



## 웹 크롤링 - 데이터 수집

## 쉽게 배워 제대로 활용하기!

### 1. 단순 반복을 줄여주고 빠르게 처리하는 웹 크롤링!

웹 크롤링은 단순 반복을 줄여주고 빠르게 처리할 수 있으며, 일의 효율을 높여주고 인간이 좀 더 창의적인 활동에 집중할 수 있도록 도와주는 좋은 도구이다.

### 2. 초보자도 해볼 수 있는 수준!

꼭 알아야 할 핵심 개념은 기본 예제로 최대한 쉽게 설명하여 처음 배우는 분들도 쉽게 해볼 수 있는 수준이다.

### 3. 실습 환경

- ✓ 파이썬 문법으로 작성한 소스코드를 실행하는 **파이썬3**
- ✓ 파이썬을 개발하는 도구인 **파이참**
- ✓ 크롤러를 만들 때 크롬에 있는 개발자 도구를 활용하기 위한 **크롬(Chrome)**

## 1장. 웹 기초

1절. HTTP

2절. URL

3절. HTML

## 2장. 크롤러 만들기

1절. 개발환경 설치하기

2절. urllib 패키지

3절. 뷰티풀썬 사용 방법

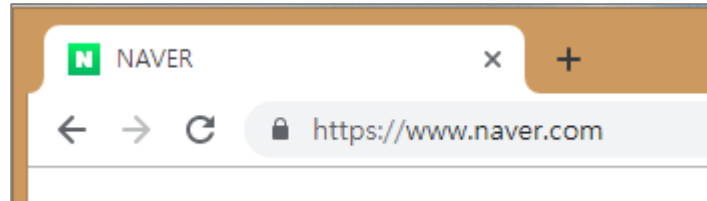
4절. 네이버에서 특정 데이터 가져오기

5절. 네이버 메뉴 이름 뽑아내기

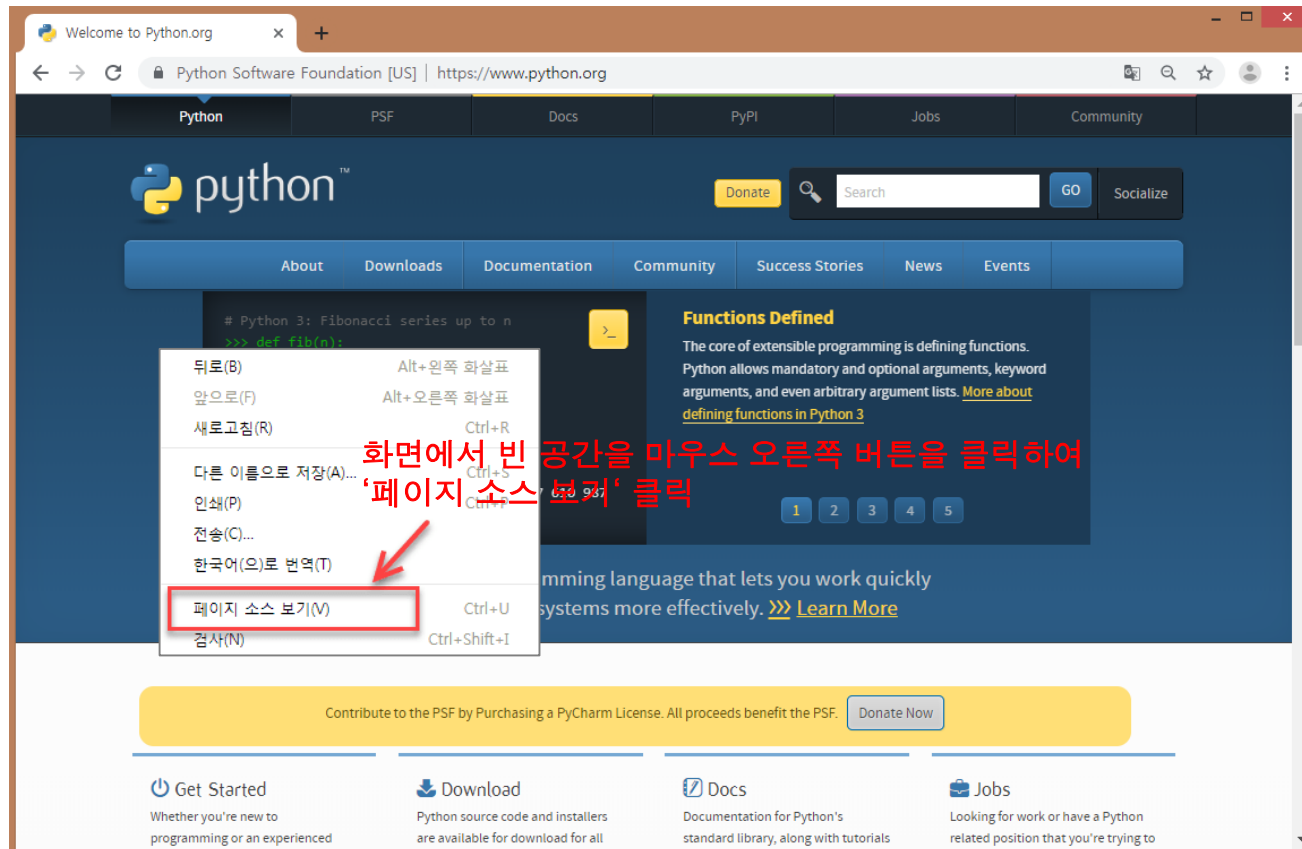
6절. 네이버 뉴스 제목 가져오기

- HTTP는 Hyper Text Transfer Protocol의 약자
  - '하이퍼 텍스트'는 마우스로 클릭하면 다른 페이지로 이동하는 기능을 말함.
  - HTTP는 HTML로 작성되어 있는 하이퍼 텍스트를 전송하기 위한 프로토콜임.
- 프로토콜은 '약속, 규칙, 규약'이라는 뜻
  - A라는 지점에서 B라는 지점으로 데이터를 보낼 때 어떻게 보낼 것인지에 대한 규칙임.
  - 인터넷 주소를 쓸 때 앞에 <http://www.google.com> 처럼 http를 붙임.

- URL(Uniform Resource Locator)은 인터넷 주소
  - 보통 웹 브라우저에서 주소 표시줄에 나오는 주소를 의미함.
  - 정확히는 네트워크상에서 자원의 위치를 알려주는 주소임.
  - URL이 <https://www.naver.com> 일 때, https 라는 규약(프로토콜)으로 [www.naver.com](https://www.naver.com)라는 주소의 실제 위치에 접속해서 정보를 가져오겠다는 의미임.

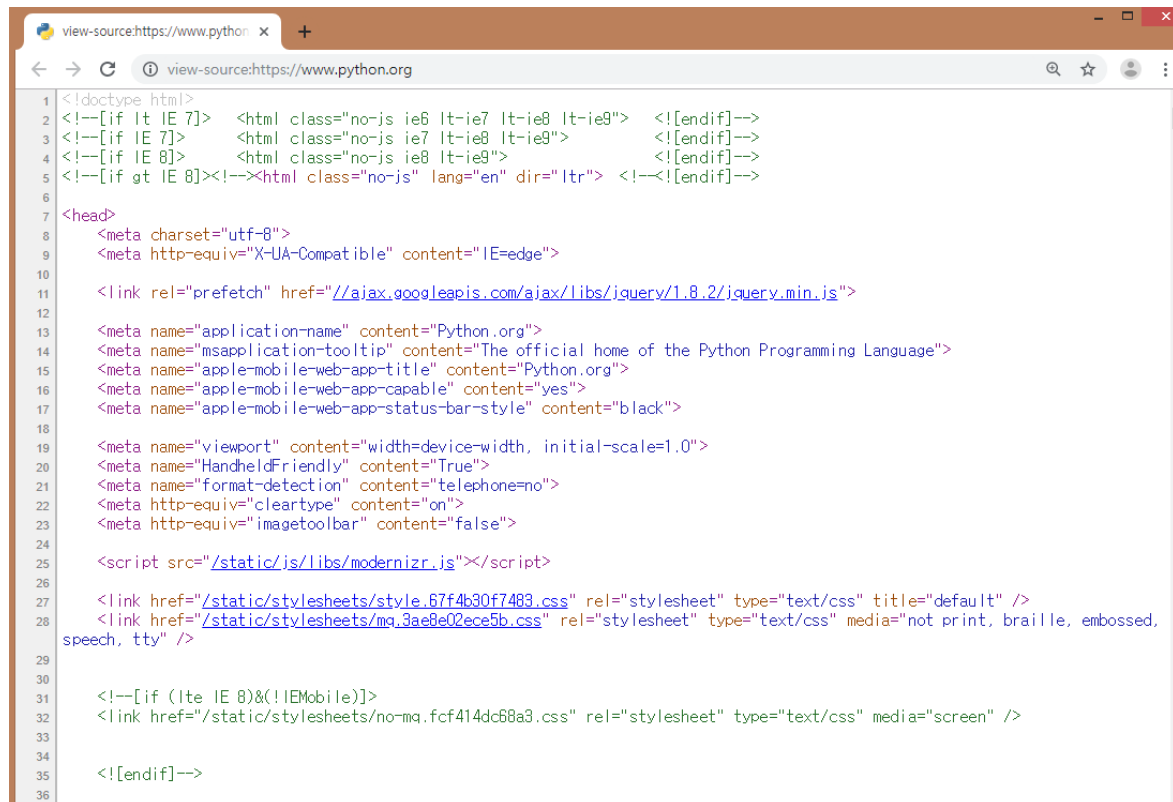


- HTML은 Hyper Text Markup Language의 약자
  - 마크업 언어라는 것은 일종의 문법임.
  - 우리가 검색하고 방문하는 웹 페이지들은 웹 페이지를 작성하기 위한 어떤 문법에 의해서 작성되는 데 , HTML도 문법들 중 한 종류임.



### ■ 웹 페이지 소스코드

- HTML 역시 웹 프로그램을 만들기 위한 프로그래밍 언어임.
- 글씨, 이미지, 웹 페이지에서 연결하려는 특정 주소 등 다양한 정보를 담으며 태그(tag)와 함께 사용됨.
- 태그는 <head> <link> <html> 등과 같이 사용하는 일종의 약속된 키워드임.

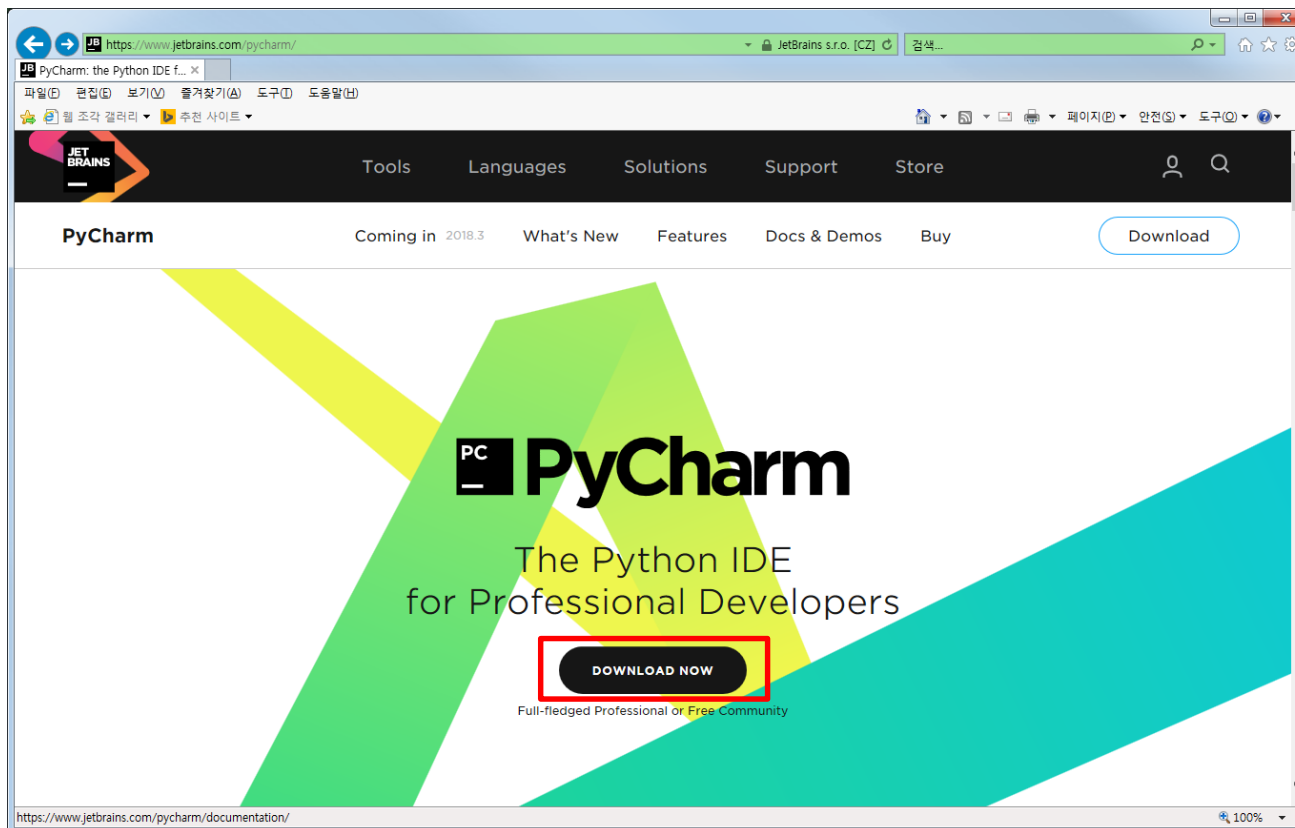


```
1 <!doctype html>
2 <!--[if lt IE 7]> <html class="no-js ie6 lt-ie7 lt-ie8 lt-ie9"> <![endif-->
3 <!--[if IE 7]> <html class="no-js ie7 lt-ie8 lt-ie9"> <![endif-->
4 <!--[if IE 8]> <html class="no-js ie8 lt-ie9"> <![endif-->
5 <!--[if gt IE 8]><!--><html class="no-js" lang="en" dir="ltr"> <!--<![endif-->
6
7 <head>
8 <meta charset="utf-8">
9 <meta http-equiv="X-UA-Compatible" content="IE=edge">
10
11 <link rel="prefetch" href="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js">
12
13 <meta name="application-name" content="Python.org">
14 <meta name="msapplication-tooltip" content="The official home of the Python Programming Language">
15 <meta name="apple-mobile-web-app-title" content="Python.org">
16 <meta name="apple-mobile-web-app-capable" content="yes">
17 <meta name="apple-mobile-web-app-status-bar-style" content="black">
18
19 <meta name="viewport" content="width=device-width, initial-scale=1.0">
20 <meta name="HandheldFriendly" content="True">
21 <meta name="format-detection" content="telephone=no">
22 <meta http-equiv="cleartype" content="on">
23 <meta http-equiv="imagetoolbar" content="false">
24
25 <script src="/static/js/libs/modernizr.js"></script>
26
27 <link href="/static/stylesheets/style.67f4b30f7483.css" rel="stylesheet" type="text/css" title="default" />
28 <link href="/static/stylesheets/mq.3ae8e02ece5b.css" rel="stylesheet" type="text/css" media="not print, braille, embossed,
29 speech, tty" />
30
31 <!--[if (lte IE 8)&(!IEMobile)]>
32 <link href="/static/stylesheets/no-mq.fcf414dc68a3.css" rel="stylesheet" type="text/css" media="screen" />
33
34
35 <![endif-->
36
```

