18. 정규식(Regular Expression)

정규식 혹은 정규표현식은 특정한 규칙을 가진 문자열의 집단을 표현하는 방식이다.

Python 은 정규식을 지원하는 re module 을 제공한다. 사용 방법은 다음과 같다.

1. re.compile 을 이용하여 정규식을 compile 하여 pattern 객체를 얻음.
2. pattern 객체를 이용하여 문자열의 검색을 수행.

- 검색 method :

search() - 문자열 전체를 검색하여 정규식과 match 되는지 조사 (문자열의 아무곳이나 match)

match = re.search(pattern, str)

검색하여 match 되는 문자열이 있으면 match object 를 반환하고, 없으면 None 이 반환된다.

match 다음에 즉시 if 문으로 match 성공 여부를 check.

match 에는 match 결과가 저장되고 match.group() method 를 통해 matching text 를 가져온다.

In [2]:

```
import re

str = 'an example word:cat!!'                    # find word:xxx

match = re.search(r'word:\w\w\w', str)      # r - raw string : backslash 를 특수문자로 인식하지
match
```

Out[2]:

<re.Match object; span=(11, 19), match='word:cat'>

In [3]:

```
if match:
    print('found - ', match.group())
else:
    print('Not found')
```

found - word:cat

정규식의 기본 pattern

meta 문자

- meta 문자는 특수한 의미를 가지므로 위와 같이 자기 자체로 match 시키지 않는다.

. ^ $ * + ? { } [ ] \ | ( )

- special character 는 \를 앞에 두면 된다. ex) \., \, \@

single character match

- a, X, 9 ← 영문 대소문자, 숫자는 자기 자체로 match 시킨다.

  ex) re.search(r'12g', 'p12g') --> match='12g'

- . (period) 는 아무 문자(any character) 와도 match 가 된다. (단, \n (new line) 제외)
- \w 는 word character (a-z, A-Z, 0-9) 와 match 된다. (single character)
- \W - NOT \w
- \b 는 word 와 non-word 의 boundary 이다
- \s 는 white space (space, new line, return, tab) 와 match 된다. (single white space)
- \S 는 non-white space
- \d 는 decimal digit (0-9) 와 match 된다.
- \D - NOT \d

quantifier (나타나는 횟수)

-
  - : 바로 앞 문자가 0 회 이상 나타남
-
  - : 바로 앞 문자가 1 회 이상 나타남
- ? : 바로 앞 문자가 0 회 혹은 1 회 나타남 (optional)

position

- ^ : string 의 처음을 의미,
- $: string 의 끝을 의미한다.
- [ ] 내의 ^ 는 NOT 의미, ex) [^A-Z] : A-Z 을 제외한 문자

grouping

- ()

사용 method

- re.compile(pattern) : 패턴 문자열 pattern을 패턴 객체로 컴파일한다
- re.search(pattern, string) : string에서 pattern과 매치하는 텍스트를 탐색한다 (처음 한개 매치)
- re.sub(pattern, repl, string) : string에서 pattern과 매치하는 텍스트를 repl로 치환한다

In [4]:

```
import re

text = "문의사항이 있으면 032-232-3245 으로 연락주시기 바랍니다."

regex = re.compile(r'(\d{3})-(\d{3}-\d{4})')
matchobj = regex.search(text)

# same as above
#matchobj = re.search(r'(\d{3})-(\d{3}-\d{4})', text)

areaCode = matchobj.group(1)
num = matchobj.group(2)

fullNum = matchobj.group()

print(areaCode)
print(num)
print(fullNum)
```

```
032
232-3245
032-232-3245
```

search

- 첫번째 match 되는 string 검색

In [5]:

```
re.search(r'12g', 'p12gabc12g')
```

Out[5]:

```
<re.Match object; span=(1, 4), match='12g'>
```

In [6]:

```
match = re.search(r'iii', 'piiig')

match.group()
```

Out[6]:

```
'iii'
```

In [7]:

```
match = re.search(r'igs', 'piiig')

type(match)
```

Out[7]:

```
NoneType
```

In [8]:

```
match = re.search(r'..g', 'piiig')

match.group()
```

Out[8]:

