

데이터 과학을 위한
파이썬
프로그래밍



15. XML과 JSON

목차

1. XML의 이해
2. Lab: XML 파싱
 - 연습 문제 2번
3. JSON의 이해
4. Lab: JSON 데이터 분석
 - 연습 문제 1번
5. 연습문제

01

XML의 이해

01. XML의 이해: XML의 개념

■ XML의 개념

- XML(eXtensible Markup Language)은 확장적인 마크업(markup) 언어라는 뜻으로, 데이터의 구조와 의미를 설명하는 태그를 사용하여 어떤 데이터의 속성과 값을 표현하는 언어이다.
- 즉, 시작 태그와 종료 태그 사이에 어떤 값이 있고, 그 값은 그 태그의 이름으로 만들어진 어떤 데이터의 속성과 값이다.

01. XML의 이해: XML 표현하기

■ XML 표현하기

- XML의 구조는 다음과 같이 간단하다

```
<?xml version="1.0"?>
<학생>
  <이름>한재일</이름>
  <학번>20105503</학번>
  <나이>26</나이>
  <학과>산업경영공학과</학과>
  <성별>남성</성별>
</학생>
```

01. XML의 이해

여기서 잠깐! 현재 XML 상황

- 현재 XML은 10년 전보다 그 중요성이 매우 떨어졌다. 가장 큰 이유는 XML이 데이터베이스보다 좀 더 자유롭게 데이터를 저장하기 위해 만들어졌는데, 이 기능이 최근 많이 사용하는 JSON보다 훨씬 무겁기 때문이다.
- 그래도 여전히 기존의 오래된 시스템, 흔히 레거시(legacy)라고 하는 시스템에서는 기기 간 또는 컴퓨터와 스마트폰 등의 다른 기기 간의 정보를 XML로 주고받고 있다.

xml과 html의 차이점과 유사점

1) XML은 데이터 저장과 전송을 목적으로 만들어진 언어.

태그 사용. XML은 미리 정의된 태그가 없다.

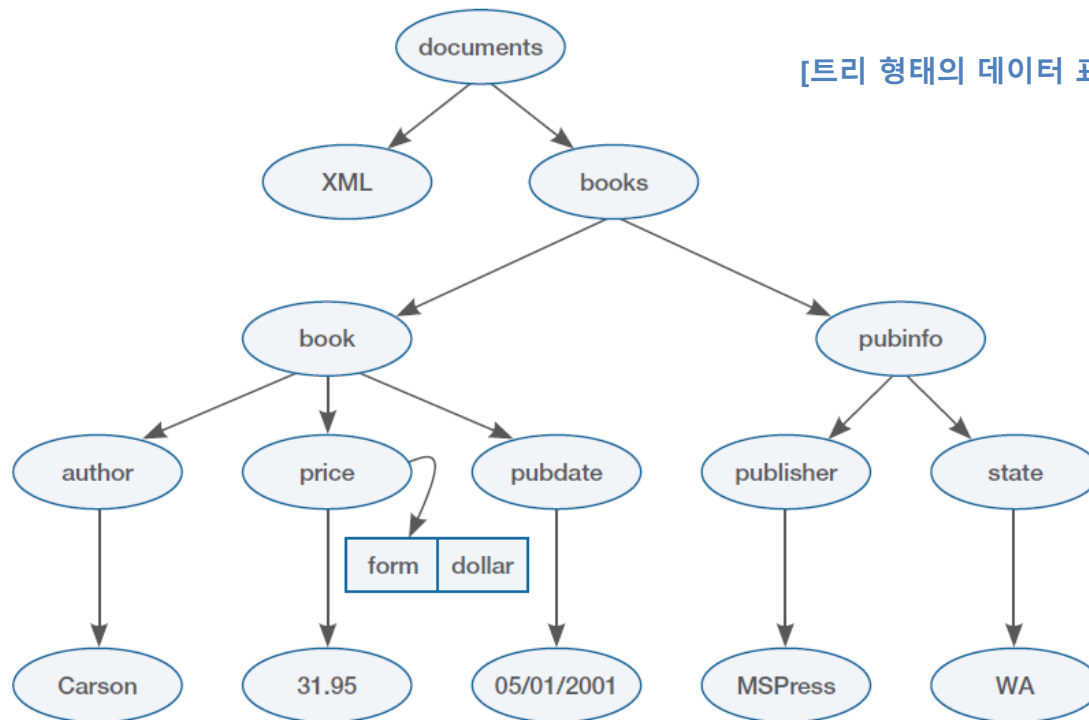
2) HTML은 data를 웹 상에 표현하기 위한 언어.

태그 사용. 미리 정의된 태그가 있다..

01. XML의 이해

■ XML 문서

- XML로 정보를 표현할 때 가장 기본적인 방법은 트리 형태로 표현하는 것이다. 이는 HTML과 완전히 같으며 모든 태그 기반의 언어가 지닌 공통적인 특징이다.



