2020 혁신성장 청년인재 집중양성 사업

빅데이터 수업 2주차

Web Crawling and Scraping

데이터 수집 1

#Python #Node.js #WebBot

시작

A table of Contents

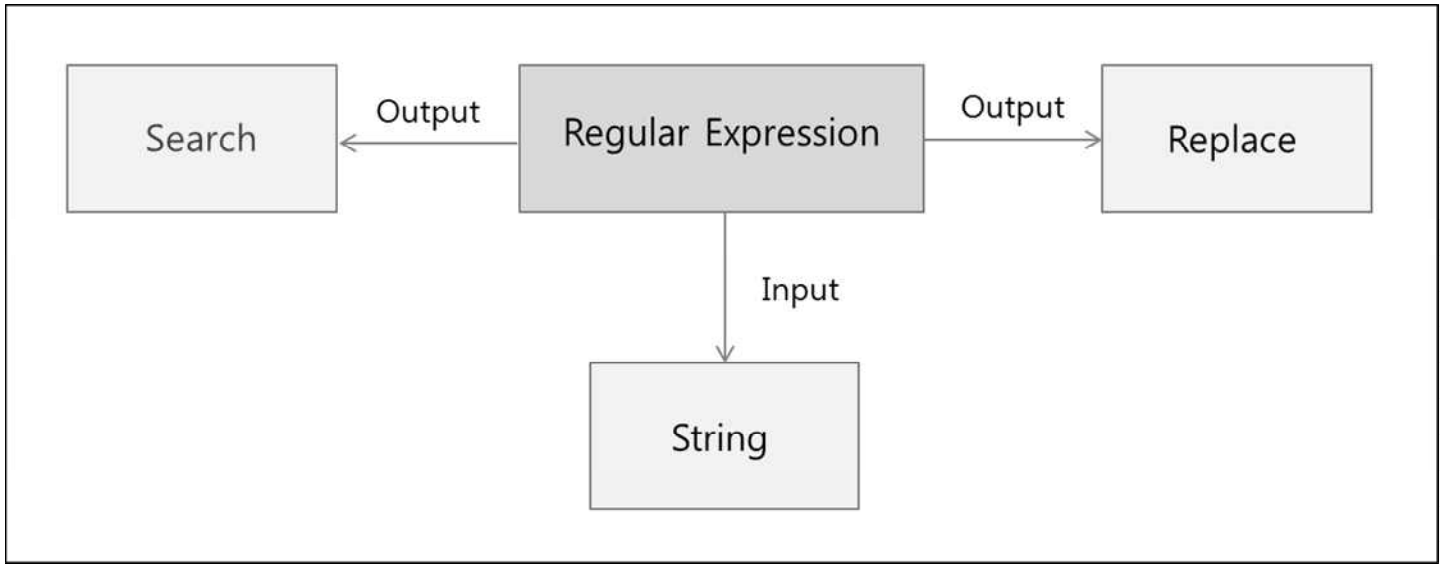
Part 1

표현 정규식

표현 정규식

서버에 다녀 오지도 않고 어떻게 처리 했을까?

