Data Collection:
Data Formats and Scrapping

2020.9

References:
- KPC, DSAC(Data Scientist Academy & Certificate) Manual, 2019
- many internet sites

File Handling

- **파일 다루기**
  - File open/close:

    `f = open("file_name", "rwa"); f.close()`

  - File read/write:

    `data=f.read(); line=f.readline(); f.write(data)`

- **With 문과 함께 사용하기**
  - 자동으로 file close 됨.

```
f = open("foo.txt", 'w')
f.write("Life is too short, you need python")
f.close()
```

```
with open("foo.txt", "w") as f:
    f.write("Life is too short, you need python")
```

Folder Handling

- **폴더 관리**
  - import os
  - os.getcwd() # get current working directory
  - os.chdir('my_dir') # change directory
  - os.listdir # show file names
  - os.mkdir('test')
  - os.rename('test','new_one')
  - os.remove('out.csv')
  - os.rmdir('new_one')

Popular Data Formats

- **CSV (Comma Separated Values)**
  - csv.reader() or pandas.read_csv()
  - DataFrame.to_csv()
- **JSON (JavaScript Object Notation)**
  - **Lightweight data exchange format (inspired by JavaScript), text itself**
  - json.dump(fp, obj), json.dumps(obj) # json encoding
  - json.load(fp, obj), json.loads(obj) # json decoding
  - Pandas.io.Json_normalize()
- **HTML (Hyper Text Markup Language)**
  - bs4.BeautifulSoup()
  - lxml.html(), and many more..
- **Plain text messages**
  - Token-based text transform (CountVectorizer(), etc.)
  - Word Embedding

CSV (Comma Separated values)

- Pandas 함수 이용
  - `pd.read_csv('test.csv', nrows=2)` # 상위 2개의 행만
  - `df.to_csv` # csv file 로 출력
- csv 패키지 이용
  - Example:

```
import csv
f = open("./data/coffee.csv")
for row in csv.reader(f):
print(row)
```

JSON(JavaScript Object Notation)

- JSON
  - JavaScript Object Notation)은 속성-값 쌍(attribute–value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽고 쓰기 쉽게 텍스트를 사용하는 개방형 표준 포맷이다.
  - 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.
- Example

```
{
    "이름": "홍길동",
    "나이": 25,
    "성별": "여",
    "주소": "서울특별시 양천구 목동",
    "특기": ["농구", "도술"],
    "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
    "회사": "경기 수원시 팔달구 우만동"
}
```

JSON(JavaScript Object Notation)

- JSON package
  - **JSON Encoding**: Python Object (dict, list, tuple 등) 를 JSON 문자열로 변경 (ex) json.dumps(result)
  - **JSON Decoding**: JSON 문자열 -> Python type (dict, list, tuple, 등)

    (ex) json.loads(obj)
- JSON format normalize
  - pandas.io.json_normalize(): normalize semi-structured JSON data into a flat table (for 문을 사용하지 않고도 JSON 데이터를 손쉽게 DataFrame으로 전환할 수 있음)

HTML(HyperText Markup Language)

- **HTML**
  - Standard Markup language for creating web pages and web applications
  - With CSS(Cascading STYPE sheets) and Javascript, it forms a triad of cornerstone technologies for WWW.
  - HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
  - The World Wide Web Consortium (W3C), has encouraged the use of CSS over explicit presentational HTML since 1997.

