python

■ 정규 표현식

```
data = """
park 800905-1049118
kim  700905-1059119
"""

result = [] -> 빈 리스트
for line in data.split("₩n"):
    word_result = []
    for word in line.split(" "):
        if len(word) == 14 and word[:6].isdigit() and word[7:].isdigit():
            word = word[:6] + "-" + "*******"
        word_result.append(word)
    result.append(" ".join(word_result))
print("₩n".join(result))
```

isdigit = 숫자인가?를 의미

■ 메타문자(Meta Characters)

※ 메타 문자란 원래 그 문자가 가진 뜻이 아닌 특별한 용도로 사용하는 문자를 말한다.

. ^ $ * + ? { } [ ] ₩ | ( )

^ = not의 의미, 사용할 때 주의 해야함

● [자주 사용하는 문자 클래스]

[0-9] 또는 [a-zA-Z] 등은 무척 자주 사용하는 정규 표현식이다. 이렇게 자주 사용하는 정규식은 별도의 표기법으로 표현할 수 있다. 다음을 기억해 두자.

₩d - 숫자와 매치, [0-9]와 동일한 표현식이다.

₩D - 숫자가 아닌 것과 매치, [^0-9]와 동일한 표현식이다.

₩s - whitespace 문자와 매치, [₩t₩n₩r₩f₩v]와 동일한 표현식이다. 맨 앞의 빈 칸은 공백문자(space)를 의미한다.

₩S - whitespace 문자가 아닌 것과 매치, [^ ₩t₩n₩r₩f₩v]와 동일한 표현식이다.

₩w - 문자+숫자(alphanumeric)와 매치, [a-zA-Z0-9_]와 동일한 표현식이다.

₩W - 문자+숫자(alphanumeric)가 아닌 문자와 매치, [^a-zA-Z0-9_]와 동일한 표현식이다.

대문자로 사용된 것은 소문자의 반대임을 추측할 수 있다.

a.b = a와 b사이에 아무거나 한 글자

a[.]b = a와 b사이에 .(도트)문자를 의미

반복 (*)
다음 정규식을 보자.

ca*t
이 정규식에는 반복을 의미하는 * 메타 문자가 사용되었다. 여기에서 사용한 *은 * 바로 앞에 있는 문자 a가 0부터 무한대로 반복될 수 있다는 의미이다.

※ 여기에서 * 메타 문자의 반복 개수가 무한대라고 표현했는데 사실 메모리 제한으로 2억 개 정도만 가능하다고 한다.

즉 다음과 같은 문자열이 모두 매치된다.

반복 (+)
반복을 나타내는 또 다른 메타 문자로 +가 있다. +는 최소 1번 이상 반복될 때 사용한다. 즉 *가 반복 횟수 0부터라면 +는 반복 횟수 1부터인 것이다.

다음 정규식을 보자.

ca+t
위 정규식의 의미는 다음과 같다.
"c + a(1번 이상 반복) + t"

반복 ({m,n}, ?)
{m,n} = m~n회 반복
{m} = m회 반복을 의미

match()	**문자열의 처음부터 정규식과 매치되는지 조사한다.**
search()	**문자열 전체를 검색하여 정규식과 매치되는지 조사한다.**
findall()	**정규식과 매치되는 모든 문자열(substring)을 리스트로 돌려준다.**
finditer()	**정규식과 매치되는 모든 문자열(substring)을 반복 가능한 객체로 돌려준다.**