표현 정규식

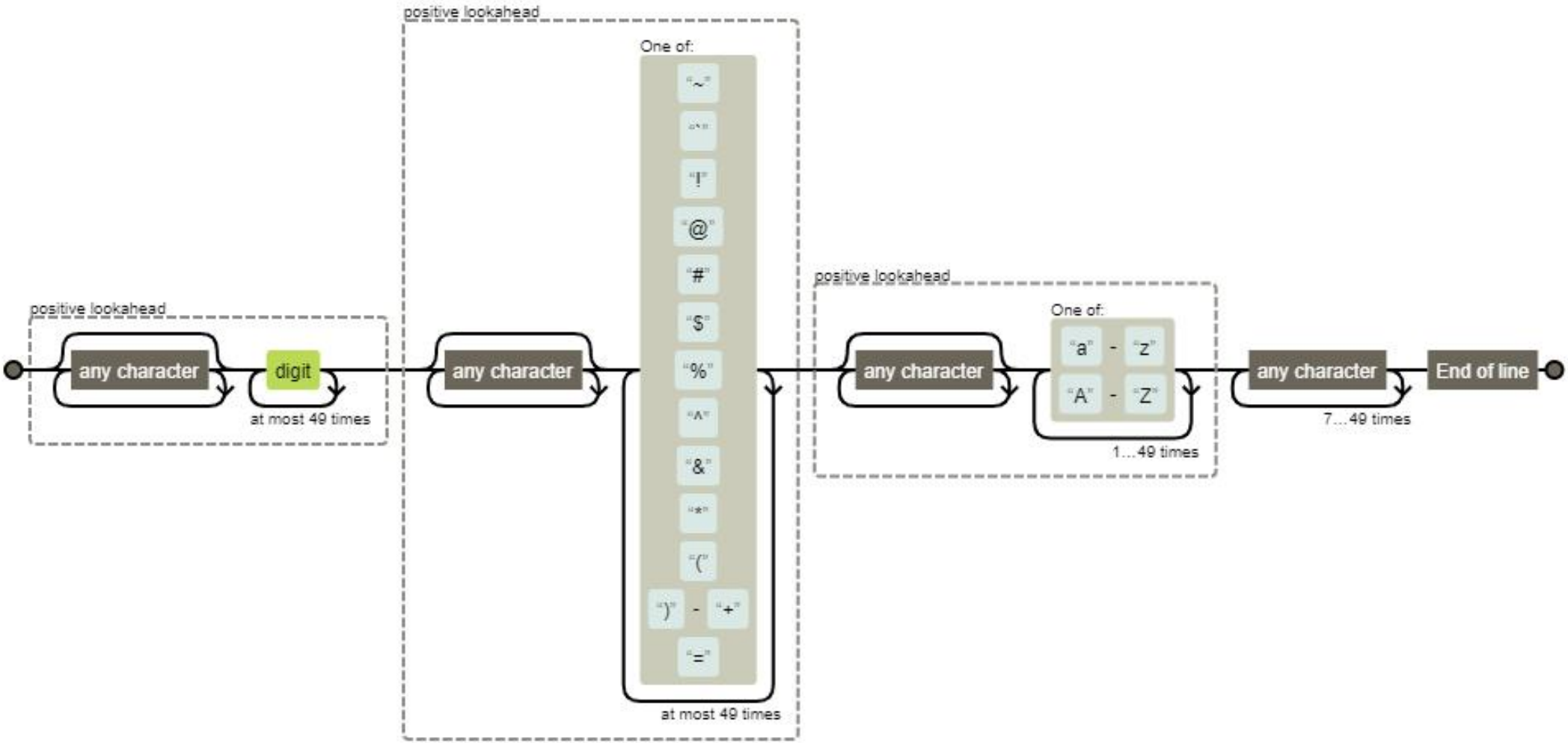

Search
Output
Regular Expression
Output
Replace
Input
String

표현 정규식

/(?=.*₩d{1,50})(?=.*[~`!@#$%₩^&*()-+=]{1,50})(?=.*[a-zA-Z]{2,50}).{8,50}$/

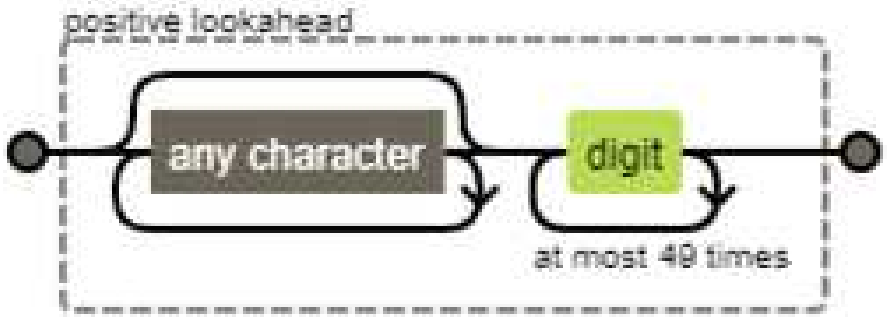
숫자, 특문 각 1회 이상, 영문은 2개 이상 사용하여 8자리 이상 50자리 이하로 입력

표현 정규식


positive lookahead
any character
digit
at most 49 times
positive lookahead
any character
One of:
"~"
"`"
"!"
"@"
"#"
"$"
"%"
"^"
"&"
"*"
"("
")" - "+"
"="
at most 49 times
positive lookahead
any character
One of:
"a" - "z"
"A" - "Z"
1…49 times
any character
7…49 times
End of line

