

Chapter 8-2

Regular Expression

Contents

- Regular Expression
- Case Study: MPAA Rating Pattern
- Case Study: Social Security Number
- Case Study: DNA Sequencing
- Case Study: Web Searches and Enron Legal Documents
- Computational Problem

Regular Expression

■ Python에서의 Regular Expression

```
import re
pattern = r'apple'
string = 'pineapple, cherry, apple'
match = re.findall(pattern, string)
print(match)
```

regular expression 모듈 import

regular expression 문자열 앞에는
r을 붙임

string에서 pattern(regular expression)과
일치하는 부분을 검색하여 리스트로 반환

match 리스트에 저장된 문자열을 출력

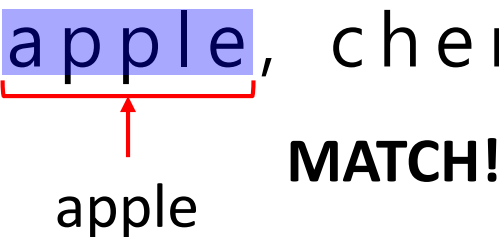
```
['apple', 'apple']
>>>
```

Regular Expression

- Python에서의 Regular Expression

```
import re
pattern = r'apple'
string = 'pineapple, cherry, apple'
match = re.findall(pattern, string)
print(match)
```

pineapple, cherry, apple



MATCH!

```
['apple', 'apple']
>>>
```

Case Study: MPAA Rating Pattern

■ Finding MPAA Rating Pattern

```
import re
str = 'This is a NC-17 rated movie'
pattern = r'G|PG|PG-13|R|NC-17'
match = re.findall(pattern, str)

if len(match) > 0:
    print('MPAA Rating:', match)
```

← 패턴 형식: *r'regular expression'*

← 패턴에 맞는 문자열이 있을 경우 True

```
MPAA Rating: ['NC-17']
>>>
```

Case Study: Social Security Number

■ Checking Social Security Number Pattern

```
import re
ssn = input('Enter a SSN: ')

pattern = r'(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)-
(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)-
(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)'

match = re.findall(pattern, ssn)

if len(match) == 1 and len(ssn) == 11:
    print(ssn, 'is valid SSN pattern')
else:
    print(ssn, 'is not valid SSN pattern')
```

```
Enter a SSN:123-45-6789
123-45-6789 is valid SSN
>>>
```

```
Enter a SSN:12-345-6789
12-345-6789 is not valid SSN
>>>
```

Case Study: Social Security Number

- Checking Social Security Number Pattern – Simple Pattern

```
import re
ssn = input('Enter a SSN: ')

pattern = r'\d{3}-\d{2}-\d{4}'

match = re.findall(pattern, ssn)

if len(match) == 1 and len(ssn) == 11:
    print(ssn, 'is valid SSN pattern')
else:
    print(ssn, 'is not valid SSN pattern')
```

```
Enter a SSN:123-45-6789
123-45-6789 is valid SSN
>>>
```

```
Enter a SSN:12-345-6789
12-345-6789 is not valid SSN
>>>
```

Case Study: DNA Sequencing

■ Finding DNA Sequencing

Searching a DNA Database

CAGACTTTCAGAACTGTCAGTTCCCCGGATTTTACCCATCACATTTTGCTACTACTTTC
TACTACTATATACTTTTCCAATTTTCATACGGGTACTATTATCCATACTCTACTATTAC



Case Study: DNA Sequencing

■ Finding DNA Sequencing

```
import re
dna = 'CAGACTTTCAGAACTGTCAG...skip...TTAC'
pattern = r'(AC.*GAA.*AG)'
match = re.findall(pattern, dna)

print('Found', len(match), 'subsequence(s)')
print(match)

found 1 subsequence(s)
['ACTTTCAGAACTGTCAG']
>>>
```

Case Study: Web Searches and Enron Legal Documents

- Finding All CAPS Word Pattern - Incorrect

All CAPS Word Pattern (Incorrect)

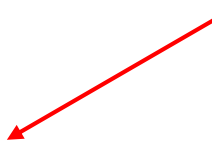
[A-Z]+

```
import re
string = 'Subject: Enron - Livingston County \
PRIVILEGED AND CONFIDENTIAL'
pattern = r'[A-Z]+'
match = re.findall(pattern, string)

print('Found', len(match), 'result(s)')
print(match)
```

원하지 않은 단일 대문자가 추출된 원인은?

```
Found 7 result(s)
['S', 'E', 'L', 'C', 'PRIVILEGED', 'AND', 'CONFIDENTIAL']
>>>
```



Case Study: Web Searches and Enron Legal Documents

- Finding All CAPS Word Pattern – Correct Pattern?

