
# 정규식 활용과 문자열

한국기술교육대학교
온라인평생교육원

## 학습목표

- 입력된 데이터값을 검증할 수 있는 정규식을 만들 수 있다.
- 정규식 패턴을 이용하여 원하는 형식의 데이터를 추출하거나 검증에 활용할 수 있다.

## 학습내용

- 정규식과 정규식 적용 함수
- 정규식 패턴 만들기와 활용

# 정규식과 정규식 적용 함수

## 1 정규표현식(Regular Expression)

### 1 정규표현식이란?

| 정규표현식 | ▪ 특정한 규칙을 가진 문자열의 패턴을 표현하는 데 사용하는 표현식<br>▪ 정규식, Regex로 부르기도 함 |
|---|---|

- 파이썬뿐만 아니라 자바 스크립트, 리눅스 스크립트 등 다른 언어에서도 많이 사용

# 정규식과 정규식 적용 함수

## 1 정규표현식(Regular Expression)

### 2 정규표현식을 사용하는 경우

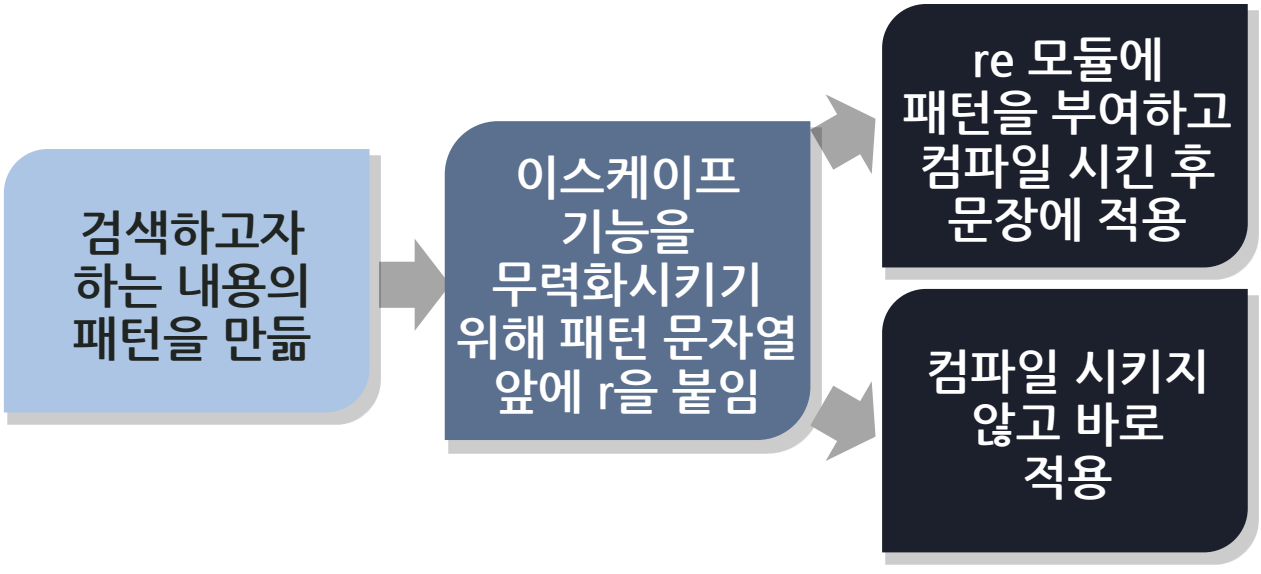
1. 사용자가 우편번호나 전화번호, 이메일 주소를 맞는 형식으로 입력했는지 확인
2. 크롤링한 자료에서 이메일이나 전화번호 등 추출
3. 로그파일에서 특정 에러메시지가 들어간 라인을 찾을 때
4. 텍스트에서 특정 문자열을 검색하거나 치환할 때