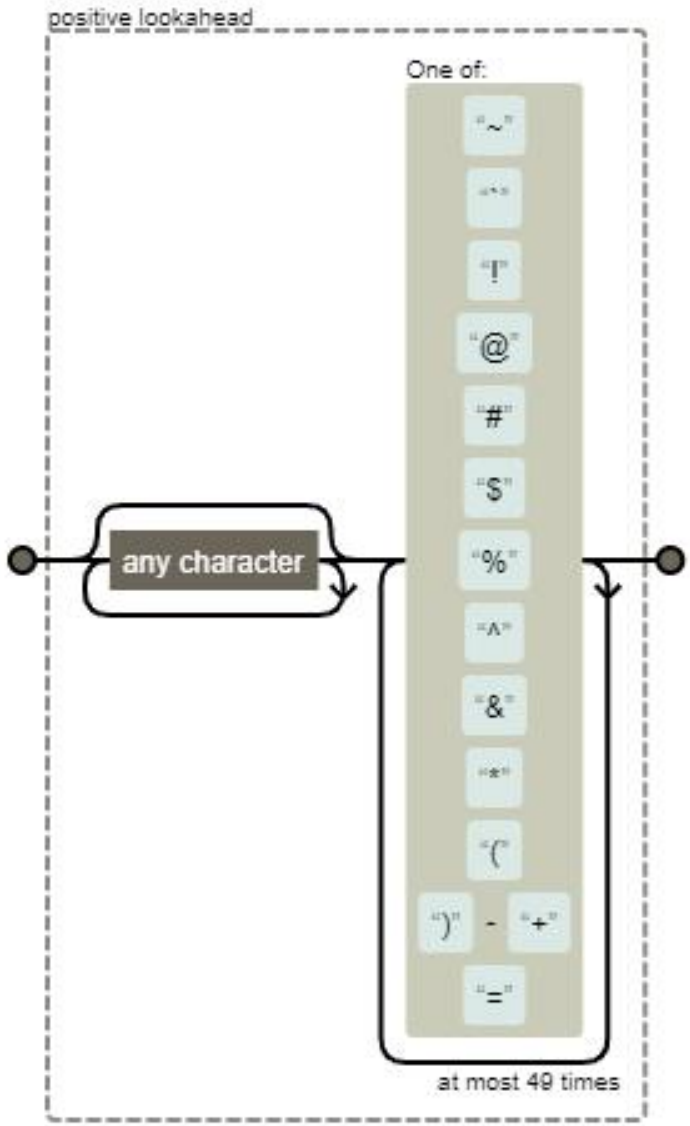
표현 정규식

/(?=.*₩d{1,50})
/



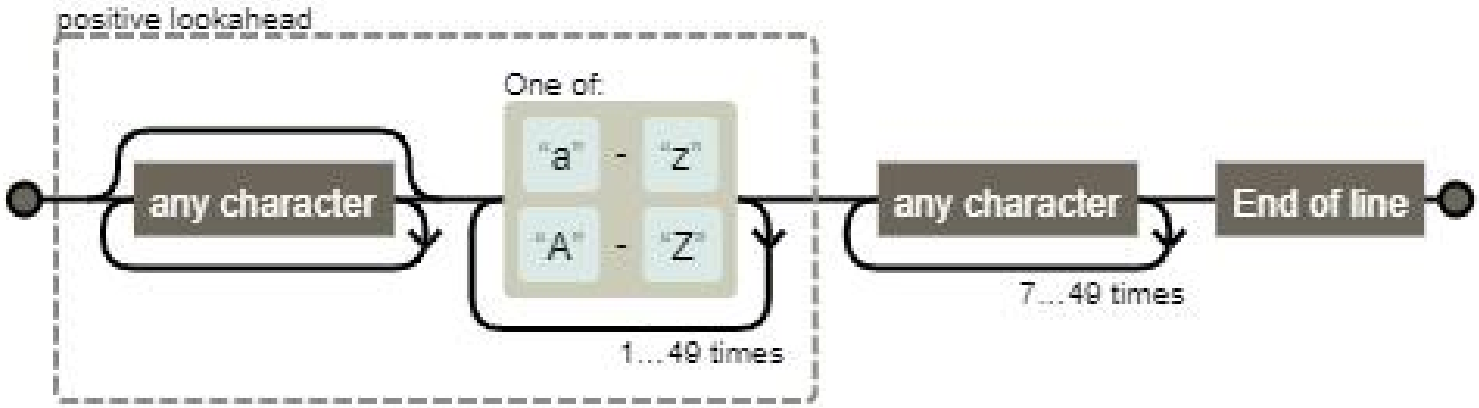
표현 정규식

(?=.*[~`!@#$%₩^&*()-+=]{1,50})

