문자열 데이터 처리하기

이준원
SKT Data Scientist

학습 목표

1. 문자열을 다룰 수 있다.
2. 메소드를 활용할 수 있다.
3. 문자열을 포매팅할 수 있다.
4. 정규표현식을 사용할 수 있다.

K-Digital 훈련원

CONTENTS

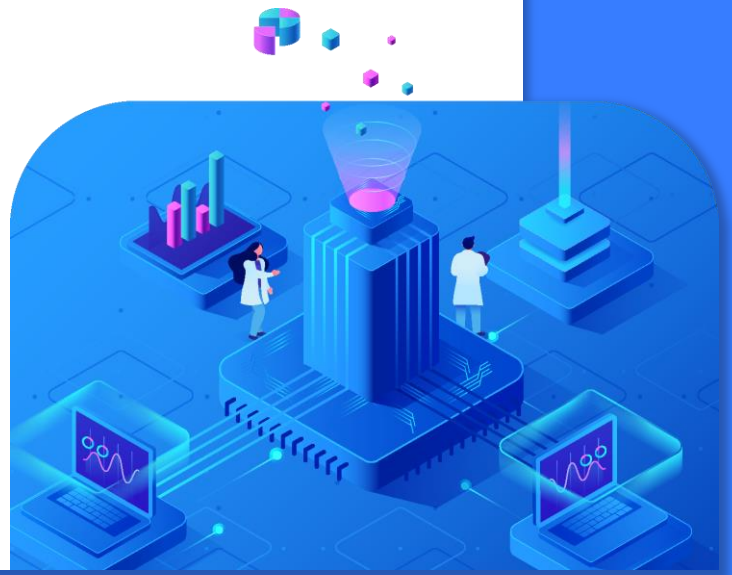
I

문자열 다루기

CHAPTER

- 파이썬 문자열 활용하기
- 문자열 추출하기

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



파이썬 문자열

- 파이썬 문자열 선언은 작은 따옴표, 큰 따옴표를 이용하여 선언
- 변수 word, voca에 문자열을 저장하여 사용
- 문자열의 길이는 특수문자를 포함한 문자의 개수

```
# python console

word1 = 'python'                    # 길이 = 6
word2 = "data analysis"             # 길이 = 13
```

파이썬 문자열

- 인덱스를 사용하여 문자열을 추출할 수 있음
- 하나의 문자는 길이가 1인 문자열

```
# python console

word1 = 'python'                    # 길이 = 6
```

인덱스	0	1	2	3	4	5
문자열	p	y	t	h	o	n
음수 인덱스	-6	-5	-4	-3	-2	-1

문자열 추출하기

```
# 문자열 추출하기

word1 = 'python'
word2 = 'data analysis'
```

```
print(word1[0])
```

```
p
```

```
print(word2[0])
```

```
d
```

```
print(word1[0:4])
```

```
pyth
```

```
print(word2[-1])
```

```
s
```

```
print(word2[-4:-1])
```

```
ysi
```

문자열 추출하기

```
word1 = 'python'
word2 = 'data analysis'
```

```
print(len(word1))
```

```
6
```

```
print(len(word2))
```

```
13
```

```
print(word1[:4])
```

```
pyth
```

```
print(word1[2:6])
```

```
thon
```

```
print(word2[:])
```

```
data analysis
```

```
print(word2[::2])
```

```
dt nlss
```

II

메소드 활용하기

CHAPTER

- 문자열 메소드 종류
- 문자열 메소드 활용방법

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다

문자열 메소드

- 문자열을 내가 원하는 방법으로 변환하고 싶을 때 사용
- 파이썬 문자열이 제공하는 다양한 메소드가 존재함
- 예시
  - 대소문자 변환 (lower, upper)
  - 문자 개수 반환 (count)
  - 특정 문자열 찾기 (find)
  - 특정 문자열 교체하기 (replace)
  - 문자열의 앞뒤 공백 제거하기 (strip)
  - 문자열을 구분자로 나누기 (split)

문자열 메소드

```
# 문자열 메소드

word = 'Big Data Analysis'
```

```
print(word.lower())
```

```
big data analysis
```

```
print(word.upper())
```

```
BIG DATA ANALYSIS
```

```
print(word.count('a'))
```

```
3
```

```
print(word.startswith('Big'))
```

```
True
```

```
print(word.endswith('analysis'))
```

```
False
```

