

파이썬 실전프로젝트(1)

# 축구선수 시장가치 분석 1부

BeautifulSoup 웹크롤링 & Pandas 데이터분석



# ORIENTATION

프로젝트 소개

# Transfermarkt.com

국민일보 | 4일 전 | 네이버뉴스

문체부 "손흥민 경제적 파급효과 2조원"

손흥민의 이달 기준 시장가치는 **트랜스퍼마켓** 기준 1206억원으로 추산된다. 이 분석에 따르면 유럽 5개국 500명을 표본조사한 수출 증대·유발 효과가 1조 1220억...

[글로벌-스포츠 24] 올해 몸값 폭락 선수 '워스트 5' 누구?...1억2,0...

일본의 '풋볼 채널' 편집부가 데이터 전문사이트 **트랜스퍼마크트(transfermarkt)** 몸값이 폭락한 선수 톱5를 소개했다. ...

부산일보 | 2020.05.24. | 네이버뉴스

손흥민 이적료 866억, 일본 선수 5명 합한 것보다 많다

유럽에서 뛰는 선수들의 이적료를 다루는 전문 매체 **트랜스퍼마크트(Transfermarkt)**는 24일(한국시간) 홈페이지를

연합뉴스 | 2018.06.17. | 네이버뉴스

[카드뉴스] '몸값 1천억원' 한국, '1조원' 독일 이길 수 있을까  
독일에 기반을 둔 축구선수 이적 전문 사이트인 **트랜스페르마크트(transfermarkt)**

t) 기준으로는 프랑스가... 자료 / 트랜스페르마크트(transfermarkt) 몸값이 승...

# 최종 결과물 : Top 50 선수 정보































https://www.transfermarkt.com/spieler-statistik/wertvollstespieler/marktwertetop

tm NEWS TRANSFERS & RUMOURS MARKET VALUES COMPETITIONS FORUMS MY TM LIVE 2

Worldwide

Show

Compact Detailed Gallery

#	Player	Age ↓	Night.	club	Market value ↓
One	 <b>Kylian Mbappé</b> Centre-Forward	22			€180.00m
2	 <b>Neymar</b> Left Winger	28			€128.00m
3	 <b>Sadio Mané</b> Left Winger	28			€120.00m
4	 <b>Mohamed Salah</b> Right Winger	28			€120.00m
5	 <b>Harry Kane</b> Centre-Forward	27			€120.00m
6	 <b>Kevin De Bruyne</b> Attacking Midfield	29			€120.00m
7	 <b>Trent Alexander-Arnold</b> Right-Back	22			€110.00m
8	 <b>Raheem Sterling</b> Left Winger	26			€110.00m ↓
9	 <b>João Félix</b> Second Striker	21			€100.00m ↑
	 <b>Erling Haaland</b>				



	number	name	position	age	nation	team	value
0	1	Kylian Mbappé	Centre-Forward	22	France	Paris Saint-Germain	€180.00m
1	2	Neymar	Left Winger	28	Brazil	Paris Saint-Germain	€128.00m
2	3	Sadio Mané	Left Winger	28	Senegal	Liverpool FC	€120.00m
3	4	Mohamed Salah	Right Winger	28	Egypt	Liverpool FC	€120.00m
4	5	Harry Kane	Centre-Forward	27	England	Tottenham Hotspur	€120.00m
5	6	Kevin De Bruyne	Attacking Midfield	29	Belgium	Manchester City	€120.00m
6	7	Trent Alexander-Arnold	Right-Back	22	England	Liverpool FC	€110.00m
7	8	Raheem Sterling	Left Winger	26	England	Manchester City	€110.00m
8	9	João Félix	Second Striker	21	Portugal	Atlético Madrid	€100.00m
9	10	Erling Haaland	Centre-Forward	20	Norway	Borussia Dortmund	€100.00m

자동 저장 켜짐 player50.xlsx Gil Jay

파일 홈 삽입 그리기 페이지 레이아웃 수식 데이터 검토 보기 도움말 표 디자인 퀴리

B2 : X ✓ fx Kylian Mbappé

	A	B	C	D	E	F	G
1	number	name	position	nation	age	team	value
2	1	Kylian Mbappé	Centre-Forward	France	22	Paris Saint-Germain	€180.00m
3	2	Neymar	Left Winger	Brazil	28	Paris Saint-Germain	€128.00m
4	3	Sadio Mané	Left Winger	Senegal	28	Liverpool FC	€120.00m
5	4	Mohamed Salah	Right Winger	Egypt	28	Liverpool FC	€120.00m
6	5	Harry Kane	Centre-Forward	England	27	Tottenham Hotspur	€120.00m
7	6	Kevin De Bruyne	Attacking Midfield	Belgium	29	Manchester City	€120.00m
8	7	Trent Alexander-Arnold	Right-Back	England	22	Liverpool FC	€110.00m
9	8	Raheem Sterling	Left Winger	England	26	Manchester City	€110.00m
10	9	João Félix	Second Striker	Portugal	21	Atlético Madrid	€100.00m
11	10	Erling Haaland	Centre-Forward	Norway	20	Borussia Dortmund	€100.00m
12	11	Jaden Sanchez	Right Winger	England	20	Borussia Dortmund	€100.00m

# 다룰 내용들

- 웹크롤링이란? : 미리 알아 두면 좋은 것들
  - 웹 스크래핑과 웹 크롤링의 차이
  - 로봇배제표준(*robots.txt*)
  - 웹의 동작 방식과 구조 : 딱 필요한 만큼만!
  - *http, html, css, java script*
  - *Chrome* 개발자 도구 사용법
- 크롤링에 필요한 Python 라이브러리 활용법
  - *Requests*
  - *BeautifulSoup*
- [코드 실습] [transfermarkt.com](http://transfermarkt.com) 크롤링

# 다루지 않는 내용들...

- 파이썬 문법
  - 변수, 리스트, 반복문 정도만 알면 됨
- 주피터 노트북 사용법
  - 다른 파이썬 *IDLE, vscode, pycharm* 등등

# WEB CRAWLING

웹크롤링이란?  
미리 알아 두면 좋은 것들

# Web Scraping vs. Web Crawling

## Web scraping

From Wikipedia, the free encyclopedia



*For broader coverage*

**Data scraping** is a technique in which a computer program extracts data from human-readable output coming from another program.



