

정 규 세 션 6 주 차

ToBig's 11기 유기윤

크롤링 : 웹 데이터 수집

BeautifulSoup 라이브러리를 중심으로

contents

Unit 01 | 소개

Unit 02 | Beautiful Soup

Unit 03 | 실용적 팁 (다이나믹 페이지 / API 이용)

Unit 04 | 과제 설명

Unit 01 | 정의

- 크롤링이란? 인터넷에 존재하는 데이터를 수집 및 저장



위키백과
우리 모두의 백과사전

Unit 01 | 정의

- 크롤링이란? 인터넷에 존재하는 데이터를 수집 및 저장

천 개의 페이지를 복.붙 해야한다면?

I love
CTRL+C &
CTRL+V

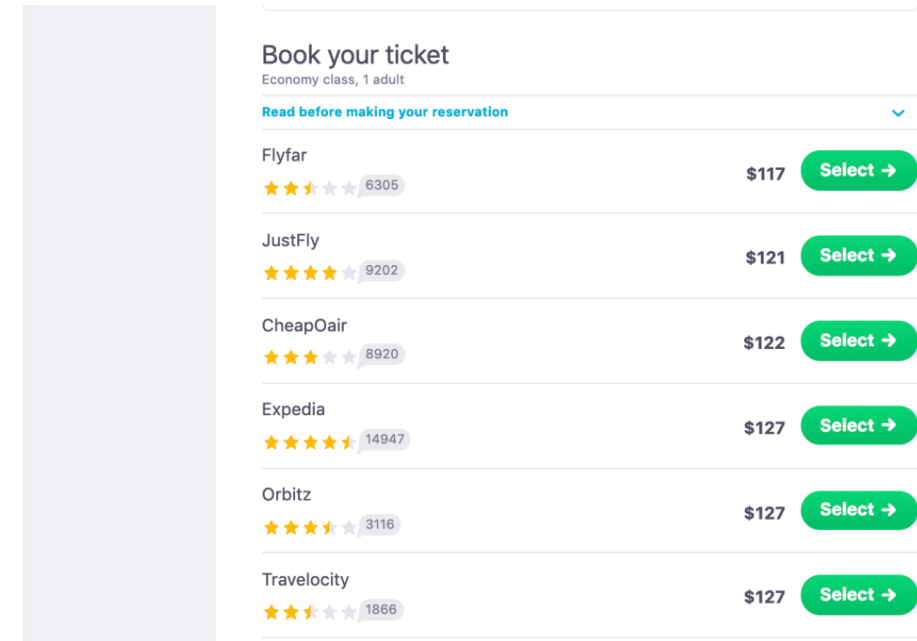
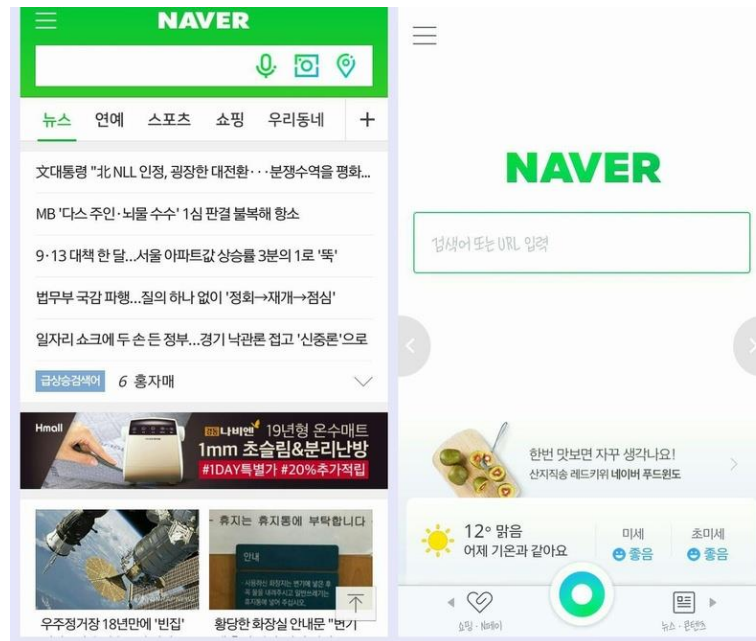
위키백과
우리 모두의 백과사전

Unit 01 | 정의

- 크롤링이란? 인터넷에 존재하는 데이터를 시스템적으로 자동화되어 수집 및 저장
- 크롤러 (Crawler) : 웹데이터를 시스템적으로 수집하는 **프로그램**
- 함께 사용되는 용어 :
 - Scraping,
 - Spider,
 - Bot

Unit 01 | 정의

- 사용 사례 : 검색 포털, 뉴스 포털, 예약 플랫폼 등
 - 구글, 네이버 등 검색포털 사이트는 모두 크롤링 기업
 - 스카이스캐너, 부킹닷컴 등 최저가 플랫폼 사이트 역시 가격정보를 크롤링해서 제공



Unit 01 | 정의

- 사용 사례: 주가 예측, 투빅스 프로젝트?!