# 정규식과 정규식 적용 함수

## 1 정규표현식(Regular Expression)

### 3 파이썬에서의 정규식 사용

- 파이썬에서 정규식 지원을 위해 re 모듈 사용




```
import re

pattern = r'비'
text = "하늘에 비가 오고 있습니다. 어제도 비가 왔고
오늘도 비가 오고 있습니다"
regex = re.compile(pattern) #패턴을 컴파일시킴
result = regex.findall(text)    #matiching이 이루어진 모든
문자열의 리스트를 반환
print(result)
```

# 정규식과 정규식 적용 함수

## 1 정규표현식(Regular Expression)

### 4 정규식 예제

- 우리나라의 우편번호는 6자리에서 5자리로 체계가 바뀜
- 정수값을 입력 받아 데이터가 우편번호 형식에 맞는지 확인

- 우편번호 패턴 방식 : /d{5}$ ➡ 정수 5개만 가능

```
#파일명 : exam13_1.py
import re

zipcode = input("우편번호를 입력하세요")
pattern = r ' ₩d{5}$’
regex = re.compile(pattern)
result = regex.match(zipcode)
if result != None:
    print("형식이 일치합니다.")
else:
    print("잘못된 형식입니다.")
```

- 정수 5자리만 입력 가능
- 5자리의 정수 입력 : “형식이 일치합니다.”라고 출력
- 그 밖에 문자 입력, 자릿수 불일치 : “잘못된 형식입니다.”라고 출력

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

| | |
|---|---|
| match (pattern, string) | ▪ 문자열의 시작 부분부터 매칭이 되는지 검색<br>▪ 매칭 후 매칭에 대한 정보를 저장한 객체를 반환 |
| search (pattern, string) | ▪ 문자열에 패턴과 매칭되는 곳이 있는지 검색<br>▪ 매칭 후 매칭에 대한 정보를 저장한 객체를 반환 |
| findall (pattern, string) | ▪ 정규식과 매치되는 모든 문자열을 리스트로 반환 |
| finditer (pattern, string) | ▪ 정규식과 매치되는 모든 문자열을 iterator 객체(매칭 객체)로 반환 |
| sub(pattern, replace, string, count=0, flag=0) | ▪ 정규식과 매치되는 모든 문자열을 대체 문자열로 교체<br>▪ 결과를 str 타입으로 반환 |

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 1 match 함수

match(pattern, string, flag=0)

- 문자열의 시작 부분부터 패턴과 매칭이 되는지 검색하는 함수
- 검색 결과 : matchObject 인스턴트로 반환
  일치하는 패턴이 없으면 None을 반환

**MatchObject 주요 멤버 함수**

- group() : 매칭된 문자열을 반환
  0, 1, 2 파라미터를 주면 해당 문자열을 반환
  패턴이 () 등을 이용해 그룹으로 되어 있으면 자동으로 분리
- start() : 매칭된 문자열 시작 위치
- end() : 매칭된 문자열 종료 위치
- span() : 매칭된 문자열 시작과 종료 위치를 튜플로 반환

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 1 match 함수

```
import re

text1 = "I like star"
text2 = "star is beautiful"
pattern = "star"
print(re.match( pattern, text1))
print(re.match( pattern, text2))
matchObj = re.match( pattern, text2)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )
```

- match 함수는 첫 부분에 star가 와야 함
- 이 문장에서 패턴을 못 찾아냄

- 그룹 함수를 통해 단어 추출, 첫 번째 패턴의 시작 위치 및 단어의 종료 위치, 단어 위치 값을 튜플로 나타냄

```
<re.Match object; span=(7, 11), match='star'>
<re.Match object; span=(0, 4), match='star'>
star
7
11
(7, 11)
star
0
4
(0, 4)
```

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 2 search 함수

search(pattern, string, flag=0)