### ■ 파이참 설치하기

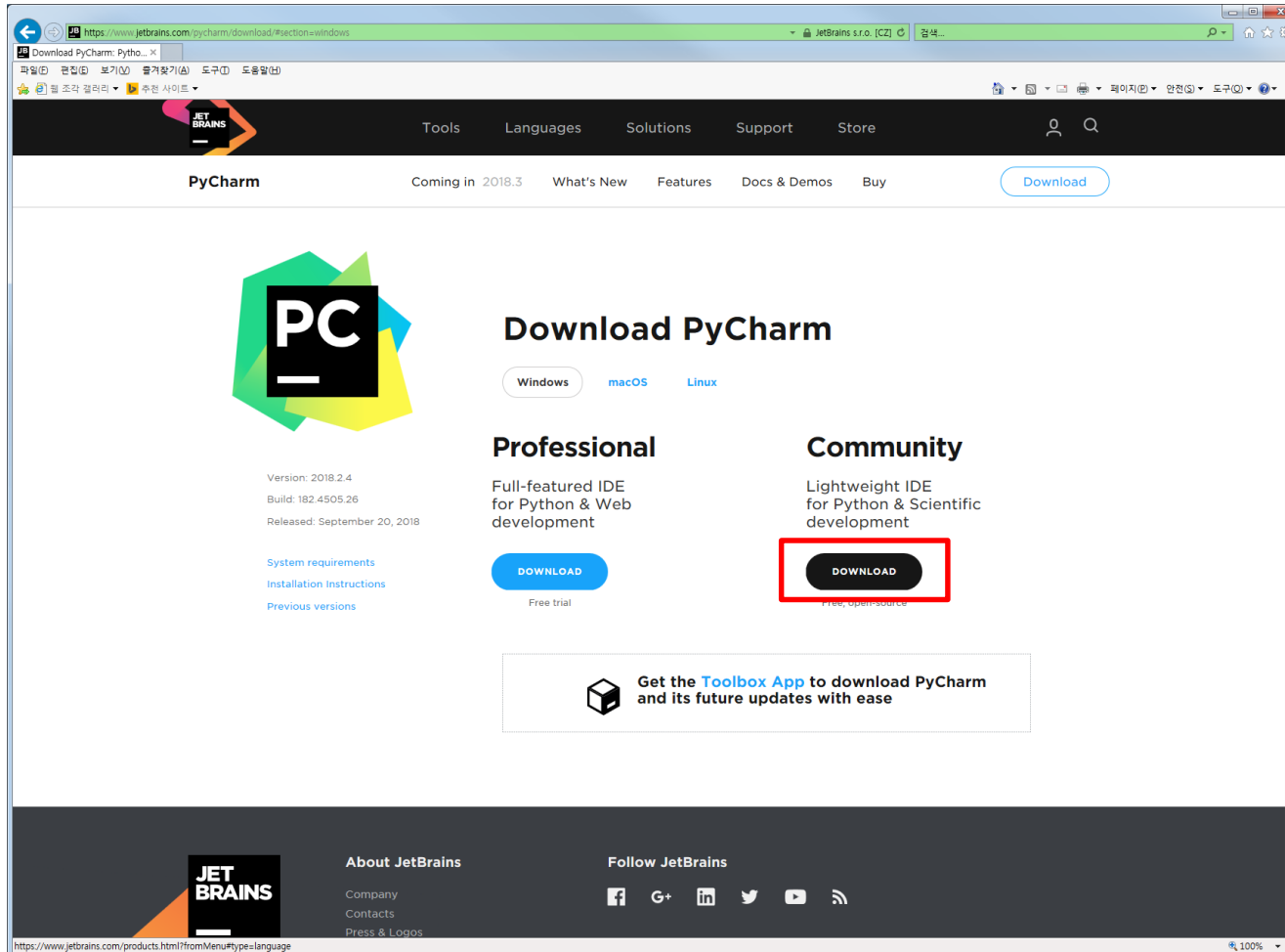
파이썬을 개발하는 도구로 편한 '라이브러리 설치' 기능과 코딩 소스코드를 매끄럽게 작성할 수 있게 도와주는 '자동 완성 기능' 등이 있음.

- 파이참 홈페이지([www.jetbrains.com/pycharm/](https://www.jetbrains.com/pycharm/))에서 Download를 클릭함

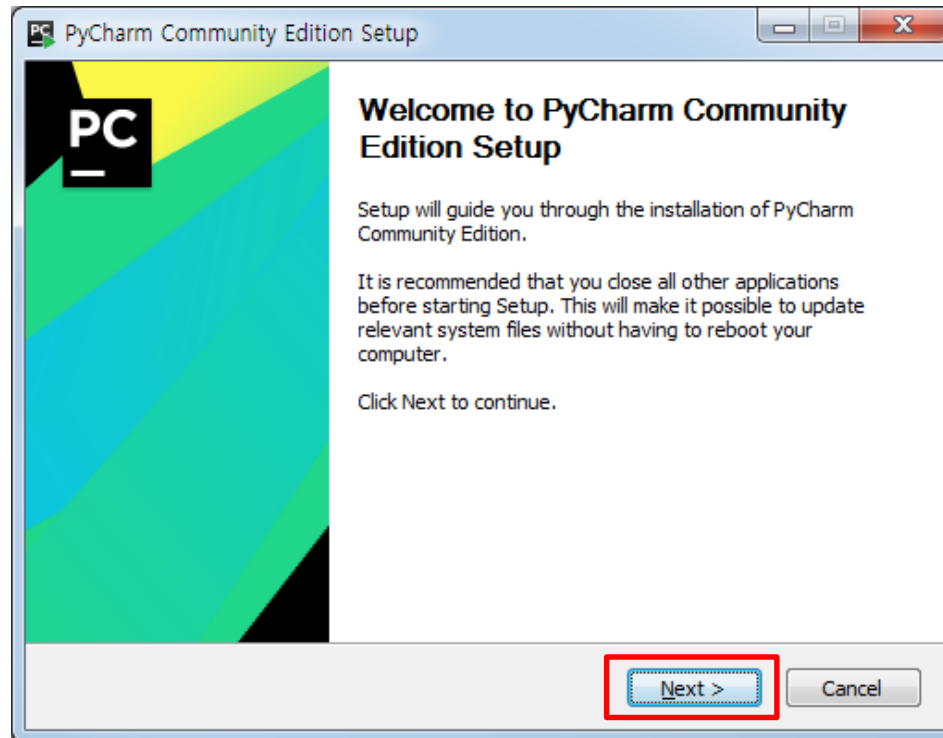




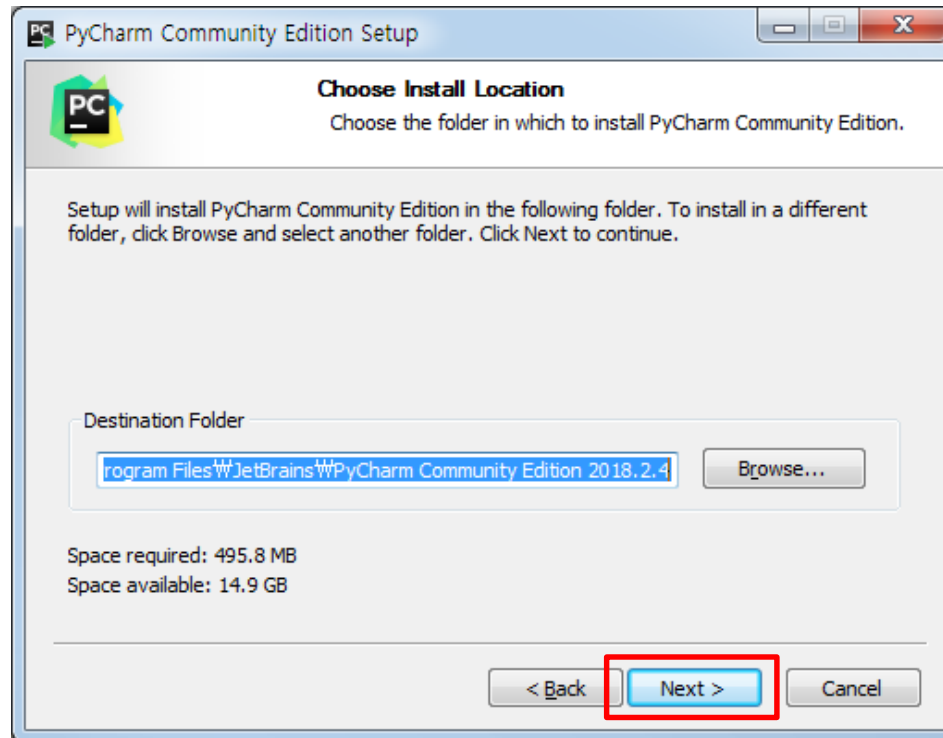
- Community의 Download를 클릭하여 다운로드함



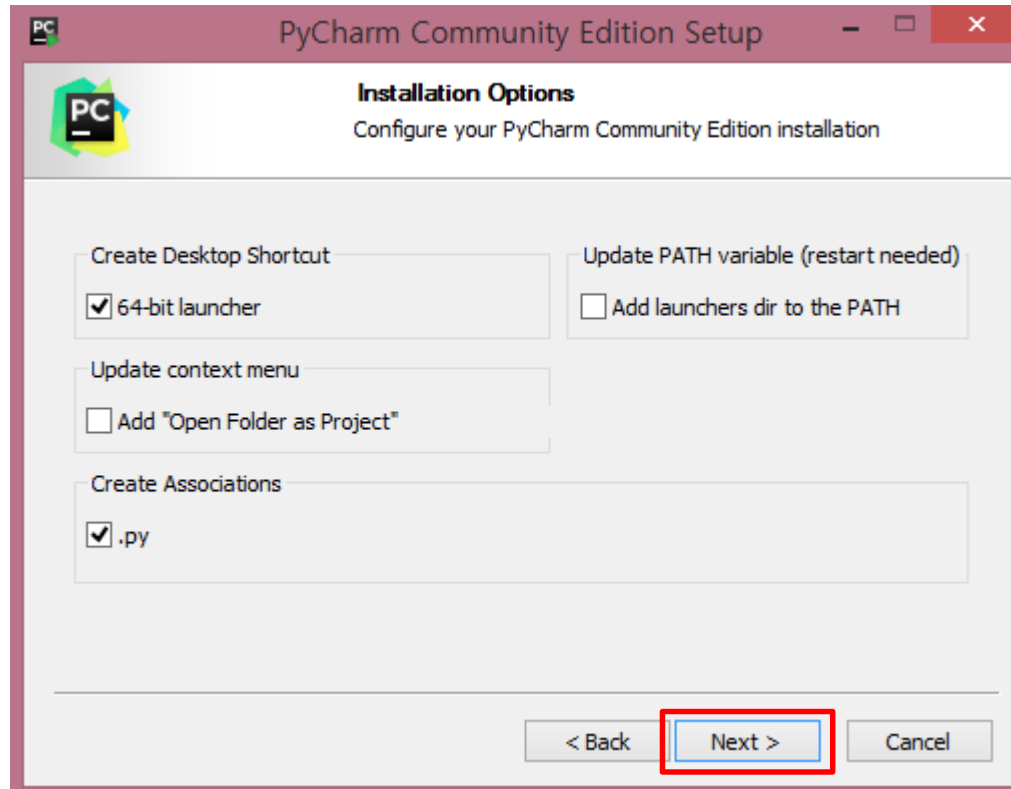
- Next 버튼 클릭하여 설치 파일 설치하기



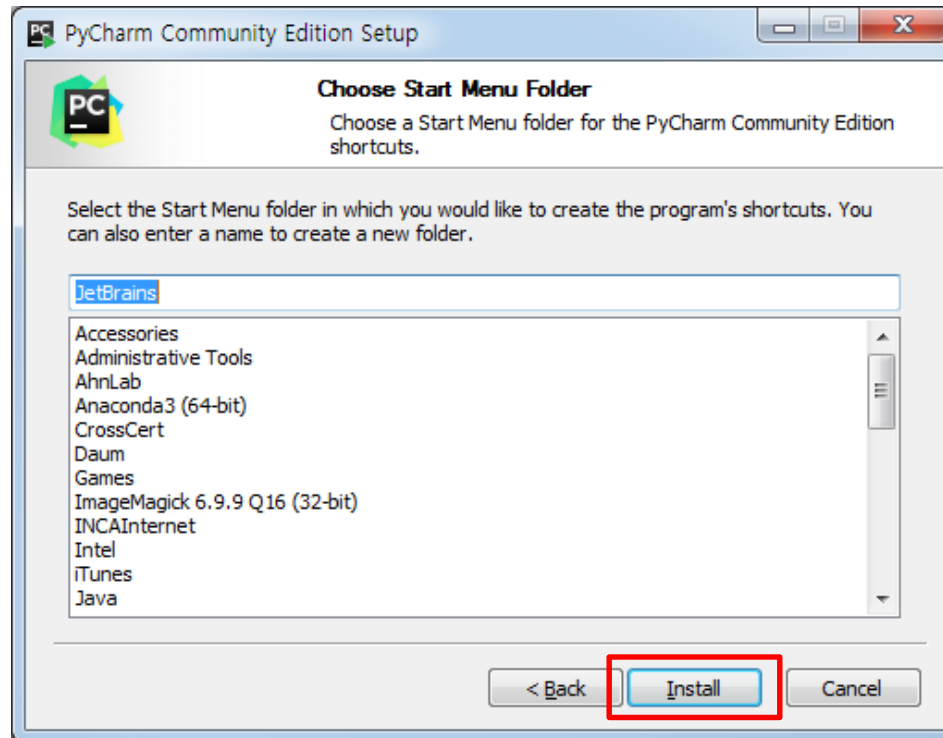
- Next 버튼 클릭하여 파이참 설치 경로 지정하기



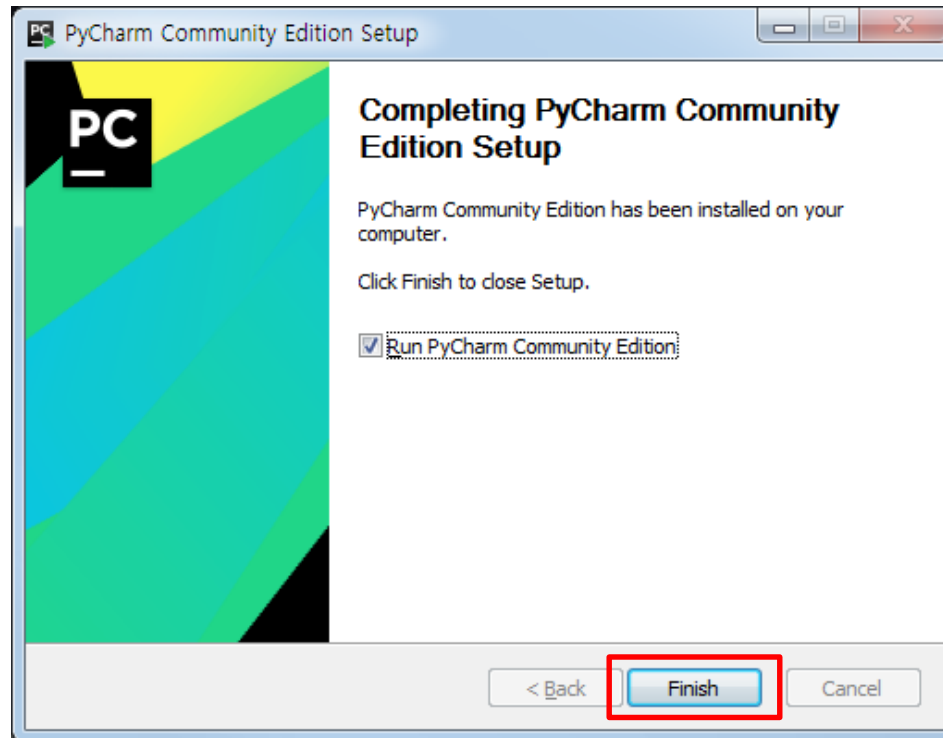
- '64-Bit launcher'와 '.py' 옵션 설정 후 Next 버튼 클릭