Stock prediction: Integrating text mining approach using real-time news

GPC

"a"

"b"

"c"

Tobig's 8th Conference**투광맨**

온라인 쇼핑몰 상품정보 자동생성 모델

Unit 01 | 정의

• 합법일까?

 https://varvy.com/robots.txt

```
User-agent: *  
Disallow: /folder/  
Disallow: /file.html  
Disallow: /image.png
```

- Robots.txt 를 통해서 사이트가 크롤링을 허용하는 범위 확인가능
- 다만, 명시해놓지 않은 사이트도 많고 권고사항이라는 것을 유의
- 사실에는 저작권이 없는 반면, 개인 의견에는 저작권이 있다
- 모든 사이트는 봇에 의한 과도한 접속은 공격으로 인식하여 IP를 차단!

Unit 01 | 정의

- Robots.txt 예시. 네이버에 비해 자세하게 크롤링 허용 범위를 명시해놓은 구글.

www.naver.com/robots.txt

```
User-agent: *  
Disallow: /  
Allow : /$
```

www.google.com/robots.txt

```
User-agent: *  
Disallow: /search  
Allow: /search/about  
Allow: /search/static  
Allow: /search/howsearchworks  
Disallow: /sdch  
Disallow: /groups  
Disallow: /index.html?  
Disallow: /?  
Allow: /?hl=  
Disallow: /?hl=*&  
Allow: /?hl=*&gws_rd=ssl$  
Disallow: /?hl=*&*gws_rd=ssl  
Allow: /?gws_rd=ssl$  
Allow: /?pt1=true$  
Disallow: /imgres  
Disallow: /u/  
Disallow: /preferences  
Disallow: /setprefs  
Disallow: /default  
Disallow: /m?  
Disallow: /m/
```