문자열 메소드

```
# 문자열 메소드

word1 = 'python coding'
word2 = ' Big Data Analysis '
```

```
# 일치하는 문자의 첫번째 인덱스를 반환합니다
print(word1.find('o'))
```

```
4
```

```
print(word1.isalpha())
```

```
False
```

```
# 공백을 구분자로 문자열을 나눕니다
print(word1.split(' '))
```

```
['python', 'coding']
```

```
# 앞뒤 공백을 제거합니다
print(word2.strip())
```

```
Big Data Analysis
```

문자열 메소드 (join)

- join 메소드는 문자열을 연결하여 새로운 문자열을 반환
- join 메소드 앞에 지정된 문자(' ')를 사용하여 여러 개의 문자열을 해당 문자로 연결해줌

```
word1 = 'kpu'
word2 = 'digital'
word3 = 'data'

vocab1 = '1st'
vocab2 = 'class'

# 공백을 사용해서 여러개의 문자열을 하나로 합치기
sentense = ' '.join([word1, word2, word3, vocab1, vocab2])
print(sentense)

comma_sentense = ','.join([word1, word2, word3, vocab1, vocab2])
print(comma_sentense)
```

```
kpu digital data 1st class
kpu,digital,data,1st,class
```

문자열 메소드 (splitlines)

- splitlines 메소드는 여러 행을 가진 문자열을 분리하여 리스트로 반환
- 블로그, 뉴스와 같은 긴 텍스트 데이터를 다룰 때 문자열 데이터를 라인 별로 나눠서 조작함

```
multi_line_string = """맑고 포근한 봄 날씨가 이어지고 있습니다.
하지만 내일은 전국에 강풍을 동반한 요란한 비가 내릴 것으로 보여 주의가 필요합니다.
자세한 날씨, 기상캐스터 연결해 알아봅니다. 홍나실 캐스터!
오늘은 봄기운이 완연하다고요?
"""
multi_line_split = multi_line_string.splitlines()
multi_line_split
```

```
['맑고 포근한 봄 날씨가 이어지고 있습니다.',
 '하지만 내일은 전국에 강풍을 동반한 요란한 비가 내릴 것으로 보여 주의가 필요합니다.',
 '자세한 날씨, 기상캐스터 연결해 알아봅니다. 홍나실 캐스터!',
 '오늘은 봄기운이 완연하다고요?']
```

```
# 개별 라인별로 문자열을 조작하기
for line in multi_line_split:
    print(line[:20])
```

```
맑고 포근한 봄 날씨가 이어지고 있습
하지만 내일은 전국에 강풍을 동반한
자세한 날씨, 기상캐스터 연결해 알아
오늘은 봄기운이 완연하다고요?
```

문자열 메소드 (replace)

- replace 메소드는 특정 문자열을 빼거나 교체할 때 사용
- 특수문자, 고유명사 등의 불필요한 문자열을 제거하는 전처리에 많이 사용됨

```
multi_line_string = """맑고 포근한 봄 날씨가 이어지고 있습니다.
하지만 내일은 전국에 강풍을 동반한 요란한 비가 내릴 것으로 보여 주의가 필요합니다.
자세한 날씨, 기상캐스터 연결해 알아봅니다. 홍나실 캐스터!
오늘은 봄기운이 완연하다고요?
"""
replace_and_split = multi_line_string.replace(".", "") \
                                     .replace("!", "") \
                                     .replace(",", "").splitlines()
```

```
replace_and_split
```

```
['맑고 포근한 봄 날씨가 이어지고 있습니다',
 '하지만 내일은 전국에 강풍을 동반한 요란한 비가 내릴 것으로 보여 주의가 필요합니다',
 '자세한 날씨 기상캐스터 연결해 알아봅니다 홍나실 캐스터',
 '오늘은 봄기운이 완연하다고요?']
```

III

문자열 포매팅하기

CHAPTER

- 문자열 포매팅이란
- 포매팅 방법

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다

문자열 format

- 문자열을 편리하게 출력할 수 있는 기능
- 출력할 문자열의 형식을 지정하거나 변수를 조합하여 새로운 문자열을 출력하는 방법

- 예시
  - 이름: 김진수 → {name}
  - 성별: 남자 → {gender}
  - 나이: 20세 → {age}
  - 주거지: 서울 → {location}

