UNISTUDY

파이썬

25강. 정규식 표현식

1. 정규 표현식

- 정규 표현식(Regular Expression)은 특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어이다. 텍스트 편집기와 프로그래밍 언어에서 문자열의 검색과 치환을 위한 용도로 쓰이고 있다.
- 예를 들면 자연어를 대상으로 원하는 단어만 추출하기 위해서는 단어가 되기 위한 일정한 패턴 (Pattern)이 존재한다. 이러한 패턴을 표준화된 텍스트 형식으로 나타낸 것을 정규 표현식이라 한다. 이때 텍스트를 구성하는 하나의 문자를 메타문자라고 한다

2. 메타문자

- 정규표현식에서 일정한 의미를 가지고 있는 특수문자를 메타문자((Meta characters)라고 한다. 메타 문자는 대부분의 프로그래밍 언어에서 표준으로 사용되며, 정규 표현식에서 사용되는 주요 메타문자는 같다.

2. 메타문자

메타문자	정규 표현식	의미
.	.x 또는 x.	임의의 한 문자가 x앞이나 뒤에 오는 패턴 지정
^	^x	x로 시작하는 문자열(접두어 패턴 지정)
$	x$	x로 끝나는 문자열(접미어 패턴 지정)
*	x*	x가 0번 이상 반복
+	x+	x가 1번 이상 반복
?	x?	x가 0 또는 1개 존재
|	abc|ABC	abc 또는 ABC 두 개 중 하나 선택
[ ]	[x]	x문자 한 개 일치
[^]	[^x]	x문자 제외(부정)
{n}	x{n}	x가 n번 연속
{n,}	x{n,}	x가 n번 이상 연속
{m, n}	x{m, n}	x가 m~n 사이 연속

2. 메타문자

- 역슬래시(₩)로 시작하는 이스케이프 문자를 이용하여 정규표현식의 메타문자로 사용할 수 있다. 정 규 표현식의 메타문자로 사용하는 이스케이프 문자와 같다. 이스케이프 문자를 정규 표현식의 메타문자로 사용하기 위해서는 역슬래시(₩)를 하나 더 붙이거나 문자 앞부분에 'r' 문자를 붙 여서 이스케이프 문자로 해석하지 않도록 해야 한다.

메타문자	의미
₩s	공백문자(white space)
₩b	문자와 공백 사이
₩d	숫자 [0-9]와 같다.
₩w	단어 [0-9a-zA-Z_]와 같다. 영문자+숫자+_(밑줄)
₩n	줄바꿈 문자
₩t	탭 문자

3. 정규 표현식 모듈

- 파이썬에서는 메타문자를 이용하여 패턴(정규 표현식)을 만들고, 이러한 패턴을 특정 문자열에 적용하 여 문자열을 처리할 수 있는 re 모듈을 제공한다. re 모듈에서 제공하는 주요 내장 함수와 기능에 대한 설명이다.

함수(파라미터)	기능
compile(pattern, flags=0)	패턴을 컴파일하여 Pattern 객체 반환
escape(pattern)	문자열에서 특수 문자를 이스케이프 처리
findall(pattern, string, flags=0)	string에서 패턴과 일치하는 모든 문자열을 리스트로 반환
finditer(pattern, string, flags=0)	string에서 패턴과 일치하는 모든 문자열을 반복자를 반환

3. 정규 표현식 모듈

fullmatch(pattern, string, flags=0)	패턴을 모든 string에 적용하여 Match 개체를 반환, 일치하는 항목이 없으면 None 반환
match(pattern, string, flags=0)	string의 처음부터 패턴을 적용하여 Match 객체를 반환, 일치하는 것이 없으면 None 반환
search(pattern, string, flags=0)	문자열을 스캔하여 패턴과 일치하는지 확인하고 일치하는객체 리턴
split(pattern, string, maxsplit=0,flags=0)	string을 대상으로 패턴과 일치하는 문자열을 분할하여 부분 문자열이 포함된 리스트 반환
sub(pattern, repl, string, count=0,flags=0)	string에서 패턴과 일치하는 문자열을 repl로 대체하여 문자열 반환
subn(pattern, repl, string, count=0,flags=0)	문자열에서 패턴과 일치하는 문자열을 repl로 대체하여(new_string, 숫자)형식의 튜플 반환
template(pattern, flags=0)	템플릿 패턴을 컴파일하여 Pattern 객체 반환