❖ **Example**

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

```
<!-- HTML5 -->
<!-- Decribes web page -->

<!-- Brower page title -->

<!-- Visible page content -->
```

HTML

- **Elements**
  - Building blocks of HTML pages
  - (ex) <head></head>: 시작을 알리는 태그, <title> 은 <head> 안에서 제목 부여.
  - <body></body>: 실제 웹검색기 화면에 출력되는 부분
  - <h1></h1> : 제목 텍스트의 크기 (<h1> ~ <h6>)
  -   : paragraphs (단락)
  - Tags such as <img /> and <input /> directly introduce content into the page.
  - A link to Wikipedia! : create a link
- **Attributes** of an element
  - id: document-wide unique identifier for an element
  - class: classifying similar elements
  - style: assign presentational properties
  - title: attach subtextual explanation to an element
  - lang: identifies a natural language to an element

HTML Tags

- **Some of the Basic Tags**
  - Headings: <h1><h2><h3>
  - Paragraphs: <p>
  - Images: <img src="logo.jpg" alt="site_logo" width="100" height="100"> ; alternative text, size (width and height)
  - Links: Here
  - Tables: <table border="1"cellpadding="5" cellspacing="5">
    <tr> <td>One</td><td>Two</td> </tr> ; row and data </table>
  - Divisions: This is a DIV container ; define a section in a document (you can think of it like a container or a building block.)
  - Lists: ; defines unordered list, ordered list, a list item
  - Line breaks:
  - Bold text: …
  - Italic text: …

CSS(Cascading Style Sheets)

- **CSS**
  - CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
  - Enable the separation of presentation and content, including layout, colors, and fonts. It can improve content accessibility, provide more flexibility and control, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file.

- **Three ways to add CSS in HTML**
  - Inline – by using the style attribute in HTML elements

    (ex) <h1 style="color:blue;">This is a Blue Heading</h1>
  - Internal – by using a <style> element in the <head> section

    (ex) next page
  - External – by using an external CSS file (the most common way)

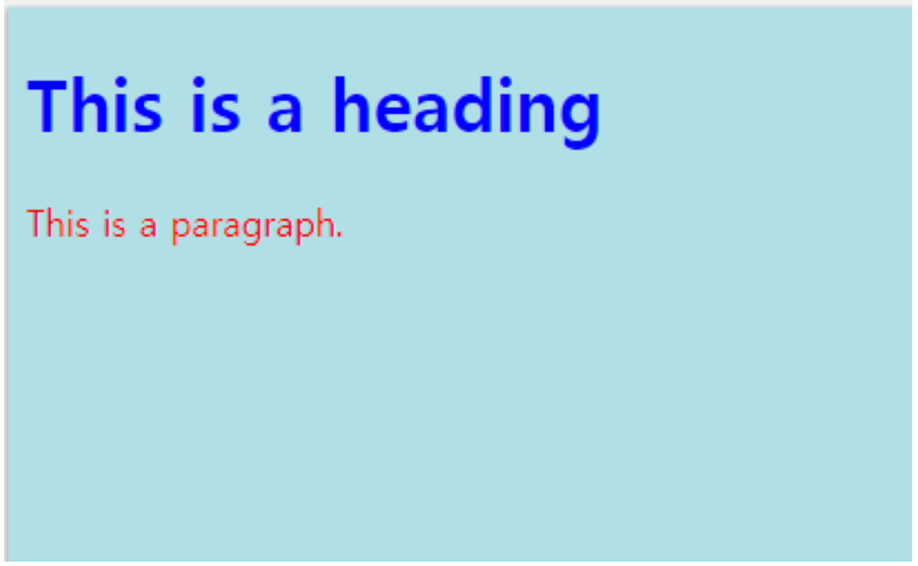
    (ex) next page

Internal CSS


```html
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



external CSS

- An external style sheet is used to define the style for many HTML pages.
- With an external style sheet, you can change the look of an entire web site, by changing one file!
- To use an external style sheet, add a link to it in the <head> section of the HTML page:

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

In the “styles.css” file:

```css
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

Example (HTML only)


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>
        안녕하세요.
    </h1>
    <h2>
        LKT Programmer 입니다.
    </h2>
</body>
</html>
```

Colored by Color Scripter

안녕하세요.

LKT Programmer 입니다.

Example (CSS)


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>

    <style>
        h1 {
            background-color : red;
        }
        h2 {
            background-color : green;
        }
    </style>

</head>
<body>
    <h1>
        안녕하세요.
    </h1>
    <h2>
        LKT Programmer 입니다.
    </h2>
</body>
</html>
```


안녕하세요.