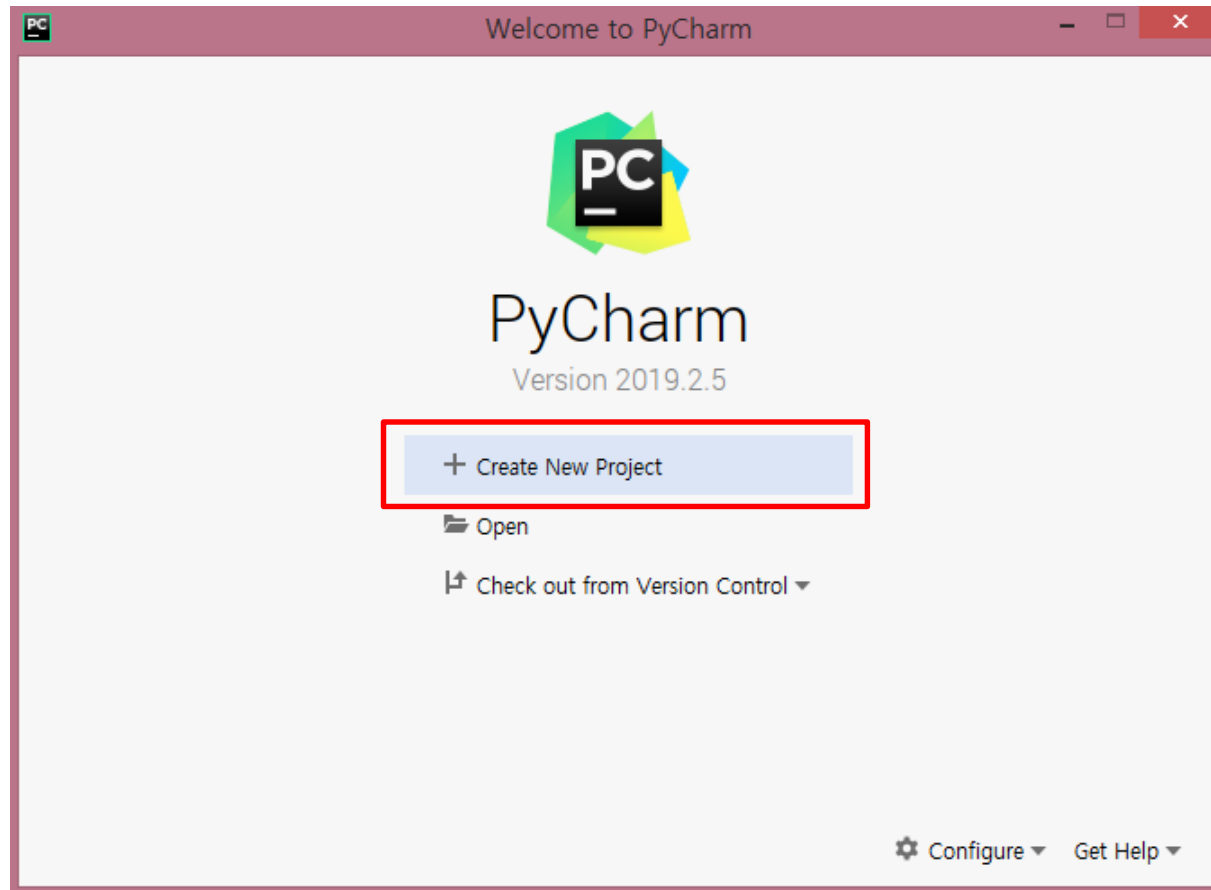
- 파이참을 시작 메뉴에 추가하기 위해 Install 버튼 클릭



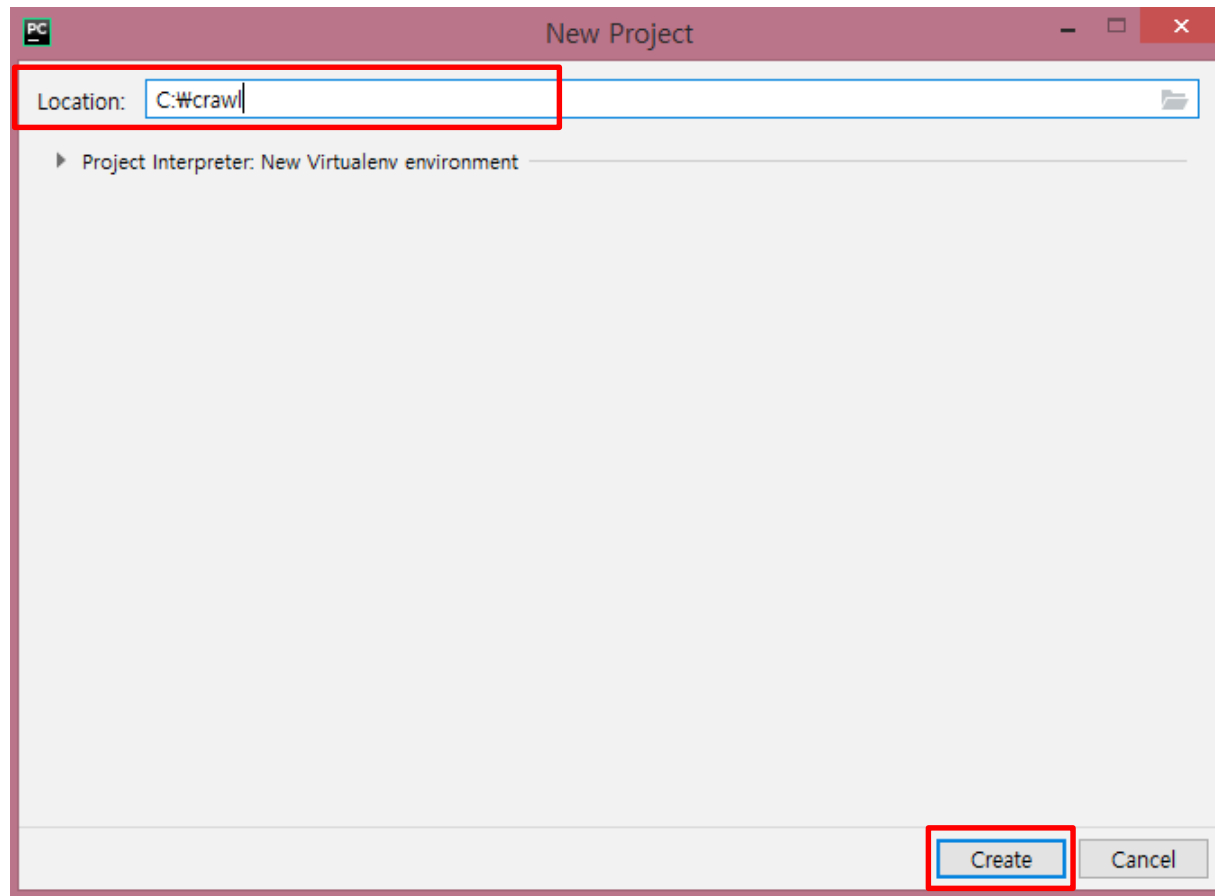
- Finish 버튼 클릭하여 설치완료



- 프로젝트 만들기
  - Create New Project을 클릭함

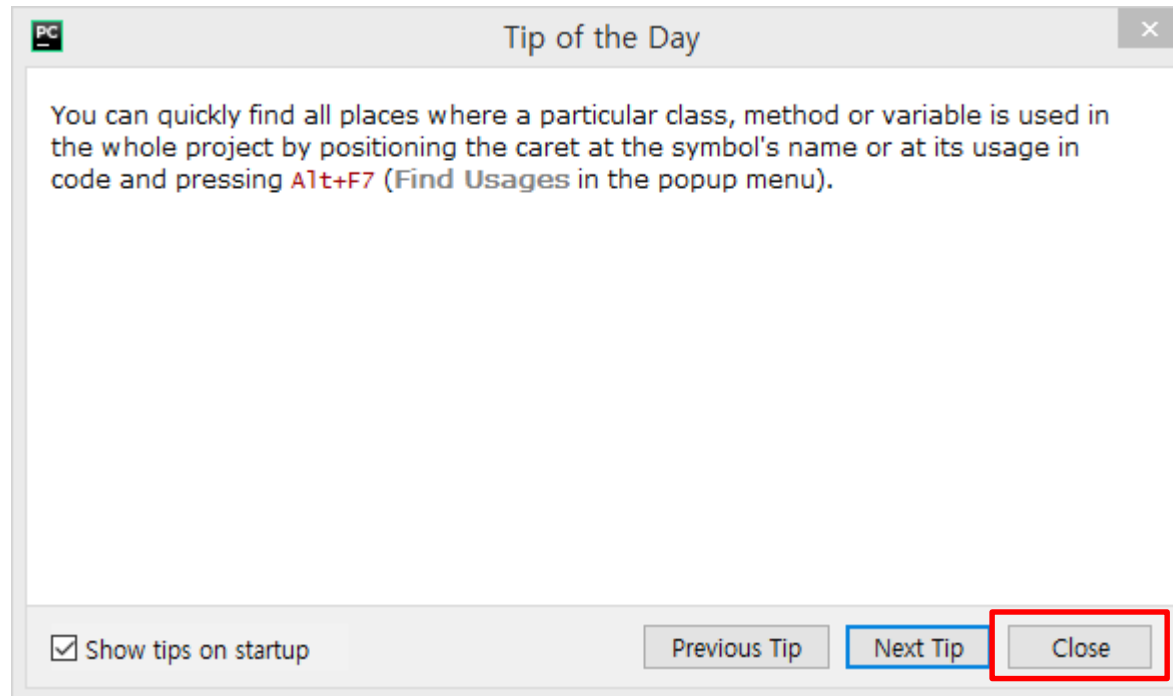


- 프로젝트가 생성될 위치 지정하고, Create 버튼 클릭
  - 'C:\crawl' 에 프로젝트를 만듦

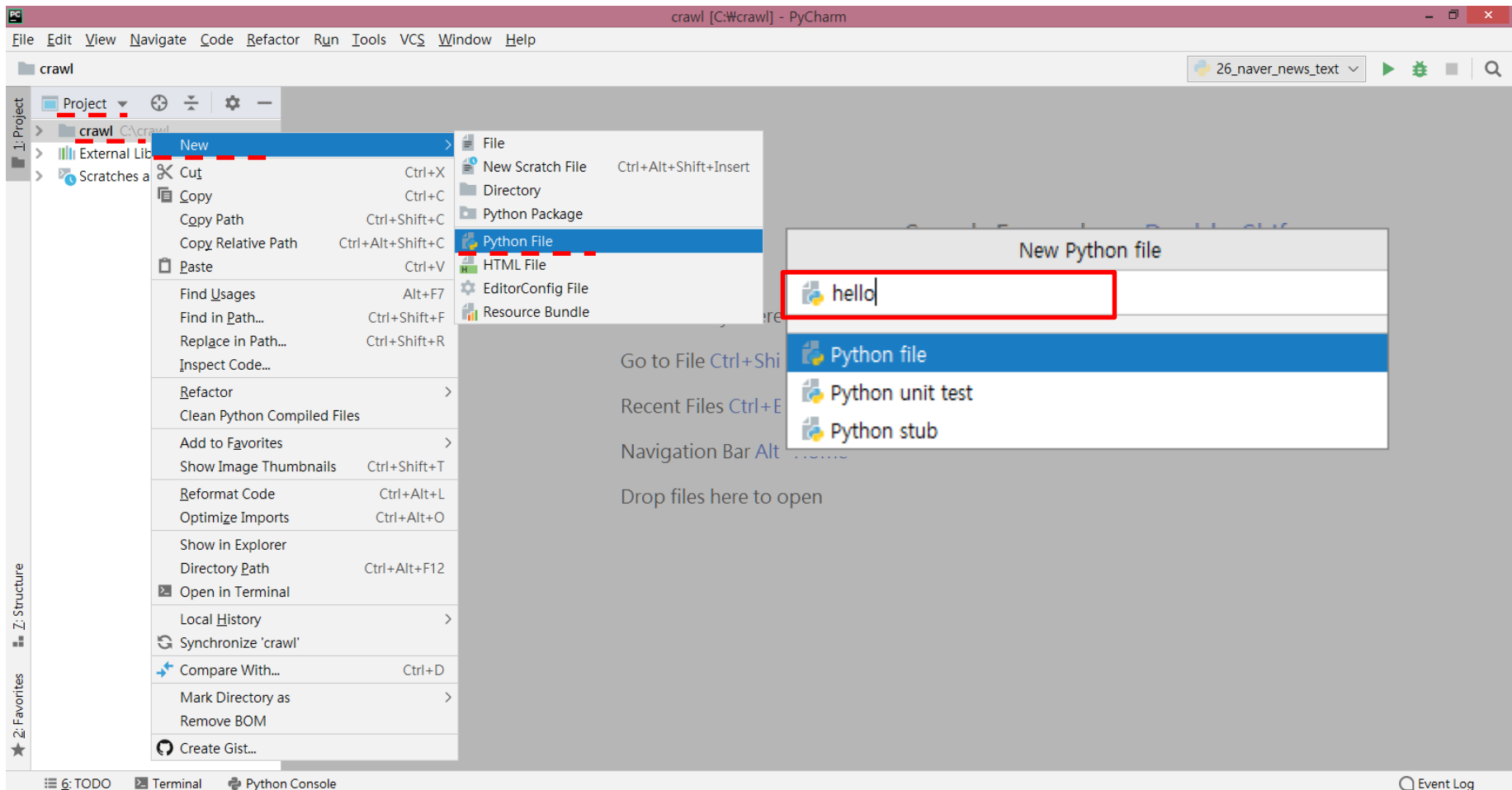




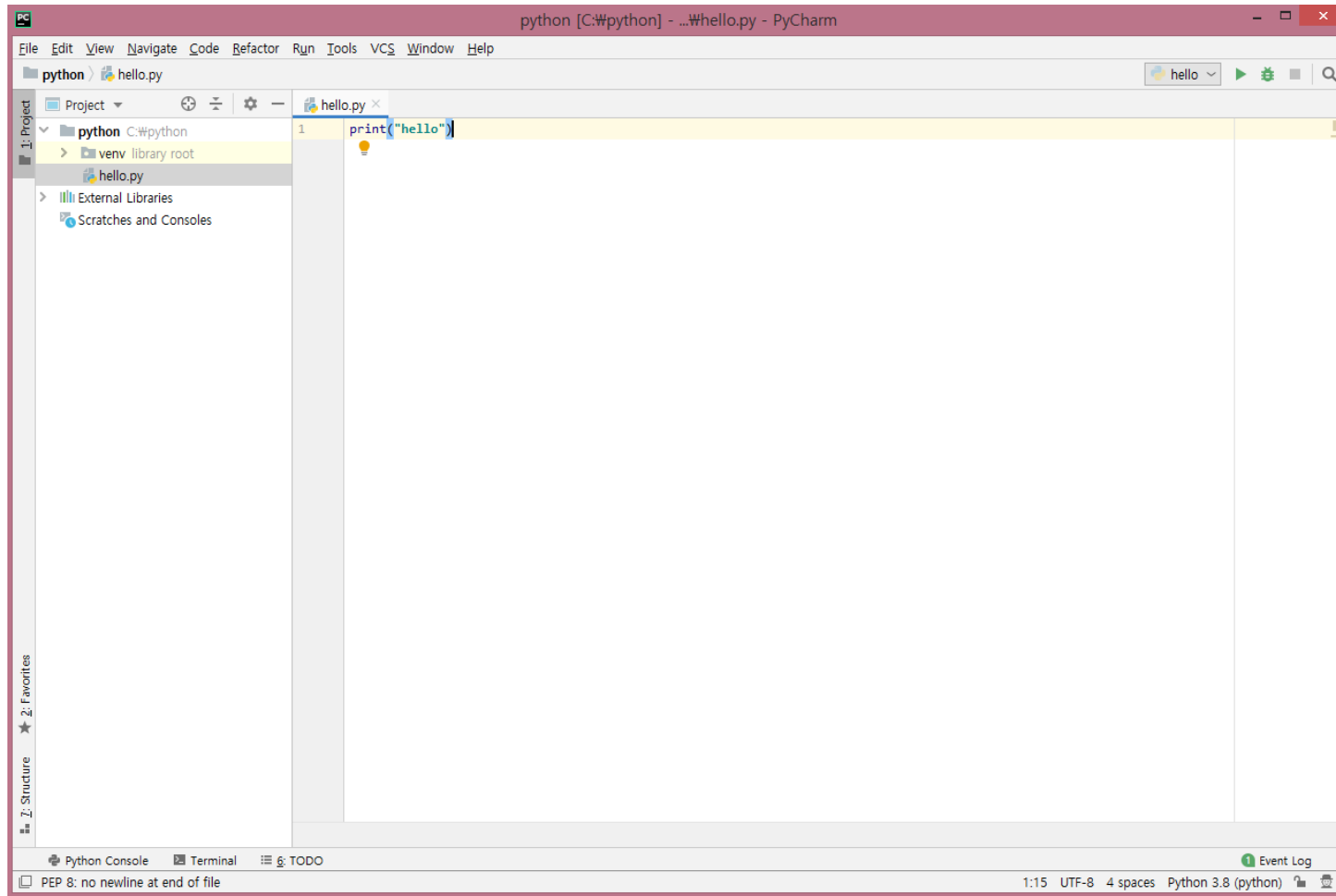
- 'Close' 버튼 클릭



- “hello” 출력하는 프로그램 작성하기
  - Project → crawl → New → Python File 클릭 후, 프로그램명(hello) 입력