문자열 format

- 문자열을 삽입할 위치를 {}로 지정하고 format 메소드를 사용함
- 이 때 {}를 placeholder라고 지칭함

```
name = '김진수'
sentense = '안녕하세요, 제 이름은 {}입니다.'

print(sentense.format(name))
```

```
안녕하세요, 제 이름은 김진수입니다.
```

```
print(sentense.format('이지원'))
```

```
안녕하세요, 제 이름은 이지원입니다.
```

문자열 format

- 여러 개의 문자열을 format하고 싶을 때는 인덱스 개념을 사용
- 단어 개수만큼 인덱스를 사용해 0, 1, 2, …로 늘려줌

```
# 인덱스를 사용한 format

sentense = """안녕하세요, 제 이름은 {0}입니다.
저의 성별은 {1}이고, 나이는 {2}세입니다.
"""

name = '김진수'
gender = '남자'
age = '20'

print(sentense.format(name, gender, age))
```

```
안녕하세요, 제 이름은 김진수입니다.
저의 성별은 남자이고, 나이는 20세입니다.
```

```
print(sentense.format('이지원', '여자', '30'))
```

```
안녕하세요, 제 이름은 이지원입니다.
저의 성별은 여자이고, 나이는 30세입니다.
```

문자열 format

- placeholder에 변수명을 지정할 수 있음
- format 메소드에도 변수명을 지정하여 전달해야 함

```
# 변수명을 사용한 문자열 format

sentense = """안녕하세요, 제 이름은 {name}입니다.
저의 성별은 {gender}이고, 나이는 {age}세입니다.
"""

_name = '김진수'
_gender = '남자'
_age = '20'

print(sentense.format(name=_name, gender=_gender, age=_age))
```

```
안녕하세요, 제 이름은 김진수입니다.
저의 성별은 남자이고, 나이는 20세입니다.
```

```
print(sentense.format(name='이지원', gender='여자', age='30'))
```

```
안녕하세요, 제 이름은 이지원입니다.
저의 성별은 여자이고, 나이는 30세입니다.
```

숫자 데이터 format

- python의 숫자 데이터도 placeholder를 사용할 수 있음
- 쉼표를 넣어서 숫자를 표현할 수도 있음 (ex. 480,000)

```
sentense = '현재 온도는 {}도 입니다.'

print(sentense.format(20.14))
```

```
현재 온도는 20.14도 입니다.
```

```
sentense = '2020년 A 회사의 매출은 {:,}원입니다.'

print(sentense.format(12040000))
```

```
2020년 A 회사의 매출은 12,040,000원입니다.
```

숫자 데이터 format

- 소수의 경우 소수점 이하의 숫자 개수를 지정하여 출력할 수 있음
  - %를 사용하여 백분율로 환산 가능
- 숫자를 정해진 자리수로 표현할 수 있음 (ex. 23 → 00023)

```
sentense = '현재 우리 제품의 불량률은 {0:.4}이며, 퍼센트로 나타내면 {0:.5%}입니다.'

print(sentense.format(0.007414123))
```

```
현재 우리 제품의 불량률은 0.007414이며, 퍼센트로 나타내면 0.74141%입니다.
```

```
sentense = '당신의 사번은 {0:05d}입니다.'

print(sentense.format(142))
```

```
당신의 사번은 00142입니다.
```

% 연산자로 format

- % 연산자를 사용하여 문자열을 format하는 방법
- 삽입할 값이 정수라면 %d, 문자열이면 %s, 실수라면 %f를 사용
  - %.3d (ex. 030), %.2f (ex. 90.15) 의 응용도 가능

```
sentense = '당신의 이름은 %s입니다.' % '이지원'

print(sentense)
```

```
당신의 이름은 이지원입니다.
```

```
sentense = '당신의 나이는 %d세입니다.' % 30

print(sentense)
```

```
당신의 나이는 30세입니다.
```

```
sentense = '당신의 이름은 %s이고, 점수는 %.2f점입니다.' % ('이지원', 90.155)

print(sentense)
```

```
당신의 이름은 이지원이고, 점수는 90.16점입니다.
```

f-stirngs 사용 format

- python 3.6 버전에 새로 도입된 format
- 문자열 앞에 f를 붙이는 것이 특징이며 기존 방법보다 빠른 속도로 문자열을 처리함

```
name = '이지원'
sentense1 = f'내 이름은 {name}입니다.'

print(sentense1)
```

```
내 이름은 이지원입니다.
```

```
lat = '10.323'
lon = '79.124'
sentense2 = f'현재 위도는 {lat}, 경도는 {lon}입니다.'

print(sentense2)
```

```
현재 위도는 10.323, 경도는 79.124입니다.
```

IV

정규표현식 사용하기

CHAPTER

- 정규표현식 개념 및 문법
- 정규표현식 메소드 활용

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다

정규표현식이란?