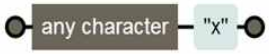
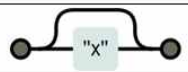
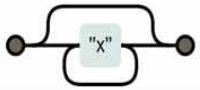
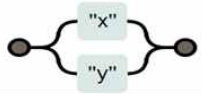
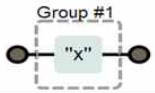
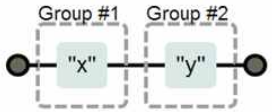
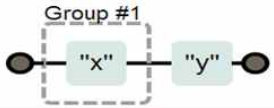
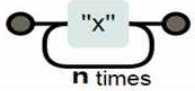
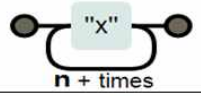
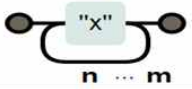


표현 정규식

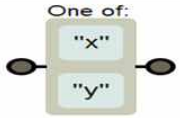
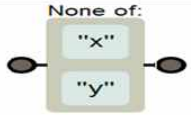
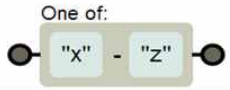
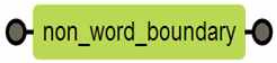
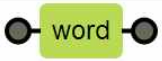
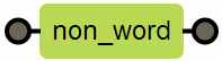
/(?=.*[a-zA-Z]{2,50}).{8,50}$/



표현 정규식

정규표현식	표현	설명
^x		문자열이 x로 시작합니다.
x$		문자열이 x로 끝납니다.
.x		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
x+		x가 1번이상 반복합니다.
x?		x가 존재하거나 존재하지 않습니다.
x*		x가 0번이상 반복합니다.
x|y		x 또는 y를 찾습니다. (or연산자를 의미합니다.)
(x)		( )안의 내용을 캡쳐하며, 그룹화 합니다.
(x)(y)		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡쳐합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
(x)(?:y)		캡쳐하지 않는 그룹을 생성할 경우 ?:를 사용합니다. 결과값 배열에 캡쳐하지 않는 그룹은 들어가지 않습니다.
x{n}		x를 n번 반복한 문자를 찾습니다.
x{n,}		x를 n번이상 반복한 문자를 찾습니다.
x{n,m}		x를 n번이상 m번이하 반복한 문자를 찾습니다.

표현 정규식

정규표현식	표현	설명
[xy]		x,y중 하나를 찾습니다.
[^xy]		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
[x-z]		x~z 사이의 문자중 하나를 찾습니다.
₩^		^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
₩b		문자와 공백사이의 문자를 찾습니다.
₩B		문자와 공백사이가 아닌 값을 찾습니다.
₩d		숫자를 찾습니다.
₩D		숫자가 아닌 값을 찾습니다.
₩s		공백문자를 찾습니다.
₩S		공백이 아닌 문자를 찾습니다.
₩t		Tab 문자를 찾습니다.
₩v		Vertical Tab 문자를 찾습니다.
₩w		알파벳 + 숫자 + _ 를 찾습니다.
₩W		알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

표현 정규식

```
[A-Za-z0-9_]{1,}
@
[A-Za-z0-9-]{1,}(₩.[A-Za-z0-9-]{1,})*₩
.
[A-Za-z]{2,6}
```