**Citation.** Please help [improve this article](#) by adding citations to challenged and removed.

[books](#) • [scholar](#) • [JSTOR](#) (June 2017) (*Learn how and when to remove this template*

**Web scraping**, **web harvesting**, or **web data extraction** is [data scraping](#) used for extracting data from websites. Web scraping software may access the [World Wide Web](#) directly using the [Hypertext Transfer Protocol](#), or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a [bot](#) or [web crawler](#). It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local [database](#) or spreadsheet, for later [retrieval](#) or [analysis](#).

- 웹 스크래핑 : 웹에 있는 데이터를 수집하는 기술
- 웹 크롤링과 거의 같은 뜻으로 사용



# 크롤링을 하기 전 마음 자세

- 사이트에 무리가 가지 않도록...
  - 쉬엄쉬엄 *time.sleep(1)*
- 저작권에도 신경 쓰면서...
- 하지 말라는 건 하지 말고..
  - 로봇배제표준 *robots.txt*

# 로봇 배제 표준이란?

## 로봇 배제 표준

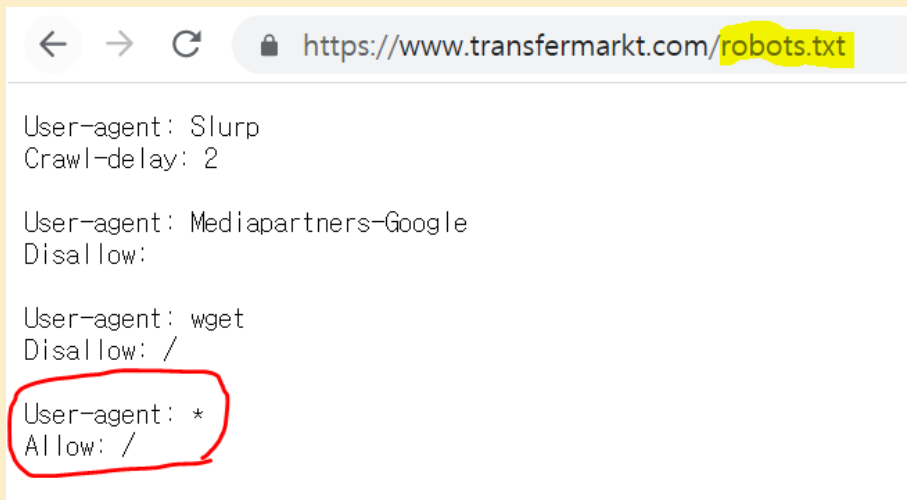
위키백과, 우리 모두의 백과사전.

"Robots.txt"는 이 문서를 가리킵니다. 위키백과의 Robots.txt의 파일을 보실려면, [미디어위키:Robots.txt](#) 와 [ko.wikipedia.org/robots.txt](https://ko.wikipedia.org/robots.txt) 를 참조하시길 바랍니다.

로봇 배제 표준(robots exclusion standard), 로봇 배제 프로토콜(robots exclusion protocol)은 웹 사이트에 로봇이 접근하는 것을 방지하기 위한 규약으로, 일반적으로 접근 제한에 대한 설명을 robots.txt에 기술한다.

이 규약은 1994년 6월에 처음 만들어졌고, 아직 이 규약에 대한 RFC는 없다.

이 규약은 권고안이며, 로봇이 robots.txt 파일을 읽고 접근을 중지하는 것을 목적으로 한다. 따라서, 접근 방지 설정을 하였다고 해도, 다른 사람들이 그 파일에 접근할 수 있다. robots.txt 파일은 항상 사이트의 루트 디렉토리에 위치해야 한다.<sup>[1]</sup>



```
← → ↻ 🔒 https://www.transfermarkt.com/robots.txt

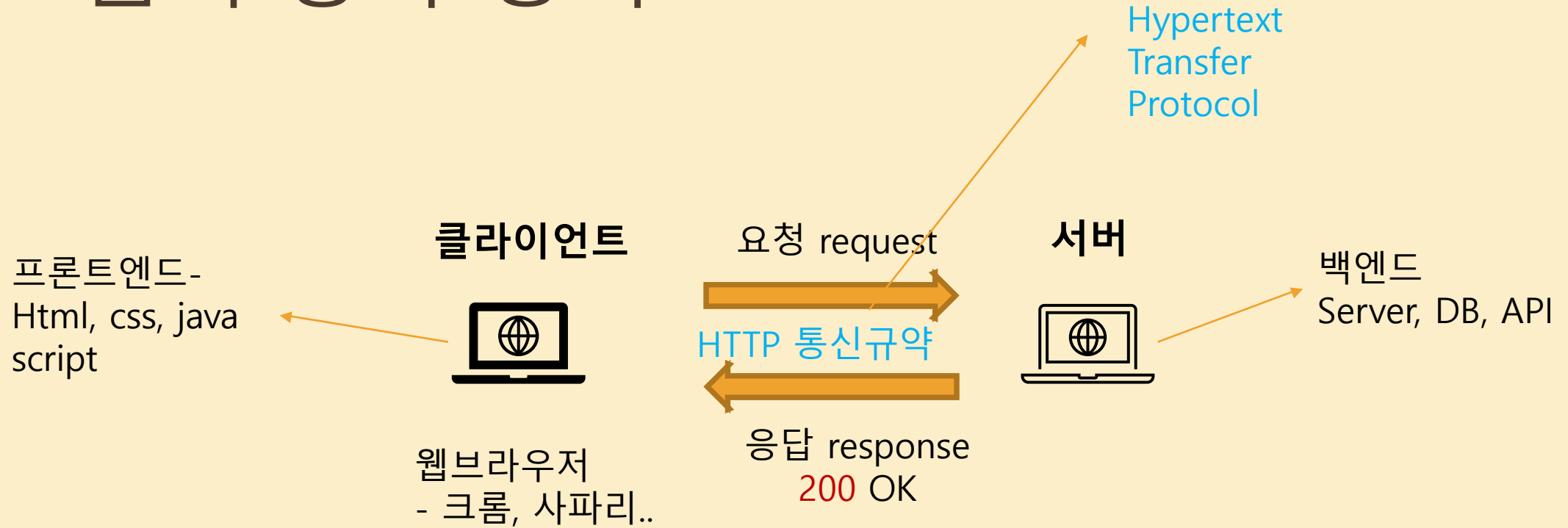
User-agent: Slurp
Crawl-delay: 2

User-agent: Mediapartners-Google
Disallow:

User-agent: wget
Disallow: /

User-agent: *
Allow: /
```

# 웹의 동작 방식





Page



Structure

- 뼈대, 구조



Styling

- 예쁘게...