- 문자열의 패턴과 매칭되는 부분이 있는지 검색하는 함수
- 일치하는 패턴이 여러 개 있더라도 맨 처음에 검색된 부분을 반환

**match 함수**

문자열의 시작 부분이 일치해야 함

**search 함수**

문자열의 시작부터가 아니라 중간부터 있더라도 검색 진행

**MatchObject 주요 멤버 함수**

- group() : 매칭된 문자열을 반환
  0, 1, 2 파라미터를 주면 해당 문자열을 반환
  패턴이 () 등을 이용해 그룹으로 되어 있으면 자동으로 분리
- start() : 매칭된 문자열 시작 위치
- end() : 매칭된 문자열 종료 위치
- span() : 매칭된 문자열 시작과 종료 위치를 튜플로 반환

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 2 search 함수

```
#파일명 : exam13_3.py
import re

text1 = "I like star, red star, yellow star"
text2 = "star is beautiful"

pattern = "star"
print(re.search( pattern, text1))
print(re.search( pattern, text2))

matchObj = re.search( pattern, text1)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )
matchObj = re.search( pattern, text2)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )
```

- match 함수는 첫 부분에 star가 와야 함
- search 함수는 이 문장에서 패턴을 찾아냄

- 그룹함수를 통해 단어 추출
- 첫 번째 패턴의 시작 위치 및 단어의 종료 위치, 단어 위치 값을 튜플로 함

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 3 findall 함수

findall(pattern, string, flag=0)

- 문자열에 패턴과 매칭되는 부분에 대하여 string 리스트로 반환
- 반환값 : 일치하는 문자열들의 리스트
- 특정한 패턴과 일치하는 문자열만 추출할 때 사용
  ➡ 예) 전화번호만 추출하거나 이메일만 추출하고자 할 때

| | |
|---|---|
| 전화번호 체크 패턴 | r"₩d{3}-₩d{4}-₩d{4}" |
| 이메일 체크 패턴 | r"₩b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+₩.[a-zA-Z]{2,4}₩b" |

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 3 findall 함수

```
#파일명 : exam13_4.py
import re
#전화번호만 추출
text = """
    phone : 010-0000-0000 email:test1@nate.com
    phone : 010-1111-1111 email:test2@naver.com
    phone : 010-2222-2222 email:test3@gmail.com
    """
print()
print("--- 전화번호 추출하기 ---")
phonepattern = r"₩d{3}-₩d{4}-₩d{4}“

matchObj = re.findall( phonepattern, text)

for item in matchObj:
    print(item)
print("--- 이메일 추출하기 ---")
emailpattern = r"₩b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+
₩.[a-zA-Z]{2,4}₩b"

matchObj = re.findall( emailpattern, text)
for item in matchObj:
    print(item)
print()
```

- text에 기술된 전화번호와 이메일에서 원하는 이메일과 전화번호 추출
- findall 함수는 원하는 패턴의 데이터를 str의 list 형태로 반환

₩b가 앞에 있을 때 : 뒤에 오는 패턴으로 시작하라는 의미

- 이메일의 경우 test123@hanmail.net 형태인데 앞쪽에는 영문자와 숫자가 하나 이상 올 수 있음
- @와 도메인, .(도트) 뒤에 영문자가 옴
- 최소 2문자~최대 4문자까지 올 수 있음

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 3 findall 함수

```
--- 전화번호 추출하기 ---
010-0000-0000
010-1111-1111
010-2222-2222
--- 이메일 추출하기 ---
test1@nate.com
test2@naver.com
test3@gmail.com
```

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 4 finditer 함수

finditer(pattern, string, flag=0)

- 문자열에 패턴과 매칭되는 부분에 대하여 매칭 객체를 리스트로 반환
- 반환값은 문자열이 아니라 매칭 객체라는 부분에서 find 함수와 차이가 남
- 특정한 패턴과 일치하는 문자열만 추출할 때 사용
  ➡ 예) 전화번호만 추출하거나 이메일만 추출하고자 할 때

**MatchObject 주요 멤버 함수**

- group() : 매칭된 문자열을 반환
  0, 1, 2 파라미터를 주면 해당 문자열을 반환
  패턴이 () 등을 이용해 그룹으로 되어 있으면 자동으로 분리
- start() : 매칭된 문자열 시작 위치
- end() : 매칭된 문자열 종료 위치
- span() : 매칭된 문자열 시작과 종료 위치를 튜플로 반환