표현 정규식

```
₩b₩w+
@
[₩dA-Za-z-]+(₩.[₩dA-Za-z-]+)*
₩.
[A-Za-z]{2,6}₩b
```

표현 정규식

remaper.ku@gmail.com

이런 이메일 주소가 가능하게 정규화를 다시 해보자

표현 정규식

1) 숫자만 : ^[0-9]*$

2) 영문자만 : ^[a-zA-Z]*$

3) 한글만 : ^[가-힣]*$

4) 영어 & 숫자만 : ^[a-zA-Z0-9]*$

5) E-Mail : ^[a-zA-Z0-9]+@[a-zA-Z0-9]+$

6) 휴대폰 : ^01(?:0|1|[6-9]) - (?:₩d{3}|₩d{4}) - ₩d{4}$

7) 일반전화 : ^₩d{2,3} - ₩d{3,4} - ₩d{4}$

8) 주민등록번호 : ₩d{6} ₩- [1-4]₩d{6}

9) IP 주소 : ([0-9]{1,3}) ₩. ([0-9]{1,3}) ₩. ([0-9]{1,3}) ₩. ([0-9]{1,3})

표현 정규식

```
import requests
import re
#regular expression(정규표현식)을 사용하기 위한 라이브러리

url = ''

res = requests.get(url)
html = res.text

pattern = r'정규표현식'
find_url = re.findall(pattern, html)
```

표현 정규식

re 라이브러리 주요 함수

Method	Contents
match()	문자열 처음부터 정규식과 매치되는 것을 확인
search()	문자열 전체를 검색하여 정규식과 매치 확인
findall()	정규식과 매치되는 모든 문자열을 리스트로 반환
finditer()	정규식과 매치되는 모든 문자열을 반복 가능한 객체로 반환

re 라이브러리 메타문자

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

표현 정규식

re 라이브러리 메타문자

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

예를 들어 일반 문자인 a는 문자 'a'에 매칭하지만, 여는 소괄호 (는 문자 '('와 매칭하지 않는다.

그럼 찾고자 하는 문자열에 소괄호가 있으면 어떻게 하나?

위의 문자들의 앞에 백슬래시 \를 붙여 주면 일반 문자처럼 한 글자에 매칭된다. 예를 들어 \(는 문자 '('와 매칭된다.

표현 정규식

```
import re

#match는 시작 문자를 비교해 준다.
matchObj = re.match('a', 'a')
print(matchObj)

print(re.match('a', 'aba'))
print(re.match('a', 'bbb'))
print(re.match('a', 'baa'))
```

표현 정규식

```
import re

# search는 문자를 비교해준다.
matchObj = re.search('a', 'a')
print(matchObj)

print(re.search('a', 'aba'))
print(re.search('a', 'bbb'))
print(re.search('a', 'baa'))
```

표현 정규식

```
import re

#findall은 문자를 비교해서 리스트로 반환해 준다
matchObj = re.findall('a', 'a')
print(matchObj)

print(re.findall('a', 'aba'))
print(re.findall('a', 'bbb'))
print(re.findall('a', 'baa'))
print(re.findall('aaa', 'aaaa'))
```

표현 정규식

Method	Contents
group()	일치된 문자열을 반환한다.
start()	일치된 문자열의 시작 위치를 반환한다.
end()	일치된 문자열의 끝 위치를 반환한다.
span()	일치된 문자열의 (시작 위치, 끝 위치) 튜플을 반환한다.

표현 정규식

syntax	long syntax	inline flag	meaning
re.I	re.IGNORECASE	(?i)	대소문자 구분 없이 일치
re.M	re.MULTILINE	(?m)	^와 $는 개행문자 위치에서 일치
re.S	re.DOTALL	(?s)	마침표는 개행문자와 일치
re.A	re.ASCII	(?a)	{₩w, ₩W, ₩b, ₩B}는 ascii에만 일치
re.U	re.UNICODE	(?u)	{₩w, ₩W, ₩b, ₩B}는 Unicode에 일치
re.L	re.LOCALE	(?L)	{₩w, ₩W, ₩b, ₩B}는 locale dependent
re.X	re.VERBOSE	(?x)	정규표현식에 주석을 달 수 있음

표현 정규식

```
import re

html = "<html>…</html>"
body = re.search('<body.*/body>', html, re.I | re.S)
if (body is None):
    print("No <body> in html")
    exit()

body = body.group()

body = re.sub('<script.*?>.*?</script>', '', body, 0, re.I | re.S)
text = re.sub('<.+?>', '', body, 0, re.I | re.S)
nospace = re.sub(' | |\t|\r|\n', '', text)
print(nospace)
```