[트리 형태의 데이터 표현(XML)]

01. XML의 이해

- XML 문서 • 그림의 구조적인 정보를 XML로 나타내면 다음과 같다.

```
<?xml version="1.0"?>
```

```
<books>
```

```
<book>
```

```
<author>Carson</author>
```

```
<price format="dollar">31.95</price>
```

```
<pubdate>05/01/2001</pubdate>
```

```
</book>
```

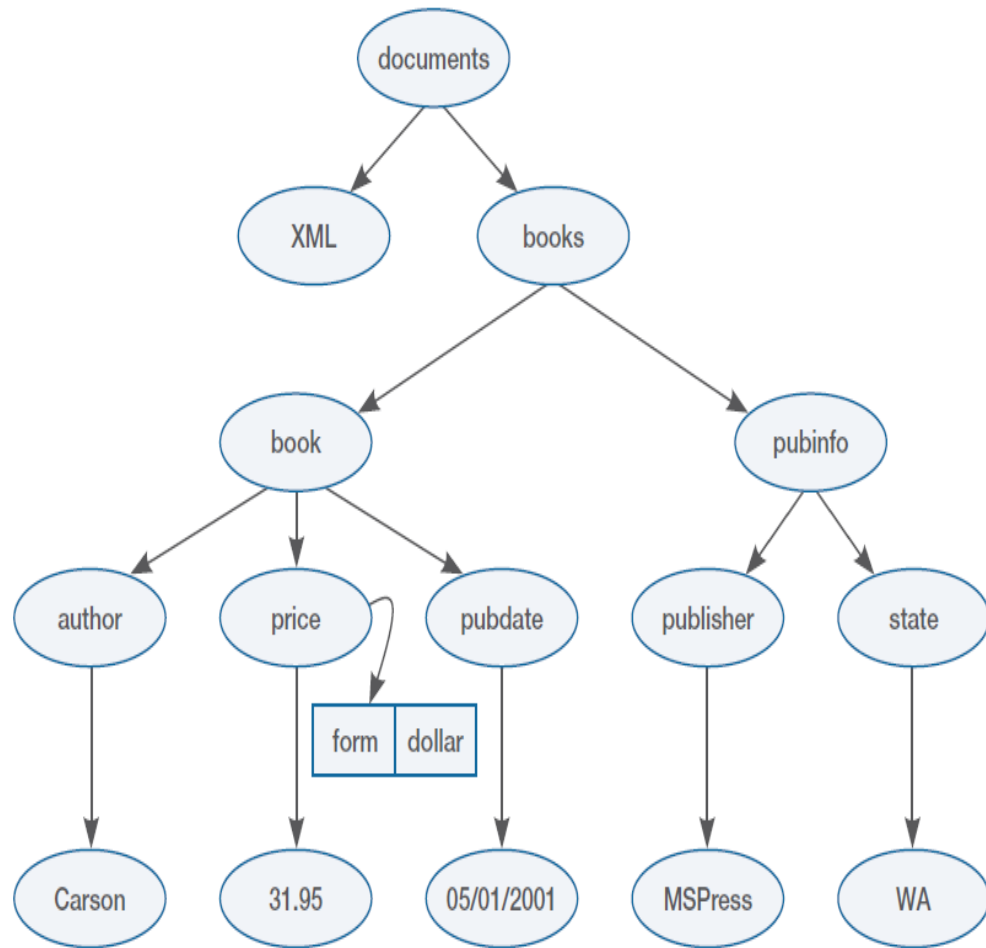
```
<pubinfo>
```

```
<publisher>MSPress</publisher>
```

```
<state>WA</state>
```

```
</pubinfo>
```

```
</books>
```



01. XML의 이해

■ XML 문서

- 간단히 딕셔너리로 생각하면 다음과 같은 방식으로 표현할 수 있다.

```
{books : [{book : {author: carson, price: 31.95, pubdate: 05/01/2001} ] }]}
```

02

Lab: XML 파싱

02. Lab: XML 파싱: BeautifulSoup 모듈 개요

■ BeautifulSoup 모듈 개요

- BeautifulSoup 모듈은 일종의 래퍼(wrapper)로, 기존 파싱 기능이 있는 다른 라이브러리를 쉽게 사용할 수 있도록 한다. 전통적인 파이썬 XML 파서(XML parser)에는 lxml과 html5lib 등이 있으며, BeautifulSoup 모듈은 이를 차용하여 데이터를 쉽고 빠르게 처리한다.

