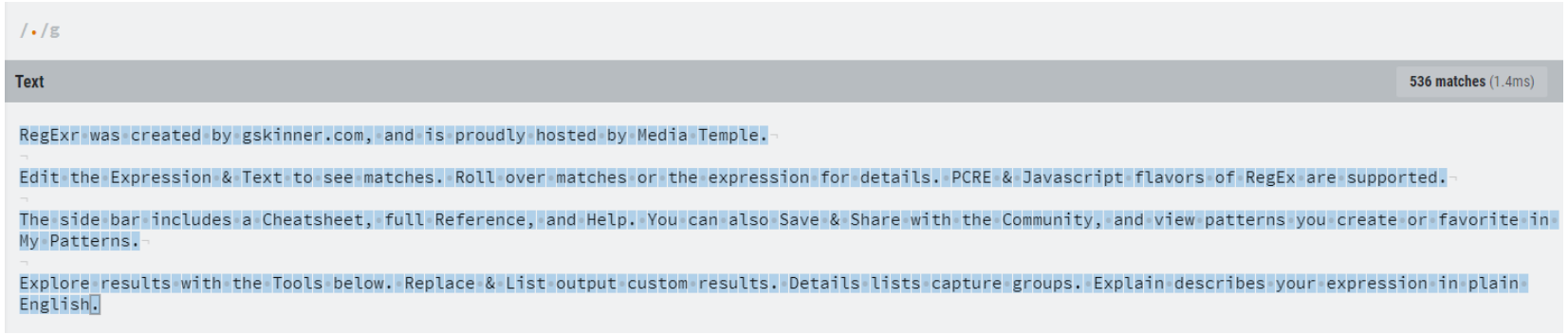
문자 그대로 찾기

- 정규식을 사용해서 원하는 문자열을 찾는 경우 일반적으로 우리가 검색하는 것처럼 문자 그대로 사용해서 문자열을 찾을 수 있습니다. 아래 예제처럼 'and'라는 글자로 찾게 되면 'and' 글자만 선택되는 것을 볼 수 있습니다.



하나의 문자만 찾는 방법이 아니라 문자열 패턴을 사용하여 찾는 방법을 설명하겠습니다.

메타 문자는 특별한 의미를 가지고 있는 약속된 문자를 뜻합니다. 정규 표현식에서 사용하는 . 은 모든 글자를 의미합니다.



I 메타 문자의 종류

정규식에서 메타 문자를 알고 있어야 패턴을 생성할 수 있습니다. 이제 기본적으로 우리가 기억해야 할 메타 문자를 소개하겠습니다.

아래 작은 표에는 dg로 시작해서 dooooog로 끝나는 텍스트가 있습니다. 이 예제를 가지고 설명하겠습니다.

dg, dog, doog, dooog, doooog, dooooog, doooooog

메타 문자	설명	예제
*	0회 이상 반복을 허용	do*g -> dg, dog, doog, dooog, doooog, dooooog, doooooog
+	1회 이상 반복을 허용	do+g -> dog, doog, dooog, doooog, dooooog, doooooog
?	0회나 1회 반복을 허용	do?g -> dg, dog
{m}	m회 반복을 허용	do{3}g -> dooog
{m,n}	m회부터 n회까지 반복을 허용	do{2,4}g -> doog, dooog, doooog

I 메타 문자의 종류

아래 메타 문자는 위치 및 그룹을 표현합니다.

메타 문자	설명	예제
.	줄바꿈 문자를 제외한 모든 문자	d.g -> dog 매칭
^	문자열의 시작과 매칭 []안에서 사용되면 반대 문자열을 의미 [^a]는 a가 아닌 문자	^dogs -> dogs and cats cats and dogs
$	문자열의 마지막에 매칭	dogs$ -> dogs and cats cats and dogs
[]	문자 집합을 나타낸다. [abc]의 의미는 a,b,c 중 한 문자를 의미한다. [a-c] 이렇게 –를 사용해서 표현할 수도 있다.	d[a-z]g -> dag, dbg, dcg, ddg .... dxg, dyg, dzg
|	a|b는 a또는 b의 의미이다.	d(a|o)g -> dag, dog
()	정규식을 그룹으로 묶는다.	