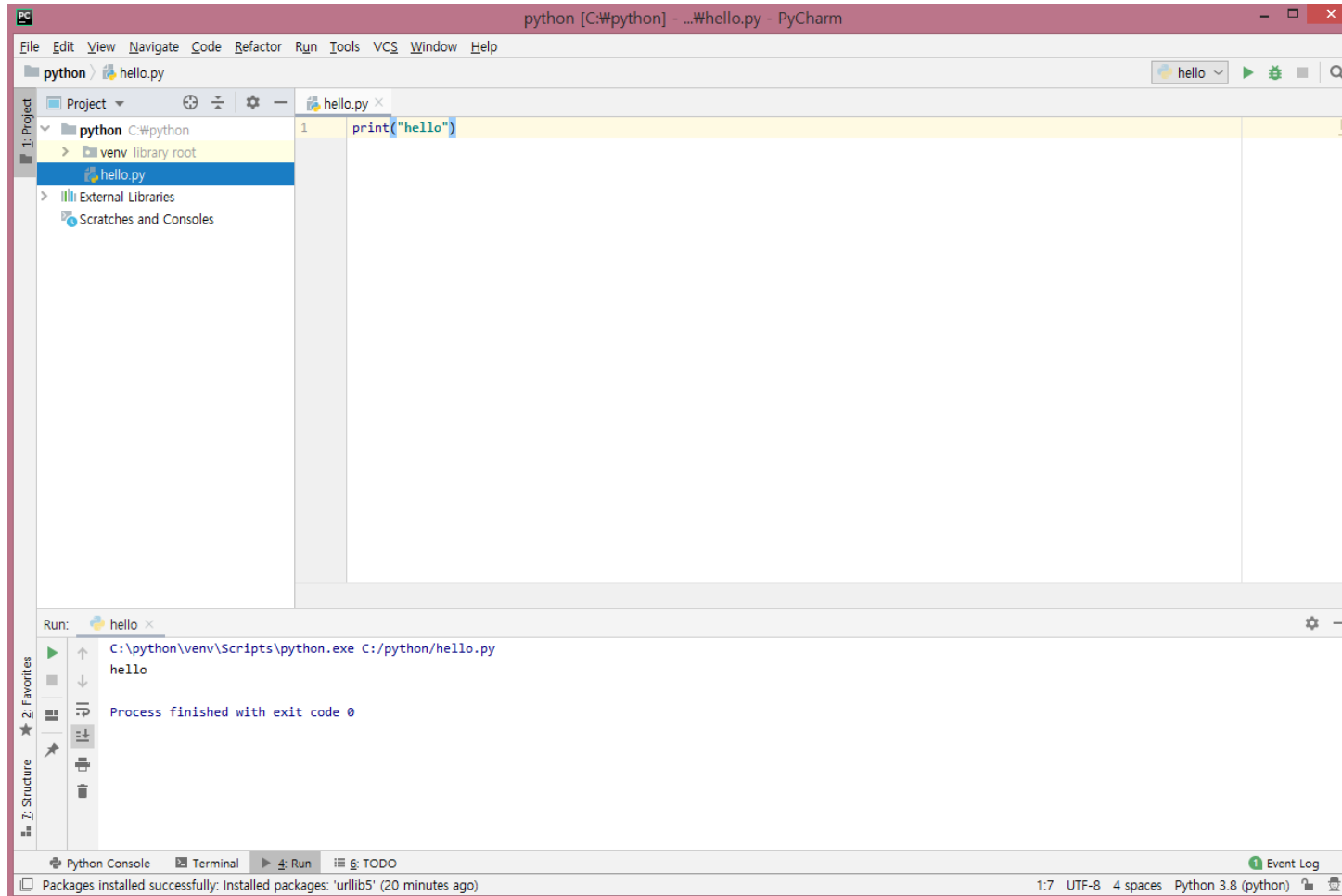


- 소스코드 작성하기



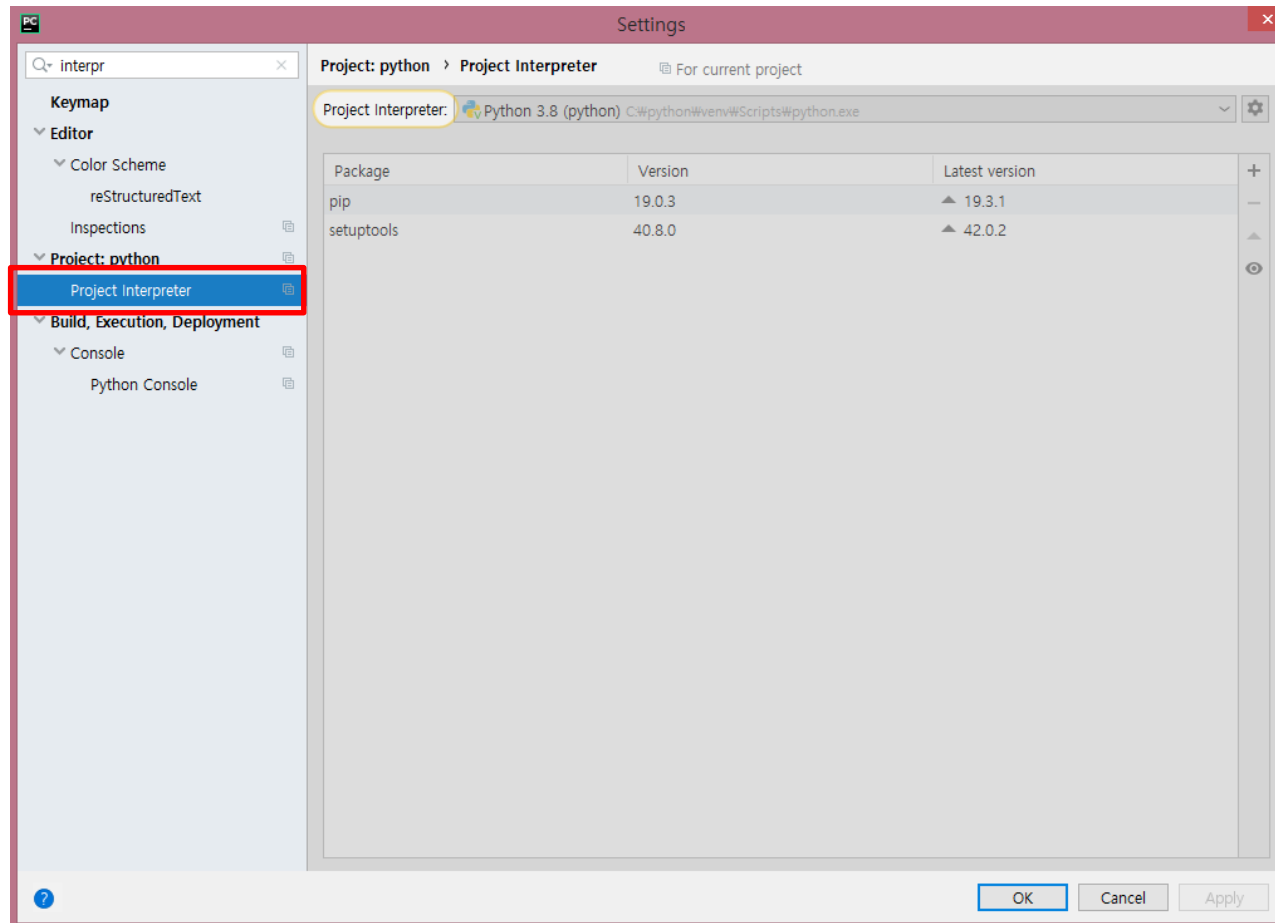
- 파이썬 프로그램 실행하기

- 상단의 메뉴에서 [Run → Run...]을 눌러서 실행할 수 있음

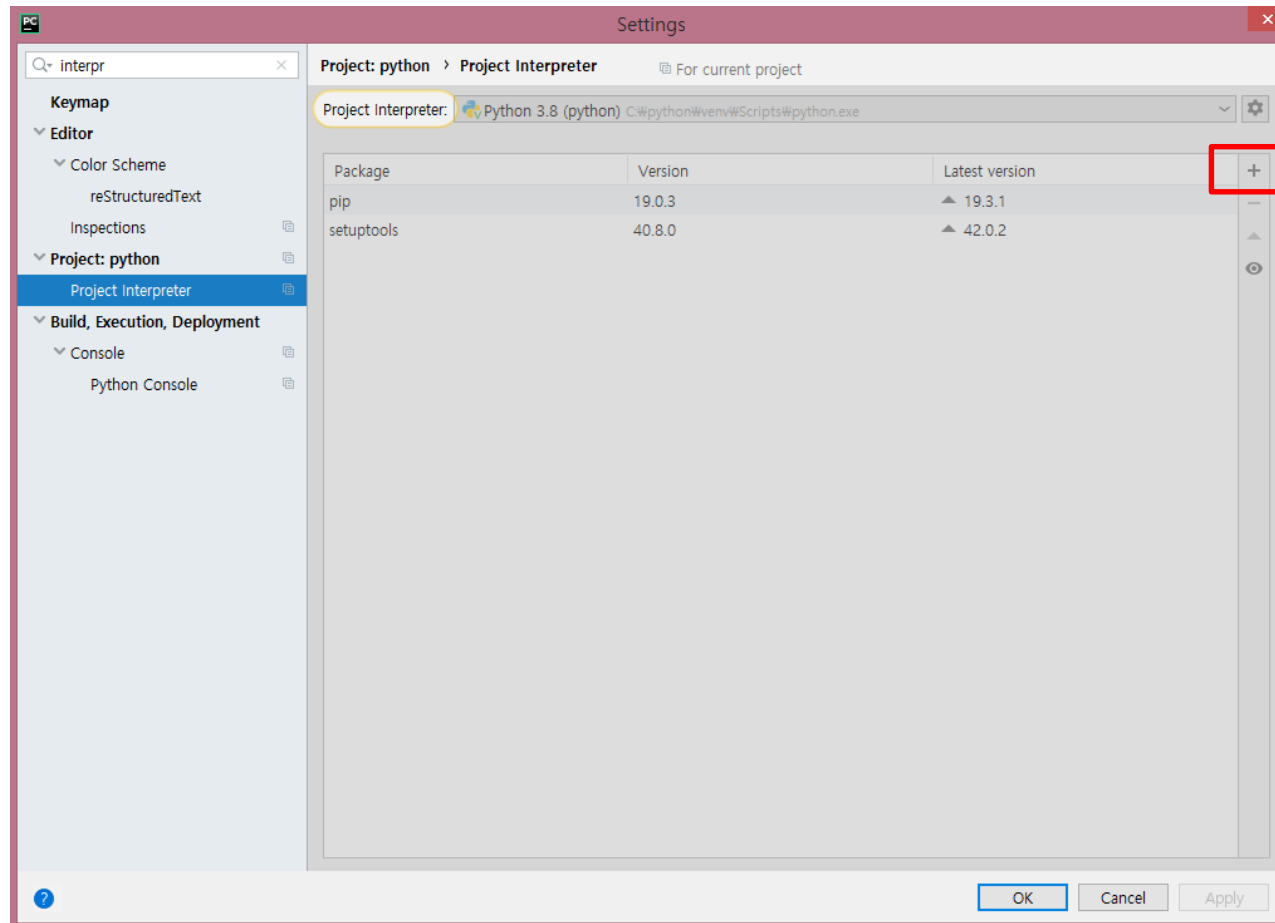


### ■ 패키지 추가하기

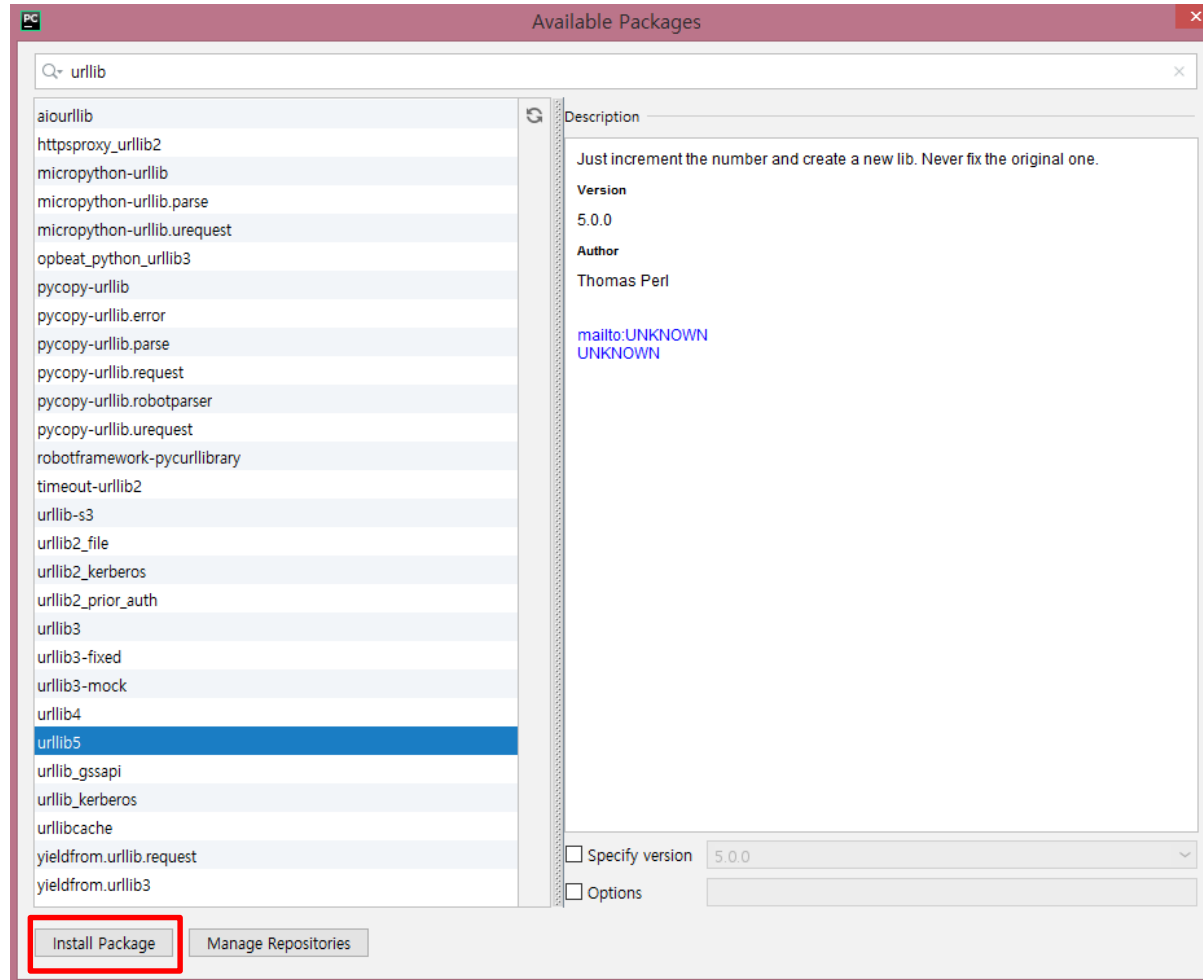
- 메뉴 File -> Settings -> Project Interpreter 클릭