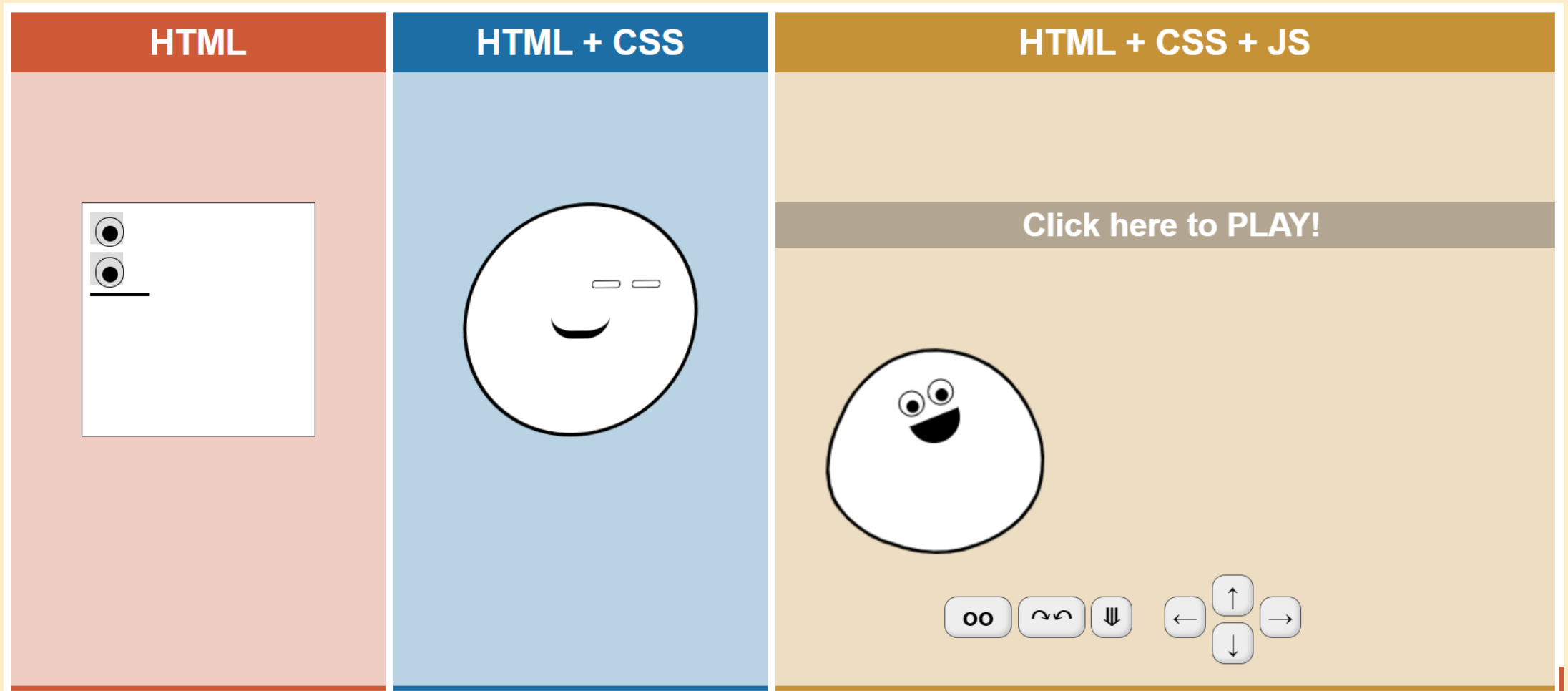


Functionality

- 살아 움직이게

# 직관적인 이해

- <https://html-css-js.com/>



# Html의 요소와 속성

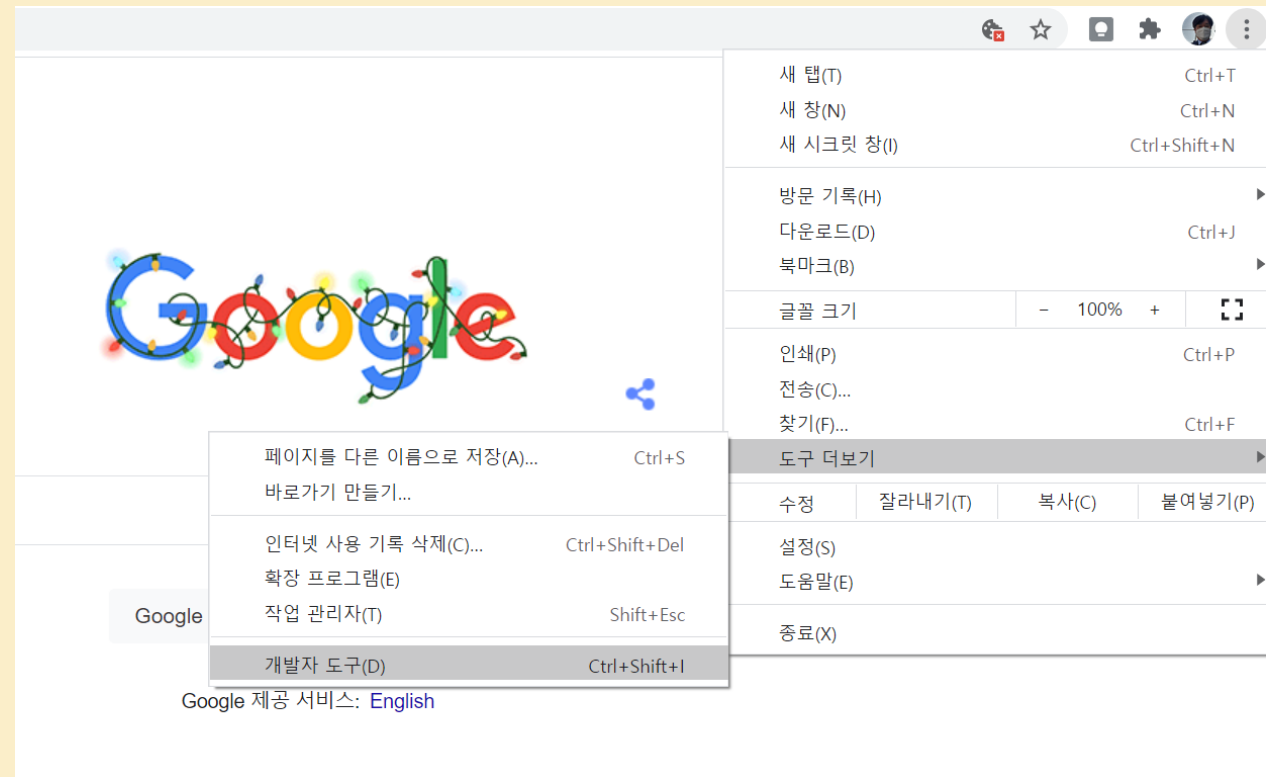
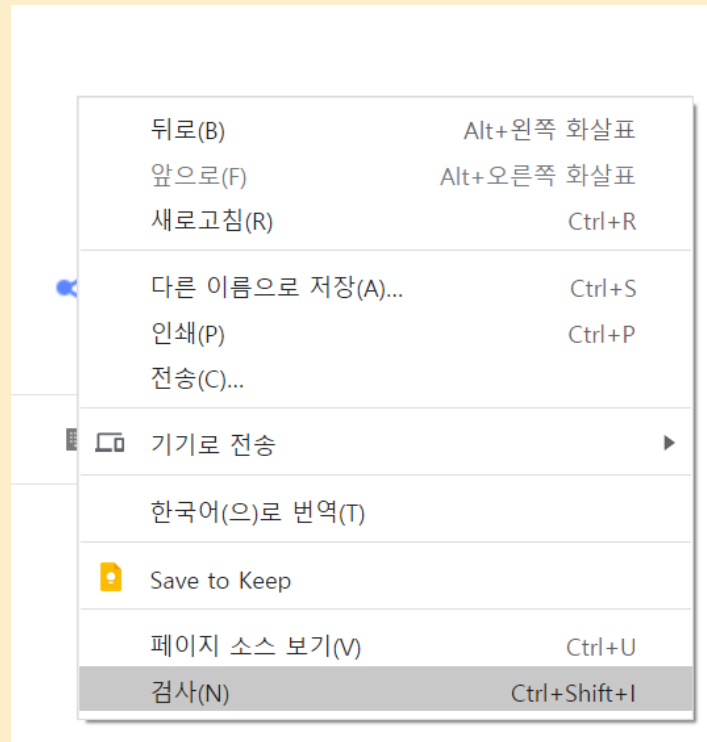
- HTML elements (요소) : 시작 tag와 끝 tag안에 내용 (예, <h1></h1>)
  - <br>, <img> 등은 닫는 태그가 없음
- HTML attributes (속성) : 요소의 추가적인 정보 제공 (예, href=, src=, width=)

```
<!DOCTYPE html>
<html>
<body>
<h1>안녕하세요?</h1>
<h2>반갑습니다.</h2>
<a
href="https://search.naver.com/search.naver?where=image&sm=tab_jum&query=%EA%B3%
A0%EC%96%91%EC%9D%B4/">고양이 사진 보러 가기</a>
<br>
<p>우리 집 고양이 코코도 보여드릴게요. </p>

</body>
</html>
```

# Chrome 개발자 도구

- Chrome에서 오른쪽 마우스 -> 검사(N)
- 또는 F12
- 또는 : -> 도구 더보기-> 개발자 도구