Unit 01 | 소개

• 브라우징 vs. 크롤링 과정

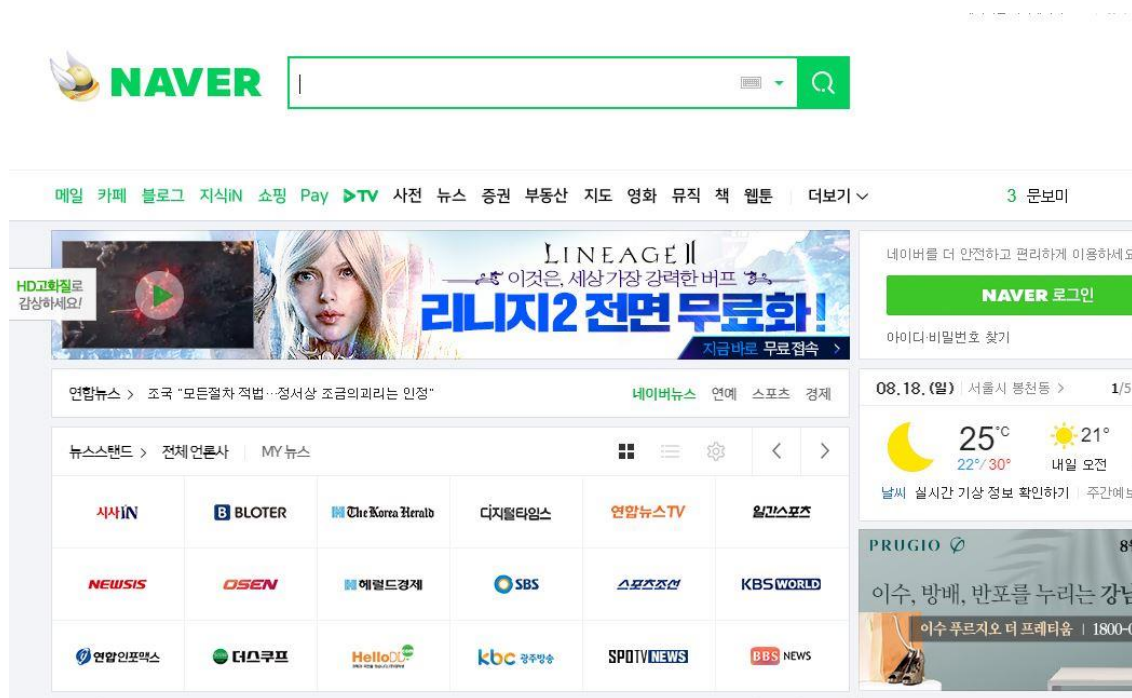
크롬 / IE → URL 입력 → HTML 저장 및 해석

쥬피터 / 커맨드 창 → Requests 라이브러리 → 사이트의 HTML 저장 → HTML 내에서 원하는 정보 찾기 → 정보 저장

Beautiful Soup 라이브러리 사용

Unit 01 | 소개

• 브라우징 vs. 크롤링 과정



```
<html lang="ko">
<head>
  <meta charset="utf-8">
  <meta name="Referren" content="origin">
  <meta http-equiv="Content-Script-Type" content="text/javascript">
  <meta http-equiv="Content-Style-Type" content="text/css">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=1100">
  <meta name="apple-mobile-web-app-title" content="NAVER">
  <meta name="robots" content="index,nofollow">
  <meta name="description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요">
  <meta property="og:title" content="네이버">
  <meta property="og:url" content="https://www.naver.com/">
  <meta property="og:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
  <meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요">
  <meta name="twitter:card" content="summary">
  <meta name="twitter:title" content="네이버">
  <meta name="twitter:url" content="https://www.naver.com/">
  <meta name="twitter:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
  <meta name="twitter:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요">
  <link rel="shortcut icon" type="image/x-icon" href="/favicon.ico?1">
  <link rel="stylesheet" type="text/css" href="https://pm.pstatic.net/css/main/v190814.css">
  <link rel="stylesheet" type="text/css" href="https://pm.pstatic.net/css/webfont/v170623.css">
  <link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/sstatic/search/pc/css/api_atcmp_190612.css">
  <script type="text/javascript" src="https://pm.pstatic.net/js/c/nlog/v181107.js"></script>
  <script type="text/javascript" src="https://ssl.pstatic.net/tveta/libs/assets/js/common/min/probe.min.js"></script>
  <script type="text/javascript">...</script>
</head>
<title>NAVER</title>
<style>
  #_nx_kbd .setkorhelp a { display:none; }
</style>
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/tveta/libs/assets/css/pc/main/min/rollingboard_imageolling_332.min.css?20180510">
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/tveta/libs/assets/css/pc/main/min/timeboard autoplay_740.min.css?20180914">
<script type="text/javascript" charset="utf-8" src="https://pm.pstatic.net/js/c/nmain/v190807.js"></script>
<script type="text/javascript" charset="utf-8" src="https://ssl.pstatic.net/tveta/libs/assets/js/pc/main/min/pc.veta.core.min.js?20180330"></script>
```

Unit 01 | HTML

- HTML 이란?
 - HyperText Markup Language 라는 의미의 **웹페이지를 위한 프로그래밍 언어**
 - 특징적으로는 제목, 단락, 목록 등과 같은 구조적 문서를 만들며, 이때 <> 런 모양의 **“태그”**란 것이 사용된다.
 - HTML 처리 장치의 행동에 영향을 주는 **자바스크립트**와 외관과 배치를 정의하는 **CSS** 같은 스크립트를 포함하거나 불러올 수 있다 (위키백과)

Unit 01 | HTML

```
<html>  
  
<h1>This is a Heading</h1>  
<p>This is a paragraph.</p>  
|  
</html>
```



This is a Heading

This is a paragraph.

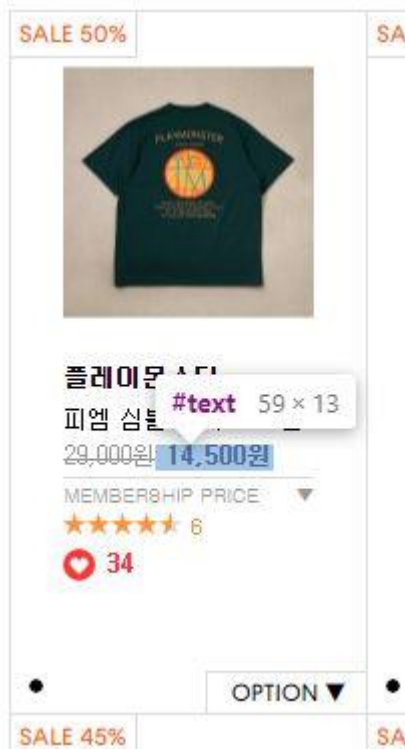
CSS 를 추가한 예시

```
<p style="color: blue">Lorem ipsum dolor.</p>
```