- 인터프리터에서 라이브러리 추가하기
  - 우측의 + 버튼을 누름



- 라이브러리 검색하여 설치하기
  - 라이브러리 선택 후 Install Package 누름

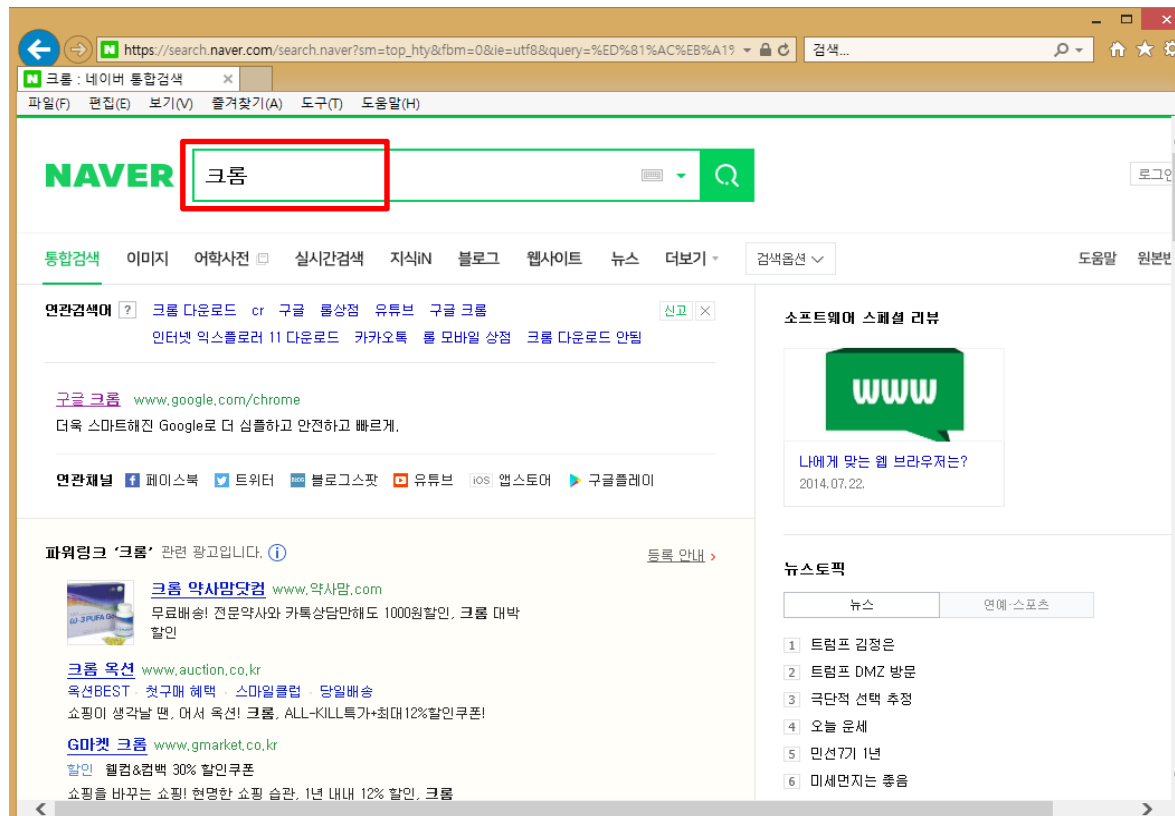


### ■ 크롬(Chrome) 설치하기

크롬에 있는 개발자 도구가 크롤러를 만들 때 필요한 도구임.

#### ■ 설치 페이지로 이동하기

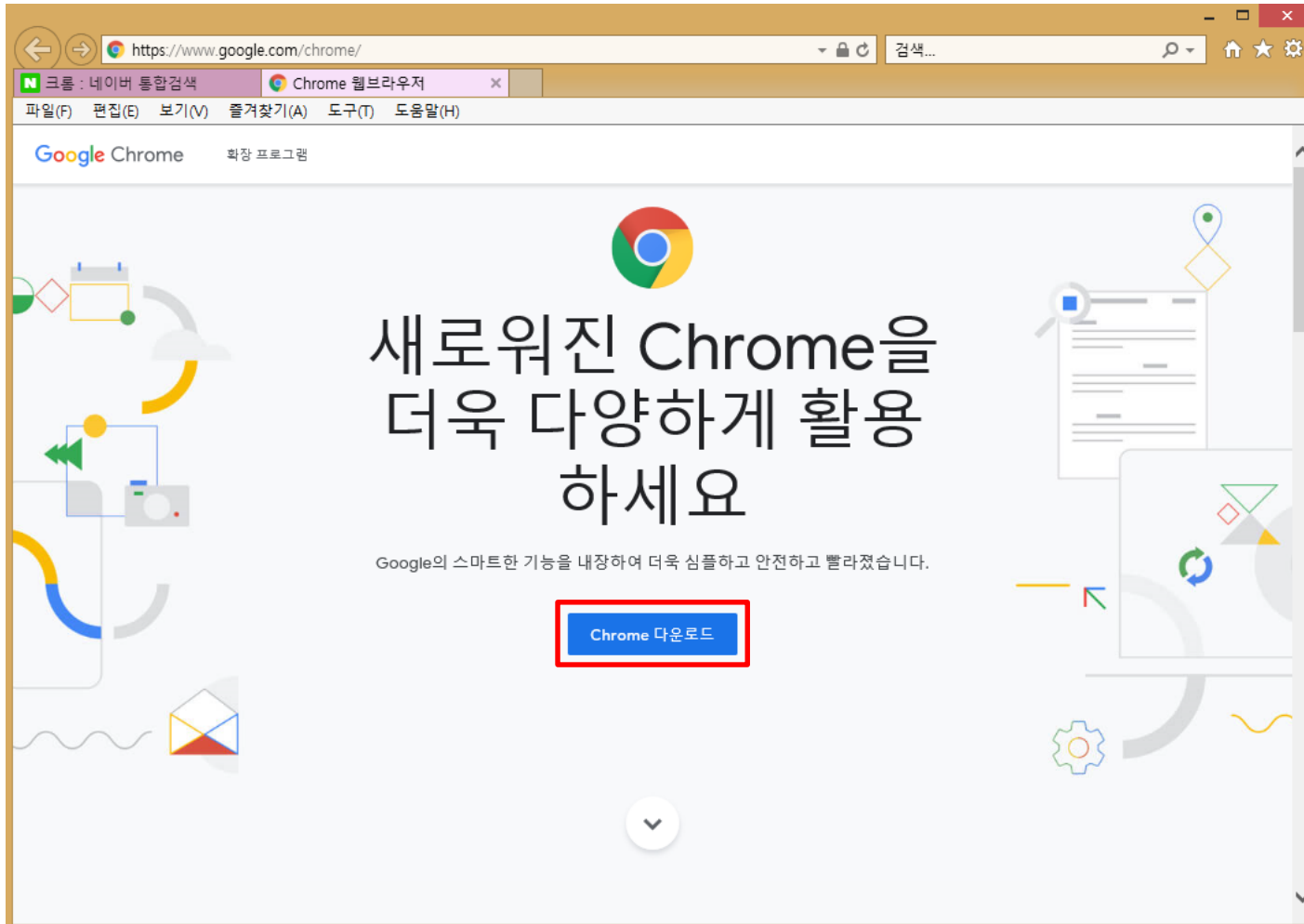
- [www.google.com/chrome](http://www.google.com/chrome)





### ■ 크롬 다운로드하기

- 웹 페이지의 다운로드 버튼을 누르고, <NEXT> 버튼을 누르면 설치됨.



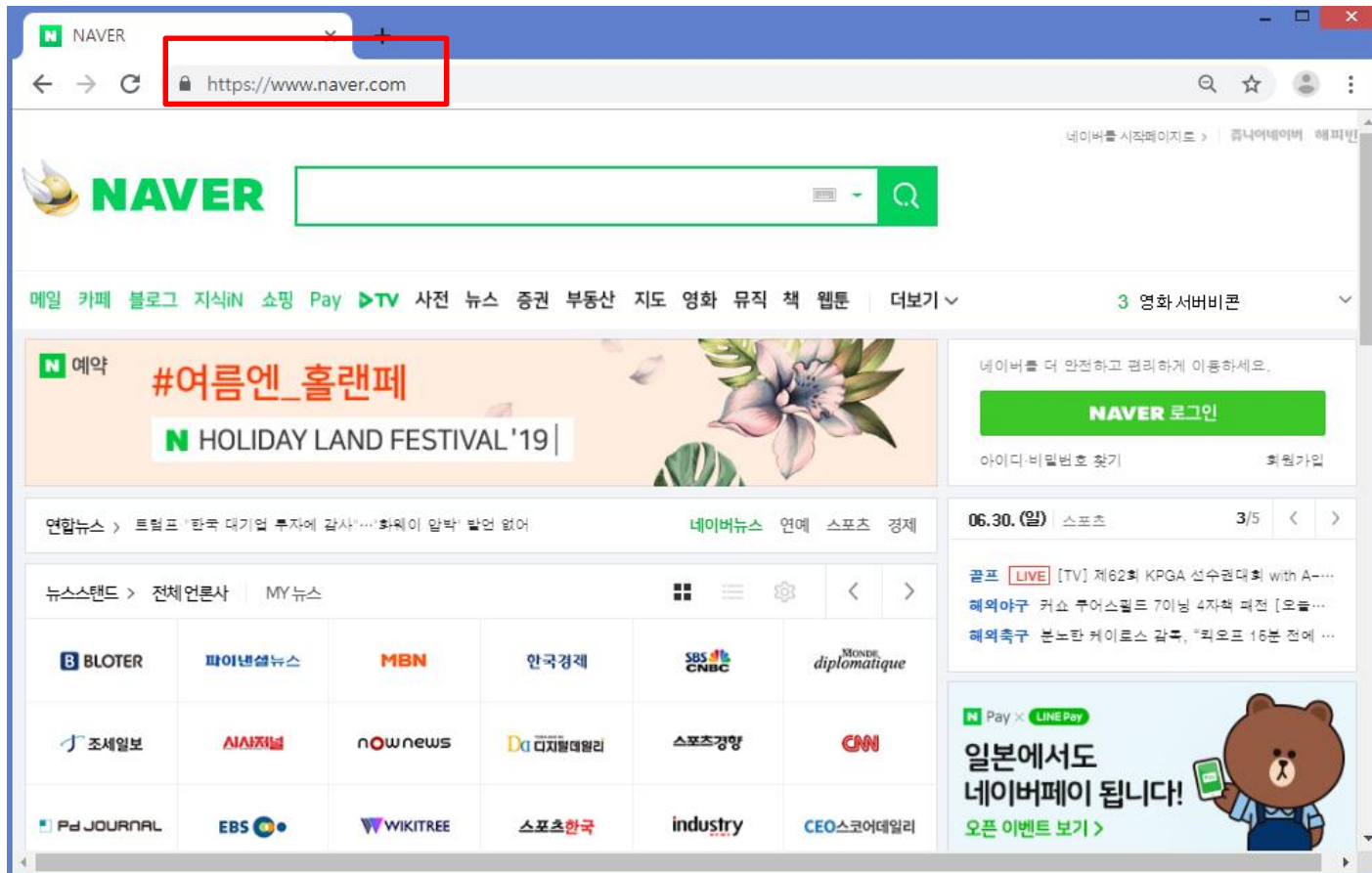
- urllib는 파이썬에서 인터넷 상의 데이터를 받아 오는 기능들이 들어 있는 패키지임. urllib에 인터넷 주소(URL)를 넣고 실행하면 데이터를 텍스트 형태로 받아옴. 데이터를 받아오는 것을 '크롤링'이라고 함. urllib는 크롤링을 하는 라이브러리임.

### ■ urllib 설치하기

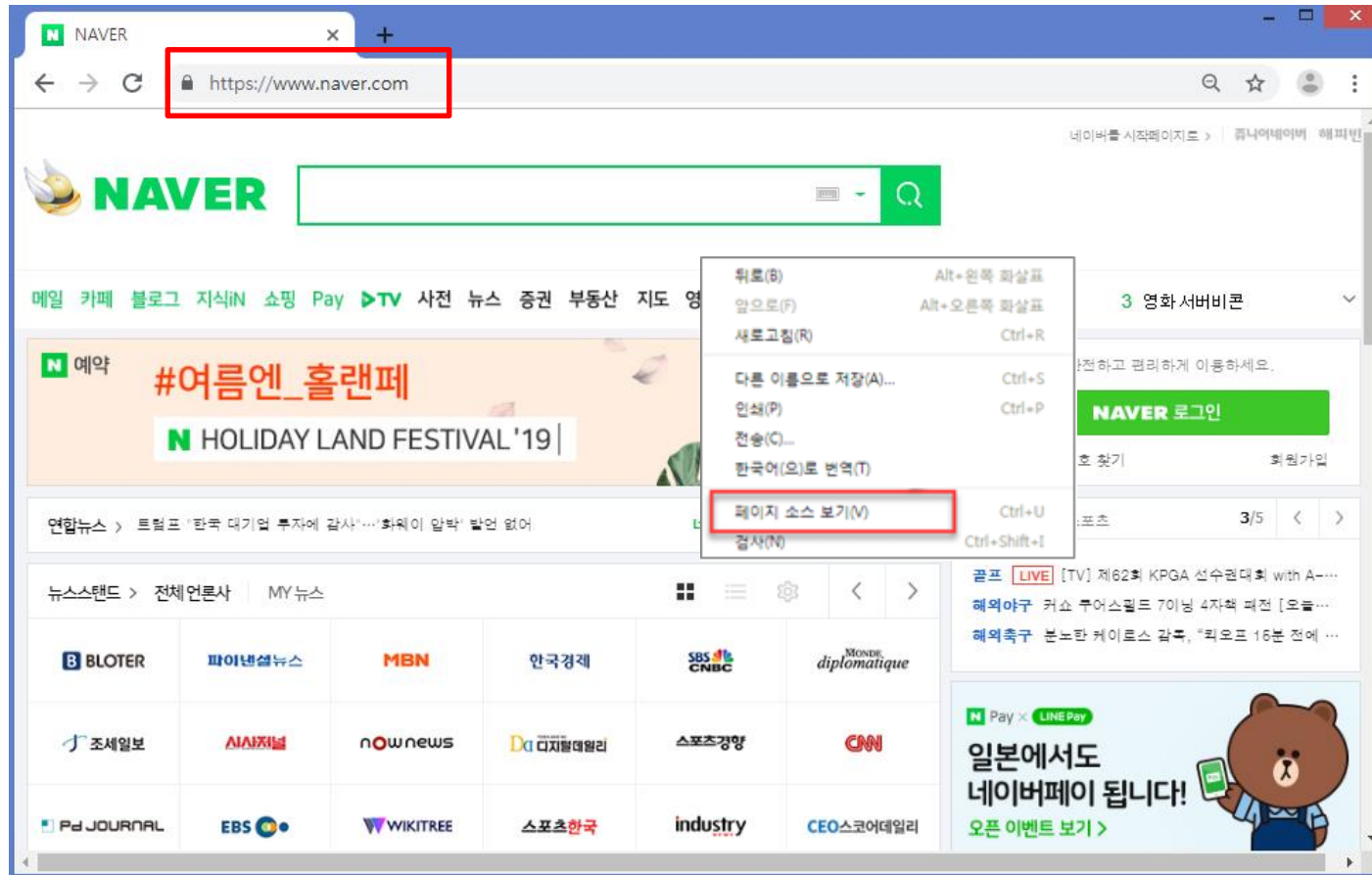
- urllib는 기본적으로 내장되어 있기 때문에 파이썬이 설치되어 있다면 임포트(import)할 수 있음.
- 기본으로 제공하는 urllib 패키지 이외에 urllib5 등 다른 버전을 사용하고자 하는 경우에는 Project Interpreter에서 package를 검색하여 설치함
- [파이참] 메뉴에서 [File → settings → Project Interpreter] 패키지 검색하여 설치

### ■ 네이버 첫 페이지 받아오기

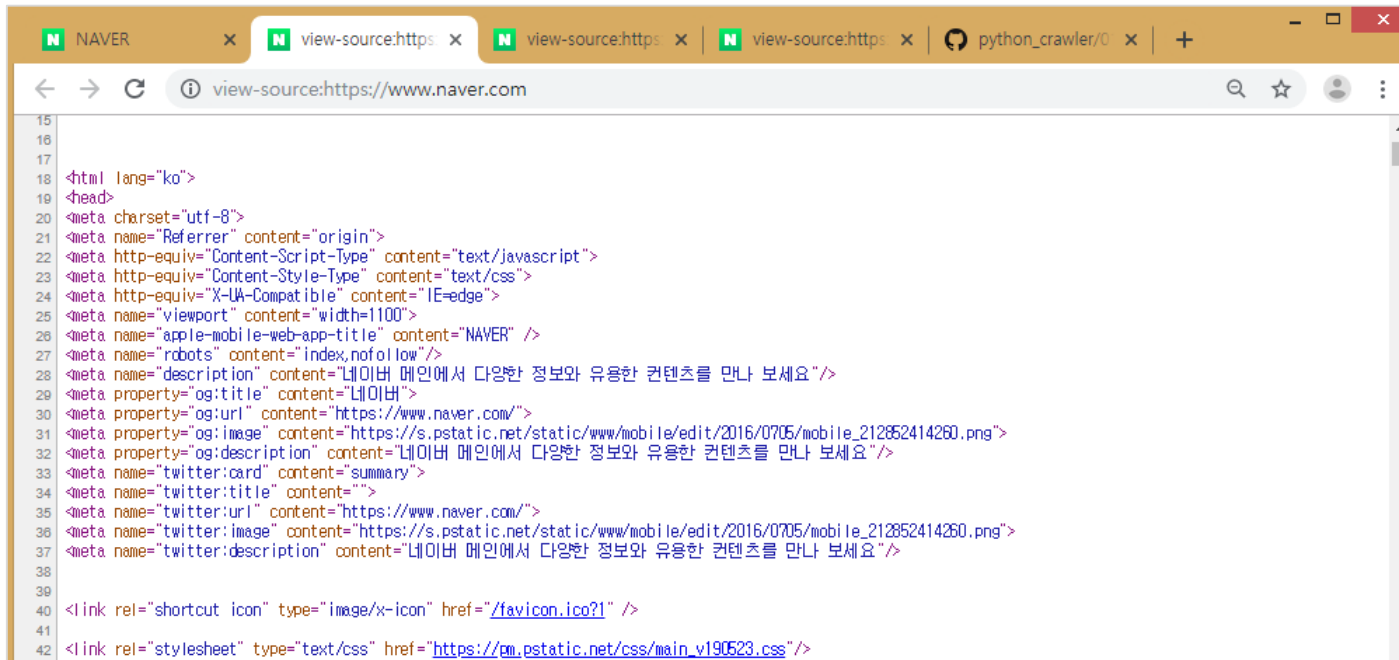
- 크롬을 켜고 주소창에 [www.naver.com](https://www.naver.com)을 입력해서 네이버로 들어감.