LKT Programmer 입니다.

CSS Tags

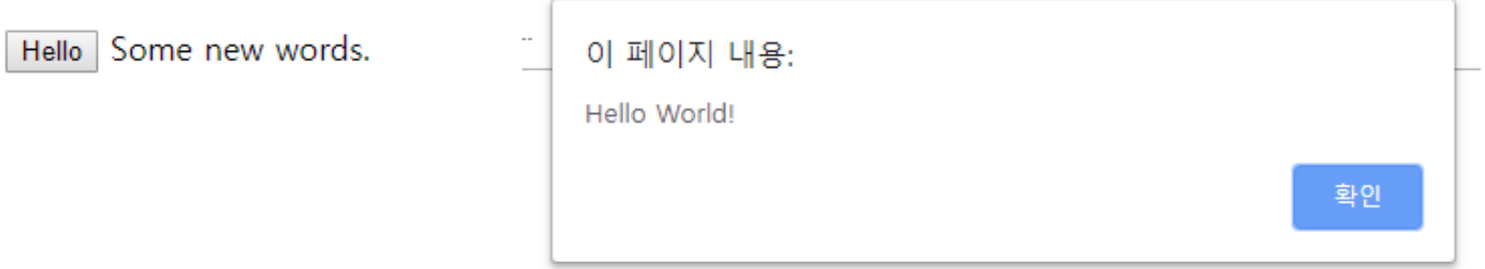
- **Some attributes**
  - CSS Fonts: color, font-family, font-size
  - CSS Border: border
  - CSS Padding: padding ; defines a padding(space) between the text and the border
  - CSS Margin: margin ; defines a margin(space) outside the border
  - id attribute: I am different ; defines a style with the specific id
  - Class attribute: I am different ; a style for the specific class
  - External reference:

<link rel="stylesheet" href=https://www.w3schools.com/html/styles.css>

JavaScript(JS)

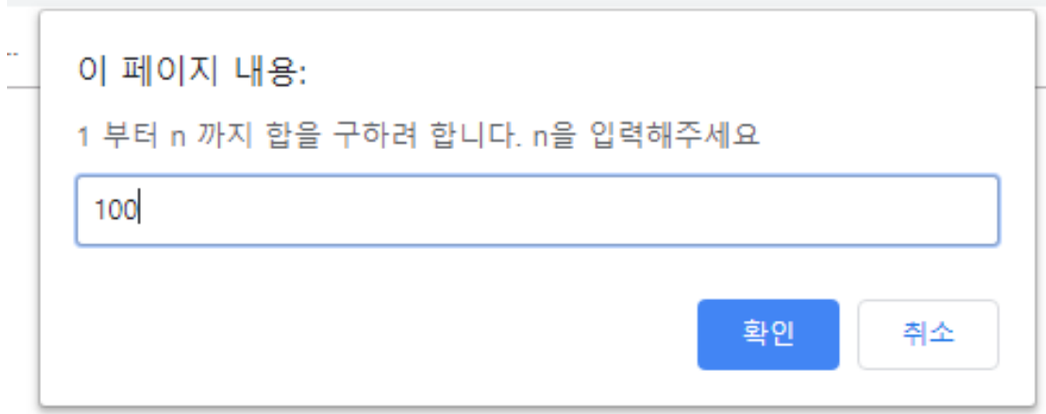
❖ Example of Web page containing JavaScript (HTML 5 syntax) and the DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <button id="hellobutton">Hello</button>
    <script>
        document.getElementById('hellobutton').onclick = function() {
            alert('Hello world!');                      // Show a dialog
            var myTextNode = document.createTextNode('Some new words.');
            document.body.appendChild(myTextNode);      // Append "Some new words" to the page
        };
    </script>
  </body>
</html>
```



JS example(2)