Lorem ipsum dolor

Unit 01 | HTML

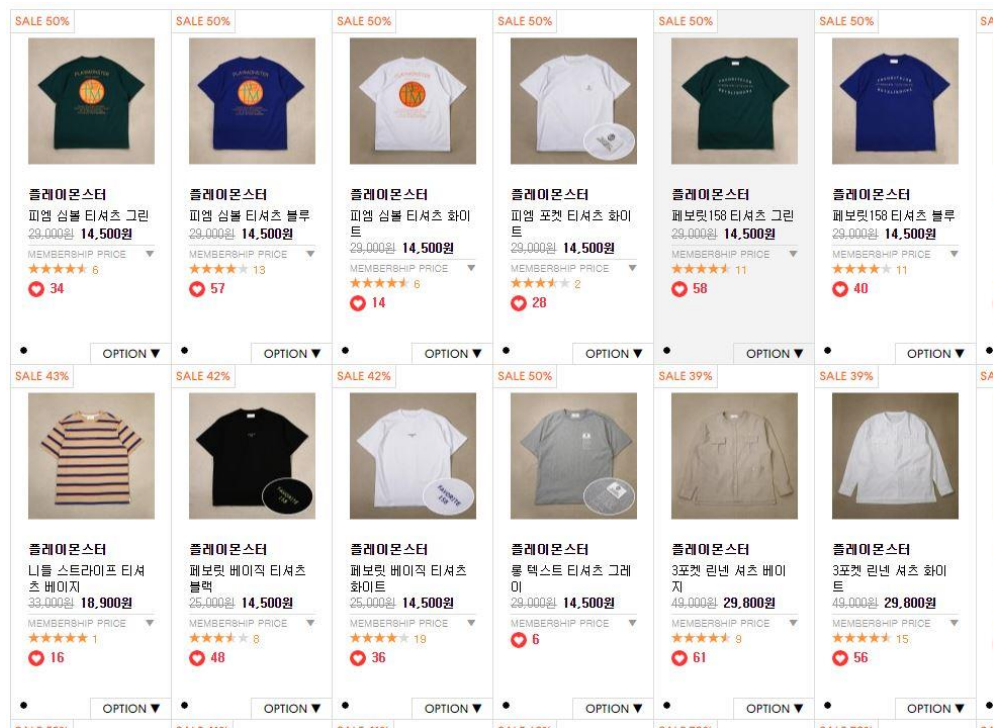


```
<div class="article_info">
  <p class="item_title">
    <a href="/playmonster">플레이몬스터</a>
  </p>
  <p class="list_info">...</p>
  <p class="price"> == $0
    <del>29,000원</del>
    " 14,500원
    "
  </p>
  <p class="membership" onclick="viewMemberPrice2('sPrice_1000384',
    '1000384', '0');">...</p>
  <div class="member_price" id="sPrice_1000384" style="display:none; z-
    index: 9999;">...</div>
  <p class="point">...</p>
  <p class="txt_cnt_like">...</p>
</div>
```

Unit 01 | HTML

- 내가 원하는 정보가 어디에 있는가?
 - 브라우저를 통해 HTML 문서 상 어디에 있는지 확인 가능
 - 크롬의 경우, 우클릭 후 “**검사**”
 - 사파리의 경우, 디벨로퍼 모드 키고 우클릭 후 “**소스보기?**”
 - <https://www.webnotes.com/view-webpage-source-css-and-html-in-safari/>

Unit 01 | HTML



다른 옷들에 대한 가격에 대한 정보는
어느 태그에 있을까??

Unit 01 | HTML

```
<p class="price">  
<del> 할인전 가격</del>  
    가격정보  
</p>
```

- 옵션 1: Cntrl + F 를 이용해서 class="price" 를 찾은 뒤 가격정보를 손으로 스크랩
- 옵션 2 : BeautifulSoup 라이브러리를 이용!

contents

Unit 01 | 소개

Unit 02 | Beautiful Soup

Unit 03 | 실용적 팁 (다이나믹 페이지 / API 이용)

Unit 04 | 과제 설명

Unit 02 | BeautifulSoup

- BeautifulSoup 이란?
 - 파이썬으로 쓰여진 **HTML Parser**; 즉, 구조를 분석하고 HTML 로 쓰여진 문서를 용이하게 읽는데 도움을 주는 라이브러리
 - 성능은 느린편이지만 **사용이 쉬운 편**
 - 다른 대안으로는 regular expression, Lxml, **Scrapy** 등 ..