[파서의
성능 비교]

파서	파이썬 2.7		파이썬 3.2	
	속도(KB/s)	성공률(%)	속도(KB/s)	성공률(%)
BeautifulSoup 3.2(SGMLParser)	221	100	—	—
html5lib(BS3 treebuilder)	253	99	—	—
BeautifulSoup 4.0 + lxml	255	100	2140	96
html5lib(lxml treebuilder)	270	99	—	—
BeautifulSoup 4.0 + html5lib	271	98	—	—
BeautifulSoup 4.0 + HTMLParser	299	59	1705	57
html5lib(simpлетree treebuilder)	332	100	—	—
HTMLParser	5194	52	3918	57
lxml	179252	100	14258	96

참고: Parser Library

Parser	Typical usage	Advantages	Disadvantages
Python's <code>html.parser</code>	<code>BeautifulSoup(markup, "html.parser")</code>	<ul style="list-style-type: none">• Batteries included• Decent speed• Lenient (As of Python 2.7.3 and 3.2.)	<ul style="list-style-type: none">• Not as fast as <code>lxml</code>, less lenient than <code>html5lib</code>.
<code>lxml</code> 's HTML parser	<code>BeautifulSoup(markup, "lxml")</code>	<ul style="list-style-type: none">• Very fast• Lenient	<ul style="list-style-type: none">• External dependency C
<code>lxml</code> 's XML parser	<code>BeautifulSoup(markup, "lxml-xml")</code> <code>BeautifulSoup(markup, "xml")</code>	<ul style="list-style-type: none">• Very fast• The only currently supported XML parser	<ul style="list-style-type: none">• External dependency C
<code>html5lib</code>	<code>BeautifulSoup(markup, "html5lib")</code>	<ul style="list-style-type: none">• Extremely lenient• Parses pages the same way a web browser does• Creates valid HTML5	<ul style="list-style-type: none">• Very slow• External Python dependency

02. Lab: XML 파싱: BeautifulSoup 모듈 설치

■ BeautifulSoup 모듈 설치 [링크](#)

1) BeautifulSoup4가 설치되어 있는지 확인한다.

```
pip show beautifulsoup4
```

2) Parser library, lxml은 별도로 설치해야 한다.

```
pip install lxml    # 미리 설치 추천.
```

3) BeautifulSoup4를 설치한다.

```
pip install beautifulsoup4
```

3) 파이썬 수행 시 임포트하여 사용하고자 한다.

```
from bs4 import BeautifulSoup
```

[참조: 웹 크롤링에 사용하는
Beautiful Soup\(뷰티플 수프\) 사용
법과 예제\(1\) 예제 \(2\)](#)

02. Lab: XML 파싱

■ BeautifulSoup 모듈 사용법

[BeautifulSoup 모듈의 주요 코드]

목적	코드	설명
객체 생성	<code>soup = BeautifulSoup(books_xml, "lxml")</code>	xml 문서를 분석하는 새로운 객체를 생성, books_xml은 문자열형, lxml은 파서의 이름
태그 검색	<code>soup.find_all("author")</code>	필요한 태그를 검색하여 여러 개를 반환하는 함수, 여기서는 'author'라는 이름의 태그를 검색하여 반환

02. Lab: XML 파싱

==> z_beautifulSoup1.py

■ BeautifulSoup 모듈 사용법

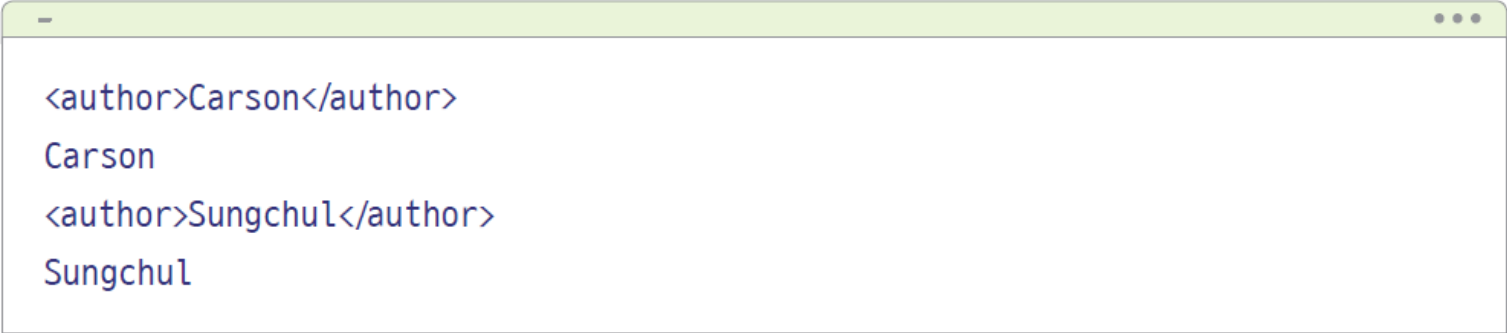
- BeautifulSoup 모듈을 사용하는 코드를 만들기 위해 소스 파일에서 'books.xml'을 작업 폴더에 가져오고, [코드 15-1]을 작성해 보자.