```
<!DOCTYPE html>
<html>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <body>
    <script>
        var sum=0;
        n=parseInt(prompt("1 부터 n 까지 합을 구하려 합니다. n을 입력해주세요",""));
        for(var i=1; i<=n; i++){
        sum+=i;
        }
        document.write("1부터 " + n + "까지의 합은 " + sum + "입니다.");
    </script>
    </body>
</html>
```



1부터 100까지의 합은 5050입니다.

BeautifulSoup parser

- Python 내장 html.parser or lxml's HTML parser
  - `BeautifulSoup(html,"html.parser")` or `BeautifulSoup(html,"lxml")`
  - 일반적으로 lxml 이 빠르고 flexible 하다고 알려짐 (정확하게 html 로 마크업이 안되어 있을 경우에는 lxml 사용)
  - (ex) Google Finance 의 예 - </tr>, </td> 가 없음. 이때 "lxml" parser 사용

```
<div id=prices class="gf-tablewrapper sfe-break-bottom-16">
<table class="gf-table historical_price">
    <tr class=bb>
        <th class="bb lm lft">Date
        <th class="rgt bb">Open
        <th class="rgt bb">High
        <th class="rgt bb">Low
        <th class="rgt bb">Close
        <th class="rgt bb">Volume
    <tr>
        <td class="lm">Feb 28, 2014
        <td class="rgt">100.71
        <td class="rgt">100.71
        <td class="rgt">100.71
        <td class="rgt rm">0
</table>
```

BeautifulSoup

- Python library for pulling data out of HTML and XML files (parsing HTML)
- functions
  - find()
  - find_all()
  - prettify()
  - select_one() ; query by CSS
  - select(*id* or *tags* or *class* or *tag.class* or *id>tag.class* ..)
  - more …

Requests module (to access URL)

- **Requests**
  - Can send URL's query strings by arguments as a dictionary od strings