I 특수한 문자에 관해서 알아보겠습니다.

정규식에서 자주 사용하는 특수 문자들이 있습니다. 예를 들어 띄어쓰기, Tab, Enter 같은 키등에 매칭되는 특수문자들이 존재합니다. 아래는 특수 문자들의 목록입니다.

메타 문자	설명
\ \	역슬래시 문자 자체를 의미한다.
\ d	모든 숫자와 매칭된다. [0-9]와 동일
\ D	숫자가 아닌 모든 문자와 매칭된다. [^0-9]와 동일
\ s	화이트 스페이스 문자와 매칭된다.
\ S	화이트 스페이스 문자가 아닌 것과 매칭된다.
\ w	숫자 또는 문자와 매칭된다. [a-z0-9A-Z]
\ W	숫자 또는 문자가 아닌 것과 매칭[^a-z0-9A-Z]
\ b	단어의 경계를 포현 단어는 영문자나 숫자의 연속 문자열로 가정
\ B	\ b와 반대로 단어의 경계가 아님을 표현

I 수량자에 대해서 알아보겠습니다.

정규식은 처음부터 끝까지라는 생각을 가진 탐욕스러운 녀석(?)입니다. 이를 공식적으로 Greedy Matching이라고 합니다. 이 탐욕스러운 녀석을 매칭 특수 문자를 사용하여 덜 탐욕스러운 녀석으로 만들어 줄수 있습니다.(Non-greedy Matching)

수량자	설명
*?	*와 같으나 문자열을 최소로 매칭
+?	+와 같으나 문자열을 최소로 매칭
??	?와 같으나 문자열을 최소로 매칭
{m,n}?	{m, n}와 같으나 문자열을 최소로 매칭

수량자를 사용한 예제

```
print (re.search(r'href="(.*)"', '<a href="index.html">HERE</a><font size="10">').group())
```

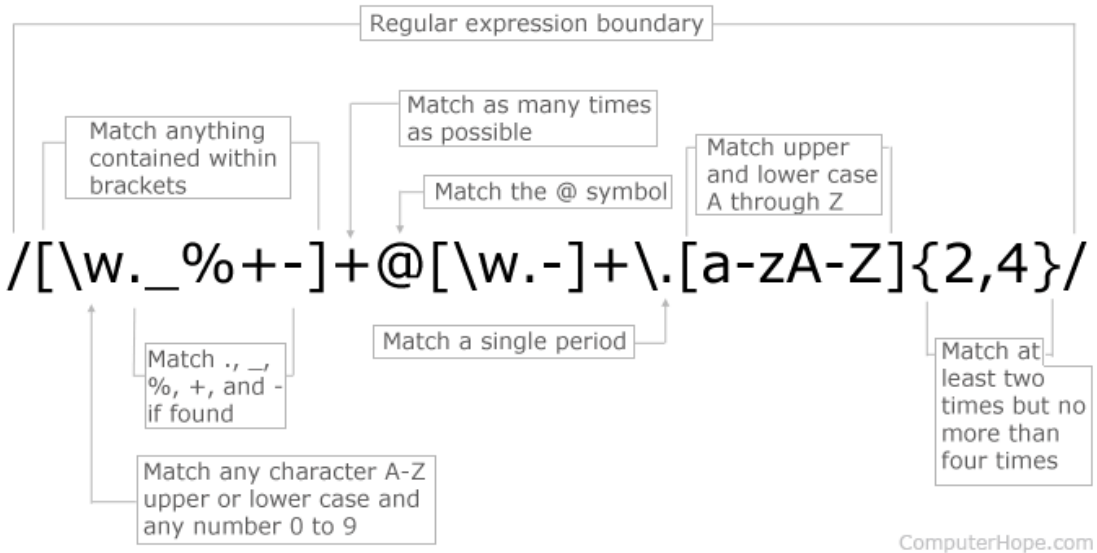
```
href="index.html">HERE</a><font size="10"
```

```
print (re.search(r'href="(.*?)"', '<a href="index.html">HERE</a><font size="10">').group())
```