코드 15-1 beautifulsoup1.py

```
1 from bs4 import BeautifulSoup
2
3 with open("books.xml", "r", encoding = "utf8") as books_file:
4     books_xml = books_file.read()           # 파일을 문자열로 읽어 오기
5
6 soup = BeautifulSoup(books_xml, "lxml")    # lxml 파서를 사용해 데이터 분석
7
8 # author가 들어간 모든 요소의 값 추출
9 for book_info in soup.find_all("author"):
10     print(book_info)
11     print(book_info.get_text())           # 해당 요소에서 값 추출
```

02. Lab: XML 파싱: BeautifulSoup 모듈 사용법

■ BeautifulSoup 모듈 사용법



```
<author>Carson</author>  
Carson  
<author>Sungchul</author>  
Sungchul
```

- ➡ 다운로드한 파일의 내용을 3~4행에서 읽어 와 문자열 파일로 변환한다. 그리고 6행에서 해당 문자열 정보를 받아오면서 lxml을 사용해 파싱한 BeautifulSoup의 객체를 생성한다. 9행의 find_all() 함수를 사용하여 author가 포함된 요소들을 받아 오고, 마지막으로 해당 요소에서 get_text() 함수를 사용해 해당 값의 결과를 출력한다.

xml 파일의 parsing 사례

z_beautifulSoup1.py

```
for info in soup.find_all(Tag):
    print(info, end=' : ')
    print(type(info))
    print(info.get_text(), end=' : ') # 해당 요소에서 값 추출
    print(type(info.get_text()), '\n')
```

추출하고자 하는 요소의 Tag 지정

Tag = "author"

<author>Carson</author> : <class 'bs4.element.Tag'>

Carson : <class 'str'>

<author>Sungchul</author> : <class 'bs4.element.Tag'>

Sungchul : <class 'str'>

추출하고자 하는 요소의 Tag 지정

Tag = 'publisher'

<publisher>MSPress</publisher> : <class 'bs4.element.Tag'>

MSPress : <class 'str'>

<publisher>Gachon</publisher> : <class 'bs4.element.Tag'>

Gachon : <class 'str'>

실행결과

02. Lab: XML 파싱: USPTO XML 데이터

z_beautifulSoup2.py

■ USPTO XML 데이터

- 미국 특허청(USPTO)의 특허 데이터는 XML로 제공된다. 이번 Lab에서는 특정 특허 XML 문서로부터 필요한 정보를 가져온다. 분석할 특허는 등록번호 '08621662'인 'Adjustable shoulder device for hard upper torso suit'이며, 다음 링크에 접속하면 자세히 볼 수 있다.

<https://patents.google.com/patent/US20120260387>

코드 15-2 beautifulsoup2.py

```
1 import urllib.request
2 from bs4 import BeautifulSoup
3
4 with open("US08621662-20140107.XML", "r", encoding="utf8") as patent_xml:
5     xml = patent_xml.read()           # 파일을 문자열로 읽어 오기
6
7 soup = BeautifulSoup(xml, "lxml")    # lxml 파서 호출
8
9 # invention-title 태그 찾기
```

02. Lab: XML 파싱

■ USPTO XML 데이터

```
10 invention_title_tag = soup.find("invention-title")
11 print(invention_title_tag.get_text())
```



Adjustable shoulder device for hard upper torso suit

- 5행에서 파일을 읽어 와 문자열로 변환한다. 7행에서 BeautifulSoup 객체를 생성하고, 10~11행에서 invention-title만 추출할 수 있도록 find() 함수를 사용하여 필요한 정보를 추출한다.

02. Lab: XML 파싱

■ USPTO XML 데이터

- 이 태그들은 각각 publication-reference와 application-reference의 하위 태그이며, 둘 모두 이름이 같다. 이 경우 어떻게 처리해야 할까?

같은 이름의
태그

```
<publication-reference>  
  <document-id>  
    <country>US</country>  
    <doc-number>08621662</doc-number>  
    <kind>B2</kind>  
    <date>20140107</date>  
  </document-id>  
</publication-reference>
```

```
<application-reference appl-type="utility">  
  <document-id>  
    <country>US</country>  
    <doc-number>13175987</doc-number>  
    <date>20110705</date>  
  </document-id>  
</application-reference>
```

02. Lab: XML 파싱

■ USPTO XML 데이터

- 정규 표현식에서 배웠듯이 구조적으로 두 번 접근하는 방식을 사용한다. 다음과 같이 먼저 publication-reference에 접근하여 요소 값을 획득한 후, 해당 태그 정보에서 다시 필요한 정보를 추출할 수 있다. 다음 코드를 [코드 15-2] 뒷부분에 이어 입력하면 문제를 해결할 수 있다

```
publication_reference_tag = soup.find("publication-reference")
p_document_id_tag = publication_reference_tag.find("document-id")
p_country = p_document_id_tag.find("country").get_text()
p_doc_number = p_document_id_tag.find("doc-number").get_text()
p_kind = p_document_id_tag.find("kind").get_text()
p_date = p_document_id_tag.find("date").get_text()

application_reference_tag = soup.find("application-reference")
a_document_id_tag = application_reference_tag.find("document-id")
a_country = a_document_id_tag.find("country").get_text()
a_doc_number = a_document_id_tag.find("doc-number").get_text()
a_date = a_document_id_tag.find("date").get_text()
```

BeautifulSoup로 선언된 객체를 통해 수행 가능한 여러가지 method와 멤버 변수에 대해 익힌다.