All CAPS Word Pattern

`\W[A-Z]+\W`

```
import re
string = 'Subject: Enron - Livingston County \
PRIVILEGED AND CONFIDENTIAL '
pattern = r'\W[A-Z]+\W'
match = re.findall(pattern, string)

print('Found', len(match), 'result(s)')
print(match)
```

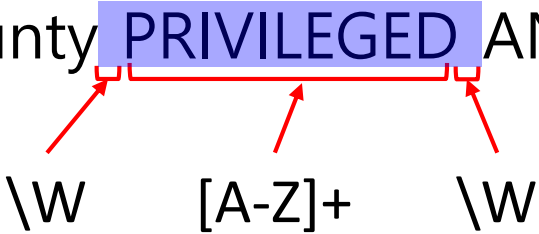
여전히 결과가 제대로 나오지 않으므로
패턴이 잘 못 정의 되었음을 확인할 수 있음

```
Found 2 result(s)
[' PRIVILEGED ', ' CONFIDENTIAL ']
>>>
```

Case Study: Web Searches and Enron Legal Documents


- Finding All CAPS Word Pattern – **Not working as Expected**

... County **PRIVILEGED** AND CONFIDENTIAL



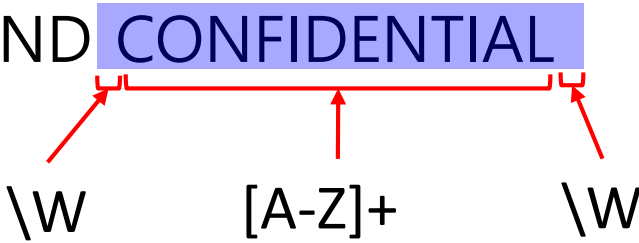
MATCH!

... County **PRIVILEGED** AND CONFIDENTIAL



NO MATCH!

... County **PRIVILEGED** AND **CONFIDENTIAL**



MATCH!

Case Study: Web Searches and Enron Legal Documents

■ Finding All CAPS Word Pattern – Correct Pattern!

All CAPS Word Pattern

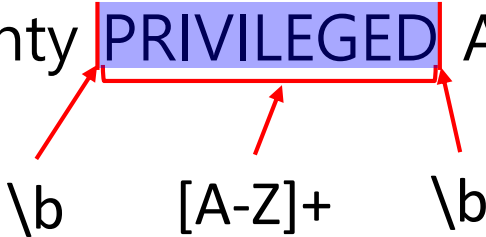
`\b[A-Z]+\b`

- \W대신 \b를 사용하여 단어의 경계를 일치(match) 시킬 수 있음
- \b : 단어 경계 패턴. 단어의 시작과 끝에 일치되는 정규식 표현. 단어 앞 뒤의 공백, 마침표 등의 문자에 일치되지 않고 가상의 단어 경계선에 일치되는 특징을 가진다 → 단어 추출 시 용이

Case Study: Web Searches and Enron Legal Documents

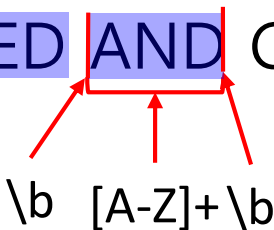
- Finding All CAPS Word Pattern – Correct Pattern for Python

... County **PRIVILEGED** AND CONFIDENTIAL



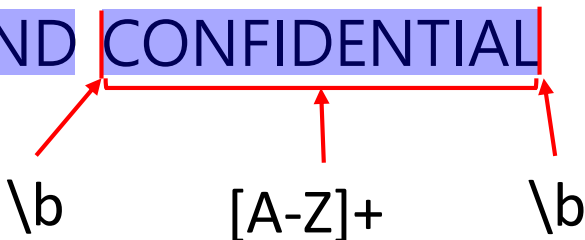
MATCH!

... County **PRIVILEGED AND** CONFIDENTIAL



MATCH!

... County **PRIVILEGED AND CONFIDENTIAL**



MATCH!