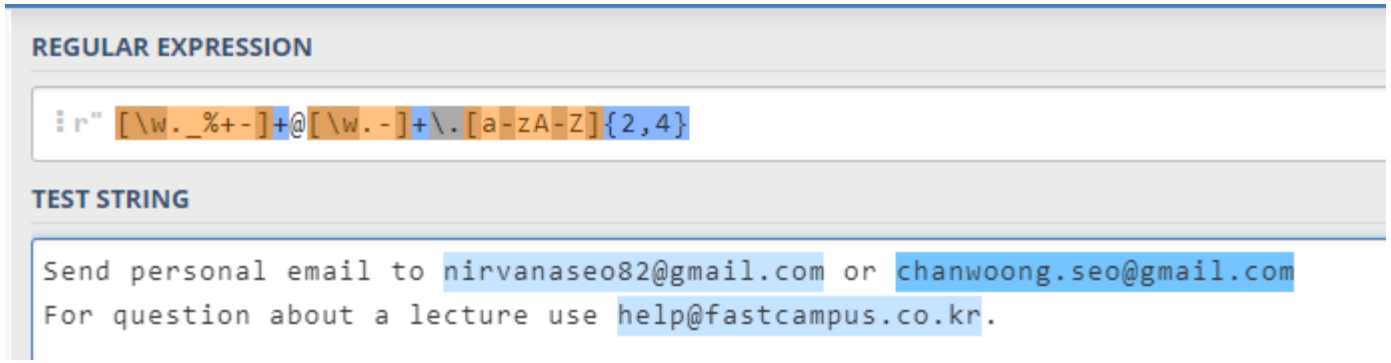
```
href="index.html"
```

I 정규식 예제를 보며 잠깐 쉬어가겠습니다.

아래 정규식 예제는 이메일 주소를 추출하는 정규식 표현입니다. 설명을 듣지 않고 이해가 된다면 정규식에 소질이 있는 것으로....



출처 : https://www.computerhope.com/jargon/r/regex.htm



I 이제부터는 re 모듈에 대해서 알아보겠습니다.

- 강력한 기능을 가진 정규식을 파이썬에서는 표준 라이브러리 re 모듈이 제공합니다.지금까지 배운 내용을 바탕으로 search() 함수, match() 함수를 사용하여 값을 가져오는 방법에 대해서 알아보겠습니다.
- search() 함수와 match() 함수는 두 함수 모두 매칭되는 객체를 반환하지만, 두 함수는 차이가 있습니다. match() 함수는 문자열이 시작되는 곳부터 일치하는지 검사하지만, search() 함수는 부분적으로 일치하는 문자열이 있는지만 검사합니다.

```
re.search("₩d+", ' 12345 ')
```

```
<re.Match object; span=(1, 6), match='12345'>
```

- match() 함수로 같은 패턴을 검색하면 12345 앞에 공백문자가 존재하기 때문에 찾지 못합니다.

```
re.match("₩d+", ' 12345 ')
```

- 만약 공백 문자가 없다면 match() 함수도 찾을수 있습니다.

```
re.match("₩d+", '12345 ')
```

```
<re.Match object; span=(0, 5), match='12345'>
```

I search() 함수 사용법에 대해서 알아보겠습니다.

search() 함수를 사용하여 반환 된 search 객체에서 group() 이나 groups() 메소드를 사용하면 정규식과 매칭된 문자열을 추출할 수 있습니다. 아래 표에 정리했습니다.