BeautifulSoup methods

`find_all(tag)`

지정하는 태그(tag)를 검색해서 여러 개의 태그를 반환하는 함수

`get_text()`

해당 태그 사이에 있는 값을 반환한다.

html 문서의 tag해석

`<a ...> ... ` => 해석 Tag a

`` ==> xx는 Tag a의 attribute, attribute의 값은 yy

A tag may have any number of attributes.

The tag `<b id="boldest">` has an attribute “id” whose value is “boldest”.

You can access a tag’s attributes by treating the tag like a dictionary

`print(soup.b[id])` => 수행 결과 => boldest

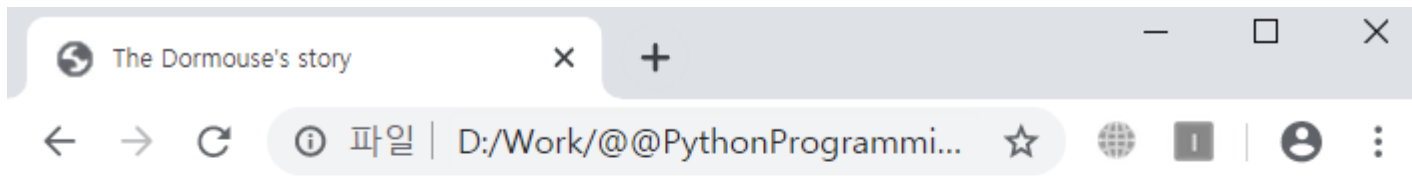
HTML parsing 메소드 링크

z_beautifulSoup3.py

실험용 파일, index.html 파일의 표현 - 파이참과 크롬에서 보았을 경우..

```
1 <html><head><title>The Dormouse's story</title></head>
2 <body>
3   <p class="title"><b>The Dormouse's story</b></p>
4
5   <p class="story">Once upon a time there were three little sisters; and their names were
6   <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
7   <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
8   <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
9   and they lived at the bottom of a well.</p>
10
11  <p class="story">...</p>
```

Index.html in PyCharm



The Dormouse's story

Once upon a time there were three little sisters; and their names were [Elsie](#), [Lacie](#) and [Tillie](#); and they lived at the bottom of a well.

Index.html in Chrome

HTML parsing 메소드 링크

z_beautifulSoup3.py

문자열을 소스 내에서 정의하여 사용하던지 아니면 파일에서 읽어온다.
결과는 같다.

```
from bs4 import BeautifulSoup
html_doc = """
<html><head><title>The Dormouse's story</title></head>
```

소스프로그램

```
<p class="title"><b>The Dormouse's story</b></p>
```

```
<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
```

```
<p class="story">...</p>
```

```
"""
```

```
# 파일에서 읽고자 하면 아래의 주석처리를 해제하시오.
```

```
#with open('index.html', "r", encoding="utf8") as fp:
```

```
# html_doc = fp.read() # 파일을 문자열로 읽어 오기
```

```
soup = BeautifulSoup(html_doc, "lxml")
```

```
print(soup.prettify())
```

← Parser

HTML parsing 메소드

링크

z_beautifulSoup3.py

```
<html>
<head>
  <title>
    The Dormouse's story
  </title>
</head>
<body>
  <p class="title">
    <b>
      The Dormouse's story
    </b>
  </p>
  <p class="story">
    Once upon a time there were three little sisters; and their names were
    <a class="sister" href="http://example.com/elsie" id="link1">
      Elsie
    </a>
    ,
    <a class="sister" href="http://example.com/lacie" id="link2">
      Lacie
    </a>
    and
    <a class="sister" href="http://example.com/tillie" id="link3">
      Tillie
    </a>
    ;
    and they lived at the bottom of a well.
  </p>
  <p class="story">
    ...
  </p>
</body>
</html>
```

print(soup.prettify())를 수행하면 정렬하여 화면에 출력한다.

03

JSON의 이해

03. JSON의 이해: JSON의 개념

■ JSON의 개념

```
{
  "dataTitle":"JSON Tutorial!",
  "swiftVersion":2.1

  "users":[
    {
      "name":"John",
      "age":25
    },
    {
      "name":"Mark",
      "age":29
    },
    {
      "name":"Sarah",
      "age":22
    }
  ],
}
```

- JSON은 XML보다 데이터 용량이 적고 코드로의 전환이 쉽다는 측면에서 XML의 대체재로 가장 많이 활용되고 있다.
- JSON은 파이썬의 딕셔너리형과 매우 비슷하여, 키-값의 쌍으로 구성되어 있다.
- Key='users', value=리스트형이 있고, 그 안에 'name'과 'age'라는 2개의 키가 또 하나의 딕셔너리형으로 있는 것을 확인할 수 있다.

03. JSON의 이해: JSON과 XML

[XML와 JSON에 관하여](#)

■ JSON과 XML

- XML과 비교할 때 JSON의 장점은 일단 코드가 간결하고, 코드의 전환이 쉽다는 점이다. 그리고 코드의 간결함 때문에 용량의 절약이라는 가장 큰 장점이 있다.