# 라이브러리 실습

requests  
BeautifulSoup



# 크롤링에 필요한 라이브러리



- 웹페이지(html) 읽어오기
- 빠르다
- 정적 웹페이지



- 원하는 데이터 가져오기 (스크래핑)



- 웹페이지 자동화
- 느리다
- 동적 웹페이지 (스크롤, 로그인...)

# Requests

- <https://requests.readthedocs.io/en/master/>

## Requests: HTTP for Humans™

Release v2.25.1. ([Installation](#))

downloads 1G license Apache 2.0 wheel yes python 2.7 | 3.5 | 3.6 | 3.7 | 3.8 | 3.9

**Requests** is an elegant and simple HTTP library for Python, built for human beings.

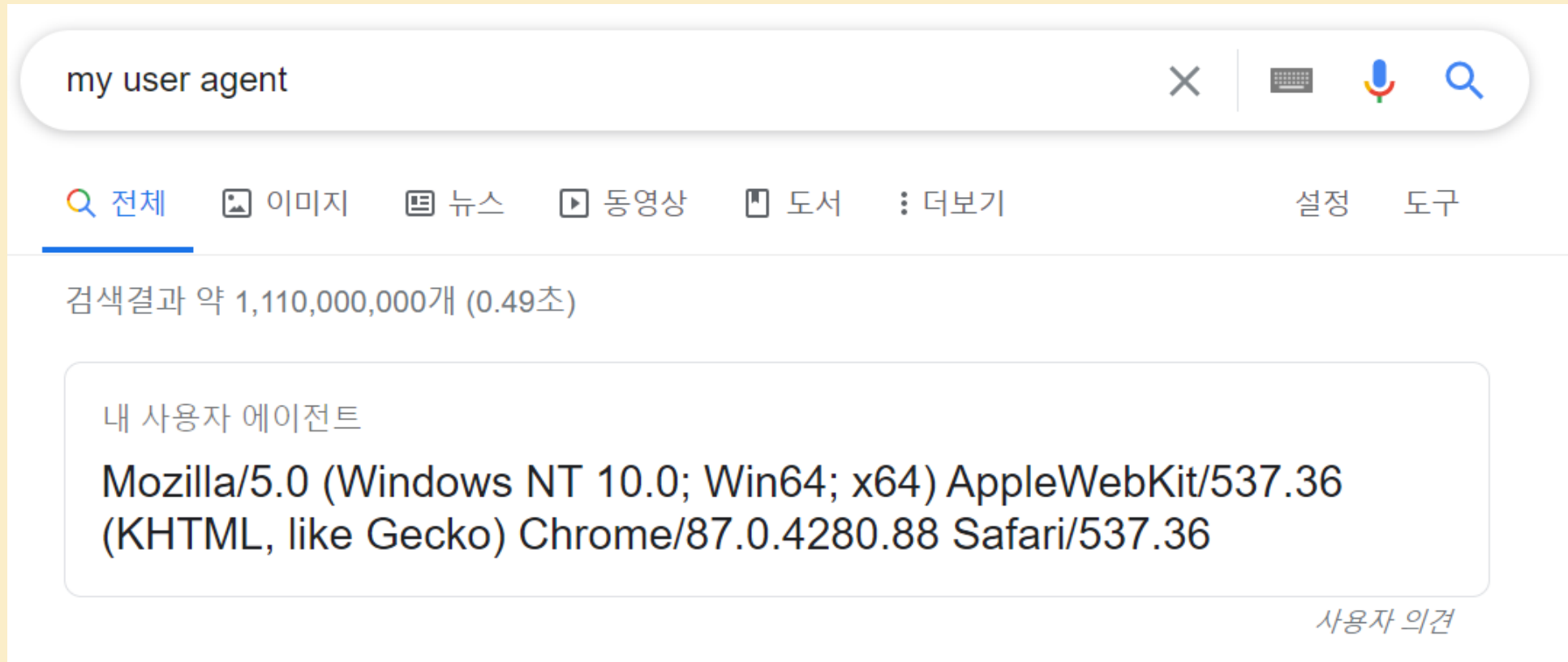
---

**Behold, the power of Requests:**

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type": "User" ... '
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