Unit 02 | BeautifulSoup

- BeautifulSoup 을 이용한 크롤링 과정

Requests

라이브러리를 이용해서 사이트 요청

저장된 HTML 을 BS 로 파싱

Beautifulsoup 객체에서 원하는 태
그 검색

```
Import requests  
From bs4 import BeautifulSoup as bs  
  
response = requests.get(my_url)  
  
soup = bs(response.text, 'html.parser')  
  
  
price = Body.find(price_tag)
```

Unit 02 | BeautifulSoup

• Requests 과정

- requests.get(url, headers)
- Headers 는 사이트에게 보여지는 **접속자**
의 인상말!
- 브라우저 종류, 버전, OS 등의 정보가 담겨
있음 *"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36"*
- 필수는 아니지만 대부분 사이트는 봇 같은
header를 자동적으로 차단

```
Import requests
```

```
From bs4 import BeautifulSoup as bs
```

```
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1)  
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36'}
```

```
response = requests.get(my_url, headers)
```

```
soup = bs(response.text, 'html.parser')
```

```
price = Body.find(price_tag)
```

Unit 02 | BeautifulSoup

- 파싱 과정

- response.text 로 서버에 요청한 것을 HTML 변환
- 이때 변환과정에서 beautifulsoup 이 사용된다
- 'lxml' parser 도 설치한 뒤 사용 가능 (속도 향상)

```
Import requests  
From bs4 import BeautifulSoup as bs  
  
response = requests.get(my_url)  
soup = bs(response.text, 'html.parser')  
  
price = Body.find(price_tag)
```

Unit 02 | BeautifulSoup

• 원하는 태그를 찾는 과정

```
<span class="excitingNote" color="black"  
      Tag      attribute >first line!</span> attribute
```

`soup.find(tag, attribute)` : 문서 내 가장 첫번째

return `bs4.element.Tag` (없다면 `None`)

`Soup.findall(tag, attribute)` : 문서 내 부합하는 모든 내용

return `bs4.element.ResultSet` (없다면 `empty list`)

Tag 들로 이뤄진 리스트!

Unit 02 | BeautifulSoup

Soup.find('span')

returns

```
<span class="excitingNote">first line!</span>
```

Soup.findAll('span')

returns

```
<span class="excitingNote">first line!</span>  
<span class="excitingNote">second line!</span>  
<span class="excitingNote">third line!</span>  
<span class="excitingNote">fourth line!</span>
```


Unit 02 | 소개

Soup.find('span')
returns

```
<span class="excitingNote">first line!</span>
```

실습으로 ~

Soup.findAll('span')
returns

```
<span class="excitingNote">first line!</span>  
<span class="excitingNote">second line!</span>  
<span class="excitingNote">third line!</span>  
<span class="excitingNote">fourth line!</span>
```

Unit 02 | BeautifulSoup

• 텍스트 외 정보 추출하기 (Attribute)

Attribute

```
<span class="excitingNote" color="black"  
>first line!</span>
```

```
<a href="/moviedb/main?movieId=93252"  
class="name_movie #title">어벤져스:  
엔드게임</a>
```

Unit 02 | BeautifulSoup

• 텍스트 외 정보 추출하기 (Attribute)

- 동일하게 find 또는 findAll 을 이용
- 위 함수가 return 하는 bs4.element.Tag 객체에 마치 **딕셔너리 key** 값을 호출하듯이 이용
- bs4.element.Tag[attribute_name]

```
<a  
href="/moviedb/main?movieId=93252"  
class="name_movie #title">어벤져스:  
엔드게임</a>
```

```
response = requests.get(my_url)
```

```
soup = bs(response.text, 'html.parser')
```

```
Movie_url = soup.find('a')['href']
```

