정규식을 이용한 패턴추출

2019년 6월 3일 월요일 오후 2:14

- ❖ re.search() 는 해당 문자열이 대상 정규식을 만족시키는지를 True/False로 리턴
- ❖ 만약 매칭된 문자열을 추출하고 싶으면 re.findall() 을 사용
- ❖ re.findall() 을 사용하면 정규식에 매칭되는 부분 문자열을 모은 리스트를 리턴

패턴 추출하기

다음 코드에서 '[0-9]+'은 0부터 9까지 문자가 1번 이상 반복되는 패턴을 의미합니다. 이것은 즉, 정수로 이루어진 데이터를 찾는 것입니다. 또한 findall 메서드는 x라는 문자열에 존재하는 패턴('[0-9+]')을 모두 리스트로 저장해주는 기능을 합니다.

따라서, 이 코드는 x라는 문자열에서 정수 형태의 데이터를 모두 추출하여 y에 저장을 하는 코드입니다.

```
import re
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[0-9]+',x)
print(y)
# ['2', '19', '42']
```

다음 코드는 'A','E','I','O','U'로 이루어진 패턴을 찾아 출력하는 코드이지만 x 문자열에 해당되는 패턴이 없어서 빈 리스트를 출력합니다. 여기서 알 수 있는 사실은 정규 표현식에서는 소문자와 대문자를 구분한다는 사실입니다.

```
import re
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[AEIOU]+',x)
print(y)
# [] (빈 리스트 출력)
```

탐욕적 방식의 패턴 찾기

만약 다음 문장에서 '^F.+:'라는 패턴과 일치하는 부분을 찾는다면,

x = 'From: Using the : character'

- From:
- From: Using the :

이라는 두 가지 부분이 모두 패턴과 일치하게 됩니다.

이럴 때는 다음과 같이 가장 긴 패턴을 찾아주는데, 이것을 '탐욕적 방식의 패턴 찾기'라고 부릅니다. 일치하는 여러 패턴이 있을 경우 가장 긴 것을 선택한다는 의미입니다.

```
import re
x = 'From: Using the : character'
y = re.findall('^F.+:', x)
print(y)
# ['From: Using the :']
```

비탐욕적 방식의 패턴 찾기

물론 탐욕적이지 않은 방식으로 패턴을 찾을 수도 있습니다.

다음 코드에서처럼 패턴 뒤에 '?'(물음표)를 붙여주면 여러 대상 중 가장 짧은 것을 선택하게 됩니다.

```
import re
```

```
x = 'From: Using the : character'
y = re.findall('^F.+?:', x)
print(y)
# ['From:']
```

원하는 부분만 추출하기

다음 코드를 실행하면 '@' 문자 앞 뒤로 공백이 아닌 문자가 오는 문자열 패턴을 찾아줍니다. 따라서 다음과 같이 이메일 주소의 패턴이 추출됩니다.

```
x = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('\S+@\S+',x)
print(y)
# ['stephen.marquard@uct.ac.za']
```

그리고 다음과 같이 From으로 시작하는 이메일 주소 패턴에서 이메일 주소 부분만 추출할 수도 있습니다. 소괄호를 사용해서 말이죠.

괄호는 매칭에 비포함 추출될 문자열의 시작 지점과 끝 지점을 지정하는 역할

```
x = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From (\S+@\S+)',x)
print(y)
# ['stephen.marquard@uct.ac.za']
```

출처: <https://www.edwith.org/python-network-data/lecture/24452/>