Case Study: Web Searches and Enron Legal Documents

- Finding All CAPS Word Pattern – **Correct Pattern!**

All CAPS Word Pattern

`\b[A-Z]+\b`

```
import re
string = 'Subject: Enron - Livingston County \
PRIVILEGED AND CONFIDENTIAL '
pattern = r'\b[A-Z]+\b'

match = re.findall(pattern, string)

print('Found', len(match), 'result(s)')
print(match)

Found 3 result(s)
['PRIVILEGED', 'AND', 'CONFIDENTIAL']
>>>
```

Case Study: Web Searches and Enron Legal Documents

- Sequence of at least 3 All CAPS Words

Sequence of at least 3 All CAPS Words

`\W([A-Z]+\W){3,}`

```
import re
string = 'Subject: Enron - Livingston County \
PRIVILEGED AND CONFIDENTIAL '
pattern = r'\W([A-Z]+\W){3,}'
```

```
match = re.search(pattern, string)
```

```
print(match.group())
```

참고: {3,}과 같이 연속된 패턴을 찾을 경우에 findall() 대신 search()와 group() 함수를 사용합니다

PRIVILEGED AND CONFIDENTIAL

```
>>>
```


Computational Problem

The Problem

A Metric Conversion Program

다음은 자동차의 최고 속도 기록에 대한 텍스트이다. 아래 텍스트를 출력 후, 본문에 사용된 미국식 속도단위(mph)를 표준 국제단위계 (International System of Units)인 km/h로 변환하여 출력하세요

```
Ruf CTR recorded 212.509 mph. Tested in  
1988. McLaren F1 was able to reach 240 mph  
in 1993. In 2005, Bugatti Veyron 16.4  
recorded 253.81 mph In 2010, Bugatti  
Veyron SS reached 257.87 mph
```

A Metric Conversion Program

Problem Analysis

- 본문에서 속도를 표기한 문자열의 패턴을 분석하여 정규식을 작성
- 본문에서 정규식에 일치되는 문자열을 모두 검색
- 검색된 문자열의 속도와 형식을 mph에서 km/h로 변환
- 변환된 문자열을 본문에 대체

Ruf CTR recorded **212.509 mph**. Tested in 1988. McLaren F1 was able to reach **240 mph** in 1993. In 2005, Bugatti Veyron 16.4 recorded **253.81 mph** In 2010, Bugatti Veyron SS reached **257.8 mph**

A Metric Conversion Program

Problem Analysis

본문에서 속도를 표기한 문자열 파악 및 패턴을 분석

2	1	2	.	5	0	9		m	p	h	
2	4	0		m	p	h					
2	5	3	.	8	1		m	p	h		
2	5	7	.	8		m	p	h			

- ⊖ 1개 이상의 digit으로 패턴이 시작됨
- ⊖ digit 패턴 이후, 0 혹은 1개의 dot이 있음
- ⊗ dot 패턴 이후 0개 이상의 digit이 있음
- ④ digit 패턴 이후 0개 이상의 공백이 있음
- ⑤ 공백 패턴 이후 'mph'문자열로 종료 됨

