정규 표현식

정규 표현식

- **정규 표현식(Regular Expression)**
  - 어떤 문자열 안에서 특정한 규칙을 가진 패턴을 찾기 위한 도구

    - 이메일 주소만 찾아내고 싶을 때
    - 전화번호 형식인지 확인하고 싶을 때
    - 주민등록번호처럼 숫자 13자리를 추출하고 싶을 때

정규 표현식

- **파이썬에서 정규 표현식**
  - 파이썬에서는 re라는 내장 모듈을 사용

```
import re
```

  - 정규 표현식 패턴
    - r'...' (raw string)
    - 문자열 앞의 r은 raw 문자열, 이스케이프 문자를 그대로 인식
    - 예: ₩n은 줄바꿈이 아니라 백슬래시 + n 문자 그대로 처리

정규 표현식

- 자주 쓰는 함수

함수	설명	예시
re.match()	문자열의 **처음**부터 패턴이 맞는지 확인	이름이 "홍길동"으로 시작하나요?
re.search()	문자열 **전체** 중에서 패턴이 있는지 확인	문자열 어디에든 숫자가 있나요?
re.findall()	패턴과 **일치하는 모든 값**을 리스트로 반환	문장 안에 있는 모든 숫자를 뽑아줘!

정규 표현식

- 정규 표현식 기호 (기초)

기호	의미	예시
.	아무 문자 1개	a.c → abc, axc 가능
₩d	숫자 1개 (0~9)	₩d₩d → 숫자 2자리
₩w	문자 1개 (영문자, 숫자, _)	₩w+ → 단어 찾기
₩s	공백 문자 (스페이스, 탭 등)	a₩s+b → "a b", "a b"
^	문자열의 시작	^Hello → "Hello..."
$	문자열의 끝	world!$ → "...world!"
+	앞의 문자가 **1번 이상 반복**	₩d+ → 숫자 여러 개
*	0번 이상 반복	a* → "", "a", "aaa" 가능

정규 표현식

- 정규 표현식 기호 (기초)

패턴	의미	
.	임의의 한 문자	
₩d	숫자(0~9)	
₩D	숫자가 아닌 문자	
₩w	문자, 숫자, 밑줄(_)	
₩W	문자, 숫자가 아닌 것	
₩s	공백 문자	
^	문자열의 시작	
$	문자열의 끝	
`a	b`	a 또는 b
[]	문자 집합 (ex. [abc])	
*	0회 이상 반복	
+	1회 이상 반복	
?	0 또는 1회 반복	
{n}	n회 반복	
{n,m}	n~m회 반복	

정규 표현식

- **활용 예제**
  - re.match(): 문자열 시작 부분에서 일치하는지 확인

```
import re

result = re.match(r'Hello', 'Hello, world!')
print(result.group())  # Hello
```

정규 표현식

- **활용 예제**
  - 특정 단어로 시작하는지 확인: re.match()

```
import re

text = "Hello world!"
result = re.match(r'Hello', text)
if result:
    print("Hello로 시작합니다!")  # 출력됨
```

정규 표현식

- **활용 예제**
  - re.search(): 문자열 전체 중 첫 일치 항목

```
import re

result = re.search(r'\d+', 'My age is 29')
print(result.group())  # 29
```

정규 표현식

- **활용 예제**
  - 이메일 주소 찾기

```
import re

text = "문의: hello@example.com"
result = re.search(r'\w+@\w+\.\w+', text)
print(result.group())  # hello@example.com
```

정규 표현식

- **활용 예제**
  - re.findall(): 모든 일치 항목을 리스트로 반환

```
import re

result = re.findall(r'\w+', 'Hello world! 123')
print(result)  # ['Hello', 'world', '123']
```

정규 표현식

- **활용 예제**
  - 숫자 찾기 (₩d+)

```
import re

text = "오늘은 2025년 6월 11일입니다."
numbers = re.findall(r'\d+', text)
print(numbers)  # ['2025', '6', '11']
```

정규 표현식

- **활용 예제**
  - 숫자만 뽑기

```
import re

str_test = "학생 점수: 수학 95점, 영어 88점, 과학 92점"

# 숫자만 뽑기
점수들 = re.findall(r'\d+', str_test)
print(점수들)  # ['95', '88', '92']
```

정규 표현식

- **활용 예제**
  - re.sub(): 패턴을 다른 문자열로 치환

```
import re

text = 'The price is $100'
new_text = re.sub(r'\$\d+', 'FREE', text)
print(new_text)  # The price is FREE
```