# User agent

- 'my user agent'로 검색하여 정보를 확인하고 headers에 넣어준다.
- > 내가 어떤 브라우저를 사용하여 접속했는지 (가짜로) 알려주기 위해...



# Requests 코드

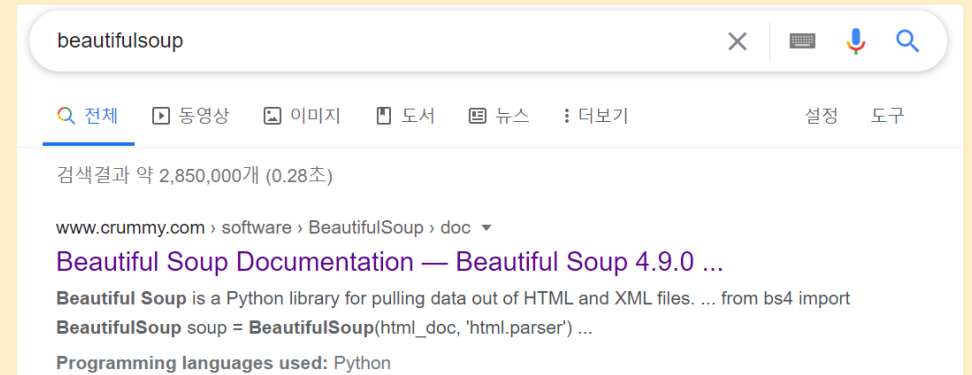
```
1 import requests
```

```
1 headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 W  
2 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}  
3  
4 url = "https://www.transfermarkt.com/spieler-statistik/wertvollstespieler/marktwertetop"  
5  
6 r = requests.get(url, headers=headers)  
7 r.status_code
```

```
] : 200
```

```
headers = {'User-Agent' : '유저정보'}  
url = '접속할 사이트'  
requests.get(url, headers = headers)
```

# Beautifulsoup



← → ↺ <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Beautiful Soup 4.9.0 documentation » BeautifulSoup Documentation

## Table of Contents

- Beautiful Soup Documentation
  - Getting help
- Quick Start
- Installing BeautifulSoup
  - Problems after installation
  - Installing a parser
- Making the soup
- Kinds of objects
  - Tag
    - Name
    - Attributes
      - Multi-valued attributes
  - NavigableString
  - BeautifulSoup
  - Comments and other special strings
- Navigating the tree
  - Going down

## Beautiful Soup Documentation ¶


Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of BeautifulSoup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers BeautifulSoup version 4.9.2. The examples in this documentation should work the same way in Python 2.7 and Python 3.8.

You might be looking for the documentation for BeautifulSoup 3. If so, you should know that BeautifulSoup 3 is no longer being developed and that support for it will be dropped on or after December 31, 2020. If you want to learn about the differences between BeautifulSoup 3 and BeautifulSoup 4, see [Porting code to BS4](#).

This documentation has been translated into other languages by BeautifulSoup users:



# Quick Start만 이해하면 오케이~!

## Quick Start

Here's an HTML document I'll be using as an example throughout this document. It's part of a story from *Alice in Wonderland*:

```
html_doc = """<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 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>
"""
```

Running the “three sisters” document through Beautiful Soup gives us a `BeautifulSoup` object, which represents the document as a nested data structure:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

print(soup.prettify())
# <html>
# <head>
# <title>
#   The Dormouse's story
# </title>
# </head>
# <body>
# <p class="title">
```

# Quick Start만 이해하면 오케이~!

```
soup.title
# <title>The Dormouse's story</title>

soup.title.name
# u'title'

soup.title.string
# u'The Dormouse's story'

soup.title.parent.name
# u'head'

soup.p
# <p class="title"><b>The Dormouse's story</b></p>

soup.p['class']
# u'title'

soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.find(id="link3")
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

One common task is extracting all the URLs found within a page's <a> tags:

```
for link in soup.find_all('a'):
    print(link.get('href'))
# http://example.com/elsie
# http://example.com/lacie
# http://example.com/tillie
```

```
1 html_doc = """<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>"""
```

```
1 from bs4 import BeautifulSoup
2 soup = BeautifulSoup(html_doc, 'html.parser')
3
4 print(soup.prettify())
```

```
<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 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>
</body>
</html>
```



# 첫번째 tag의 정보를 가져오기

```
<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>
```

```
1 soup.p
```

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

```
1 soup.find('p')
```

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

# 첫번째 tag의 속성 정보를 가져오기

```
<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>
  ;
  nd they lived at the bottom of a well.
</p>
```

```
1 soup.a['href']
```

```
http://example.com/elsie
```

```
1 soup.find('a')['href']
```

```
'http://example.com/elsie'
```

# find()와 find\_all()

```
1 soup.find('a')
```

```
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

```
1 soup.find_all('a')
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,  
  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,  
  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

- Find\_all()의 결과물은 리스트 타입!

# 텍스트만 가져오기

## .text

```
1 soup.find('a')
```

```
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

```
1 soup.find_all('a')
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,  
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,  
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

```
1 soup.find('a').text
```

```
'Elsie'
```

```
1 a_list = soup.find_all('a')  
2  
3 for i in a_list:  
4     print(i.text)
```

```
Elsie
```

```
Lacie
```

```
Tillie
```



- find\_all()로 가져온 정보는 반복문을 통해 하나씩 가져올 수 있다.