Unit 02 | BeautifulSoup

• 부모, 형제 관계 이용하기

- HTML 문서는 구조적으로 작성이 가능하고
- 한 태그안에 다른 태그가 속할 수 있다.
- bs4.element.tag.**next_sibling** : 다음 형제태그

parent : 부모 태그

descendants : 모든 자식 태그

등등 생각할 수 있는건 다 있음

부모, 자식 관계

```
<hr class="hide">
▼ <div id="dkContent" class="cont_movie" role="main">
  <h2 id="dkBody" class="screen_out">박스오피스 본문</h2>
  ▼ <div id="cMain">
    ▼ <div id="mArticle">
      <strong class="screen_out">기간 선택</strong>
      ▶ <div class="wrap_menu">...</div>
      <h3 class="screen_out">
        연간
      </h3>
      <!-- 메뉴 선택에 따라 텍스트 변경 필요 -->
      ▶ <div class="wrap_calendar">...</div>
      ▼ <ul class="list_movie #movie">
        ▶ <li>...</li>
        ▼ <li>
```

형제 관계

Unit 02 | BeautifulSoup

- 정규식 이용하기 (Regular Expression)

- 정규식은 정규화된 즉, 패턴이 있는 텍스트 데이터를 추출하는 데 사용된다.
- 가령, 010-9292-2929 같은 전화번호는 **(3개의 숫자)-(4개의 숫자)-(4개의 숫자)** 으로 정규화 되어 있으므로 `(\Wd[3])-(\Wd[4])-(\Wd[4])` 의 정규식 (regex) 으로 추출 가능
- re 라이브러리를 이용하고 `re.compile(my_regex)` 으로 표현 함
- 정규식이 익숙하지 않으면 몰라도 된다! (사용이 가능하다는것만 알고있자)**
- 실습!**

Unit 02 | BeautifulSoup

• 실제 크롤링 팁

- 내가 찾고 싶은 정보가 어떤 태그에 존재하는지 확인
- 반복적으로 찾을 수 있는 정보인지 확인 (패턴이 보이는지 확인)
- URL 에서 패턴이 보이는지 확인
- URL 을 바꿔가면서 내가 찾고 싶은 태그를 찾아주면 된다.
- 예시로 확인해봅시다! (<https://movie.daum.net/boxoffice/yearly>)

Unit 02 | BeautifulSoup

• 예외 처리 (Try - Except)

- 크롤링 중 많은 과정에서 오류가 일어날 수 있음
- 인터넷 끊긴다거나, 내가 찾고 있는 태그가 없다거나... 등등
- 이런 오류가 발생하면 **중간에 코드가 멈추므로** 그동안 크롤링한 데이터가 날라갈 수도 있으므로 try-except 문을 이용

```
If response.status_code != 200:
```

```
    에러문 발생 (print('Error!'))
```

```
    break
```

```
try :
```

```
    사이트 요청 시도
```

```
    태그 찾기
```

```
except Exception as e:
```

```
    print(e)
```

Unit 02 | BeautifulSoup

• 예외 처리 (Try - Except)

- 오류가 일어날만한 부분들에 개별적으로 적용하는 것이 가장 이상적
- 그래야지 어떤 오류가 발생했는지 알 수 있다.
- Except 문에는 항상 예외를 어떻게 처리할 것인지 명시

```
data = []  
cnt = 0  
for movie in movie_table:  
    try :  
        data.append(movie.find('가끔 가다 없는 태그').text)  
    except Exception as e:  
        data.append('NA')  
        print(e)  
        print('Error occurred at 가끔가다 없는 태그 :' cnt)  
    Cnt+= 1
```


Contents

Unit 01 | 소개

Unit 02 | Beautiful Soup

Unit 03 | 실용적 팁 (다이나믹 페이지 / API 이용)

Unit 04 | 과제 설명

Unit 03 | 실용적 팁

- 지금까지는 **정적**인 (Static) 페이지만 다뤘지만 사이트 미관/사용자 편의를 위해 Javascript 를 통해 **동적**으로 (Dynamic) 만들어진 페이지도 있음.
- 이 경우, **유저 움직임에** 따라 HTML 을 전송받는다.
- 예시 : 인스타그램
- 스크롤을 해야지 더 많은 게시글을 볼 수 있다. **브라우저가 아닌 파이썬에서 웹을 접속하는 우리에게**는 큰 문제!

Unit 03 | 실용적 팁

- 동적인 페이지 크롤링

- 해결책 1 : 브라우저를 이용하자!
- Selenium : 실제 브라우저를 생성해서 파이썬 커맨드로 **마우스/키보드를** 조정할 수 있음
- PhantomJS : Headless browser (머리없는?? 브라우저)
 - **눈에 보이지는 않지만** 브라우저 형태를 실행시켜 Selenium 과 동일하게 작동
- 둘 다 **느리지만**, 크롤링을 가능!