메소드	내 용	메소드	내용
group( )	매칭된 전체 문자열을 반환	start()	문자열의 시작 위치
group(n)	n번째 그룹 문자열을 반환	end()	문자열의 끝 위치
groups( )	매칭된 전체 그룹 문자열을 튜플 형식으로 반환	span()	(시작, 끝) 위치 튜플로 반환

```
re.search("₩d+", ' 12345 ').group()
```

```
'12345'
```

```
re.search("₩d+", ' 12345 ').group(0)
```

```
'12345'
```

```
re.search("₩d+", ' 12345 ').start()
```

```
1
```

```
re.search("₩d+", ' 12345 ').end()
```

```
6
```

```
re.search("₩d+", ' 12345 ').span()
```

```
(1, 6)
```

I 정규식 연습을 해보겠습니다.

아래는 강사가 좋아하는 명언을 정리한 글입니다. 맨 왼쪽은 YYYYMM 형식의 날짜로 되어 있고 그 다음엔 영어로 된 명언과 저자, 번역글, 한글로 표현한 이름식으로 되어 있습니다.

```
201901 Dost thou love life? Then do not squander time, for that is the stuff life is made of. (Benjamin Franklin) 그대는 인생을 사랑하는가? 그렇다면 시간을 낭비하지 말라, 시간이야말로 인생을 형성하는 재료이기 때문이다. (벤자민 프랭클린)
201902 Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (알버트 아인슈타인)
201903 Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin)  인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (찰리 채플린)
201904 Dream as if you'll live forever. Live as if you'll die today. (James Dean) 영원히 살 것처럼 꿈꾸고 오늘 죽을 것처럼 살아라. (제임스 딘)
201905 Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (데드풀)
```

Life로 시작하는 글을 추출해 보겠습니다.

```
rt = re.search("Life.*", txt)
```

```
rt.group()
```

```
'Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (알버트 아인슈타인) '
```

Life로 시작하는 명언이 하나만 존재하는 것은 아닌데, 출력이 하나밖에 되지 않았습니다. 의도랑 다르게 출력이 되었습니다.

I compile() 함수를 알아봅시다.

match()나 search() 함수를 사용하면 사용할 때마다 정규식을 표현해야 하고, 전체 문장에서 둘 이상의 값을 찾을 수도 없다. 그래서 compile() 함수를 이용하여 정규식을 정규식 객체로 변환하고, 이 객체를 사용하여 match()나 search() 메소드를 사용할수도 있고, findall() 메소드를 사용하여 전체 문장에서 정규식 패턴에 부합되는 문자열을 모두 추출할 수 있다.

```
rt2 = re.compile("Life.*")
```

```
rt2.findall(txt)
```

```
['Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (알버트 아인슈타인) ',
 'Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin)  인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (찰리 채플린) ',
 'Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (데드풀)']
```

I 예제에서 영어 명언만 추출해 봅시다.

예제에서 영어로 된 명언만 추출해 보겠습니다.

[A-Za-z] 알파벳 a~z, A~Z로 시작되고 .(모든 글자) + 하나 이상으로 있고, [A-Za-z] 알파벳으로 끝나고 [\.] 마침표로 끝나는 모든 패턴을 추출합니다.

```
rt2_1 = re.compile("[A-Za-z].+[A-Za-z][₩.₩?]")
```

```
rt2_1.findall(txt)
```

```
['Dost thou love life? Then do not squander time, for that is the stuff life is made of.',
 'Life is like riding a bicycle. To keep your balance you must keep moving.',
 'Life is a tragedy when seen in close-up, but a comedy in long-shot.',
 "Dream as if you'll live forever. Live as if you'll die today.",
 'Life is an endless series of trainwrecks with only brief commercial like breaks of happiness.']
```

201901 Dost thou love life? Then do not squander time, for that is the stuff life is made of. (Benjamin Franklin) 그대는 인생을 사랑하는가? 그렇다면 시간을 낭비하지 말라, 시간이야말로 인생을 형성하는 재료이기 때문이다. (벤자민 프랭클린)
201902 Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (알버트 아인슈타인)
201903 Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin) 인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (찰리 채플린)
201904 Dream as if you'll live forever. Live as if you'll die today. (James Dean) 영원히 살 것처럼 꿈꾸고 오늘 죽을 것처럼 살아라. (제임스 딘)
201905 Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (데드풀)

한글만 추출하고 싶다면 [A-Za-z] 대신 [가-힣]을 사용하시면 됩니다.