# 실전 크롤링



[transfermarkt.com](https://transfermarkt.com)


# 실전 크롤링 - transfermarkt.com

https://www.transfermarkt.com/spieler-statistik/wertvollstespieler/marktwertetop



 .with 

f | t | i | r

Enter search term  

NEWS TRANSFERS & RUMOURS **MARKET VALUES** COMPETITIONS FORUMS MY TM LIVE  LOGIN

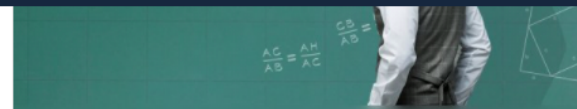
**LATEST NEWS**

- > Latest market value updates
- > Biggest market value increases
- > Greatest decreases in market value
- >   What's my value?

**STATS**

- > Most valuable clubs
- > **Most valuable players**
- > Most valuable matches
- > The most valuable internationals
- > Most valuable XI
- > Record market values
- > Highest ever market value by age
- > Market value at the end of career
- > Market Value Development during loan

Main position:



크롬 개발자 도구  
: 모든 것은 "odd"와 "even" 클래스 안에

tm

NEWS

TRANSFERS & RUMOURS

MARKET VALUES

COMPETITIONS

FORUMS

MY TM

LIVE

2

Year:

All

Nationality:

All

Confederation:

Worldwide

Show

Compact

Detailed

Gallery

tr.odd 523.33 × 44.67

		Age ↓	Night.	club	Market value ↑
One	<div> <div>Kyllian Mbappé</div> <div>Centre-Forward</div> </div>	22			€180.00m
2	<div> <div>Neymar</div> <div>Left Winger</div> </div>	28			€128.00m
3	<div> <div>Sadio Mané</div> <div>Left Winger</div> </div>	28			€120.00m
4	<div> <div>Mohamed Salah</div> <div>Right Winger</div> </div>	28			€120.00m
	<div> <div>Harry Kane</div> <div>Striker</div> </div>				

Achieving New Year's goal

Pagoda Language School [Here](#)

LATEST RUMOURS

Go to the rumour mill >

LATEST TRANSFERS

player	Transfer	Fee
<div> <div>Joe Lumley</div> <div>Goalkeeper</div> </div>		End of loan
<div> <div>Gidi Kanyuk</div> <div>Attacking Mi...</div> </div>		free transfer May 31, 2021
<div> <div>A. Palocevic</div> <div>Attacking Mi...</div> </div>		End of loan
<div> <div>Guido Herr...</div> <div>Striker</div> </div>		End of

[illegible]

<td> 태그 안에 모든 정보가 차례로..

```
▼<tbody>
  ▼<tr class="odd">
    ▼<td class="zentriert"> == $0
      ▼<font style="vertical-align: inherit;">
        <font style="vertical-align: inherit;">One</font>
      </font>
    </td>
```

```
</td>
▼<td class="hauptlink"> == $0
  <a class="spielprofil_tooltip tooltipstered" id="342229" href="/kylian-mbappe/profil/spieler/342229">Kylian Mbappé</a>
</td>
```

```
</td>
▼<tr>
  <td>Centre-Forward</td>
</tr>
```



# <td> 태그 안에 모든 정보가 차례로..

```
▼<td class="zentriert">
  ▼<font style="vertical-align: inherit;">
    <font style="vertical-align: inherit;">22</font>
  </font>
</td>
▼<td class="zentriert"> </td>
```

```
▼<td class="zentriert">
  
</td>
```

```
▼<td class="zentriert">
  ▼<a class="vereinprofil_tooltip tooltipstered" id="583" href="/fc-paris-saint-germain/startseite/verein/583">
    
  </a> == $0
</td>
```

```
▼<td class="rechts hauptlink">
  <b>€180.00m</b>
  <span title="€180.00m" class="icons_sprite grey-block-ten">
    &nbsp;</span>
</td>
```

# 크롤링 설계

1단계 : 'tr' 태그의 클래스가 'odd'와 'even'인 것을 find\_all로 모두 찾는다.

```
player_info= soup.find_all('tr', class_=["odd","even"])
```

2단계 : 필요한 정보를 담은 빈 리스트를 각각 만든다.

```
number =[ ]
```

```
name = [ ]
```

```
position =[ ]
```

```
age =[ ]
```

```
nation = [ ]
```

```
team =[ ]
```

```
value=[ ]
```

# 크롤링 설계

3단계 : 반복문으로 'td' 태그인 것을 모두 찾는다.

```
for info in player_info:  
    player = info.find_all('td')
```


































4단계 : append함수로 리스트에 요소들을 추가한다.

```
number.append(player[0].text)  
name.append(player[3].text)  
position.append(player[4].text)  
age.append(player[5].text)  
nation.append(player[6].img['title'])  
team.append(player[7].img['alt'])  
value.append(player[8].text)
```































# 크롤링 설계

5단계 : 2번째 페이지까지 크롤링하기  
페이지 규칙 발견하기!

https://www.transfermarkt.com/spieler-statistik/wertvollstespieler/marktwertetop?page=2

tm NEWS TRANSFERS & RUMOURS MARKET VALUES COMPETITIONS FORUMS MY TM LIVE					
Compact Detailed Gallery					
#	Player	Age ↑	Night.	club	Market value ↑
26	 <b>Matthijs de Ligt</b> Centre-Back	21			€75.00m ↑
27	 <b>Andrew Robertson</b> Left-Back	26			€75.00m ↑
28	 <b>Thibaut Courtois</b> Goalkeeper	28			€75.00m ↑
29	 <b>Marc-André ter Stegen</b> Goalkeeper	28			€75.00m ↑
30	 <b>Roberto Firmino</b> Centre-Forward	29			€72.00m ▬
31	 <b>Lautaro Martinez</b> Centre-Forward	23			€70.00m ▬
32	 <b>Federico Valverde</b> Central Midfield	22			€70.00m ↑
33	 <b>Frenkie de Jong</b> Central Midfield	23			€70.00m ↓
34	 <b>Sergej Milinkovic-Savic</b> Central Midfield	25			€70.00m ↑
35	 <b>Jose Gimenez</b> Centre-Back	25			€70.00m ↑
36	 <b>Bernardo Silva</b> Right Winger	26			€70.00m ↓

https://www.transfermarkt.com/spieler-statistik/wertvollstespieler/marktwertetop?page=1

tm NEWS TRANSFERS & RUMOURS MARKET VALUES COMPETITIONS FORUMS MY TM LIVE					
Compact Detailed Gallery					
#	Player	Age ↑	Night.	club	Market value ↑
One	 <b>Kylian Mbappé</b> Centre-Forward	22			€180.00m ▬
2	 <b>Neymar</b> Left Winger	28			€128.00m ▬
3	 <b>Sadio Mané</b> Left Winger	28			€120.00m ▬
4	 <b>Mohamed Salah</b> Right Winger	28			€120.00m ▬
5	 <b>Harry Kane</b> Centre-Forward	27			€120.00m ▬
6	 <b>Kevin De Bruyne</b> Attacking Midfield	29			€120.00m ▬
7	 <b>Trent Alexander-Arnold</b> Right-Back	22			€110.00m ▬
8	 <b>Raheem Sterling</b> Left Winger	26			€110.00m ↓
9	 <b>João Félix</b> Second Striker	21			€100.00m ↑
10	 <b>Erling Haaland</b> Centre-Forward	20			€100.00m ↑