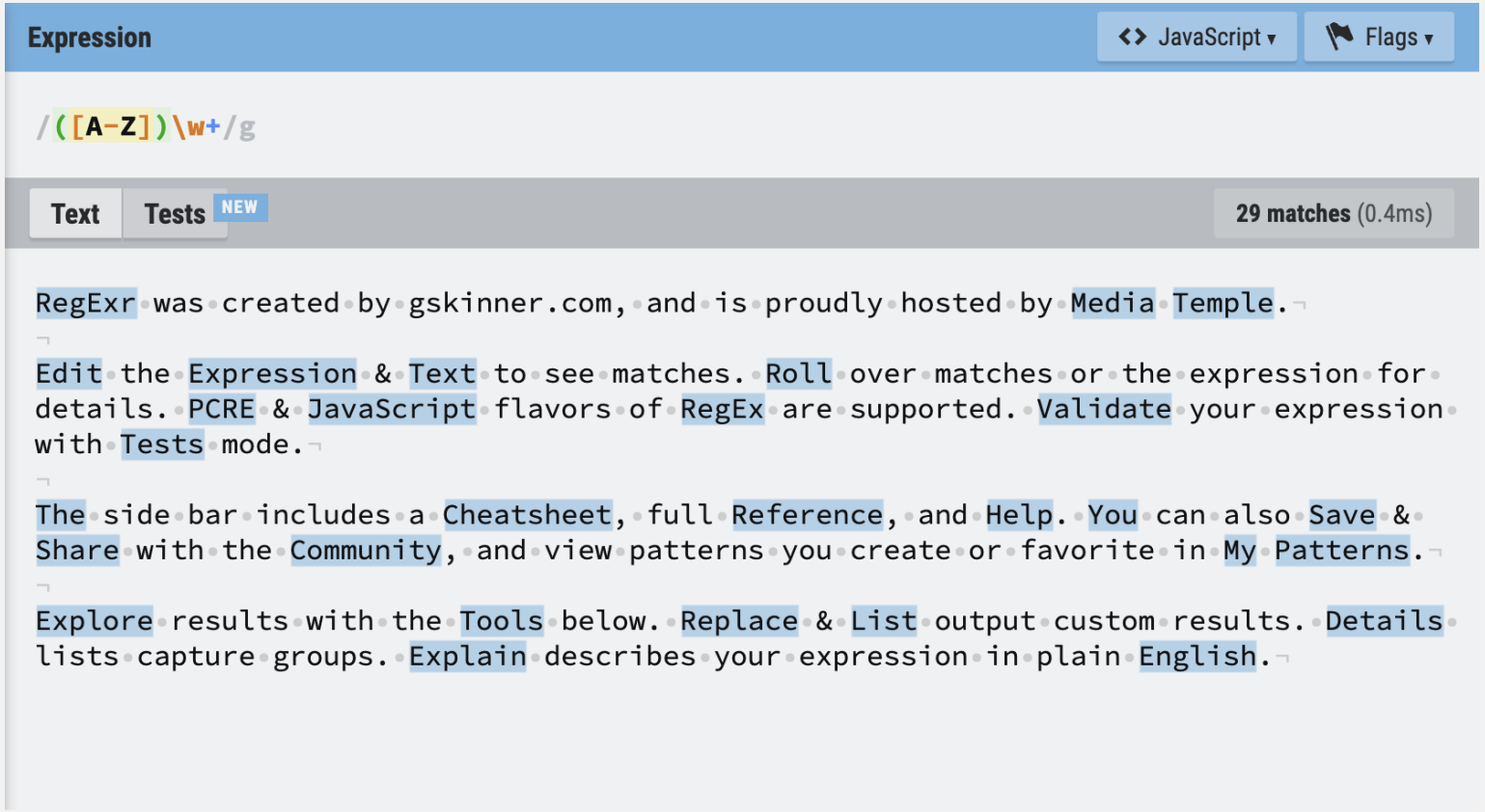
- 아주 많은 문자열 가운데 내가 원하는 패턴의 문자열을 추출하려면?

  ex) 'I like a car and she wants a carpet' 이라는 문자열에서 car로 시작하는 단어 추출

- 내가 찾고자 하는 문자열에 해당하는 정규식 패턴을 만들어야 함

- https://regexr.com 와 같은 웹사이트에서 테스트 가능
  - 정규식 패턴
  - 추출 대상 문자열

정규표현식이란?