Unit 03 | 실용적 팁

• Selenium

- 실제 브라우저가 생성되어 코드가 제대로 작동되는지 쉽게 확인 가능
- 태그 정보 뿐만 아니라 **CSS selector, xpath** 등 여러가지로 원하는 element 찾기 가능
 - 소스코드에서 우클릭 후 선택가능
- Find_element 를 복수형으로 쓰면 (find_elements) BeautifulSoup 의 findAll 과 동일
- 반환되는 건 Selenium 객체이지만 위에서 본 bs4.Tag 와 유사하다
- 다만, 호출되는 방법이 약간씩 다르니 [documentation](#) 참고!
- 실습

```
find_element_by_id  
find_element_by_name  
find_element_by_xpath  
find_element_by_link_text  
find_element_by_partial_link_text  
find_element_by_tag_name  
find_element_by_class_name  
find_element_by_css_selector
```

Unit 03 | 실용적 팁

• 해결책 2: 다른 능력자들의 도움 (구글링 또는 깃허브)

huaying/instagram-crawler: Get Instagram posts/profile ... - GitHub

<https://github.com/huaying/instagram-crawler> ▼ 이 페이지 번역하기

Get Instagram posts/profile/hashtag data without using Instagram API - huaying/instagram-crawler.

Topic: instagram-crawler · GitHub

<https://github.com/topics/instagram-crawler> ▼ 이 페이지 번역하기

GitHub is where people build software. More than 40 million people use GitHub to discover, fork, and contribute to over 100 million projects.

timgrossmann/instagram-profilecrawl: quickly crawl the ... - GitHub

<https://github.com/timgrossmann/instagram-profilecrawl> ▼ 이 페이지 번역하기

quickly **crawl** the information (e.g. followers, tags etc...) of an **instagram** profile. - timgrossmann/instagram-profilecrawl.

261 repository results

Sort: Best match ▼

iammrhelo/InstagramCrawler Archived

Python

★ 312

A non API python program to *crawl* public photos, posts or followers

instagram selenium python crawler

MIT license Updated on 3 May

huaying/instagram-crawler

Python

★ 227

Get *Instagram* posts/profile/hashtag data without using *Instagram* API


instagram instagram-crawler likers python

MIT license Updated 9 days ago

timgrossmann/instagram-profilecrawl

Python

★ 569

 quickly *crawl* the information (e.g. followers, tags etc...) of an *instagram* profile.

crawler python instagram automation

MIT license Updated on 3 Jul

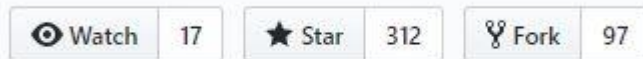
Unit 03 | 실용적 팁

• 깃허브 이용시 주의사항!

• 이용 전, Readme.md 확인!

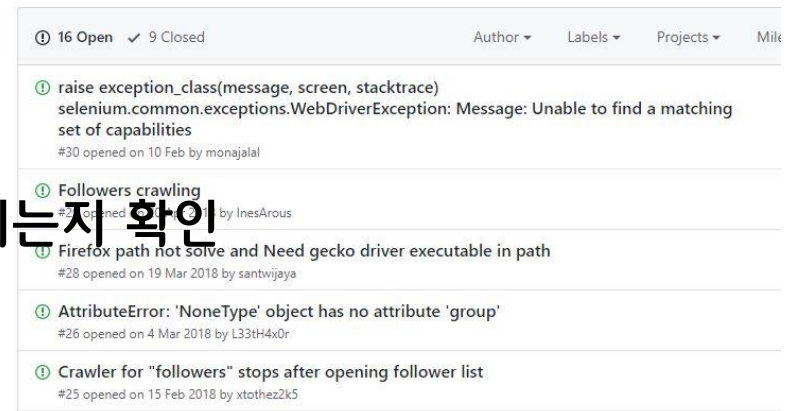
- Repo의 사용설명서: 본 repo 의 목적, 필요사항, 설치법 등 모든게 적혀있음
- 내가 필요한 정보를 크롤링할 수 있는지 가장 먼저 확인

• 별의 개수 확인!



• Issue 를 확인!