- 네이버 접속하고 마우스 오른쪽 버튼을 클릭해서 '페이지 소스보기'를 선택



- 네이버 첫 페이지의 소스코드로 HTML 형식으로 되어 있음.



```
15
16
17 <html lang="ko">
18 <head>
19 <meta charset="utf-8">
20 <meta name="Referrer" content="origin">
21 <meta http-equiv="Content-Script-Type" content="text/javascript">
22 <meta http-equiv="Content-Style-Type" content="text/css">
23 <meta http-equiv="X-UA-Compatible" content="IE=edge">
24 <meta name="viewport" content="width=1100">
25 <meta name="apple-mobile-web-app-title" content="NAVER" />
26 <meta name="robots" content="index,nofollow" />
27 <meta name="description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"/>
28 <meta property="og:title" content="네이버">
29 <meta property="og:url" content="https://www.naver.com/">
30 <meta property="og:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
31 <meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"/>
32 <meta name="twitter:card" content="summary">
33 <meta name="twitter:title" content="">
34 <meta name="twitter:url" content="https://www.naver.com/">
35 <meta name="twitter:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png">
36 <meta name="twitter:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"/>
37
38
39
40 <link rel="shortcut icon" type="image/x-icon" href="/favicon.ico?1" />
41
42 <link rel="stylesheet" type="text/css" href="https://pm.pstatic.net/css/main_v190523.css"/>
```

- 웹 브라우저는 텍스트 형태로 되어 있는 HTML 문서를 읽어서 우리가 보기 좋게 그려 주는 렌더링(rendering) 기능을 하는 프로그램임.
- 크롤링을 한다는 것은 이 텍스트 형태의 데이터를 받아오는 것을 말하며, 받아온 데이터에서 내가 필요한 것을 뽑아내는 것을 '파싱'이라고 함.
- 요즘에는 크롤링, 파싱, 스크래핑이 '인터넷에서 무언가 데이터를 받아서 필요한 정보만 뽑아 내는 것'이라는 뜻으로 같이 사용됨.

- 파이참 실행하여 네이버 첫 페이지의 데이터를 크롤링하기

## 결과

- 이 텍스트를 웹 브라우저에서 해석해서 초록색으로 배치가 잘 된 화면을 보임.
- 웹은 HTML 형태로 되어 있어서 이 HTML 텍스트를 받아온 다음에 여기에서 우리가 필요한 정보들을 파싱할 수 있음.

### ■ 크롤링의 과정

- 페이지를 불러온다.
  - urllib.request를 써서 불러온다.
  - urlopen(url)을 쓴다.
- 파싱을 한다.
  - bs4.BeautifulSoup으로 파싱한다.
  - 개발자 도구를 켜서 원하는 데이터가 있는 위치를 찾는다.

#### ■ BeautifulSoup(뷰티풀속)

- 뷰티풀속은 데이터를 추출하는 데 필요한 기능이 들어 있는 라이브러리이며, 파싱 라이브러리라고도 함.
  - 모래(반도체원료, 규모) = `html(BeautifulSoup)`
- 뷰티풀속은 텍스트를 단계별로 접근을 할 수 있게 데이터를 구조화 시켜줌.
- HTML 소스코드(문서)는 웹 브라우저가 화면을 그리기 위한 내용들이 많이 들어가 있지만, 이 중에서 우리가 필요한 단어나 내용은 아주 조금이므로 파싱은 html 소스코드에서 필요한 내용만 뽑아내는 것임.

#### ■ library(라이브러리)

- 많이 사용하는 기능들을 다른 사람 또는 회사가 만들어서 올려놓은 것
- 라이브러리는 `import`를 해서 사용함
- 내장 라이브러리와 외장 라이브러리가 있음
- 내장 라이브러리는 `import`해서 사용하고, 외장 라이브러리는 설치한 후 `import`해서 사용함



#### ■ 뷰티풀솅 설치하기

- BeautifulSoup(뷰티풀솅)은 외장 라이브러리 이므로 따로 설치를 해야 함.
- [파이참] 메뉴에서 [File → settings → Project Interpreter → bs4] 패키지 검색하여 설치

#### ■ 뷰티풀솅 사용 방법

- 임포트한 후에 HTML형식의 문자열(string)을 BeautifulSoup()에 넣어주고 find()를 이용해 필요한 부분만 뽑아낼 수 있음.

```
# beautiful_soup.py
import bs4
html_str = '<html> <div>hello</div> </html>'
bs_obj = bs4.BeautifulSoup(html_str, "html.parser") -----;
                                     html 형식으로 된 데이터를 구조화 시켜서 오브젝트로 리턴함
print(type(bs_obj)) -----bs_obj 변수의 타입이 뷰티풀솅4라는 뜻
print(bs_obj)
print(bs_obj.find("div")) -----div 이라는 태그만 뽑아냄
print(bs_obj.find("div").text) -----div 태그 안에 들어 있는 텍스트를 뽑아냄
```

#### 결과

```
<class 'bs4.BeautifulSoup'>
<html> <div>hello</div> </html>
<div>hello</div>
hello
```

- HTML 다운 예제로 .find() 기능 사용

```
# bs4_2.py
import bs4

html_str = """ -----소스코드에 텍스트를 여러 줄 넣기 위해서는 """따옴표 3개를 사용함
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul") -----ul 이라는 태그를 뽑아 냄
print(ul)
```

#### 결과

```
<ul>
<li>hello</li>
<li>bye</li>
<li>welcome</li>
</ul>
```

- 여기에서 'hello'만 뽑아내려면 ul에서 li를 한 번 더 찾아주면 됨.

```
# fild_li.py
import bs4

html_str = """
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")  -----ul 이라는 태그를 뽑아 냄
li = ul.find("li")      -----ul에서 li 이라는 태그를 찾아 줌.
print(li)               .find()는 만나는 첫 번째 태그를 리턴해 주는 함수
```

결과

```
<li>hello</li>
```

- <li> </li> 태그 안에 있는 'hello'만 뽑아내려면 .text 속성을 사용함.

```
# find_li_text.py
import bs4

html_str = """
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""
bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")
li = ul.find("li")
print(li.text)  -----> 태그 안에 들어 있는 텍스트를 뽑아서 출력함
```

결과

hello

#### ■ findAll() 사용하기

- .findlAll()은 조건에 해당하는 모든 요소를 리스트 [ ] 형태로 추출해주는 기능임.

```
# find_all.py
import bs4

html_str = """
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")
lis = ul.findAll("li") -----li 태그를 모두 뽑아서 리스트 [ ] 형태로 리턴해 줌
print(lis)
```

결과

```
[<li>hello</li>, <li>bye</li>, <li>welcome</li>]
```

#### ■ 인덱스로 데이터 접근하기

- 인덱스(index)로 리스트 안에 있는 데이터에 접근할 수 있으며, 인덱스는 0부터 시작함.

```
# find_all_index.py
import bs4

html_str = """
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")
lis = ul.findAll("li")
print(lis[1])
```

첫 번째 있는 요소를 뽑으려면 0번을 쓰고,  
-----두 번째 요소를 뽑으려면 1번을 사용함

결과

<li>bye</li>

- 맨 마지막 줄을 `print(list[1].text)`로 바꾸면 'bye'만 출력됨

```
# find_all_index_text.py
import bs4

html_str = """
<html>
  <body>
    <ul>
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")
lis = ul.findAll("li")
print(lis[1].text)          -----두 번째 li 태그 안에 들어 있는 텍스트를 뽑아서 출력함
```

결과

bye



#### ■ 태그와 속성 그리고 속성값

- HTML은 <html>로 시작해서 </html>로 끝나는 문서이며, HTML은 태그(tag)와 속성(property)으로 구성되어 있음.
- 태그란 html 문서를 표현 할 때 쓰는 화면 구성을 하는 표시들로 문서에 있는 특정 부분에 꼬리표를 붙여준다는 뜻으로 tag라고 씀.
  - <html>, <div>, <ul>, <li>, <a>, <span> 등 있음.
- <ul class="greet">
  - 속성(property) : class
  - 속성값 : "greet"
- <a href="https://www.google.com/">
  - 속성(property) : href
  - 속성값 : "https://www.google.com/"

#### ■ 데이터 뽑을 때 class 속성 이용하기

- HTML코드에 ul이 두 개 있는 경우, class 속성을 필터링 조건에 추가해서 추출함.

```
# property_class.py
import bs4

html_str = """
<html>
  <body>
    <ul class="greet">
      <li>hello</li>
      <li>bye</li>
      <li>welcome</li>
    </ul>
    <ul class="replay">
      <li>ok</li>
      <li>no</li>
      <li>sure</li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul", {"class": "replay"})
print(ul)
```

----- .find()는 첫 번째로 만나는 ul 태그만을 리턴함.  
class 속성을 필터링 조건에 추가해서 추출함.

#### 결과

```
<ul class="replay">  
<li>ok</li>  
<li>no</li>  
<li>sure</li>  
</ul>
```

- .find()를 사용할 때, "ul" 말고 {"class": "replay"} 조건을 하나 더 추가해주면 뽑고 싶은 요소에 조금 더 정확하게 접근할 수 있음.
- class를 이용하면 같은 태그가 있을 때 특정 블록을 선택해서 뽑아낼 수 있음.

#### ■ 속성값 뽑아내기

- a태그에서 링크를 뽑아낼 때는 href라는 속성의 속성값을 뽑아내야 함.

```
# property_href.py
import bs4

html_str = """
<html>
  <body>
    <ul class="ko">
      <li>
        <a href="https://www.naver.com/">네이버</a>
      </li>
      <li>
        <a href="https://www.daum.net/">다음</a>
      </li>
    </ul>
  </body>
</html>
"""

bs_obj = bs4.BeautifulSoup(html_str, "html.parser")
atag = bs_obj.find("a")
print(atag)          -----.find()는 첫 번째로 만나는 a태그만을 리턴함
```

#### 결과

```
<a href="https://www.naver.com/">네이버 </a>
```

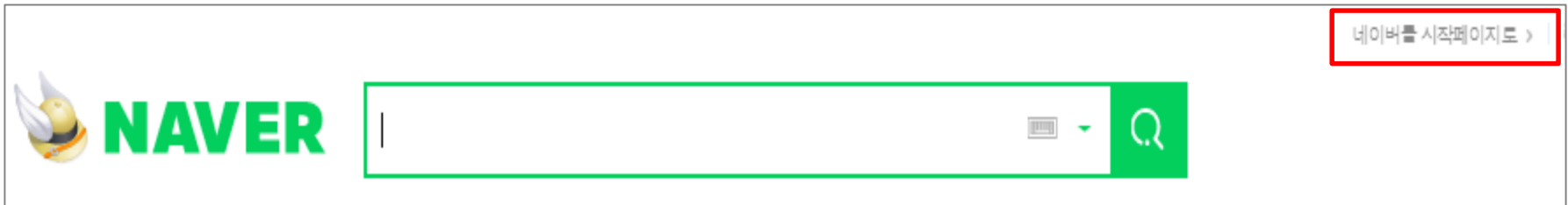
- a태그에 들어 있는 텍스트인 '네이버'를 뽑으려면 atag.text를 사용함
- href속성의 속성 값인 네이버의 주소 <https://www.naver.com/>을 뽑기 위해서는 atag['href'] 이렇게 대괄호 안에 '를 붙인 속성을 넣어주면 '속성값'을 뽑아 낼 수 있음.

```
---생략---  
bs_obj = bs4.BeautifulSoup(html_str, "html.parser")  
atag = bs_obj.find("a")  
print(atag['href'])  -----a태그에서 대괄호 안에 '를 붙인 속성을 넣어주면 속성값을 뽑아 냄
```

#### 결과

```
https://www.naver.com/
```

- 네이버 첫 페이지에서 맨 오른쪽 위에 있는 '네이버를 시작페이지로'라는 단어 두 개만 뽑아보기



- 웹에서 데이터를 받아오려면 request라는 요청을 보내서 받아와야 함.
- 파이썬에서 웹의 특정 주소로 요청을 보내는 기능은 urllib.request의 urlopen() 이라는 함수를 사용함.

## 결과

- 네이버 첫 페이지를 받아오면, `b'<!doctype html>` 시작해서 `</html>`로 끝나는 데이터가 받아와 짐.