```
> payload = {'key1': 'value1', 'key2': 'value2'}
> r = requests.get('https://httpbin.org/get', params=payload)
> print(r.url)
  https://httpbin.org/get?key2=value2&key1=value1
> r.text
  '{\n  "args": {\n    "key1": "value1", \n    "key2": "value2"\n  }, \n  "headers": {\n
"Accept": "*/*", \n    "Accept-Encoding": "gzip, deflate", \n    "Host": "httpbin.org", \n
"User-Agent": "python-requests/2.22.0", \n    "X-Amzn-Trace-Id": "Root=1-5f72bcc5-
847076b614ca532c66eea09a"\n  }, \n  "origin": "223.194.34.53", \n  "url":
"https://httpbin.org/get?key1=value1&key2=value2"\n}\n`
> r.json()
  {'args': {'key1': 'value1', 'key2': 'value2'},
   'headers': {'Accept': '*/*',
   'Accept-Encoding': 'gzip, deflate',
  'Host': 'httpbin.org',
  'User-Agent': 'python-requests/2.22.0',
  'X-Amzn-Trace-Id': 'Root=1-5f72bcc5-847076b614ca532c66eea09a'},
  'origin': '223.194.34.53',
  'url': 'https://httpbin.org/get?key1=value1&key2=value2'}
```

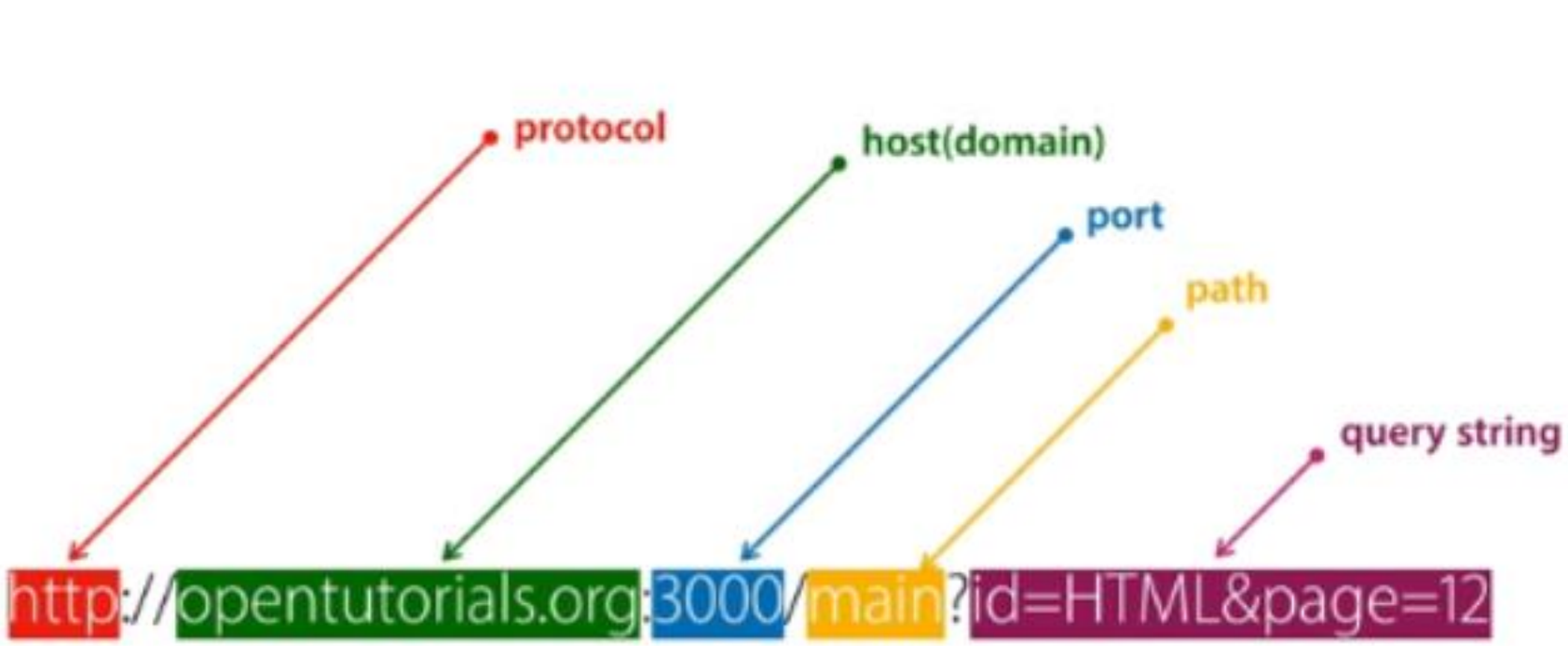
json decoded

Requests module

- Useful and good to access url
  - Requests.get() ; try to get or retrieve data from a specified source
  - Requests.post()
  - Requests.put()
  - requests.delete()
- Regardless of whether GET / POST, you never have to encode parameters again, it simply takes a dictionary as an argument and is good to go:
  - userdata = {firstname:”John”, “lastname”:”Doe”, “passwd”:”jdoe123”}
  - resp = request.post(‘http://www.mywebsite.com/user’, data=userdata)
- Plus, it even has a built-in JSON decoder.
  - resp.json()
  - resp.text ; if response data is just text

URL(uniform resource locator)

- **URL (uniform resource locator)**
  - 네트워크 상에서 자원이 어디 있는지 알려주기 위한 규약



urllib: another URL handling module

- A package collecting several modules for working with URLs
- The **requests package** is recommended for a higher-level HTTP client interface.
- urllib.request
  - urlopen() ; open the url

Regular Expression (RE)

- **Regular Expression**
