정규표현식 - regex

1. 정규표현식이란?

- 문자열을 처리할때 특정 패턴으로 문자열을 처리하는 방법입니다.
- 정규표현식 함수
    - match : 문자열의 가장 앞에서 일치하는 패턴 찾기
    - search : 문자열에서 가장 첫번째로 일치하는 패턴 찾기
    - findall : 일치하는 패턴을 모두 찾기
    - split : 문자열을 특정 패턴으로 나누기
    - sub : 특정 패턴에 맞는 문자열을 대체 하기
- pattern
- 중고나라
    - 01o일이삼3구칠82 -> (정규표현식 패턴 + sub) -> 01012339782

2. 정규표현식 함수

In [1]:
```
import re
s = "fast campus datascience fighting. datascience fighting. fast campus fighting."
```

match

In [2]:
```
# 가장 앞에서 부터 일치하는 패턴 찾기
result1 = re.match("fast", s) # (패턴, 문자열)
result2 = re.match("campus", s) # (패턴, 문자열)
print(result1)
print(result2)
```

```
<re.Match object; span=(0, 4), match='fast'>
None
```

search

In [3]:
```
# 문자열에서 가장 첫번째로 일치하는 패턴을 찾기는것 입니다.
result1 = re.search("fast", s) # (패턴, 문자열)
result2 = re.search("campus", s) # (패턴, 문자열)
print(result1)
print(result2)
```

```
<re.Match object; span=(0, 4), match='fast'>
<re.Match object; span=(5, 11), match='campus'>
```

findall

In [4]:
```
# 일치하는 패턴을 모두 찾습니다.
result1 = re.findall("fast", s) # (패턴, 문자열)
result2 = re.findall("campus", s) # (패턴, 문자열)
print(result1)
print(result2)
```

```
['fast', 'fast']
['campus', 'campus']
```

split

In [5]:
```
# 패턴으로 문자열을 나누서 리스트로 만들어 줍니다.
# 여러가지 문자로 나누고 싶을때, string.split("," "%") 체이닝을 이용해서 여러번 함수를 호출해야 합니다.
# 패턴을 이용하여 한번만 함수를 호출해도 됩니다.
s1 = "fast campus datascience fighting!"
result = re.split("i", s1)
result
```

Out[5]:
```
['fast campus datasc', 'ence f', 'ght', 'ng!']
```

sub

In [6]:
```
# 일치하는 패턴을 대체
print(s)
re.sub("fast", "slow", s) # ("패턴", "해당되는 패턴을 바꿀문자열", 전체 문자열)
```

```
fast campus datascience fighting. datascience fighting. fast campus fighting.
```

Out[6]:
```
'slow campus datascience fighting. datascience fighting. slow campus fighting.'
```

3. 패턴 - pattern

- 문자
    - 숫자인지 문자인지 특수문자인지등을 구분
    - \d : 숫자
    - \D : 비숫자
    - \w : 숫자, 문자, _
    - \W : 숫자, 문자, _ 제외
    - \s : 공백문자
    - \S : 비공백문자
- 지정자
    - 범위가 몇회 반복과 같은 패턴을 구분

In [4]:
```
import string
pt = string.printable
len(pt), pt
```

Out[4]:
```
(100,
 '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}
~ \t\n\r\x0b\x0c')
```

In [14]:
```
# \d, \D - 숫자와 비숫자를 찾는 패턴
result = re.findall("\d", pt)
# result = re.findall("[0-9]", pt)
''.join(result)
```

Out[14]:
```
'0123456789'
```

In [10]:
```
result = re.findall("\D", pt)
''.join(result)
```

Out[10]:
```
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b
\x0c'
```

In [12]:
```
# 문자
result = re.findall("\w", pt)
print(''.join(result))
result = re.findall("\W", pt)
''.join(result)
```

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_
```

Out[12]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^`{|}~ \t\n\r\x0b\x0c'
```

In [11]:
```
# 공백문자
result = re.findall("\S", pt)
print(''.join(result))
result = re.findall("\s", pt)
''.join(result)
```

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Out[11]:
```
' \t\n\r\x0b\x0c'
```

4. 지정자

- []: 문자
- - : 범위
- . : 하나의 문자
- ? : 0회 또는 1회 반복
- * : 0회 이상 반복
- + : 1회 이상 반복
- {m,n} : m~n회 반복
- () : 그룹핑

In [15]:
```
pt
```

Out[15]:
```
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}
~ \t\n\r\x0b\x0c'
```

In [16]:
```
# [] : 문자
re.findall("[abc1]", pt)
```

Out[16]:
```
['1', 'a', 'b', 'c']
```

In [17]:
```
# - : 범위
re.findall("[01234567]", pt)
```

Out[17]:
```
['0', '1', '2', '3', '4', '5', '6', '7']
```

In [18]:
```
re.findall("[0-7]", pt)
```

Out[18]:
```
['0', '1', '2', '3', '4', '5', '6', '7']
```

In [19]:
```
re.findall("[a-f]", pt)
```

Out[19]:
```
['a', 'b', 'c', 'd', 'e', 'f']
```

In [20]:
```
re.findall("[a-fA-F]", pt)
```

Out[20]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'A', 'B', 'C', 'D', 'E', 'F']
```

In [18]:
```
# . : 문자하나
ls = ["123aab123", "a0b", "abc"]
for s in ls:
    result = re.findall("a.b", s)
    print(s, result)
```

```
123aab123 ['aab']
a0b ['a0b']
abc []
```

In [19]:
```python
# ? : ? 앞에 있는 패턴을 0회 또는 1회 반복
ls = ["aab", "a3b", "abc", "accb"]
for s in ls:
    # a + 어떤문자0개 또는 1개 + b
    result = re.findall("a.?b", s)
    print(s, result)
```