# 크롤링 설계

## 5단계 : 2번째 페이지까지 크롤링하기

반복문으로 차례로 크롤링

```
for i in range (1,3):

    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3496.102 Safari/537.36'}
    url = f"https://www.transfermarkt.com/marktwertetop/wertvollstespieler?ajax=yw1&page={i}"
    response = requests.get(url, headers=headers)
    response

    soup = BeautifulSoup(response.content, 'html.parser')

    player_info = soup.find_all('tr', class_=["odd", "even"])

    for info in player_info:

        player = info.find_all('td')

        number = player[0].text
        name = player[3].text
        position = player[4].text
```

# 크롤링 설계

6단계 : 판다스 데이터프레임과 csv파일로 저장한다. -> Pandas 강의에서 자세하게 다룰 예정

```
df = pd.DataFrame(  
    {"number":number,  
     "name": name,  
     "position": position,  
     "nation": nation,  
     "age": age,  
     "team": team,  
     "value": value}  
)  
df.to_csv(f'player{i}.csv', index=False)
```

# 라이브 코딩

같이 따라해봐요.

파이썬 실전프로젝트(1)

# 축구선수 시장가치 분석 2부

BeautifulSoup 웹스크래핑 & **Pandas** 데이터분석





# Pandas 데이터 분석

## Pandas 데이터 조작 및 분석을 위한 파이썬 라이브러리

pandas

X🗣️🔍

전체

뉴스

이미지

동영상

도서

더보기

설정

도구

검색결과 약 500,000,000개 (0.38초)

pandas.pydata.org

pandas - Python Data Analysis Library

pandas. pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Documentation

pandas is an open source, BSD-licensed library providing high ...

10 Minutes to Pandas

Creating a DataFrame by passing a dict of objects that can be ...

pandas.DataFrame

Arithmetic operations align on both

API reference

DataFrame - Series - Input/output - General functions - GroupBy

Package overview

Package overview¶ · Tabular data with heterogeneously-typed ...

User Guide

User Guide¶. The User Guide

Pandas

소프트웨어

영어에서 번역됨 - 컴퓨터 프로그래밍에서 pandas는 데이터 조작 및 분석을 위해 Python 프로그래밍 언어로 작성된 소프트웨어 라이브러리입니다. 특히 숫자 테이블과 시계열을 조작하기위한 데이터 구조와 연산을 제공합니다. 3 절 BSD 라이선스에 따라 출시 된 무료 소프트웨어입니다. 위키백과(영어)

원래 설명 보기 ▼

# Pandas 데이터 분석

Pandas Cheat Sheet [https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)

← → ↻ 🔒 [https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf) 1 / 2

Pandas\_Cheat\_Sheet.pdf

## Data Wrangling with pandas Cheat Sheet

<http://pandas.pydata.org>

### Tidy Data – A foundation for wrangling in pandas

In a tidy data set:

Each **variable** is saved in its own **column** & Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M \* A → F

M \* A

### Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame({
    "a" : [4 ,5, 6],
    "b" : [7, 8, 9],
    "c" : [10, 11, 12]},
    index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
```

### Reshaping Data – Change the layout of a data set

**pd.melt(df)**  
Gather columns into rows.

**df.pivot(columns='var', values='val')**  
Spread rows into columns.

**pd.concat([df1,df2])**  
Append rows of DataFrames

**pd.concat([df1,df2], axis=1)**  
Append columns of DataFrames

**df.sort\_values('mpg')**  
Order rows by values of a column (low to high).

**df.sort\_values('mpg',ascending=False)**  
Order rows by values of a column (high to low).

**df.rename(columns = {'y':'year'})**  
Rename the columns of a DataFrame

**df.sort\_index()**  
Sort the index of a DataFrame

**df.reset\_index()**  
Reset index of DataFrame to row numbers, moving index to columns.

**df.drop(columns=['Length','Height'])**  
Drop columns from DataFrame

# DataFrame 만들기 : pd.DataFrame()

## 데이터 프레임 만드는 방법 2가지

1) 딕셔너리에 칼럼이름과 리스트 넣기

```
1 df1 = pd.DataFrame(  
2     {"이름" : ['손흥민', '메시', '호날두'],  
3       "나이" : [28, 33, 35],  
4       "소속" : ['토트넘', '바르셀로나', '유벤투스']}  
5 )
```

1 df1

	이름	나이	소속
0	손흥민	28	토트넘
1	메시	33	바르셀로나
2	호날두	35	유벤투스

2) 리스트 안에 리스트 넣고 칼럼 이름은 따로 지정