정규 표현식

- **활용 예제**
  - re.sub(): 바꾸기

```
import re

text = "비밀번호는 1234입니다"
new_text = re.sub(r'\d+', '****', text)
print(new_text)  # 비밀번호는 ****입니다
```

정규 표현식

- **활용 예제**
  - re.split(): 정규식을 기준으로 문자열 분할

```
import re

text = 'apple,banana;orange grape'
words = re.split(r'[;, ]+', text)
print(words)  # ['apple', 'banana', 'orange', 'grape']
```

정규 표현식

- **활용 예제**
  - re.compile()로 만든 객체는 아래 메서드를 사용

```
import re

pattern = re.compile(r'\d+')

pattern.match('123abc')
pattern.search('abc123')
pattern.findall('abc123 def456')
```

정규 표현식

- **활용 예제**
  - re.compile(): 자주 쓰는 패턴을 저장

```
import re

pattern = re.compile(r'\d+')
print(pattern.findall("전화번호는 01012345678입니다"))
# ['01012345678']
```

정규 표현식

- 플래그 (옵션)

플래그	의미
re.IGNORECASE or re.I	대소문자 구분 없이
re.MULTILINE or re.M	여러 줄 처리 (^, $가 각 줄마다 적용)
re.DOTALL or re.S	.이 개행 문자도 포함하도록 함

```
import re

match = re.search(r'^abc', 'abc\ndef', re.M)
if match:
    print("찾았습니다:", match.group())
else:
    print("못 찾았습니다.")
```

정규식 패턴

- **정규 표현식 문법**
  - 정규 표현식을 배우기 위해 기본으로 알아야 하는 개념은 메타문자(meta-characters) 이다. 메타문자는 문자를 설명하기 위한 문자로, 문자의 구성을 설명하기 위해 원래의 의미가 아니라 다른 의미로 쓰이는 문자를 뜻한다.
  - 기본 메타문자 []
    - 먼저 대괄호 [ ]는 [ ] 블록 사이의 문자와 매칭. [ ]에는 or의 의미.
      예를 들어, [abc]는 어떤 텍스트에 a 또는 b 또는 c라는 텍스트가 있는지 확인
  - 반복 관련 메타문자 -, +, *, ?, { }
    - + : 해당 글자가 1개 이상 출현
    - {} : 출현 횟수를 조정해야 할 때 사용하는 메타문자는 중괄호
    - * : 해당 글자가 0번부터 무한대까지 반복
    - () : 묶음을 표시
      [.]는 일반적인 마침표를 뜻하고 (.)는 줄 바꿈 기호를 제외한 전체 문자를 뜻한다.
    - 메타문자 |나 ^은 or와 not의 의미, 정규 표현식의 처음과 끝에는 메타문자 ^과 $를 붙인다.

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

정규식 패턴

- **정규 표현식 문법**
  - 전화번호 찾기

```
[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]
```

```
[0-9]+-[0-9]+-[0-9]+
```

```
[0-9]{3}-[0-9]{3,4}-[0-9]{4}
```

정규식 패턴

- ^ (caret)
  - 라인의 처음이나 문자열의 처음을 표시
  - 예 : ^aa (문자열의 처음에 aa를 포함하면 참, 그렇지 않으면 거짓)
- $ (dollar)
  - 라인의 끝이나 문자열의 끝을 표시
  - 예 : aaa$ (문자열의 끝에 aaa를 포함하면 참, 그렇지 않으면 거짓)
- . (period)
  - 임의의 한 문자를 표시
  - 예 : ^a.c (문자열의 처음에 abc, adc, aZc 등은 참, aa 는 거짓)
  - 예 : a..b$ (문자열의 끝에 aaab, abbb, azzb 등을 포함하면 참)
- [] (bracket)
  - 문자의 집합이나 범위를 나타냄, 두 문자 사이의 범위는 "-" 사용.
  - []내에서 "^"이 선행되면 not을 나타냄
  - 예 : [abc] (a, b, c 중 어떤 문자, "[a-c]."과 동일)
  - 예 : [Yy] (Y 또는 y)
  - 예 : [A-Za-z0-9] (모든 알파벳과 숫자)
  - 예 : [-A-Z]. ("-"(hyphen)과 모든 대문자)
  - 예 : [^a-z] (소문자 이외의 문자)
  - 예 : [^0-9] (숫자 이외의 문자)

정규식 패턴

- {} (brace)

  {} 내의 숫자는 직전의 선행 문자가 나타나는 횟수,범위를 나타냄

  예 : a{3} (‘a’의 3번만 반복인 aaa만 해당됨)

  예 : a{3,} (‘a’가 3번 이상 반복인 aaa, aaaa, … 등을 나타냄)