```
aab ['aab']
a3b ['a3b']
abc ['ab']
accb []
```

In [20]:
```python
# * : 0회이상 반복
ls = ["ac", "abc", "abbbbc", "a3bec"]
for s in ls:
    # a + b가 0회 이상 반복 + c
    result = re.findall("ab*c", s)
    print(s, result)
```

```
ac ['ac']
abc ['abc']
abbbbc ['abbbbc']
a3bec []
```

In [21]:
```python
# + : 1회이상 반복
ls = ["ac", "abc", "abbbbc", "a3bec"]
for s in ls:
    # a + b가 1회 이상 반복 + c
    result = re.findall("ab+c", s)
    print(s, result)
```

```
ac []
abc ['abc']
abbbbc ['abbbbc']
a3bec []
```

In [22]:

```
# {m} m회 반복
# {m, n} m에서 n회 반복
# + : 1회이상 반복
ls = ["ac", "abc", "abbbbbc", "abbbbbbbbbbc"] # 5회, 10회
for s in ls:
    # a + b가 m회 ~ n회 반복 + c
    result = re.findall("ab{1,8}c", s)
    print(s, result)
```

```
ac []
abc ['abc']
abbbbbc ['abbbbbc']
abbbbbbbbbbc []
```

In [23]:
```
# () : 그룹핑
ls = ["aaa5.djfi","abdddc5","1abbbbc","a3.bec"]
for s in ls:
    # 그룹1(0~9의 숫자가 1회이상반복) + [.문자] + 그룹2(a ~ z가 2글자)
    result = re.findall("([0-9]+)[.]([a-z]{2})", s)
    print(s, result)
```

```
aaa5.djfi [('5', 'dj')]
abdddc5 []
1abbbbc []
a3.bec [('3', 'be')]
```

5. 예시 - example

In [24]:
```
# 이메일 주소 찾기
s = "저의 이메일 주소는 pdj1224@gmail.com 입니다. 또한 radajin1224@gmail.com 도 가지고 있습니다."
p = "[0-9a-zA-Z]+@[0-9a-z]+\.[0-9a-z]+"
re.findall(p, s)
```

Out[24]:
```
['pdj1224@gmail.com', 'radajin1224@gmail.com']
```

In [25]:
```
# 주민등록번호 : group 나눠서 변경 : 761211-1023334 -> 761211-*******
# () 그룹핑을 사용
s = "저의 주민번호는 761211-1023334 입니다."
p = "([0-9]{6})[-]?([0-9]{7})"
print(re.findall(p, s))
re.sub(p, "\g<1>-*******", s)
```

```
[('761211', '1023334')]
```

Out[25]:
```
'저의 주민번호는 761211-******* 입니다.'
```

- 문자열에서 전화번호를 추출해서 바꾸기
- 01o일이삼3구칠82 -> (정규표현식 패턴 + sub) -> 01012339782

In [18]:
```
s = "안녕하세요, 저의 전화번호는 영일공-48구삼삼7이사 그리고 010사팔구삼삼구삼일 입니다. 둘중에 하나로 연락하세요"

# 전화번호 패턴
p = "[0-9영공일이둘삼사오육칠팔구빵oO]{3}[-]?[0-9영공일이둘삼사오육칠팔구빵oO]{3,4}[-]?\
[0-9영공일이둘삼사오육칠팔구빵oO]{4}"

# 패턴 찾기
numbers = re.findall(p, s)
numbers
```

Out[18]:
```
['영일공-48구삼삼7이사', '010사팔구삼삼구삼일']
```

In [19]:

```
# 숫자로 바꿔주기
dic = {
    "영":0, "공":0, "일":1, "이":2, "둘":2, "삼":3, "사":4,
    "오":5, "육":6, "칠":7, "팔":8, "구":9, "빵":0, "o":0, "O":0,
}
```

In [20]:
```
result = []
for number in numbers:
    # number 영일공-48구삼삼70이사
    #"영":0,"공":0,"일":1,
    for key, value in dic.items():
        number = number.replace(key, str(value))
    number = number.replace("-", "")
    print(number)
    result.append(number)
result
```

```
01048933724
01048933931
```

Out[20]:
```
['01048933724', '01048933931']
```

In [14]:
```
import re
```

In [1]:
```
def find_tel(s):

    # 전화번호 패턴
    p = "[0-9영공일이둘삼사오육칠팔구빵oO]{3}[-]?[0-9영공일이둘삼사오육칠팔구빵oO]{3,4}[-]?[0-9영공일이둘삼사오육칠팔구빵oO]{4}"

    # 패턴 찾기
    numbers = re.findall(p, s)

    # 숫자로 바꿔주기
    dic = {
        "영":0, "공":0, "일":1, "이":2, "둘":2, "삼":3, "사":4,
        "오":5, "육":6, "칠":7, "팔":8, "구":9, "빵":0, "o":0, "O":0,
    }

    result = []
    for number in numbers:
        # number 영일공-48구삼삼70이사
        #"영":0,"공":0,"일":1,
        for key, value in dic.items():
            number = number.replace(key, str(value))
        number = number.replace("-", "")
        result.append(number)

    return result
```

In [2]:
```
import re
```

In [3]:
```
s = "안녕하세요, 저의 전화번호는 영일공-48구삼삼7이사 그리고 010사팔구삼삼구삼일 입니다. 둘중에 하나로 연락하세요"
```

In [4]:
```
find_tel(s)
```

Out[4]:
```
['01048933724', '01048933931']
```