regular expression
➡ `\d+\.? \d* \s* mph`

A Metric Conversion Program

Algorithm Analysis

- ① 정규식과 일치하는 문자열을 리스트로 추출

`['212.509 mph' '240 mph', '253.81 mph' '257.8 mph']`

- ② ① 결과에서 수치 부분만 다시 추출

`['212.509' '240', '253.81' '257.87', '267.857']`

- ③ ②를 km/h 단위 값으로 변환

`['342.0' '386.2', '408.5' '415.0', '431.1']`

- ④ 'km' 단위를 추가

`['342.0km' '386.2km', '408.5km' '415.0km', '431.1km']`

- ⑤ 본문 text에서 ①단계에서 일치된 부분을 ④의 결과로 대체

A Metric Conversion Program

Data Representation

원본 텍스트

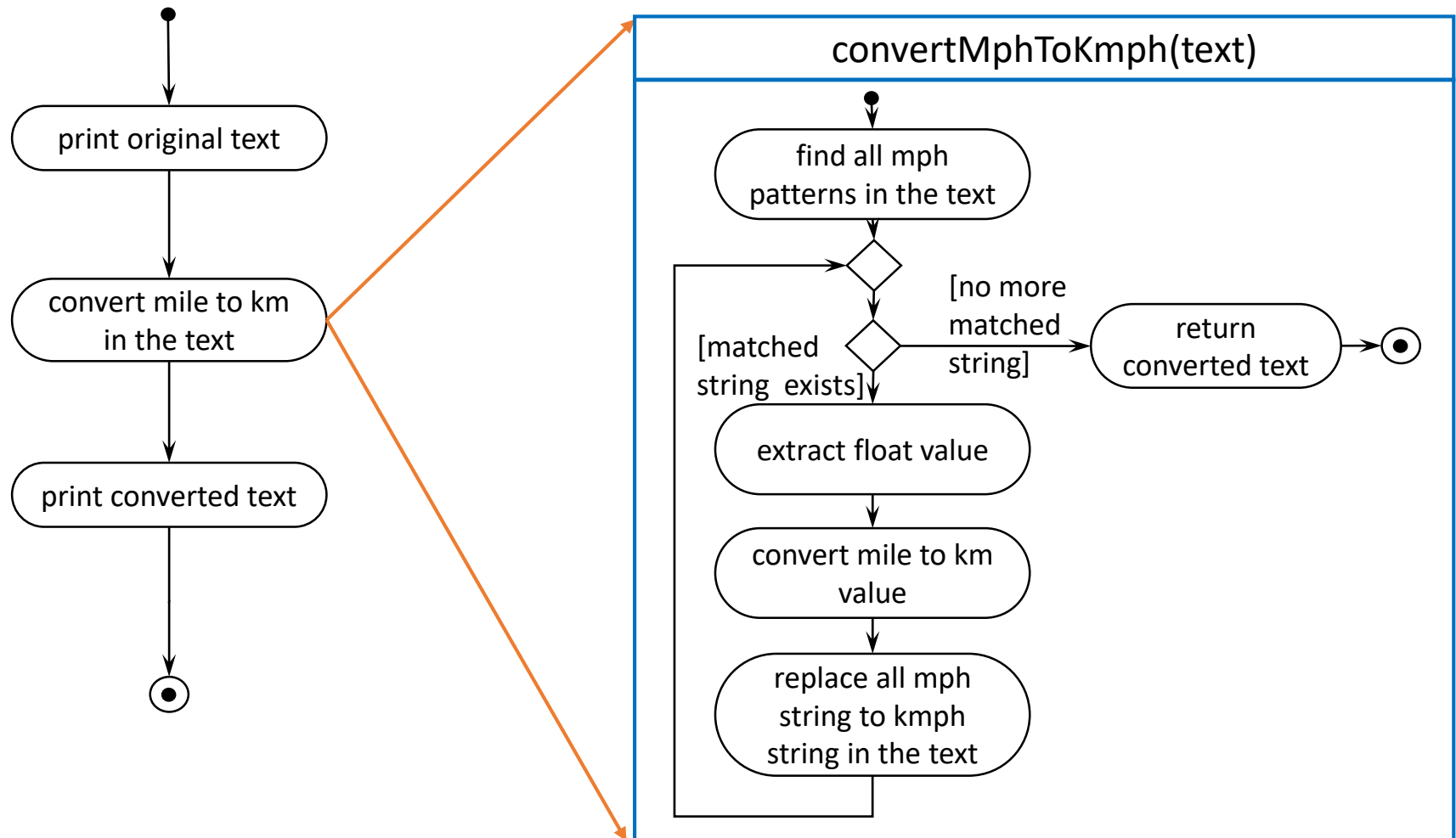
`originalText`

변환된 문자열

`convertedText`

A Metric Conversion Program

Algorithmic Thinking & Decomposition



A Metric Conversion Program

Program Design

Program Introduction

Displaying welcome messages

Processing

Applying Regular expression and string operations

Display results

Display converted text

A Metric Conversion Program

Program Implementation

```
import re
def convertMphToKmph(text):
    mphPattern = r'\d+\.\?\d*\s*mph'
    floatPattern = r'\d+\.\?\d*'
    match = re.findall(mphPattern, text)

    for mphString in match:
        value = re.findall(floatPattern, mphString)
        kmValue = float(value[0]) * 1.60934
        kmphString = format(kmValue, '.1f') + ' km/h'
        text = text.replace(mphString, kmphString)

    return text
```


A Metric Conversion Program

Program Implementation

```
originalText = 'Ruf CTR recorded 212.509 mph. Tested in 1988. \
McLaren F1 was able to reach 240 mph in 1993. \
In 2005, Bugatti Veyron 16.4 recorded 253.81 mph \
In 2010, Bugatti Veyron SS reached 257.8 mph'

print('Welcome to Metric Conversion Program')
print('\nOriginal Text')
print(originalText)
convertedText = convertMphToKmph(originalText)
print('\nConverted Text')
print(convertedText)
```

A Metric Conversion Program

Program Execution

Welcome to Metric Conversion Program

Original Text

Ruf CTR recorded 212.509 mph. Tested in 1988. McLaren F1 was able to reach 240 mph in 1993. In 2005, Bugatti Veyron 16.4 recorded 253.81 mph In 2010, Bugatti Veyron SS reached 257.8 mph

Converted Text

Ruf CTR recorded 342.0 km/h. Tested in 1988. McLaren F1 was able to reach 386.2 km/h in 1993. In 2005, Bugatti Veyron 16.4 recorded 408.5 km/h In 2010, Bugatti Veyron SS reached 414.9 km/h

>>>