표현 정규식

```
body = re.sub('(<([^>]+)>)', '', body, 0, re.I | re.S)
```

다양한 정규식 예제

"읽지 못하면 쓸 수 없고 쓸 수 없으면
생각할 수 없다"

Part 2

JSON으로 저장하기

JSON으로 저장하기

JSON (JavaScript Object Notation)

- JavaScript Object Notation라는 의미의 축약어로 데이터를 저장하거나 전송할 때 많이 사용되는 경량의 DATA 교환 형식
- Javascript에서 객체를 만들 때 사용하는 표현식을 의미한다.
- JSON 표현식은 사람과 기계 모두 이해하기 쉬우며 용량이 작아서, 최근에는 JSON이 XML을 대체해서 데이터 전송 등에 많이 사용한다.
- JSON은 데이터 포맷일 뿐이며 어떠한 통신 방법도, 프로그래밍 문법도 아닌 단순히 데이터를 표시하는 표현 방법일 뿐이다.

JSON으로 저장하기

JSON (JavaScript Object Notation)

- 서버와 클라이언트 간의 교류에서 일반적으로 많이 사용된다.
- 자바스크립트 객체 표기법과 아주 유사하다.
- 자바스크립트를 이용하여 JSON 형식의 문서를 쉽게 자바스크립트 객체로 변환할 수 있는 이점이 있다.
- JSON 문서 형식은 자바스크립트 객체의 형식을 기반으로 만들어졌다.
- 자바스크립트의 문법과 굉장히 유사하지만 텍스트 형식일 뿐이다.
- 다른 프로그래밍 언어를 이용해서도 쉽게 만들 수 있다.
- 특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서 JSON 포맷의 데이터를 핸들링할 수 있는 라이브러리를 제공한다.

JSON으로 저장하기

JSON (JavaScript Object Notation)

- 데이터를 나타낼 수 있는 방식은 여러가지가 있지만, 대표적인 것이 XML이고, 이후 가장 많이 사용되는 것이 아마도 JSON일 것이다.

XML
- 데이터 값 양쪽으로 태그가 있다.
- (HTML을 근본으로 했기에 태그라는 것이 없을 수가 없는데, 그 태그를 줄인다 해도 최소한 표현하려면 양쪽에 몇글자씩이 있어야 한다.)

JSON
- 태그로 표현하기 보다는 중괄호({}) 같은 형식으로 하고, 값을 ','로 나열하기에 그 표현이 간단하다.

JSON으로 저장하기

```
{
    "employees": [
        {
            "name": "Surim",
            "lastName": "Son"
        },
        {
            "name": "Someone",
            "lastName": "Huh"
        },
        {
            "name": "Someone else",
            "lastName": "Kim"
        }
    ]
}
```

- JSON 형식은 자바스크립트 객체와 마찬가지로 key / value가 존재할 수 있으며 key값이나 문자열은 항상 쌍따옴표를 이용하여 표기해야한다.
- 객체, 배열 등의 표기를 사용할 수 있다.
- 일반 자바스크립트의 객체처럼 원하는 만큼 중첩 시켜서 사용할 수도 있다.
- JSON형식에서는 null, number, string, array, object, boolean을 사용할 수 있다.

JSON으로 저장하기


```
{
  "1.FirstName": "kim",
  "2.LastName": "ki-hyun",
  "3.Age": 99,
  "4.University": "korea University",
  "5.Courses": [
    {
      "Classes": [
        "Probability",
        "Generalized Linear Model",
        "Categorical Data Analysis"
      ],
      "Major": "Statistics"
    },
    {
      "Classes": ["Data Structure", "Programming", "Algorithms"],
      "Minor": "ComputerScience"
    }
  ]
}
```

JSON으로 저장하기

```
import json

json_file = open("json/student_file.json", "r")
student_data = json.load(json_file)
st_json = json.dumps(student_data, indent=4)

print(st_json)

print(type(student_data))
print(type(st_json))
print(student_data["3.Age"])
student_data["3.Age"] = 37
print(student_data["3.Age"])
print(student_data["5.Courses"][0]["Classes"])
```

Part 3

코드 리뷰

코드 리뷰

날씨 데이터를 가져와 보자

주가 데이터를 가져와 보자