### ■ 뷰티풀썬에 데이터 넣기

- urlopen() 이라는 함수를 이용해 받아온 데이터를 파싱하기 위해서는 뷰티풀썬에 데이터를 넣어서 파이썬에서 가공할 수 있는 형태로 만들어 주어야 함.

```
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url) ----- html이라는 변수 안에는 웹에서 받은 텍스트가 들어감

bs_obj = bs4.BeautifulSoup(html, "html.parser") -----
bs4.BeautifulSoup() 에 웹에서 받은 텍스트를 넣으면,
파이썬에서 가공할 수 있는 html 형태의 텍스트가 bs_obj에
들어감

print(bs_obj)
```

### 결과

```
<!DOCTYPE doctype html>

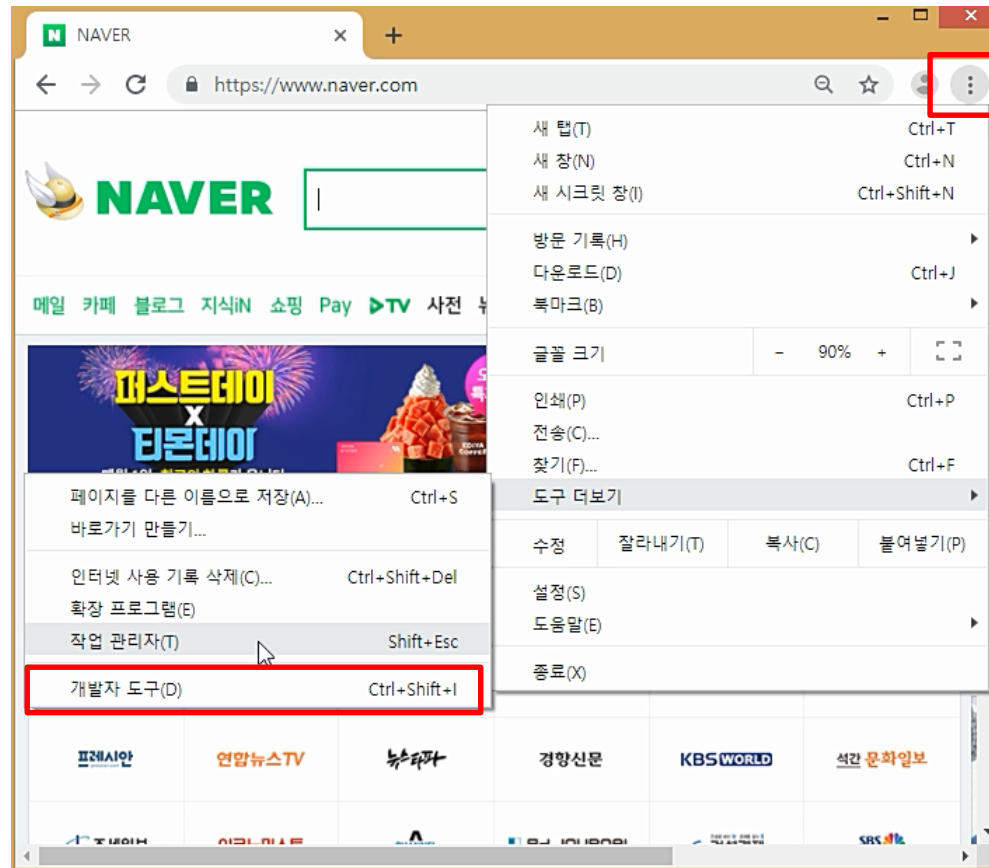
<html lang="ko">
<head>
<meta charset="utf-8"/>
---중략---
</html>
```



- `bs_obj = bs4.BeautifulSoup(html, "html.parser")`
  - `.BeautifulSoup(<받은텍스트>, <텍스트를 파싱할 파서>)`에는 총 2가지가 들어감.
  - 받은 텍스트 : 웹에서 받은 텍스트
  - 텍스트를 파싱할 파서 : 웹 문서는 대부분 HTML로 되어 있어서 "html.parser"를 사용
    - parser(파서)는 데이터를 뽑아내는(파싱) 프로그램임.
    - 파이썬이 HTML 형식으로 인식하라는 뜻임.
    - "html.parser"를 가장 많이 사용하며, "lxml"과 "xml" 등도 있음.

#### ■ 뷰티풀썬으로 필요한 부분 뽑아내기

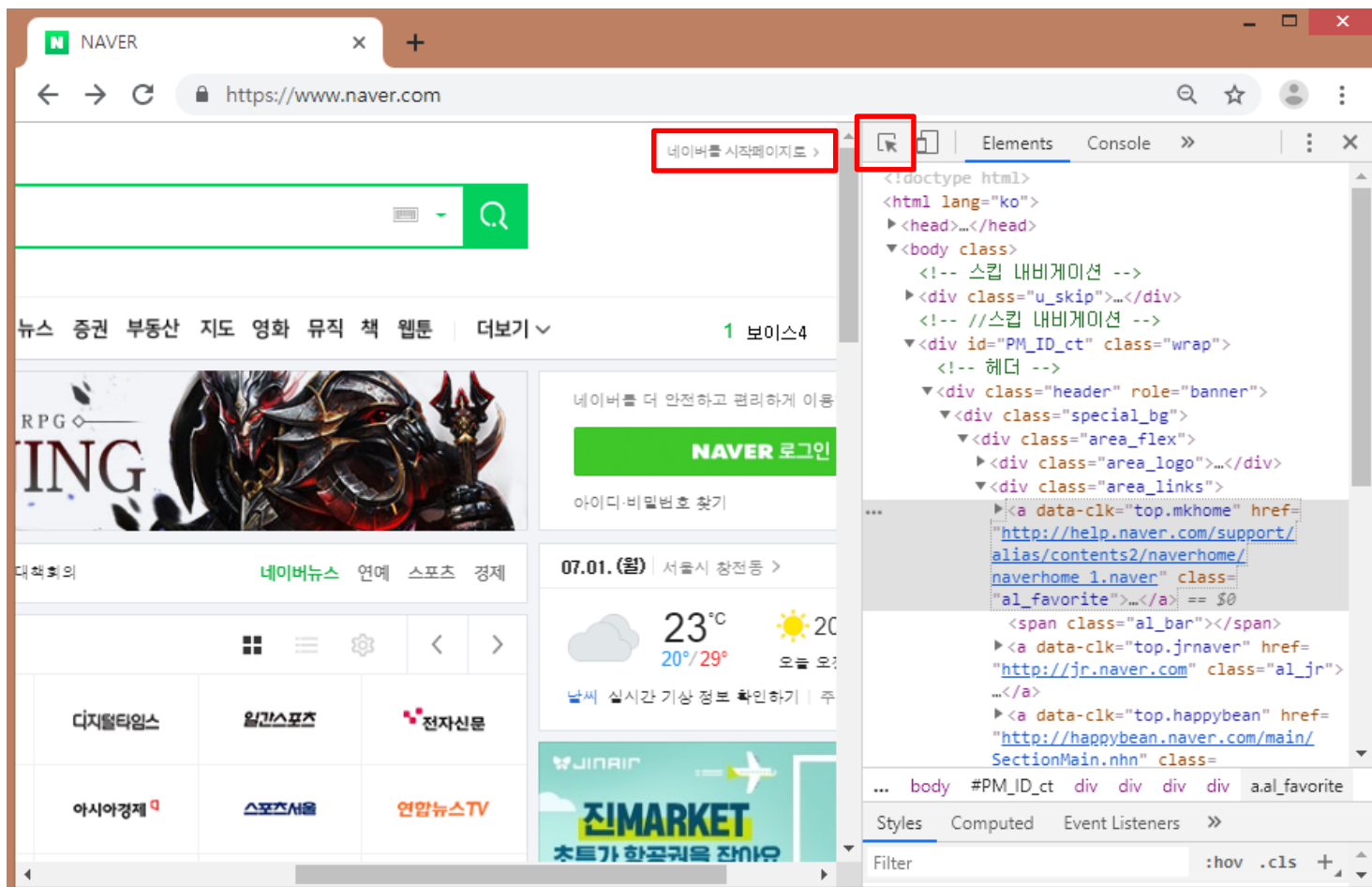
- 원하는 데이터 HTML상에 어디에 있는지 찾기 위해서는 구글 크롬의 '개발자 도구'를 이용함.
  - 크롬을 켜고 우측 상단에 ...버튼을 눌러서 [도구 더보기 → 개발자 도구]



## 4. 네이버에서 특정 글자 추출하기

### 2장. 크롤러 만들기

- 개발자 도구에서 왼쪽 위의 화살표 아이콘을 누르면 색이 파란색으로 바뀌고, 마우스를 움직여 원하는 위치를 클릭하면 소스코드에서 해당부분을 알려줌.
- '네이버를 시작페이지로' 버튼을 마우스로 찾은 후에 클릭하면, HTML 소스코드에서 어떤 부분인지 표시를 해줌.



```
<div class="area_links">
<a data-clk="top.mkhome"
href="http://help.naver.com/support/alias/contents2/naverhome/naverhome_1.n
aver" class="al_favorite">네이버를 시작페이지로<span
class="al_ico_link"></span></a>
```

- '네이버를 시작페이지로'는 <a>라는 태그 안에 텍스트가 들어 있는 형태임.
  - <a>네이버를 시작페이지로<span></span></a>
- HTML 코드를 보면, <div>라는 태그 안에 <a></a>가 들어 있는 구조임.

```
<div class="area_links">
    <a data-clk="top.mkhome" href="http://~생략~>
    </a>
</div>
```

- div 태그 안에 class="area\_links" 부분은 클래스를 가지고 파이썬의 뷰티풀썬 라이브러리를 이용해 데이터를 뽑아낼 수 있기 때문에 매우 중요함.

- `<div class="area_links"> </div>` 이 영역만 추출해보도록 하겠음.

```
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

bs_obj = bs4.BeautifulSoup(html, "html.parser") ----- bs_obj에는 전체 소스가 들어 있음

top_right = bs_obj.find("div", {"class":"area_links"}) ----- div 태그 중에서 class가 "area_links"로
print(top_right)                                           되어있는 div를 찾으라는 명령
```

### 결과

```
<div class="area_links">
<a class="al_favorite" data-clk="top.mkhome"
href="http://help.naver.com/support/alias/contents2/naverhome/naverhome_1.naver">네이버를 시
작페이지로<span class="al_ico_link"> </span> </a>
<span class="al_bar"> </span>
<a class="al_jr" data-clk="top.jrnaver" href="http://jr.naver.com"> <span class="blind">쥬니어네이
버</span> <span class="al_ico"> </span> </a>
<a class="al_happybean" data-clk="top.happybean"
href="http://happybean.naver.com/main/SectionMain.nhn"> <span class="blind">해피빈
</span> <span class="al_ico"> </span> </a>
</div>
```

- `top_right = bs_obj.find("div", {"class": "area_links"})`
  - `bs_obj`에는 전체 소스가 들어 있음.
  - `bs_obj.find("div")` 명령어는 전체에서 가장 처음 나타나는 `<div>` 태그를 뽑으라는 명령
  - `"div"` 옆에 `,`(콤마)를 찍고 `{"class": "area_links"}`를 추가하면, `div` 태그 중에서 `class`가 `"area_links"`로 되어있는 `div`를 찾으라는 명령
- 이제 필요한 '네이버를 시작페이지로' 글자만 뽑아보겠음.

```
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

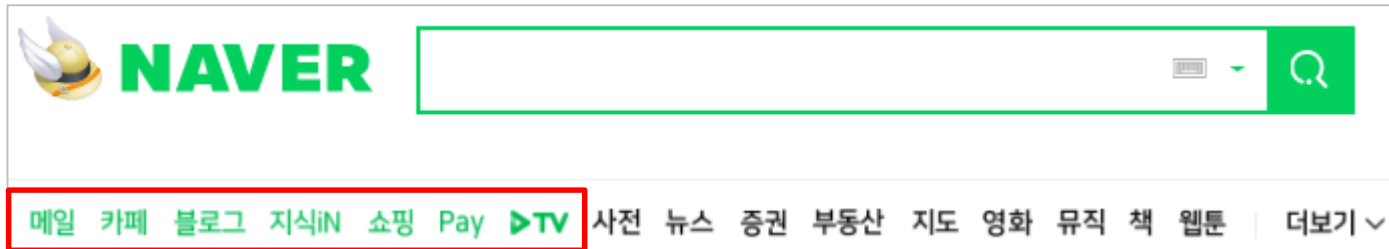
bs_obj = bs4.BeautifulSoup(html, "html.parser")

top_right = bs_obj.find("div", {"class": "area_links"})
first_a = top_right.find("a")  ----- 첫 번째 나오는 a태그를 찾음
print(first_a.text)           ----- a태그 안에 있는 text만 뽑아냄
```

결과

네이버를 시작페이지로

- 네이버 첫 페이지에서 메뉴에 있는 '메일', '카페', '블로그' 등의 글자 뽑아내기
  - 메뉴 이름을 뽑는 방법과 뉴스 페이지에서 뉴스 제목을 뽑는 방법은 아주 비슷하므로 메뉴를 뽑아 보면서 크롤링을 익힐 수 있음.

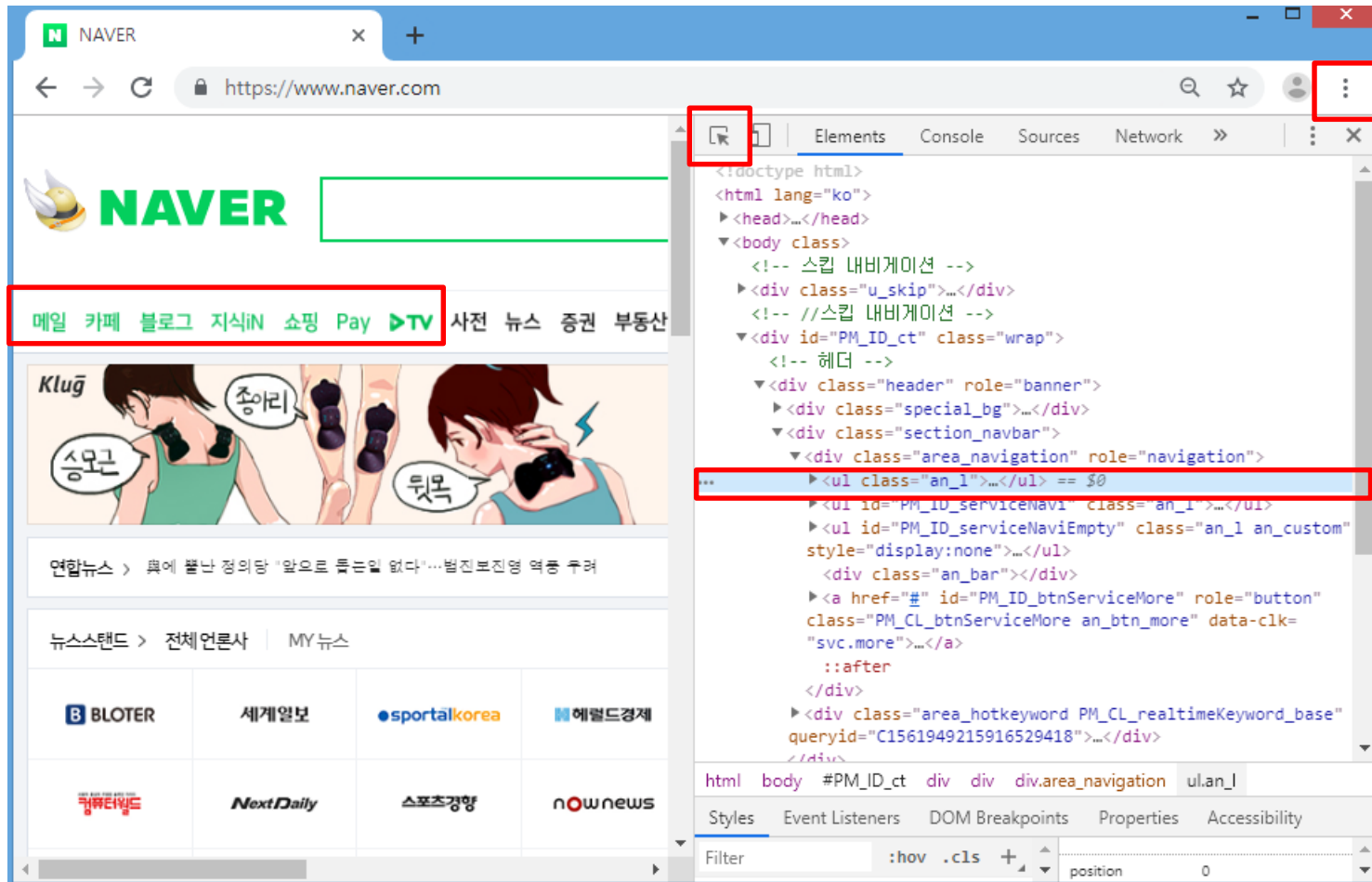


### 결과

메일  
카페  
블로그  
지식인  
쇼핑  
네이버페이  
네이버TV

### ■ 추출할 범위의 class 알아내기

- [크롬] 메뉴에서 [...버튼 → 도구 더보기 → 개발자 도구 → 왼쪽위 화살표버튼]





- `<ul>...</ul>`안에는 여러 개의 `<li>...</li>`태그가 들어 있으며, 네이버 메뉴들은 `<li class="an_item">...</li>` 구조 안에 들어 있음.

```
▼<div class="section_navbar">
  ▼<div class="area_navigation" role="navigation">
    ▼<ul class="an_l"> == $0
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ▶<li class="an_item">...</li>
      ::after
    </ul>
  ▶<ul id="PM_ID_serviceNavi" class="an_l">...</ul>
```

```
▼<ul class="an_l"> == $0
  ▼<li class="an_item">
    ▼<a href="https://mail.naver.com/" class="an_a mn_mail" data-clk="svc.mail">
      <span class="an_icon"></span>
      <span class="an_txt">메일</span>
    </a>
  </li>
  ▶<li class="an_item">...</li>
  ▶<li class="an_item">...</li>
  ▶<li class="an_item">...</li>
  ▶<li class="an_item">...</li>
  ▶<li class="an_item">...</li>
  ::after
</ul>
```

- 파이썬에서 `<ul>` 태그와 클래스(class)를 이용하여 li안에 있는 내용을 뽑아낼 예정임.