4. 문자열 처리

- 메타문자를 이용하여 패턴(정규 표현식)을 작성하고, re 모듈에서 제공하는 함수들을 이용하여 패턴을 문자열에 적용해서 패턴과 일치되는 문자열을 찾고, 일치 여부를 검사하고 다른 문자열로 치환하는 과 정을 예문으로 알아본다.

5. 문자열 찾기

- 패턴과 일치하는 문자열을 찾는 findall() 함수는 다음 형식과 같이 3개의 파라미터를 갖는다. 그 중에서 pattern은 메타문자를 이용하여 작성한 정규 표현식, string은 처리할 문자열을 의미하고, flags=0은 기본값 0을 가지고 있기 때문에 일반적으로 생략한다. findall() 함수는 패턴과 일치 되는 문자열이 있으면 해당 문자열을 리스트(list) 자료구조로 반환하고, 일치하는 문자열이 없으면 빈 리스트([])로 반환한다.

```
import re
re.findall(pattern, string[, flags=0])
```

5. 문자열 찾기

chapter07.lecture.step01_findall.py

```
import re # 모듈 추가 - 방법1
from re import findall # 모듈 추가 - 방법2

st1 = '1234 abc홍길동 ABC_555_6 이사도시'

# (1) 숫자 찾기
print(findall('1234', st1))
print(findall('[0-9]', st1))
print(findall('[0-9]{3}', st1))
print(findall('[0-9]{3,}', st1))
print(findall('\\d{3,}', st1))

# (2) 문자열 찾기
print(findall('[가-힣]{3,}', st1))
print(findall('[a-z]{3}', st1))
print(findall('[a-z|A-Z]{3}', st1))


# (3) 특정 위치의 문자열 찾기
st2 = 'test1abcABC 123mbc 45test'

# 접두어/접미어
print(findall('^test', st2)) # 접두어
print(findall('st$', st2)) # 접미어

# 종료 문자 찾기 : abc, mbc
print(findall('.bc', st2))

# 시작 문자 찾기
print(findall('t.', st2))
```

Python Console

```
['1234']
['1', '2', '3', '4', '5', '5
', '5', '6']
['123', '555']
['1234', '555']
['1234', '555']

['홍길동', '이사도시']
['abc']
['abc', 'ABC']


['test']
['st']

['abc', 'mbc']

['te', 't1', 'te']
```

5. 문자열 찾기

```
# (4) 단어 찾기(\\w) - 한글+영문+숫자
st3 = 'test^홍길동 abc 대한*민국 123$tbc'
words = findall('\\w{3,}', st3)
print(words)
```

```
['test', '홍길동', 'abc', '123', 'tbc']
```

```
# (5) 문자열 제외 : x+(x가 1개 이상 반복)
print(findall('[^^*$]+', st3)) # 특수문자 제외
```

```
['test', '홍길동 abc 대한', '민국 123', 'tbc']
```