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 4 finditer 함수

```
#파일명 : exam13_5.py
import re
#전화번호만 추출
text = """
    phone : 010-0000-0000 email:test1@nate.com
    phone : 010-1111-1111 email:test2@naver.com
    phone : 010-2222-2222 email:test3@gmail.com
    """

print()
print("--- 전화번호 추출하기 ---")
phonepattern = r"₩d{3}-₩d{4}-₩d{4}"
matchObj = re.finditer( phonepattern, text)
for item in matchObj:
    print(item.group())
    print(item.span())
print()
```

- finditer 함수는 패턴과 일치할 경우 MatchObject 형태의 객체 반복자를 반환
- 이터레이터 형태이므로 for 구문을 통해 일치한 대상에 대한 정보를 확인할 수 있음

```
--- 전화번호 추출하기 ---
010-0000-0000
(13, 26)
010-1111-1111
(60, 73)
010-2222-2222
(108, 121)
```

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 5 sub 함수

sub(pattern, replace, string, count=0, flag=0)

- 문자열에 패턴과 매칭되는 부분에 대하여 매칭 객체를 리스트로 반환함
- 반환값은 문자열이 아니라 매칭 객체라는 부분에서 find 함수와 차이가 남
- 특정한 패턴과 일치하는 문자열만 추출할 때 사용함
  ➡ 예) 전화번호만 추출하거나 이메일만 추출하고자 할 때

| | |
|---|---|
| replace | ▪ 문자열이 될 수도 있고, 함수가 될 수도 있음 |
| count | ▪ 최대 몇 번까지 교체할 것인가를 설정하는 인자<br>▪ 값이 0이면 모두 교체, 0보다 크면 지정된 횟수만큼만 교체 |

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 5 sub 함수

**MatchObject 주요 멤버 함수**

- group() : 매칭된 문자열을 반환
  0, 1, 2 파라미터를 주면 해당 문자열을 반환
  패턴이 () 등을 이용해 그룹으로 되어 있으면 자동으로 분리
- start() : 매칭된 문자열 시작 위치
- end() : 매칭된 문자열 종료 위치
- span() : 매칭된 문자열 시작과 종료 위치를 튜플로 반환

# 정규식과 정규식 적용 함수

## 2 정규식에서 자주 사용하는 함수

### 5 sub 함수

```
#파일명 : exam13_6.py
import re

text1 = "I like stars, red star, yellow star"

print()
pattern = "star"
result = re.sub( pattern, "moon", text1)
print(result)

result2 = re.sub( pattern, "moon", text1, count=2)
print(result2)
```

```
I like moons, red moon, yellow moon
I like moons, red moon, yellow star
```

# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 1 패턴들

#### 1 패턴 ^

- 시작이 이 패턴으로 이루어져야 함

- 패턴이 ^abc라면 문자열의 시작이 abc여야 함

abcde　abc　abc123　→ 모두 가능

#### 2 패턴 $

- 이 패턴으로 끝나야 함

- 패턴이 abc$라면 문자열의 끝은 abc여야 함
- match 함수는 안 됨

deabc　abc　123abc　→ 모두 가능

# 정규식 패턴 만들기와 활용

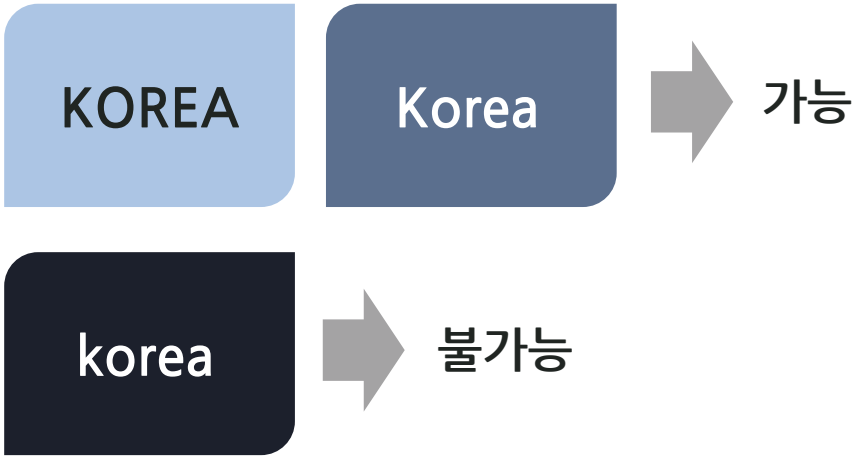
## 1 정규식 패턴 표현

### 1 패턴들

#### 3 패턴 [문자들]

- []에 속한 문자들만 해당되며, 가능한 문자열의 집합을 의미

- 패턴이 [p|P]ython라면 python, Python은 가능, PYTHON은 불가능
- [A-Z] 첫 글자가 알파벳 대문자만 가능함



# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 1 패턴들

#### 4 패턴 [^문자들]

- 피해야 할 문자들의 집합

- 이 문자들로 구성된 단어만 피함
- 패턴이 [^abc]일 경우,

# 정규식 패턴 만들기와 활용

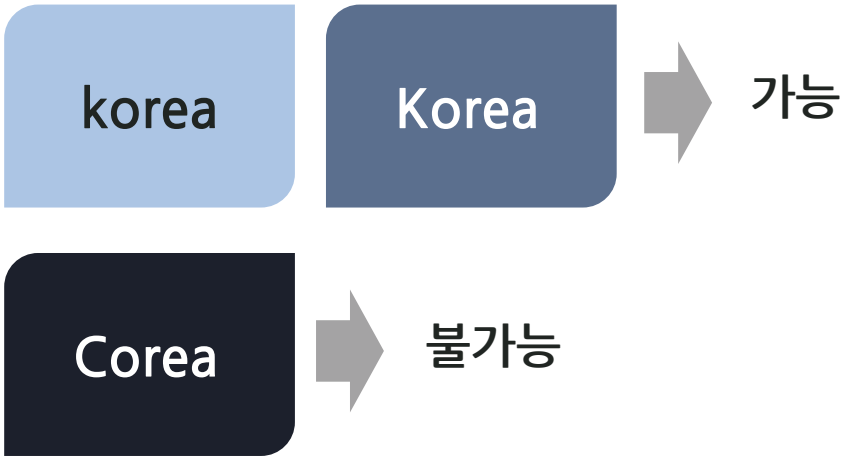
## 1 정규식 패턴 표현

### 1 패턴들

#### 5 패턴 |

- or 연산 둘 중 하나만 일치하면 됨

- 패턴이 [k|K]orea일 경우,



# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 1 패턴들

#### 6 패턴 ?, +, *

- ? : 앞의 패턴이 없거나 하나여야 함
- + : 앞의 패턴이 하나 이상 있어야 함
- * : 앞의 패턴이 0개 이상이어야 함, 반복을 의미

₩d? → 숫자가 없거나 하나만 있어야 함

₩d+ → 숫자가 하나 이상이어야 함

₩d* → 숫자가 없거나 하나 이상이어야 함

# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 1 패턴들

#### 7 패턴 {n}

- 패턴이 n번 반복해서 나타나야 함

₩d{3} ➡ 숫자가 3개 나타나야 함

#### 8 패턴{n, m}

- 패턴이 최소 n번, 최대 m번으로 나타나야 함

₩d{3-4} ➡ 숫자가 최소 3번에서 최대 4번 나타나야 함

# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 1 패턴들

#### 9 기타 패턴

**₩d**

숫자

**₩w**

문자

**₩s**

화이트 스페이스

**.**

newline을 제외한
모든 문자

**₩b**

- 단어의 시작이나 끝이어야 함
- r'₩bain' : ain으로 시작할 때
- r'ain₩b' : ain으로 끝날 때

# 정규식 패턴 만들기와 활용

## 1 정규식 패턴 표현

### 2 자주 사용하는 정규식 패턴

| | |
|---|---|
| 전화번호 | r"₩d{3}-₩d{3,4}-₩d{4}" |
| 이메일 | r"₩b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+₩.[a-zA-Z]{2,4}₩b" |
| 우편번호 | r”₩d{5,6}" |