Json

```
{
  "sibling": [
    {"firstName": "Anna", "lastName": "Clayton"},
    {"lastName": "Alex", "lastName": "Clayton"}
  ]
}
```

XML

```
<siblings>
  <sibling>
    <firstName>Anna</firstName>
    <lastName>Clayton</lastName>
  </sibling>
  <sibling>
    <firstName>Alex</firstName>
    <lastName>Clayton</lastName>
  </sibling>
</siblings>
```

04

Lab: JSON 데이터 분석

04. Lab: JSON 데이터 분석

- 파이썬에서 JSON을 사용하기 위해서는 json 모듈을 이용한다. JSON 데이터 포맷은 데이터 저장 및 읽기가 딕셔너리형과 완벽히 상호 호환되어, 딕셔너리형에 익숙한 사용자가 매우 쉽게 사용할 수 있다는 장점이 있다.

04. Lab: JSON 데이터 분석: JSON 읽기

■ JSON 읽기

- JSON을 읽기 위해서는 JSON 파일의 구조를 확인한 후, json 모듈로 읽고 딕셔너리형처럼 처리한다.

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

04. Lab: JSON 데이터 분석

■ JSON 읽기

- 앞 데이터에는 'employees' 아래에 3개의 데이터가 있다. 이를 파이썬으로 읽어 오기 위해 [코드 15-3]과 같이 입력한다.

코드 15-3 json1.py

```
1 import json
2
3 with open("json_example.json", "r", encoding="utf8") as f:
4     contents = f.read()                # 파일 내용 읽어 오기
5     json_data = json.loads(contents)    # json 파싱
6     print(json_data["employees"])      # 딕셔너리처럼 사용하기
```

<class 'dict'>

```
json_example.json - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말

{"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]}
```

```
{ "employees": [
  { "firstName": "John", "lastName": "Doe",
  { "firstName": "Anna", "lastName": "Smith",
  { "firstName": "Peter", "lastName": "Jones"
}]}
```

```
[{'firstName': 'John', 'lastName': 'Doe'}, {'firstName': 'Anna', 'lastName': 'Smith'},
{'firstName': 'Peter', 'lastName': 'Jones'}]
```


04. Lab: JSON 데이터 분석

■ JSON 읽기

- ➔ 먼저 1행에서 json 모듈을 호출하고, 3행에서 open() 함수를 사용하여 파일 내용을 가져온다. 그리고 4행에서 문자열형으로 변환하여 처리한다. 5행에서는 loads() 함수를 사용하여 해당 문자열형을 딕셔너리형처럼 변환한다. 6행에서 딕셔너리처럼 json_data["employees"]를 print() 함수로 출력하면 결과값이 출력된다.

04. Lab: JSON 데이터 분석: JSON 쓰기

■ JSON 쓰기

- 딕셔너리형으로 구성된 데이터를 json 형태의 파일로 변환하는 과정에 대해 알아보자.

코드 15-4 json2.py

```
1 import json
2
3 dict_data = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}           # 딕셔너리 생성
4
5 with open("data.json", "w") as f:
6     json.dump(dict_data, f)
```

생성된 파일: data.json

```
{ "Name": "Zara", "Age": 7, "Class": "First" }
```

- ➡ json을 쓰기 위해서는 먼저 3행처럼 데이터를 저장한 딕셔너리형을 생성하고, 6행에서 json.dump() 함수를 사용하여 데이터를 저장한다. 이때 인수는 딕셔너리형과 파일 객체가 차례대로 들어가면 완성할 수 있다. 실행 결과, 작업 폴더에 'data.json' 파일이 생성된 것을 확인할 수 있다.

JSON formatter [링크](#)

JSON Editor

```
1 { "employees": [  
2   { "firstName": "John", "lastName": "Doe"},  
3   { "firstName": "Anna", "lastName": "Smith"},  
4   { "firstName": "Peter", "lastName": "Jones"}  
5 ] }
```

Format JSON

```
1 {  
2   "employees": [  
3     {  
4       "firstName": "John",  
5       "lastName": "Doe"  
6     },  
7     {  
8       "firstName": "Anna",  
9       "lastName": "Smith"  
10    },  
11    {  
12      "firstName": "Peter",  
13      "lastName": "Jones"  
14    }  
15  ]  
16 }
```