I 예제에서 영어 명언만 추출해 봅시다.

앞의 예제를 수량자를 사용해서 추출해보겠습니다.

```
rt2_2 = re.compile("[A-Za-z].+?[A-Za-z][₩.₩?]")
```

```
rt2_2.findall(txt)
```

```
['Dost thou love life?',
 'Then do not squander time, for that is the stuff life is made of.',
 'Life is like riding a bicycle.',
 'To keep your balance you must keep moving.',
 'Life is a tragedy when seen in close-up, but a comedy in long-shot.',
 "Dream as if you'll live forever.",
 "Live as if you'll die today.",
 'Life is an endless series of trainwrecks with only brief commercial like breaks of happiness.']
```

201901 Dost thou love life? Then do not squander time, for that is the stuff life is made of. (Benjamin Franklin) 그대는 인생을 사랑하는가? 그렇다면 시간을 낭비하지 말라, 시간이야말로 인생을 형성하는 재료이기 때문이다. (벤자민 프랭클린)
201902 Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (알버트 아인슈타인)
201903 Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin) 인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (찰리 채플린)
201904 Dream as if you'll live forever. Live as if you'll die today. (James Dean) 영원히 살 것처럼 꿈꾸고 오늘 죽을 것처럼 살아라. (제임스 딘)
201905 Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (데드풀)

I 괄호안의 저자들의 이름을 출력해보겠습니다.

예제에서는 괄호안에 명언 저자들의 이름이 적혀 있습니다. 저자들의 이름만 추출해보겠습니다.

```
rt3 = re.compile('(?<=₩()[a-zA-Z₩s^)]*(?=₩))')
```

```
rt3.findall(txt)
```

```
['Benjamin Franklin',
 'Albert Einstein',
 'Charlie Chaplin',
 'James Dean',
 'Deadpool']
```

이름이 출력되었습니다. 여기서 전방탐색, 후방탐색을 알아보겠습니다.

?<= : 후방탐색, ?= : 전방탐색

전방탐색, 후방탐색을 쉽게 이야기하면 일치하는 부분을 제외하고 추출해 줍니다.

전후방 탐색하여 () 안의 [a-zA-Z\s](모든 알파벳과 공백)이 0번 이상 해당되는 값들이 출력이 됩니다. 결국 괄호 안에 글자들이 출력이 되는 것을 확인할수 있습니다.

Ire 모듈의 정규식 플래그 사용하기

정규식과 함께 사용할 수 있는 플래그가 존재하는데 같이 사용하면 더욱 더 좋은 효과를 볼 수 있습니다.

compile(), search(), match() 함수의 인수로 사용되는 내용을 아래 정리했습니다.

플래그	설명
I, IGNORECASE	대소문자를 구분하지 않고 매칭
L, LOCALE	\w와 \W, \b, \B를 현재의 Locale에 영향을 받는다.
M, MULTILINE	^가 문자열의 맨 처음, 각 줄의 맨 처음과 매칭 $는 문자열의 맨 끝, 각 줄의 끝에 매칭 이 플래그를 사용하지 않을 경우 ^는 문자열의 맨 처음만, $는 문자열의 맨 마지막만 매칭한다.
S, DOTALL	. 을 개행 문자도 (\n)도 포함하여 매칭한다. 해당 플래그를 사용하지 않으면 \n을 제외한 문자와 매칭
U, UNICODE	\w, \W, \b, \B가 유니코드 문자 특성에 의존하게 한다.
X, VERBOSE	정규 표현식을 보기 좋게 표현할 수 있다. 정규식 내 공백은 무시하며 공백을 사용할 때는 백슬래시 문자로 표현 정규식 내에서 # 문자를 사용하면 주석으로 이후의 모든 문자는 무시한다.

Ire 모듈의 정규식 플래그 사용 예시

중요 플래그에 대해서 예제를 통해서 확인 하겠습니다.