'iig'

In [9]:

```
match = re.search(r'\d\d\d', 'p123g')

match.group()
```

Out[9]:

'123'

In [10]:

```
match = re.search(r'\w\w\w', '@@abcd!!')        #영수자 연속 3 개
match.group()
```

Out[10]:

'abc'

In [11]:

```
match = re.search(r'^bo', 'rebok')
match.group()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-c1013d7e131a> in <module>
      1 match = re.search(r'^bo', 'rebok')
----> 2 match.group()

AttributeError: 'NoneType' object has no attribute 'group'
```

In [12]:

```
match = re.search(r'k$', 'book')

match.group()
```

Out[12]:

'k'

match 되는 pattern 의 반복 (Repetition)

- + → 1개 혹은 그 이상의 해당 pattern 이 왼쪽에 있음

- * → 0 혹은 그 이상의 해당 pattern 이 왼쪽에 있음
- ? → 0 혹은 1 개의 해당 pattern 이 왼쪽에 있음

Leftmost and Largest 규칙

search 는 가장 왼쪽 (leftmost) match 를 먼저 찾고 repetition 메타문자 (., *, +)를 만족하는 한 match 를 계속 찾는다.

In [13]:

```
1 import re
2
3 match = re.search(r'pi*', 'pg')
4
5 match.group()
```

Out[13]:

'p'

In [13]:

```
1 match = re.search(r'i+', 'piigiiig')
2
3 match.group()
```

Out[13]:

'ii'

In [19]:

```
1 match = re.search(r'\d\s*\d\\s*d', 'xx1 2    3xx')   #숫자1 + white space 0 개 이상 + 숫자1 + 
2
3 match.group()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-19-ad0183a9d6ba> in <module>
      1 match = re.search(r'\d\s*\d\\s*d', 'xx1 2    3xx')   #숫자1 + white space 0
 개 이상 + 숫자1 + white space 0개 이상 + 숫자1
      2
----> 3 match.group()

AttributeError: 'NoneType' object has no attribute 'group'
```

In [20]:

```
1 match = re.search(r'\d\s*\d\s*\d', 'xx12   3xx')    # 위와 동일 pattern
2 match.group()
```

Out[20]:

'12   3'

In [21]:

```
match = re.search(r'\d\s*\d\s*\d', 'xx123xx')      # 위와 동일 pattern
match.group()
```

Out[21]:

'123'

In [22]:

```
match = re.search(r'^b\w+', 'foobar')
type(match)
```

Out[22]:

NoneType

In [23]:

```
match = re.search(r'b\w', 'foobar')

match.group()
```

Out[23]:

'ba'

대괄호 ([ ])

character set 을 표시한다. 예를 들어 [abc] 는 a, b, c 와 match 된다. 단, [ ] 내의 . 는 메타문자가 아니라 실제 . 표시이다.

In [24]:

```
str = 'Please send me email to young-jea.oh@citi.com with your return address'
match = re.search(r'[\w.-]+@[\w.-]+', str)       #영수자로 시작하는 any 문자열 (-포함)

match.group()
```

Out[24]:

'young-jea.oh@citi.com'

Group 구분

matching text 를 여러 부분으로 구분할 때 사용

In [25]:

```
match = re.search(r'([\w.-]+)@([\w.-]+)', str)
```

In [26]:

```
print(match.group())
print(match.group(1))
print(match.group(2))
```

young-jea.oh@citi.com
young-jea.oh
citi.com

findall

re.search() 는 첫번째 match 하나만 return

re.findall() 은 string 에서 match 되는 것 전부 list 로 return

In [27]:

```
re.findall(r'12g', 'p12gabc12g')
```

Out[27]:

['12g', '12g']

In [28]:

```
regex = re.compile(r'\d\d\w')

regex.findall('p12gabc12g')
```

Out[28]:

['12g', '12g']

In [29]:

```
re.findall(r'\d\d\w', 'p12gabc12g')
```

Out[29]:

['12g', '12g']

In [30]:

```
str = '메인 이멜 주소는 young-jea.oh@citi.com 이고 보조 이멜 주소는 yjohhhhh@naver.com 입니다
```

