정규 표현식 (Regular Expression)

자연어처리 텍스트마이닝

정규 표현식

- 부분 문자열을 일치시키기위한 규칙으로 해석되는 문자의 조합

메타 문자	설명
.	한 개의 임의의 문자
?	앞문자가 0개 또는 1개
*	앞문자가 0개 이상
+	앞문자가 1개 이상
^	뒤의 문자로 문자열이 시작됩니다.
$	앞의 문자로 문자열이 끝납니다.

정규 표현식

메타 문자	설명
{숫자}	숫자만큼 반복
{숫자1, 숫자2}	숫자1 이상 숫자2 이하만큼 반복
{숫자,}	숫자 이상만큼 반복
[ ]	대괄호 안의 문자들 중 한 개의 문자와 매칭. [abc]라고 한다면 a 또는 b 또는 c 중 하나라도 존재하면 매치를 의미합니다. [a-z]와 같이 범위를 지정. 메타문자가 적용되지 않음
[^문자]	해당 문자를 제외한 문자를 매칭.
|	A|B와 같이 쓰이며 A 또는 B의 의미를 가집니다.

정규 표현식

- 자주 사용하는 정규식은 별도의 표기법으로 표현

문자	설명
\d	한 개의 숫자 (=[0-9])
\D	한 개의 숫자가 아닌 문자 (=[^0-9])
\w	한 개의 알파벳 문자 혹은 숫자 (=[a-zA-Z0-9])
\W	한 개의 알파벳 문자 혹은 숫자가 아닌 문자 (=[^a-zA-Z0-9])
\s	공백 문자
\S	공백이 아닌 문자

정규 표현식

문자	설명
\t	탭
\r	캐리지 리턴
\n	개행
\x	16진수
\b	단어 구분자

연습

[234cde]	2, 3, 4, c, d, e 중 하나	3, 6, b, c, o, x
[2-5c-e]	2에서 5까지 c에서 e 중 하나	2, 3, 5, a, c, b, e, f
[^2-5c-e]	2에서 5까지도 아니고 c에서 e 도 아닌 하나	2, 3, 5, a, c, b, e, f
(x)(yz)	group 1, group2	xyz, xxyz
(x|y)	x 또는 y	x, y, xy
x?	x가 없거나 1번	a, b, c, x, xx
x+	x가 1번 이상 반복	xx, xy, y
x*	x가 0번 이상 반복	xy, yxx, yyyy
x{2}	x가 2번 반복	x, xx, xy
x{2, }	x가 2번 이상 반복	x, xx, xxxxy, xyxy
x{2, 4}	x가 2번 이상 4번 이하 반복	xx, xxxxxxxx, xxxyxyxy

Python re 패키지

- re.compile() : 패턴 컴파일
- re.match() : 문자열의 첫 부분부터 정규 표현식과 매칭을 검사하여 **match object** 반환
- re.search() : 문자열 전체에서 정규 표현식과 매칭을 검사하여 **match object** 반환
- re.split() : 입력된 정규 표현식을 기준으로 문자열들을 분리하여 리스트로 리턴
- re.findall() : 매칭된 결과를 문자열 리스트로 반환
- re.finditer() : 문자열에서 정규 표현식과 매치되는 모든 경우의 문자열에 대한 이터레이터 객체를 리턴
- re.sub() : 정규 표현식 패턴과 일치하는 문자열을 찾아 다른 문자열로 대체

Match 오브젝트

- group() 매칭된 문자열

  group(0) 매치된 전체 문자열
  group(1) 첫 번째 그룹에 해당되는 문자열
  group(2) 두 번째 그룹에 해당되는 문자열
  group(n) n 번째 그룹에 해당되는 문자열

- start() 매치된 문자열의 시작 위치 반환
- end() 매치된 문자열의 끝 위치 반환
- span() 매치된 문자열의 (시작, 끝)의 튜플 반환

re.match()

문자열 처음부터 매치여부 조사. Match 객체리턴

```
>>> import re

>>> text = “I like orange! I love orange!”
>>> result = re.match(“orange”, text)
>>> print(result)
None
```

```
>>> import re

>>> text = “orange! I love orange!”
>>> result = re.match(“orange”, text)
>>> print(result)
<re.Match object; span=(0, 6), match=’orange’>
>>> print(result.group())
orange
>>> print(result.start())
0
>>> print(result.end())
6
>>> print(result.span())
(0, 6)
```

re.serch()