- MULTILINE 플래그

```
s = """Beautiful Soup, so rich and green,
Waiting in a hot tureen!
Who for such dainties would not stoop?
Soup of the evening, beautiful Soup! """
```

```
p = re.compile('^.+')
```

```
p.findall(s)
```

```
['Beautiful Soup, so rich and green,']
```

```
p = re.compile('^.+', re.M)
```

```
p.findall(s)
```

```
['Beautiful Soup, so rich and green,',
 'Waiting in a hot tureen!',
 'Who for such dainties would not stoop?',
 'Soup of the evening, beautiful Soup! ']
```

- DOTALL

```
p = re.compile('green.+Waiting', re.S)
p.findall(s)
```

```
['green,₩n₩Waiting']
```

```
p = re.compile('green.+Waiting')
p.findall(s)
```

```
[]
```

- VERBOSE

```
rt3 = re.compile('''
(?<=₩()
[가-힣₩s^)]+
(?=₩))''')
rt3.findall(txt)
```

```
[]
```

```
rt3 = re.compile('''
(?<=₩()
[가-힣₩s^)]+
(?=₩))''', re.X)
rt3.findall(txt)
```

```
['벤자민 프랭클린', '알버트 아인슈타인', '찰리 채플린', '제임스 딘', '데드풀']
```

- 플래그 중첩

```
p = re.compile('green.+Waiting', re.S | re.M)
p.findall(s)
```

```
['green,₩n₩Waiting']
```

I 치환에 대해서 알아보겠습니다.

sub() 메소드를 사용하면 찾은 값을 다른 값으로 변경할 수 있습니다.

메소드 사용법 : sub(replacement, string[, count = 0])

```
rt2_3 = re.compile("(?<=₩()[가-힣₩s^)]*(?=₩))")
rt2_3.sub("서찬웅", txt)
```

```
"201901 Dost thou love life? Then do not squander time, for that is the stuff life is made of. (Benjamin Franklin) 그대는 인생을 사랑하는가? 그렇다면 시간을 낭비하지 말라, 시간이야말로 인생을 형성하는 재료이기 때문이다. (서찬웅) ₩n201902 Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (서찬웅) ₩n201903 Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin) 인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (서찬웅) ₩n201904 Dream as if you'll live forever. Live as if you'll die today. (James Dean) 영원히 살 것처럼 꿈꾸고 오늘 죽을 것처럼 살아라. (서찬웅) ₩n201905 Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (서찬웅)"
```

```
rt2_3 = re.compile("(?<=₩()[가-힣₩s^)]*(?=₩))")
rt2_3.sub("서찬웅", txt, 2)
```

```
"201901 Dost thou love life? Then do not squander time, for that is the stuff life is made of. (Benjamin Franklin) 그대는 인생을 사랑하는가? 그렇다면 시간을 낭비하지 말라, 시간이야말로 인생을 형성하는 재료이기 때문이다. (서찬웅) ₩n201902 Life is like riding a bicycle. To keep your balance you must keep moving. (Albert Einstein) 인생은 자전거를 타는 것과 같다. 균형을 잡으려면 움직여야 한다. (서찬웅) ₩n201903 Life is a tragedy when seen in close-up, but a comedy in long-shot. (Charlie Chaplin) 인생은 가까이서 보면 비극이지만 멀리서 보면 희극이다 (찰리 채플린) ₩n201904 Dream as if you'll live forever. Live as if you'll die today. (James Dean) 영원히 살 것처럼 꿈꾸고 오늘 죽을 것처럼 살아라. (제임스 딘) ₩n201905 Life is an endless series of trainwrecks with only brief commercial like breaks of happiness. (Deadpool) 인생이란 괴로움의 연속이고, 행복은 광고처럼 짧다. (데드풀)"
```

I 정리

- 정규 표현식 사용법
  - 메타문자
  - 수량자
  - 전후방 탐색
  - Non-greedy Matching
- 파이썬에서 정규 표현식 사용하는 방법
- re 모듈의 함수
  - compile()
  - sub()
- 플래그 설정

감사합니다