5. 문자열 찾기

- 문자열 찾기 예
- # (1) 숫자 찾기
- 숫자를 찾을 수 있는 다양한 패턴을 문자열에 적용하여 패턴과 일치하는 문자열을 리스트로 반환한 결과이다. 패턴에서 [0-9]는 숫자 1개와 일치되는 패턴을 의미한다. 만약 연속된 숫자를 찾기 위해서 는 반복을 의미하는 {} 메타문자를 뒤에 붙인다. 예를 들면 [0-9]{3}은 숫자가 3개 연속된 패턴을 의미한다. 숫자가 3개 이상 연속된 경우에는 [0-9]{3,}으로 숫자 뒤에 콤마(,)를 붙인다.
- # (2) 문자열 찾기
- 한글이나 영문자 등의 문자열을 찾는 패턴을 적용하여 패턴과 일치하는 문자열을 리스트로 반환한 결과이다. 패턴에서 [가-힣]{3,}는 한글 3자 이상 연속된 패턴을 의미한다. 만약 영문자인 경우에는 [a-z|A-Z]{3}으로 영문 소문자 또는 대문자가 3개 연속된 패턴을 찾는다.

5. 문자열 찾기

- # (3) 특정 위치의 문자열 찾기
- 문장의 시작이나 끝 지점 또는 특정 문자로 시작하거나 특정 문자로 끝나는 문자열을 찾는 패턴을 만드는 방법이다. 패턴에서 ^test는 메타문자 ^를 이용하여 문자열에서 접두어 test를 찾는 방법이고 test$는 메타문자 $를 이용하여 문자열의 접미어 st를 찾는 방법이다. 또한 패턴에서 .bc는 메타문자 .을 이용하여 3개 문자로 구성된 문자열 중에서 bc로 끝나는 문자열을 찾는 패턴이고, t.는 2개 문자로 구성된 문자열 중에서 t로 시작하는 문자열을 찾는 패턴이다. 메타문자 .은 문자 1개를 받을 수 있는 와일드-카드 역할을 한다.

5. 문자열 찾기

- # (4) 단어 찾기(₩₩w)
- 이스케이프 문자를 메타문자로 이용하여 패턴을 작성한 예문이다. ₩₩w는 단어(word)를 인식하는 메타문자로 사용된다. 역슬래시(₩)를 하나 더 붙인 이유는 이스케이프 문자를 정규 표현식의 메타문자 로 사용하기 위해서이다.(패턴 앞부분에 r 문자를 붙여도 된다.) 패턴에서 ₩₩w{3,}는 한글, 영문자, 숫자 중 하나가 3개 이상 연속된 것을 단어로 찾기 위한 패턴이다.
- # (5) 문자열 제외([^]
- 문자열에서 특정 문자 또는 문자열을 제외시킬 때 사용되는 패턴으로 [^^*$]+은 [^]와 + 메타문자를 이용하여 특수문자(^,*,$)를 문자열에서 제거하는 패턴이다.
- [^] 메타문자는 ^다음에 오는 문자를 제외시키는 역할을 하고, +는 기호 앞에 오는 문자가 1개 이상 연속되는 패턴의 의미를 갖는다. 이와 같은 방법은 텍스트나 자연어를 전처리하는 용도로 많이 이용된다.

6. 문자열 검사

- 주어진 문자열에서 패턴과 일치하는 문자열이 있으면 객체를 반환하고, 일치되는 문자열이 없으면 None을 반환하는 match() 함수는 다음 형식과 같이 3개의 파라미터를 갖는다. findall() 함수와 마찬가지로 pattern은 메타문자를 이용하여 작성한 정규 표현식, string은 처리할 문자열을 의미 한다. match() 함수는 문자열이 패턴과 일치되는지의 여부를 검사할 경우 사용한다. 예를 들면 주민 번호 또는 email 양식에 위배되는지를 검증할 때 사용한다.

```
import re
re.match(pattern, string[, flags=0])
```

6. 문자열 검사

chapter07.lecture.step02_match.py

```
from re import match # re 모듈 import

# (1) 패턴과 일치된 경우
jumin = '123456-3234567'
result = match('[0-9]{6}-[1-4][0-9]{6}', jumin)
print(result) 1

if result : # object
    print('주민번호 일치') # 주민번호 일치
else : # null
    print('잘못된 주민번호')

# (2) 패턴과 불일치된 경우
jumin = '123456-5234567'
result = match('[0-9]{6}-[1-4][0-9]{6}', jumin)
print(result) # None

if result : # object
    print('주민번호 일치') # 주민번호 일치
else : # null
    print('잘못된 주민번호')
```