- 문의사항들이 올라오니 어떤 문제점이 있는지, 답변을 잘 달리는지 확인



Unit 03 | 실용적 팁

- 해결책 3: 제공되는 API 가 존재하는지 확인
 - 드물지만 큰 사이트 경우, 크롤링을 위해 공식적으로 제공되는 API 가 존재하기도 합니다.
 - 대부분 트래픽 제한이 있고, 사용법이 제한적이지만 불가피하게 사용할 때도 있습니다.
 - 대표적인 예가 네이버
 - 검색결과는 크롤링 가능하지만 블로그 내에서는 불가
 - 뉴스 기사, 댓글 등은 가능
 - 다만, 아주 쉽게 차단 됨

Unit 03 | 실용적 팁

- 네이버 API 사용법

- <https://developers.naver.com/>
 - Application > 어플리케이션 등록 > Client ID 와 비밀번호 발급 뒤 사용 가능
- 헤더로 발급받은 ID 와 비밀번호를 전송한 뒤 접속 가능

```
request.add_header( "X-Naver-Client-Id" ,client_key)
```

```
request.add_header( "X-Naver-Client-Secret" ,client_secret)
```

- 하루 1000 건 제한 ☹
- 참고: <https://brunch.co.kr/@sukhyun9673/13>

Unit 03 | 실용적 팁

• 마지막 실용적 팁

- Try-except 문 적극 활용하기
 - 밤새 돌려놓은게 에러 때문에 멈춰있다면... 😞

• 크롤링 된 데이터 입출력:

- .csv .json 으로 저장하기 (pandas 를 활용하면 쉬워요)

• 참고:

<http://blog.naver.com/PostView.nhn?blogId=kiddwannabe&logNo=221274278923&parentCategoryNo=&categoryNo=35&viewDate=&isShowPopularPosts=false&from=postView>

• 차단 방지를 위해 중간중간 크롤러를 쉬자

- e.g. sleep(randint(3,5)) : 3-5초간 정지
 - from time import sleep
 - from random import randint, uniform

Unit 03 | 실용적 팁

- 오늘 다루지 못했지만 중요한 것들:
 - Scrapy : BeautifulSoup 보다 많이 사용되는 라이브러리
 - 크롤러를 class 로 구현하는 것이 특징 (따라서 입문하기에는 어려움)
 - 빠르고, 확장성이 좋아 큰 프로젝트에 유용
 - 깃허브에 공개된 것은 대부분 Scrapy 를 이용
 - Multiprocessing (멀티쓰레딩) : 여러 작업을 세분화시켜 동시에 하는 일
 - HTML 파싱 작업이 오래 걸리면 아주 유용하나
 - 요청작업이 오래 걸린다면 큰 도움이 되지 못한다

Contents

Unit 01 | 소개

Unit 02 | Beautiful Soup

Unit 03 | 실용적 팁 (다이나믹 페이지 / API 이용)

Unit 04 | 과제 설명

Unit 04 | 과제 1

• 과제 1 : 머리 식히기

Selenium 을 이용해서 인스타그램의 검색결과를 크롤링한다고 생각하고 pseudocode 를 작성해보세요.

- 필요한 정보: 5000개의 게시글. 게시글 안에는 게시글 내용, 게시자 계정 날짜, 댓글단 계정, 댓글 내용
- Pseudocode 인만큼 정확한 태그 이름 등은 신경쓰실 필요 없습니다. 다만 실제로 크롤링한다고 생각하고 코드 한 줄이 하는 작업을 자세히 명시해주세요.
- 브라우저로 소스코드를 실제로 어떻게 처리되는지 보시는게 도움될 겁니다 ☺
- 파이썬 스타일로 코드를 작성하시면 됩니다.

Unit 04 | 과제 2

- 과제 2 : 다음 영화 사이트를 다시 크롤링해보세요.

(<https://movie.daum.net/boxoffice/yearly?year>)

- 필요한 정보는 2개 년도의 [제목, 네티즌 평점, 평론가 평점, 누적 관객수] 입니다.
(2019년, 2018년)
- 누락 데이터는 적절하게 처리해주시면 됩니다. ('NA', '없음' 등등)
- 크롤링이 된 데이터는 (100개 영화에 대한 정보) csv / json 으로 제출해주세요!
- 채점기준 :
 - 모듈화가 잘 되었는가 (각 함수의 역할이 명확한가)
 - End-to-end (한 번의 실행으로 모든 작업이 이뤄지는가)

Reference

- <도서>

Web Scraping with Python: Collecting Data from the Modern Web, Ryan Mitchell

Web Scraping with Python: Scrape data from any website with the power of python, Richard Lawson

- <웹사이트>

<https://ko.wikipedia.org/wiki/HTML>

Q & A

들어주셔서 감사합니다.