  예 : a{3,5} (aaa, aaaa, aaaaa 만 해당됨)

  예 : ab{2,3} (abb와 abbb 만 해당됨)

  예 : [0-9]{2} (두 자리 숫자)

  예 : doc[7-9]{2} (doc77, doc87, doc97 등이 해당)

  예 : [^Zz]{3} (Z와 z를 포함하지 않는 5개의 문자열, abc, ttt 등)

  예 : .{3,4}er (‘er’앞에 세 개 또는 네 개의 문자를 포함하는 문자열이므로 Peter, mother 등이 해당)

- * (asterisk)

  “*” 직전의 선행 문자가 0번 또는 여러 번 나타나는 문자열

  예 : ab*c (‘b’를 0번 또는 여러 번 포함하므로 ac, abc, abbbc 등)

  예 : * (선행 문자가 없는 경우이므로 임의의 문자열 및 공백 문자열)

  예 : .* (선행 문자가 “.”이므로 하나이상의 문자를 포함하는 문자열)

  예 : ab* (‘b’를 0번 또는 여러 번 포함하므로 a, accc, abb 등)

  예 : a* (‘a’를 0번 또는 여러 번 포함하므로 k, kd, a, aa, abb 등)

  예 : doc[7-9]* (doc7, doc777, doc778989, doc 등이 해당)

  예 : [A-Z].* (대문자로만 이루어진 문자열)

  예 : like.* (직전의 선행 문자가 ‘.’이므로 like에 0 또는 하나 이상의 문자가 추가된 문자열이 됨, like, likely, liker, likelihood 등)

정규식 패턴

- + (Plus Sign)

  "+" 직전의 선행 문자가 1번 이상 나타나는 문자열

  예 : ab+c ('b'를 1번 또는 여러 번 포함하므로 abc, abcd, abbc 등)

  예 : ab+ ('b'를 1번 또는 여러 번 포함하므로 ab, abcc, abb 등)

  예 : [A-Z]+ (대문자로만 이루어진 문자열)

  예 : like.+ (직전의 선행 문자가 '.'이므로 like에 하나 이상의 문자가 추가된 문자열이 됨, likely, liker, likelihood 등, 그러나 like는 해당 안됨)

- ? (question mark)

  "?" 직전의 선행 문자가 0번 또는 1번 나타나는 문자열

  예 : ab?c ('b'를 0번 또는 1번 포함하므로 abc, abcd 만 해당됨)

- () (parenthesis)

  ()는 정규식내에서 패턴을 그룹화 할 때 사용

- | (bar)

  or를 나타냄

  예 : a|b|c (a, b, c 중 하나, 즉 [a-c]와 동일함)

  예 : yes|Yes (yes나 Yes 중 하나, [yY]es와 동일함)

  예 : korea|japan|chinese (korea, japan, chinese 중 하나)

- ₩ (backslash)

  위에서 사용된 특수 문자들을 정규식 내에서 문자로 취급하고 싶을 때 '₩'를 선행시켜서 사용하면 됨

  예 : filename₩.ext ("filename.ext"를 나타냄)

  예 : [₩?₩[₩₩₩]] ('?', '[', '₩', ']' 중 하나)

정규 표현식 예제

구분	표현식	설명
한글만 입력가능	/^[가-힣]+$/	
숫자만 입력가능	/^[0-9]*$/	
영문만 입력가능	/^[a-zA-Z]+$/	
한글+ 영문만 입력가능	/^[가-힣a-zA-Z]+$/	
아이디(영문+숫자)	/^[a-zA-Z0-9]{4,12}$/	4글자 이상, 12자 미만, 영어 대소문자, 숫자, -, _ 사용 가능
비밀번호(영문/숫자+특문)	/^(?=.*[a-zA-Z])((?=.*₩d)|(?=.*₩W)).{6,20}$/	6글자 이상 20자 미만, 최소 1개의 숫자 혹은 특수 문자 포함
이메일 주소	/^[a-z0-9_+.-]+@([a-z0-9-]+₩.)+[a-z0-9]{2,4}$/	r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+₩.[a-zA-Z0-9-.]+'
전화번호	/^₩d{2,3}-₩d{3,4}-₩d{4}$/	
핸드폰입력	/^01([0|1|6|7|8|9]?)-?([0-9]{3,4})-?([0-9]{4})$/	
url입력	/^(file|gopher|news|nntp|telnet|https?|ftps?|sftp):₩/₩/([a-z0-9-]+₩.)+[a-z0-9]{2,4}.*$/	

- **정규 표현식 문법 연습**
  - 정규 표현식 연습장 웹 사이트(http://www.regexr.com) 활용