04. Lab: JSON 데이터 분석(이후 생략...)

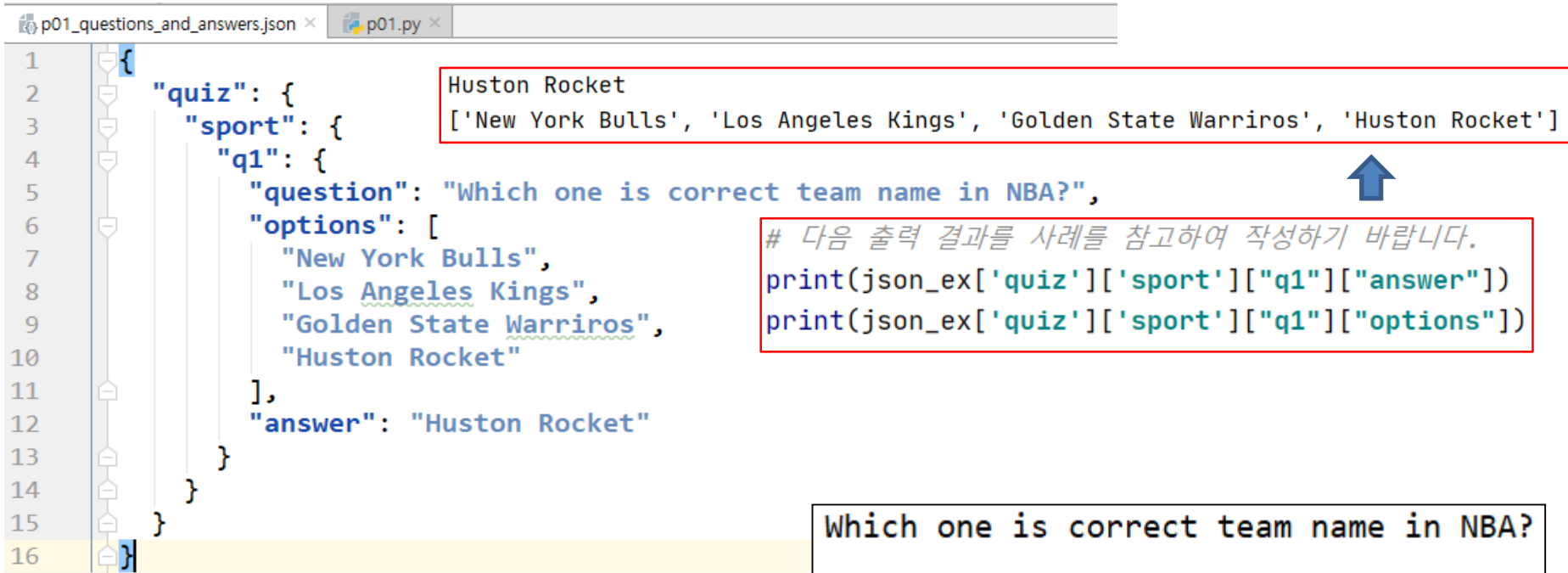
■ 트위터 데이터 읽어오기

- API(Application Programming Interface)는 일종의 함수로, 해당 회사가 제공하는 서비스를 활용하기 위한 함수를 뜻한다. 일반적으로 API 서비스는 JSON으로 데이터를 주고받기 때문에 JSON 데이터를 연습하기에 매우 좋은 환경이다.
- 이번 Lab에서는 트위터에서 제공하는 Developer API를 가지고 트위터 데이터를 수집하는 방법에 대해 설명하겠다.
- 실험을 위해서는 트위터에 계정이 있어야 가능하다....

<https://apps.twitter.com>

연습문제 1

'p01_questions_and_answers.json' 파일을 이용해 다음과 같은 결과를 출력하고자 한다.
소스 파일 'p01.py'의 빈칸에 알맞은 코드를 쓰시오.



```
p01_questions_and_answers.json x p01.py x
```

```
1 {
2   "quiz": {
3     "sport": {
4       "q1": {
5         "question": "Which one is correct team name in NBA?",
6         "options": [
7           "New York Bulls",
8           "Los Angeles Kings",
9           "Golden State Warriros",
10          "Huston Rocket"
11        ],
12        "answer": "Huston Rocket"
13      }
14    }
15  }
16 }
```

Huston Rocket
['New York Bulls', 'Los Angeles Kings', 'Golden State Warriros', 'Huston Rocket']

다음 출력 결과를 사례를 참고하여 작성하기 바랍니다.
print(json_ex['quiz']['sport']['q1']['answer'])
print(json_ex['quiz']['sport']['q1']['options'])

Which one is correct team name in NBA?

0. New York Bulls
1. Los Angeles Kings
2. Golden State Warriros
3. Huston Rocket

answer Huston Rocket

PyCharm 편집창으로 관찰한 json 파일

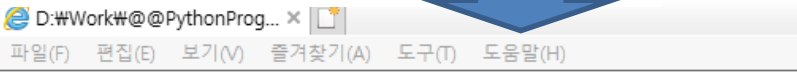
p01.py를 수행한 결과 화면

연습문제 2: p02_researchers.xml

Explorer에서 관찰

확장

```
<?xml version="1.0" encoding="UTF-8"?>
- <researchers>
+ <researcher researcherID="0001">
+ <researcher researcherID="0002">
+ <researcher researcherID="0003">
</researchers>
```



```
<?xml version="1.0" encoding="UTF-8"?>
- <researchers>
- <researcher researcherID="0001">
  <na>Tedd</na>
  <position>Professor</position>
  <degree>Doctor of Engineering</degree>
  <salary format="dollar">70000</salary>
  <day_off>Y</day_off>
</researcher>
- <researcher researcherID="0002">
  <na>Henry</na>
  <position>Senior.researcher</position>
  <degree>Master of Engineering</degree>
  <salary format="dollar">7000</salary>
  <day_off>N</day_off>
</researcher>
- <researcher researcherID="0003">
  <na>John</na>
  <position>Junior.researcher</position>
  <degree>Bachelor of Engineering</degree>
  <salary format="dollar">700</salary>
  <day_off>Y</day_off>
</researcher>
</researchers>
```