문자열 전체를 조사. 처음 검색된 최초 문자열에 대한 Match 객체리턴

```
>>> import re

>>> text = “I like orange! I love orange!”
>>> result = re.search(“orange”, text)
>>> print(result)
<re.Match object; span=(7, 13), match='orange'>
>>> print(result.group())
orange
>>> print(result.start())
7
>>> print(result.end())
13
>>> print(result.span())
(7, 13)
```

re.findall()

매치되는 모든 문자열 리스트로 리턴

```
>>> import re

>>> text = “I like orange! I love orange!”
>>> result = re.findall(“orange”, text)
>>> print(result)
[‘orange’,’orange’]
```

re.finditer()

매치되는 모든 문자열에 대한 Match 객체 리턴

```
>>> import re

>>> text = “I like orange! I love orange!”
>>> result = re.finditer(“orange”, text)
>>> for each in result:
>>>      print(each)
<re.Match object; span=(7, 13), match='orange'>
<re.Match object; span=(22, 28), match='orange'>
```

re.split()

입력된 정규 표현식을 기준으로 문자열들을 분리하여 리스트로 리턴

```
>>> import re

>>> text = “apple, orange! banana pineapple”
>>> result = re.split(“[,! ]”, text)
>>> print(result):
['apple', '', 'orange', '', 'banana', 'pineapple']
```

탐욕적 혹은 게으른

탐욕적 수량자	게으른 수량자
*	*?
+	+?
{n,}	{n,}?

```
>>> import re

>>> text = “<html><head><title>Title</title>”
>>> print(print(re.match('<.*>', text).group()):
<html><head><title>Title</title>

>>> print(re.match('<.*?>', s).group())
<html>
```

역 참조(backreferences)

- 역으로 참조 - 하위표현식을 참조
- 방법 : 하위표현식을 숫자로 참조
- ex) <H([1-6])>.*?</H\1> ⇨ <H1>Text</H1> vs <H2>Text</H3>

```
>>> import re

>>> text = “Paris in the the spriing”
>>> result = re.search(“(\b\w+)\s+\1”, text)
>>> print(result)
the the
```

re.sub()

문자열에서 정규 표현식과 일치하는 부분에 대해서 다른 문자열로 대체

```
>>> import re

>>> text = “apple, orange! banana pineapple”
>>> result = re.sub(“[^a-zA-Z]”, “”, text)
>>> print(result):
apple orange banana pineapple
```

- 역참조를 이용

```
>>> text = “park 010-1234-1234”
>>> result = (r"(\w+)\s+((\d+)[-]\d+[-]\d+)", "\g<2> \g<1>", text)
>>> print(result):
'010-1234-1234 park'
```

전방 탐색 vs 후방 탐색

- 정규식과 매치되어야 하며 조건이 통과되어도 문자열을 소비하지 않는다
- 전방 탐색 : 앞에 있는 문자열을 탐색
- **후방 탐색** : 뒤에 있는 문자열을 탐색
- 긍정형 : 일치하는 텍스트를 탐색
- 부정형 : 일치하지 않는 텍스트를 탐색

탐색 기호	설명
(?=)	긍정형 전방탐색
(?!)	부정형 전방탐색
(?<=)	긍정형 후방탐색
(?<!)	부정형 후방탐색

실습01

https://regexr.com/4chri

정상적인 이메일만 추출해주세요

실습02

https://regexr.com/4rdvb

텍스트중에 <내용> 괄호로 묶여진 텍스트를 괄호 포함 모두 제거해주세요

실습03

https://regexr.com/4rdve

1. 정규표현식을 이용 내용 을 각각 추출
2. 추출된 항목에서 과 태그를 모두제거
3. 각각 총 3개의 항목을 리스트에 넣기

결과

["네이버가 뉴스 서비스에 인공지능(AI)을 도입해 페이지 뷰(PV)를 늘리고 이용자를 끌어 모으고 있다. ",
"네이버는 5일 오전 서울 강남구 그랜드 인터컨티넨털 호텔에서 AI 콜로키움 2019를 열고 이 같은 AI 성과와 전략을 소개했다.",
"이날 기조연설에서 김광현 네이버 서치앤클로바 리더는 "AI 뉴스 추천 시스템인 에어스(AiRS)를 도입하면서 뉴스 소비량이 확대되고 있다" 고 말했다."]

심화 : 위의 1, 2 과정을 하나의 정규식으로 해결해보세요