Expression
JavaScript
Flags
/([A-Z])\w+/g
Text
Tests
NEW
29 matches (0.4ms)
RegExr was created by gskinner.com, and is proudly hosted by Media Temple.
Edit the Expression & Text to see matches. Roll over matches or the expression for details. PCRE & JavaScript flavors of RegEx are supported. Validate your expression with Tests mode.
The side bar includes a Cheatsheet, full Reference, and Help. You can also Save & Share with the Community, and view patterns you create or favorite in My Patterns.
Explore results with the Tools below. Replace & List output custom results. Details lists capture groups. Explain describes your expression in plain English.

정규표현식 특수문자

문자	설명
\d	숫자 1개를 의미함 (0-9)
\D	숫자 이외에 문자 1개를 의미함 (0-9를 제외한 모든 문자)
\s	공백이나 탭 1개를 의미함
\S	공백 문자 외에 문자 1개를 의미함
\w	알파벳, 숫자, underscore 문자 1개를 의미함 (a-z, A-Z, 0-9, _)
\W	위에 적힌 문자(\w)를 제외한 모든 문자 1개를 의미 (한국어, 일본어, 특수문자 등)

기본 정규표현식 문법

문법	예시	설명
.	...k	문자(k) 앞에 문자가 3개 있는 패턴을 찾음 (back)
^	^Da	문자열이 처음부터 Da로 일치하는 패턴을 찾음 (Data)
$	ne$	문자열의 마지막이 ne으로 끝나는 패턴을 찾음 (done)
*	ab*	a 이후 b가 0번 이상 등장하는 패턴을 찾음 (a, abbb)
+	a\w+	a 이후 알파벳이나 숫자가 1번 이상 등장하는 패턴을 찾음 (aw, a1c3)
{m}	a\d{3}	a 이후 숫자가 3개인 패턴을 찾음 (addd)
?	date?	? 앞의 문자(e)가 있거나 없는 패턴을 찾음 (dat, date)
[]	[abc]go	a, b, c 중에 1개를 포함하고 그 뒤의 문자열이 go인 패턴을 찾음 (cgo, bgo)
()	(\d{3})-(\d{4})	()에 지정한 패턴을 찾을 때 사용함 (123-9284)

정규표현식 메소드

- match() 메소드는 문자열의 처음부터 검색하여 찾아낸 패턴의 양 끝 인덱스를 반환함

```
import re

number = '123 4567 890'
```

```
match = re.match('[0-9]{3}', number)
print(match)
```

```
<re.Match object; span=(0, 3), match='123'>
```

```
match = re.match('[0-9]{10}', number)
print(match)
```

```
None
```

```
match = re.match('(\d+)\s(\d+)\s(\d+)', number)
print(match)
```

```
<re.Match object; span=(0, 12), match='123 4567 890'>
```

정규표현식 메소드

- findall() 메소드는 일치하는 패턴을 모두 찾아 리스트로 반환함

```
import re

number = '123 4567 890'
```

```
regex_list = re.findall('[0-9]{3}', number)
print(regex_list)
```

```
['123', '456', '890']
```

```
regex_list = re.findall('\d{3,4}', number)
print(regex_list)
```

```
['123', '4567', '890']
```

```
regex_list = re.findall('^abc', number)
print(regex_list)
```

```
[]
```

내용 정리

- 데이터를 잘 다루기 위해서는 문자열도 잘 처리할 수 있어야 함
- 문자열이 제공하는 다양한 메소드가 존재함
  - lower, upper, count, find, replace, strip, split
- 문자열 format을 사용하여 새로운 문자열을 편리하게 출력할 수 있음
- 정규표현식을 사용하면 내가 원하는 형태의 문자열만을 패턴으로 찾아낼 수 있음

Thank you

junwonee@gmail.com