Chrome에서 관찰

확장

```
<researchers>
▶ <researcher researcherID="0001">...</researcher>
▶ <researcher researcherID="0002">...</researcher>
▶ <researcher researcherID="0003">...</researcher>
</researchers>
```



```
<researchers>
  <researcher researcherID="0001">
    <na>Tedd</na>
    <position>Professor</position>
    <degree>Doctor of Engineering</degree>
    <salary format="dollar">70000</salary>
    <day_off>Y</day_off>
  </researcher>
  <researcher researcherID="0002">
    <na>Henry</na>
    <position>Senior.researcher</position>
    <degree>Master of Engineering</degree>
    <salary format="dollar">7000</salary>
    <day_off>N</day_off>
  </researcher>
  <researcher researcherID="0003">
    <na>John</na>
    <position>Junior.researcher</position>
    <degree>Bachelor of Engineering</degree>
    <salary format="dollar">700</salary>
    <day_off>Y</day_off>
  </researcher>
</researchers>
```

연습문제 2

- 'p02_researcher.xml' 파일
- 1) 아래 코드로 작성한 수행 결과 값을 예측하여 밝히시오.

```
from bs4 import BeautifulSoup
with open("p02_researcher.xml", "r", encoding="utf-8") as researcher_xml:
    xml = researcher_xml.read()

soup = BeautifulSoup(xml, "html.parser")
for i, element in enumerate(soup.findAll('researcher')):
    if i % 2 == 1:
        print(element['researcherid'])
        print(element.na.get_text())
        print(element.salary)
    else:
        pass
```

연습문제 2

- 'p02_researcher.xml' 파일
- 2) 다음의 결과를 출력하는 프로그램을 작성하시오.

```
0001: Tedd, Professor, Doctor of Engineering 70000 dollar
0002: Henry, Senior.researcher, Master of Engineering 7000 dollar
0003: John, Junior.researcher, Bachelor of Engineering 700 dollar
```


연습문제 3,4,5

3. XML과 JSON 파일 형식의 차이에 대해 서술하시오.

- XML: eXtensible Markup Language의 줄임말로, HTML과 동일한 구조인 XML을 사용하여 데이터를 표현한다. 태그를 사용하기 때문에 데이터의 표현이 매우 길어진다. 전통적으로 2000년대 레거시(legacy) 시스템에서 매우 많이 사용되는 방식이다.
- JSON: JavaScript Object Notation의 줄임말로, 자바스크립트에서 객체를 표현하는 언어의 기법을 빌려와 데이터를 표현한다. 키:값(key:value) 쌍을 쓰기 때문에 기존 파이썬의 딕셔너리 자료형과 유사한 형식으로 데이터를 처리할 수 있다.

4. XML에 대한 설명 중 틀린 것은? ⑤

- ① eXtensible Markup Language의 줄임말이다.
- ② 2000년대 중반 대부분의 컴퓨터들이 통신을 위해 사용된 언어이다.
- ③ HTML과 같이 태그를 사용하여 계층적인 구조로 데이터를 표현하다.
- ④ 딕셔너리와 유사한 형태로 표현이 가능하다.
- ⑤ 자바스크립트를 기반으로 발전된 언어이다.

5. 이 책에서는 BeautifulSoup을 사용하여 XML을 파싱하였다. 해당 모듈을 사용할 때 사용자 입장에서 얻게 되는 장점을 쓰시오.

마크(Markup) 언어를 손쉽게 스크래핑하게 해주어 사용자의 코드의 양이 줄어들고, 기존 정규 표현식 대비 간단히 태그(tag) 정보를 추출할 수 있음|

연습문제 7, 8

7. JSON에 대한 설명 중 틀린 것은? ③

- ① Java Script Object Notation의 줄임말이다.
- ② 최근 많은 시스템에서 데이터를 주기 위해 사용되고 있다.
- ③ 계층적인 정보는 표현할 수 없다.
- ④ dict 타입과 유사한 형태로 표현할 수 있다.
- ⑤ 자바스크립트를 기반으로 발전된 언어이다.

8. 다음 코드의 목적을 쓰시오.

기존 `dict_data`에 있는 정보를 '`data.json`' 파일에 작성하여 새로운 파일을 생성한다.

```
import json

dict_data = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
with open("data.json", "w") as f:
    json.dump(dict_data, f)
```

Thank You !