Python Console

```
<re.Match object; span=(0, 14), match='123456-3234567'>

주민번호 일치

None

잘못된 주민번호
```

6. 문자열 검사

- 문자열 검사 예
- # (1) 패턴과 일치된 경우
- 주민번호 13자리 숫자를 검사하기 위해서 match() 함수에서 '[0-9]{6}-[1-4][0-9]{6}' 패턴을 적용하였다. 즉 하이픈(-)을 기준으로 앞부분 6자리 숫자, 하이픈(-) 다음에 오는 성별 1자리 숫자, 그리고 나머지 6개의 숫자를 메타문자를 이용하여 정규 표현식으로 지정하였다. 만약 jumin 변수의 문자열 객체가 패턴과 일치하면 match 클래스의 객체를 반환한다. 객체를 반환 받은 result 변수를 if문에서 사용할 경우 객체가 반환되면 True 객체가 반환되지 않으면 False값으로 비교 판단하여 if문의 실행문을 실행한다.
- # (2) 패턴과 불일치된 경우
- 성별 1자리에 올 수 있는 숫자는 1에서 4 사이의 정수 1개가 올 수 있는 패턴으로 지정되어 있기 때문에 예문 처리 5가 오면 패턴과 일치되지 않은 주민번호로 인식하여 None이 반환된다. 따라서 객체를 반환받은 result 변수를 if문에서 사용할 경우 False값으로 비교 판단하여 '잘못된 주민번 호' 문구가 출력된다

7. 문자열 치환

- 패턴과 일치하는 문자열을 지정한 문자열로 치환하여 새로운 문자열을 반환 sub() 함수는 다음 형식 과 같이 5개의 파라미터를 갖는다. pattern은 메타문자를 이용하여 작성한 정규 표현식, repl은 치 환할 문자열, string은 처리할 문자열을 의미한다. sub() 함수는 대상 문자열에서 패턴과 일치되는 문자열을 찾아서 다른 문자열로 치환하는 경우에 사용한다. 예를 들면 자연어를 대상으로 불용어에 해 당하는 문장부호나 특수문자를 제거할 때 사용된다.

```
import re
re.sub(pattern, repl, string[, count=0,flags=0])
```

7. 문자열 치환

chapter07.lecture.step03_sub.py

```
from re import sub

st3 = 'test^홍길동 abc 대한*민국 123$tbc'

# (1) 특수문자 제거
text1 = sub('[\^*$]+', '', st3)
print(text1)

# (2) 숫자 제거
text2 = sub('[0-9]', '', text1)
print(text2)
```

Python Console

```
test홍길동 abc 대한민국 123tbc

test홍길동 abc 대한민국 tbc
```

7. 문자열 치환

- 문자열 검사 예
- # (1) 특수문자 제거
- st3 문자열을 대상으로 3개의 특수문자(^, *, $)를 제거하기 위해서 '[₩^*$]+' 패턴을 지정하고, 치환되는 문자열은 공백('')으로 지정하였다. 여기서 패턴의 내용을 살펴보면, 특수문자(^)앞에 역슬래시(₩)가 붙여진 것을 볼 수 있는데, 이러한 이유는 해당 특수문자가 메타문자의 특수문자가 아닌 일반 특수문자로 인식하기 위해서이다.
- 참고로 이스케이프 문자를 메타문자로 사용할 경우에도 '₩₩이스케이프문자' 형식으로 역슬래시를 앞에 붙였다.
- # (2) 숫자 제거
- 특수문자를 제거한 이후에 숫자를 제거하기 위해서 '[0-9]' 패턴으로 숫자를 찾아서 공백('')으로 치환하는 예문이다. 문자열에서 숫자가 제거된 것을 출력결과에서 확인할 수 있다.

THANK YOU