- 파이썬의 BeautifulSoup 라이브러리를 이용해 필요한 글자를 뽑아낼 수 있음.
  - ul 태그와 class가 "an\_l"인 ul을 전체 문서에서 찾기

```
# naver_menu.py
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

bs_obj = bs4.BeautifulSoup(html, "html.parser")

ul = bs_obj.find("ul", {"class": "an_l"})  ----- ul 태그 중에서 class가 an_l인 ul을 찾음
print(ul)
```

### 결과

```
<ul class="an_l">
<li class="an_item">
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">
<span class="an_icon"> </span> <span class="an_txt">메일</span>
</a>
</li>
---중략---
</ul>
```

- ul안에 들어 있는 각각의 li만 뽑기

```
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url)

bs_obj = bs4.BeautifulSoup(html, "html.parser")

ul = bs_obj.find("ul", {"class": "an_l"})  ----- ul 태그 중에서 class가 an_l인 ul을 찾음
for li in ul:  ----- ul 안에 있는 li들을 하나씩 꺼내서 출력
    print(li)
```

### 결과

```
<li class="an_item">
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">
<span class="an_icon"> </span> <span class="an_txt">메일</span>
</a>
</li>

<li class="an_item">
<a class="an_a mn_cafe" data-clk="svc.cafe" href="https://section.cafe.naver.com/">
<span class="an_icon"> </span> <span class="an_txt">카페</span>
---생략---
```

### ■ .findAll( )로 li만 뽑아내기

- .findAll( )은 조건에 해당하는 모든 것들을 [ ]리스트 안으로 추출해주는 함수임.
- 반복문 for를 이용해도 되지만, 중간에 빈칸이 뽑히는 경우가 있어서 .findAll( )을 써서 한 번 더 뽑아 주는 게 좋음.

```
bs_obj = bs4.BeautifulSoup(html, "html.parser")

ul = bs_obj.find("ul", {"class":"an_l"})

lis = ul.findAll("li") ----- ul 안에 있는 모든 li를 찾으라는 명령
print(lis)
```

----- [ ]리스트 안에 <li></li>가 여러 개 들어 있으며, 콤마(,)로 구분된 형태

```
[<li class="an_item">
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">
<span class="an_icon"></span> <span class="an_txt">메일</span>
</a>
</li>, <li class="an_item">
<a class="an_a mn_cafe" data-clk="svc.cafe" href="https://section.cafe.naver.com/">
<span class="an_icon"></span> <span class="an_txt">카페</span>
---중략---
</li>]
```

### ■ li 하나씩 꺼내서 출력하기

- 대괄호와 중간의 콤마(,)가 나오지 않게 하나씩 뽑아보기

```
---생략---
Bs_obj = bs4.BeautifulSoup(html, "html.parser")

ul = bs_obj.find("ul", {"class": "an_l"})

lis = ul.findAll("li") ----- ul를 바로 for문을 사용하여 출력하면 중간에 빈칸이 뽑히는 경우가 있음

for li in lis:
    print(li) ----- lis 안에 있는 li들을 하나씩 꺼내서 출력
```

### 결과

```
<li class="an_item">
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">
<span class="an_icon"></span> <span class="an_txt">메일</span>
</a>
</li>
<li class="an_item">
<a class="an_a mn_cafe" data-clk="svc.cafe" href="https://section.cafe.naver.com/">
---중략---
</li>
```

### ■ a태그 뽑아내기

```
---생략---
Bs_obj = bs4.BeautifulSoup(html, "html.parser")

ul = bs_obj.find( "ul" , { "class" : "an_l" })

lis = ul.findAll( "li" )

for li in lis:
    a_tag = li.find("a")
    print(a_tag)  ----- li안에 있는 a태그를 뽑아서 a_tag라는 변수에 넣으라는 명령
                    (점점 세밀하게 필요한 내용에 접근 )
```

### 결과

```
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">
<span class="an_icon"> </span> <span class="an_txt"> 메일 </span>
</a>
<a class="an_a mn_cafe" data-clk="svc.cafe" href="https://section.cafe.naver.com/">
---중략---
<a class="an_a mn_tvcast" data-clk="svc.tvcast" href="https://tv.naver.com/">
<span class="an_icon"> </span> <span class="an_txt"> 네이버TV </span>
</a>
```

### ■ span인 것 중에 an\_txt인 class 뽑아내기

```
<a class="an_a mn_mail" data-clk="svc.mail" href="https://mail.naver.com/">  
    <span class="an_icon"> </span>  
    <span class="an_txt">메일</span>  
</a>
```

- <a>태그 안에 <span>이 두 개가 있으므로, 태그인 span과 클래스인 an\_txt 두 조건으로 추출해야 함.

```
---생략---  
Bs_obj = bs4.BeautifulSoup(html, "html.parser")  
  
ul = bs_obj.find( "ul", { "class" : "an_l" })  
  
lis = ul.findAll( "li" )  
  
for li in lis:  
    a_tag = li.find("a")  
    span = a_tag.find("span", {"class":"an_txt"}) ----- span 태그 중에서 an_txt 클래스를 가진  
    print(span)                                           것만 뽑으라는 명령
```

### 결과

```
<span class="an_txt">메일</span>  
<span class="an_txt">카페</span>  
<span class="an_txt">블로그</span>  
<span class="an_txt">지식인</span>  
<span class="an_txt">쇼핑</span>  
<span class="an_txt">네이버페이</span>  
<span class="an_txt">네이버TV</span>
```



### ■ Text 뽑아내기

- <span> 태그가 제외된, '메일'만 들어 있는 순수한 텍스트만 뽑아내기

```
import urllib.request
import bs4

url = "https://www.naver.com/"
html = urllib.request.urlopen(url) --- html이라는 변수 안에는 웹에서 받은 텍스트가 들어감

bs_obj = bs4.BeautifulSoup(html, "html.parser") --- bs4.BeautifulSoup() 에 웹에서 받은 텍스트를
                                                    넣으면, 파이썬에서 가공할 수 있는 html 형태의
                                                    텍스트가 bs_obj에 들어감

ul = bs_obj.find("ul", {"class":"an_l"})

lis = ul.findAll("li")
for li in lis:
    a_tag = li.find("a")
    span = a_tag.find("span", {"class":"an_txt"})
    print(span.text) --- span 태그에서 text만 뽑아내는 부분
```

결과

메일  
카페  
블로그  
지식인  
쇼핑  
네이버페이  
네이버TV

- 네이버 뉴스의 오늘의 기사 제목 가져오기



**결과** ----- 매일 바뀌는 뉴스 제목을 가지고 올 수 있음

文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후  
日, 강제징용 판결에 '경제보복'...공공 얼어붙은 한일관계  
트럼프 밀착취재 美기자 "김정은, 폐병환자처럼 혈떡대"  
한국·바른미래, '北어선 사건' 국조요구서 공동 제출(종합)  
"권리당원 확보에 올인"...민주당 '7월 전쟁'으로 지역별 공천경쟁 본격화

### ■ 뉴스 페이지 불러오기

```
# naver_news.py
import urllib.request
import bs4

url = "http://news.naver.com/"
html = urllib.request.urlopen(url)

bs_obj = bs4.BeautifulSoup(html, "html.parser")

print(bs_obj)
```

--- html이라는 변수 안에는 웹에서 받은 텍스트가 들어감

--- bs4.BeautifulSoup() 에 웹에서 받은 텍스트를  
넣으면, 파이썬에서 가공할 수 있는 html 형태의  
텍스트가 bs\_obj에 들어감

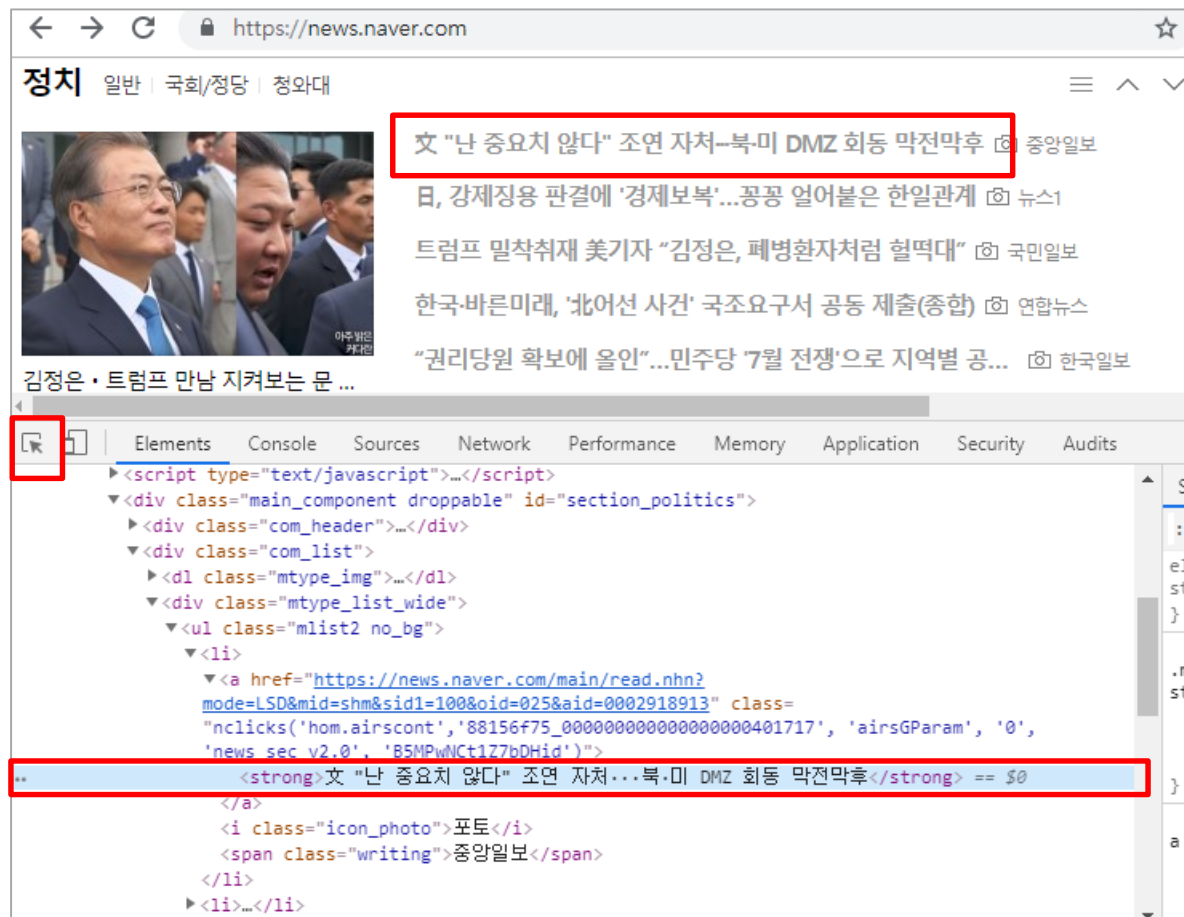
### 결과

```
<!DOCTYPE HTML>

<html lang="ko">
<head>
<meta charset="utf-8"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta contents="always" name="referrer"/>
<meta content="600" http-equiv="refresh">
<meta content="width=1106" name="viewport">
<meta content="네이버 뉴스" property="og:title"/>
---생략---
```

### ■ 뉴스 제목 위치 찾기

- [크롬] 메뉴에서 [...버튼 → 도구 더보기 → 개발자 도구 → 왼쪽위 화살표버튼]



### ■ HTML 구조 분석하기

```
<li>
  <a href="https://news.naver.com/main/read.nhn? ...생략...">
    <strong>文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후</strong>
  </a>
</li>
```

- HTML 구조를 분석해 보면, <li>안에 <a>가 들어 있고, <a>안에 <strong>이 들어 있고, <strong>안에 뉴스 제목이 있는 형태임.
- 개발자 도구에서 선택된 부분은 <ul>안에 위의 구조가 반복되고, <li>가장 안쪽에 <strong>이 있고 거기에 기사 제목이 들어 있는 구조임.
  - 선택한 범위는 ul이고, class는 'mlist2 no\_bg'임.

```
<div class="com_list">
  <dl class="mtype_img">...</dl>
  <div class="mtype_list_wide">
    <ul class="mlist2 no_bg">
      <li>...</li>
      <li>...</li>
      <li>...</li>
      <li>...</li>
      <li>...</li>
    </ul>
  </div>
</div>
```

----- ul 태그 안에 여러 개의 li들이 들어 있음.

- class로 ul 태그 찾기

```
---생략---
bs_obj = bs4.BeautifulSoup(html, "html.parser")

mlist2_no_bg = bs_obj.find("ul", {"class": "mlist2 no_bg"})
print(mlist2_no_bg)
```

-----  
bs\_obj에서 class가 'mlist2 no\_bg' 인 ul 태그를 찾음

#### 결과

```
<ul class="mlist2 no_bg">
<li>
<a class="nclicks('hom.airscont','880000D8_0000000000000000010926323',
'airsGParam', '0', 'news_sec_v2.0', 'TDSFjOT0gQflfRgg')"
href="https://news.naver.com/main/read.nhn?mode=LSD&amp;mid=shm&amp;sid1=1
00&amp;oid=025&amp;aid=0002918917">
<strong>文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후</strong>
</a>
<i class="icon_photo"> 포토 </i>
<span class="writing"> 중앙일보 </span>
</li>
---중략---
</ul>
```

- .findAll( )로 li들 뽑아내기

```
---생략---
bs_obj = bs4.BeautifulSoup(html, "html.parser")

mlist2_no_bg = bs_obj.find("ul", {"class": "mlist2 no_bg"})
lis = mlist2_no_bg.findAll("li")  ----- lis 변수를 선언하고, 'mlist2 no_bg' 에서 li 태그들을
                                   리스트[] 형태로 담으라는 명령

for li in lis:  ----- for 문을 이용해 lis에 있는 li들을 하나씩 꺼내서 출력
    print(li)
```

### 결과

```
<li>
<a class="nclicks('hom.airscont','880000D8_0000000000000000010926323',
'airsGParam', '0', 'news_sec_v2.0', 'TDSFjOT0gQflfRgg')"
href="https://news.naver.com/main/read.nhn?mode=LSD&amp;mid=shm&amp;sid1=1
00&amp;oid=025&amp;aid=0002918917">
<strong>文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후</strong>
</a>
<i class="icon_photo"> 포토 </i>
<span class="writing"> 중앙일보 </span>
</li>
<li>
---생략---
```

- strong 태그와 그 안에 있는 기사 제목 뽑아내기

```
---생략---
bs_obj = bs4.BeautifulSoup(html, "html.parser")

mlist2_no_bg = bs_obj.find("ul", {"class": "mlist2 no_bg"})
lis = mlist2_no_bg.findAll("li")

for li in lis:
    strong = li.find("strong")
    print(strong)
```

#### 결과

```
<strong>文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후</strong>
<strong>日, 강제징용 판결에 '경제보복'...꽂꽂 얼어붙은 한일관계</strong>
<strong>트럼프 밀착취재 美기자 "김정은, 폐병환자처럼 헐떡대"</strong>
<strong>한국·바른미래, '北어선 사건' 국조요구서 공동 제출(종합)</strong>
<strong>"권리당원 확보에 올인"...민주당 '7월 전쟁'으로 지역별 공천경쟁 본격화</strong>
```



### ■ Text 뽑아내기

```
import urllib.request
import bs4

url = "http://news.naver.com/"
html = urllib.request.urlopen(url)

bs_obj = bs4.BeautifulSoup(html, "html.parser")

mlist2_no_bg = bs_obj.find("ul", {"class": "mlist2 no_bg"})
lis = mlist2_no_bg.findAll("li")

for li in lis:
    strong = li.find("strong")
    print(strong.text)  # ----- strong 태그에 들어 있는 text만 뽑아냄
```

### 결과

文 "난 중요치 않다" 조연 자처...북·미 DMZ 회동 막전막후  
日, 강제징용 판결에 '경제보복'...공공 얼어붙은 한일관계  
트럼프 밀착취재 美기자 "김정은, 폐병환자처럼 혈떡대"  
한국·바른미래, '北어선 사건' 국조요구서 공동 제출(종합)  
"권리당원 확보에 올인"...민주당 '7월 전쟁'으로 지역별 공천경쟁 본격화