  - 특정한 규칙의 문자열의 집합을 표현하기 위한 형식언어
- **Meta characters**

표현식	의미
^x	문자열의 시작을 표현하며 x 문자로 시작됨을 의미한다.
x$	문자열의 종료를 표현하며 x 문자로 종료됨을 의미한다.
.x	임의의 한 문자의 자리수를 표현하며 문자열이 x 로 끝난다는 것을 의미한다.
x+	반복을 표현하며 x 문자가 한번 이상 반복됨을 의미한다.
x?	존재여부를 표현하며 x 문자가 존재할 수도, 존재하지 않을 수도 있음을 의미한다.
x*	반복여부를 표현하며 x 문자가 0번 또는 그 이상 반복됨을 의미한다.
x|y	or 를 표현하며 x 또는 y 문자가 존재함을 의미한다.
(x)	그룹을 표현하며 x 를 그룹으로 처리함을 의미한다.
(x)(y)	그룹들의 집합을 표현하며 앞에서 부터 순서대로 번호를 부여하여 관리하고 x, y 는 각 그룹의 데이터로 관리된다.
(x)(?:y)	그룹들의 집합에 대한 예외를 표현하며 그룹 집합으로 관리되지 않음을 의미한다.
x{n}	반복을 표현하며 x 문자가 n번 반복됨을 의미한다.
x{n,}	반복을 표현하며 x 문자가 n번 이상 반복됨을 의미한다.
x{n,m}	반복을 표현하며 x 문자가 최소 n번 이상 최대 m 번 이하로 반복됨을 의미한다.

Regular Expression (RE)

표현식	의미
[xy]	문자 선택을 표현하며 x 와 y 중에 하나를 의미한다.
[^xy]	not 을 표현하며 x 및 y 를 제외한 문자를 의미한다.
[x-z]	range를 표현하며 x ~ z 사이의 문자를 의미한다.
₩^	escape 를 표현하며 ^ 를 문자로 사용함을 의미한다.
₩b	word boundary를 표현하며 문자와 공백사이의 문자를 의미한다.
₩B	non word boundary를 표현하며 문자와 공백사이가 아닌 문자를 의미한다.
₩d	digit 를 표현하며 숫자를 의미한다.
₩D	non digit 를 표현하며 숫자가 아닌 것을 의미한다.
₩s	space 를 표현하며 공백 문자를 의미한다.
₩S	non space를 표현하며 공백 문자가 아닌 것을 의미한다.
₩t	tab 을 표현하며 탭 문자를 의미한다.
₩v	vertical tab을 표현하며 수직 탭(?) 문자를 의미한다.
₩w	word 를 표현하며 알파벳 + 숫자 + _ 중의 한 문자임을 의미한다.
₩W	non word를 표현하며 알파벳 + 숫자 + _ 가 아닌 문자를 의미한다.

Flag	의미
g	Global 의 표현하며 대상 문자열내에 모든 패턴들을 검색하는 것을 의미한다.
i	Ignore case 를 표현하며 대상 문자열에 대해서 대/소문자를 식별하지 않는 것을 의미한다.
m	Multi line을 표현하며 대상 문자열이 다중 라인의 문자열인 경우에도 검색하는 것을 의미한다.

Regular Expression (RE)

❖ **Examples**

- ^http : 문자열의 맨 처음에 http가 온 경우에 매치
- them$: 문자열이 them으로 끝난 경우에 them에 매치
- \bplay\b : play 의 양 끝에 단어 경계가 오는 경우에만 play에 매치
  - ("playground"의 play에는 매치하지 않음)
- \bplay\B : play뒤에 단어 경계가 아닌 것이 왔을 때 play에 매치
  - (play는 No, "playground", "playball" 은 Yes)

- /[0-9]/g : 전체에서 0~9 사이의 아무 숫자 '한 개' 찾음
- /[to]/g : 전체에서 t 혹은 o 를 찾음
- /filter/g : 전체에서 'filter' 라는 단어에 매칭되는 것을 찾음
- \b(?!\b**to**\b)\w+\b : 'to'라는 단어 빼고 다른 단어 매칭
  - (\w+ 는 match one or more word charqcters: '[a-zA-Z0-9_]'
- /^\d{3}-\d{3,4}-\d{4}$/ : 전화번호
- /^01([0|1|6|7|8|9]?)**-?**([0-9]{3,4})**-?**([0-9]{4})$/ 휴대폰번호