```
1 player_list = [  
2     ['손흥민', 28, '토트넘'],  
3     ['메시', 33, '바르셀로나'],  
4     ['호날두', 35, '유벤투스']  
5 ]  
6 df2 = pd.DataFrame(player_list, columns=['이름', '나이', '소속'])
```

1 df2

	이름	나이	소속
0	손흥민	28	토트넘
1	메시	33	바르셀로나
2	호날두	35	유벤투스

# DataFrame 저장하고 불러오기

Csv파일로 저장하기

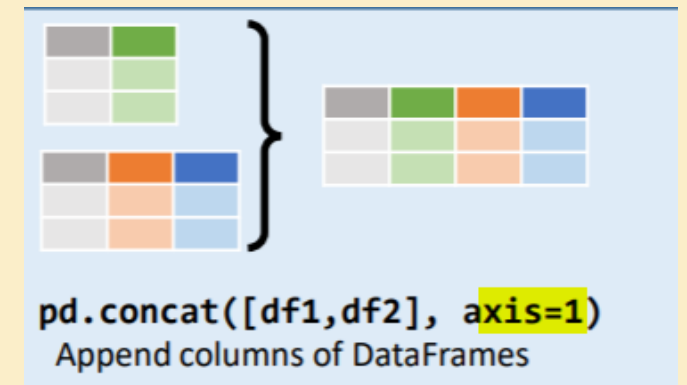
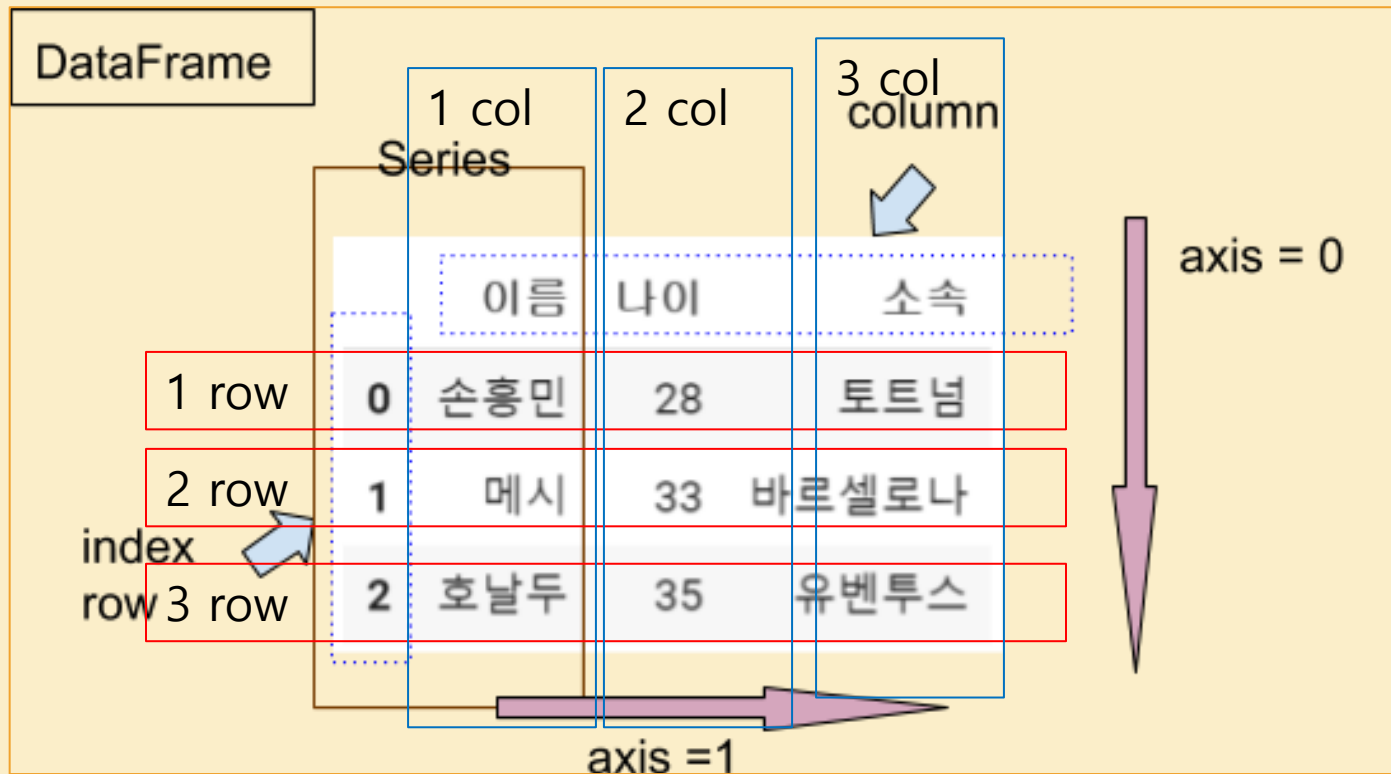
```
pd.to_csv('player50.csv', index=False)
```

Csv파일을 불러오기

```
pd.read_csv('player50.csv')
```

# DataFrame 이름 붙이기

Row는 행, Column은 열



# DataFrame 뜯어 보기와 인덱싱

명령어	설명
df.shape	행과 열의 개수 파악
df.info()	전체 정보 요약 (개수와 데이터타입)
df.head()	처음 5개 행 보여주기
df.tail()	마지막 5개 행 보여주기
df[:10] 또는 df.iloc[:10]	처음 10개 행까지 보여주기
df['name'] 또는 df.name	컬럼 이름 선택하기
df[['name', 'value']]	여러 개 컬럼 이름 선택하기
df.loc[:,['name', 'value']]	[loc]쉽표를 기준으로 행과 열의 이름으로 선택하기
df.loc[df['age']>30, ['name', 'age']]	- 슬라이싱, 조건식 등 가능

- iloc는 인덱스 번호 기준 (파이썬 인덱싱과 같음)
- loc는 이름 기준 (끝 이름 포함)

```
1 players.iloc[0:2] # 처음부터 인덱싱
```

	number	name	position
0	1	Kylian Mbappé	Centre-Forward
1	2	Neymar	Left Winger

```
1 players.loc[0:2]
```

	number	name	position
0	1	Kylian Mbappé	Centre-Forward
1	2	Neymar	Left Winger
2	3	Sadio Mané	Left Winger

# DataFrame 정렬하기와 컬럼 바꾸기

명령어	설명
<code>df.sort_index()</code>	인덱스 기준으로 정렬
<code>df.sort_values('age')</code>	컬럼 이름을 기준으로 정렬
<code>df.set_index('number', inplace=True)</code>	인덱스를 컬럼이름으로 설정
<code>df.rename(columns = {'team':'club'})</code>	컬럼 이름 변경
<code>df.rename(columns = {'team':'club'}, inplace=True)</code>	컬럼 이름 변경하고 저장
<code>df['시장가치(억)']=df['value']*13</code>	새로운 컬럼 생성
<code>df.drop(columns=['value'])</code>	컬럼 삭제

# DataFrame 통계 분석과 groupby()

명령어	설명
df['age'].mean()	특정 컬럼의 평균
df['시장가치(억)'].sum()	특정 컬럼의 합
df['nation'].mode()	특정 컬럼의 가장 많이 나타난 값(최빈값)
df.describe()	모든 수치형 데이터의 통계
df.groupby("nation")	컬럼 이름으로 그룹화
df.groupby("nation").max()	그룹화 이후 통계 확인



# 감사합니다.

- 자료 문의. [jaygil8755@gmail.com](mailto:jaygil8755@gmail.com)