In [31]:

```
match = re.findall(r'[\w.-]+@[\w.-]+', str)

match
```

Out[31]:

['young-jea.oh@citi.com', 'yjohhhhh@naver.com', 'faq@gmail.net']

In [32]:

```
match = re.findall(r'([\w.-]+)@([\w.-]+)', str)

match
```

Out[32]:

[('young-jea.oh', 'citi.com'), ('yjohhhhh', 'naver.com'), ('faq', 'gmail.net')]

In [33]:

```
match = re.search(r'([\w.-]+)@([\w.-]+)', str)

print(match.group(1))
print(match.group(2))
```

young-jea.oh
citi.com

In [34]:

```
match = re.findall(r'[가-힣]+', "한글 regex 의 범위는 가-힣까지 입니다.")
match
```

Out[34]:

['한글', '의', '범위는', '가', '힣까지', '입니다']

text cleansing

https://en.wikipedia.org/wiki/Alice%27s_Adventures_in_Wonderland (https://en.wikipedia.org/wiki/Alice%27s_Adventures_in_Wonderland)

- [2], [3] 등 특수문자와 결합된 숫자 제거
- 영수자만 남기고 모두 제거

In [35]:

```
text = "Alice's Adventures in Wonderland (commonly shortened to Alice in Wonderland) is an 1
text
```

Out[35]:

"Alice's Adventures in Wonderland (commonly shortened to Alice in Wonderland) is an 1865 novel written by English author Charles Lutwidge Dodgson under the pseudonym Lewis Carroll.[1] It tells of a young girl named Alice falling through a rabbit hole into a fantasy world populated by peculiar, anthropomorphic creatures. The tale plays with logic, giving the story lasting popularity with adults as well as with children.[2] It is considered to be one of the best examples of the literary nonsense genre.[2][3] Its narrative course, structure, characters, and imagery have been enormously influential[3] in both popular culture and literature, especially in the fantasy genre."

In [36]:

```
text1 = re.sub(r'\[\d\]', '', text)
text1
```

Out[36]:

"Alice's Adventures in Wonderland (commonly shortened to Alice in Wonderland) is an 1865 novel written by English author Charles Lutwidge Dodgson under the pseudonym Lewis Carroll. It tells of a young girl named Alice falling through a rabbit hole into a fantasy world populated by peculiar, anthropomorphic creatures. The tale plays with logic, giving the story lasting popularity with adults as well as with children. It is considered to be one of the best examples of the literary nonsense genre. Its narrative course, structure, characters, and imagery have been enormously influential in both popular culture and literature, especially in the fantasy genre."

In [31]:

```
text2 = re.sub(r'[^A-Za-z0-9 ]', '', text1)
text2
```

Out[31]:

'Alices Adventures in Wonderland commonly shortened to Alice in Wonderland is an 1865 novel written by English author Charles Lutwidge Dodgson under the pseudonym Lewis Carroll It tells of a young girl named Alice falling through a rabbit hole into a fantasy world populated by peculiar anthropomorphic creatures The tale plays with logic giving the story lasting popularity with adults as well as with children It is considered to be one of the best examples of the literary nonsense genre Its narrative course structure characters and imagery have been enormously influential in both popular culture and literature especially in the fantasy genre'

연습문제

```
### HTML 에서 BABY 이름 찾기 (https://developers.google.com/edu/python/exercises/baby-names)

- baby1998.html 에서 <tr align="right"><td>1</td><td>Michael</td><td>Jessica</td> 에
match 되는 정규표현식을 이용하여 (rank, boy-name, girl-name) tuples 추출하여 print

```
import sys
import re

def extract_name(filename):
    names = []
    f = open(filename, 'r')
    text = f.read()
    groups = re.findall(r'<td> #YOUR CODE HERE# </td>', text)
    for tup in groups:
        print(tup[0], tup[1], tup[2])

if __name__ == '__main__':
    args = sys.argv[1:]
    if not args:
        print("file 명을 parameter 로 입력 바랍니다.")
```
```

```
        sys.exit(1)

    filename = './babynames/' + args[0]
    extract_name(filename)
```