- r은 문자열 안에 사용된 ₩(이스케이프) 문자의 본래 기능을 뺏기 위해 사용

# 정규식 패턴 만들기와 활용

## 2 정규식 그룹화

### 1 정규식 그룹화란?

- 정규표현식에서 **( )괄호**는 **그룹**을 의미
- 표현식을 그룹화하여 지정하는 경우, **매칭 후 값을 출력하고자 할 때** 사용

# 정규식 패턴 만들기와 활용

## 2 정규식 그룹화

### 2 활용

**전화번호**

- 010-9000-8000 : 정수 3자리-정수 4자리-정수 4자리 형태
  ➡패턴 : ₩d{3}-₩d{4}-₩d{4}가 됨
- 국과 번호를 구분하려면 split과 같은 함수를 이용해 데이터를 나누어야 함
  ➡정규식 그룹화 사용
- (₩d{3})-(₩d{4})-(₩d{4}) 형태로 괄호를 해주면 세 개의 그룹으로 묶임
- 매칭 결과를 MatchObject로 반환받는 함수
  ➡MatchObject의 group 함수를 통해 그룹 접근 가능
- (₩d{3})-(₩d{4})-(₩d{4})
  - ✓ group(0) : 전체 번호
  - ✓ group(1), group(2), group(3)을 통해 값을 얻을 수 있음

**우편번호**

- (₩d{3})-(₩d{2,3}) 패턴을 사용

# 정규식 패턴 만들기와 활용

## 2 정규식 그룹화

### 3 예제

#파일명 : exam13_11.py

```
import re

contents = "문의 사항이 있으면 010-1234-6789 으로 연락주시기 바랍니다."
pattern = r'(₩d{3})-(₩d{4})-(₩d{4})'
regex = re.compile(pattern)
result = regex.search(contents)

if result != None:
    phone1 = result.group(1)
    phone2 = result.group(2)
    phone3 = result.group(3)
    print(phone1)
    print(phone2)
    print(phone3)
else:
    print("전화번호가 없습니다.")
```

- 전화번호 패턴을 괄호를 이용해 그룹화
- 3개의 그룹으로 나누어졌으므로 group(1), group(2), group(3)으로 각각의 번호 추출

```
010
1234
6789
```

# 학습정리

## 1. 정규식과 정규식 적용 함수

- 정규식
  - 문자열이 특정 유형을 갖고 있을 때, 각 문자열의 규칙성을 특정한 문자 형태로 작성하여 문자열을 검색하거나 추출하고자 할 때, 입력된 값의 형태가 정해진 규칙에 맞는지 검색하기 위해 사용하는 방식
- re 모듈을 이용해서 패턴과 매치된 데이터를 찾음
  - 패턴을 컴파일 후 컴파일된 객체를 이용하여 데이터를 검색할 수 있음
  - 컴파일하지 않고 re 모듈을 통해 직접 패턴을 찾을 수 있음
- 파이썬에서는 match, search, findall, finditer, sub 함수 등을 제공함
- 패턴이 일치하면 일치하는 MatchObject 객체를 주로 반환하는데, 이 객체에는 매칭되는 데이터의 위치값이나 데이터 등에 대한 정보가 있음

## 학습정리

### 2. 정규식 패턴 만들기와 활용

- 정규식을 만들기 위해서는 패턴을 만들어야 함
- 패턴은 여러 가지 문자들의 조합으로 이루어짐
- 시작을 영문 대문자만 오게 하고 싶다면 ^[A-Z]라는 패턴을 만들어야 함
- [], {}, () 등으로 원하는 패턴을 만들어서 사용할 수 있음