20190701

Chapter 01 크롤링, 스크래핑

스크래핑(scraping)

-웹사이트의 특정 정보를 추출하는 기술

-공개된 정보는 대부분 HTML형식으로 되어있기 때문에 이들 중 필요한 데이터를 저장하기 위해 데이터 가공이 필요

-최근에는 절차가 까다로워져 로그인 이후 필요한 웹페이지 접근 기술 필요

ex. HTML

크롤링 (crawling)

-프로그램이 웹사이트 정보를 정기적으로 추출하는 기술

→ 크롤링을 하는 프로그램 : 크롤러 / 스파이더

웹컴포넌트 : HTML과 HTTP:HTML

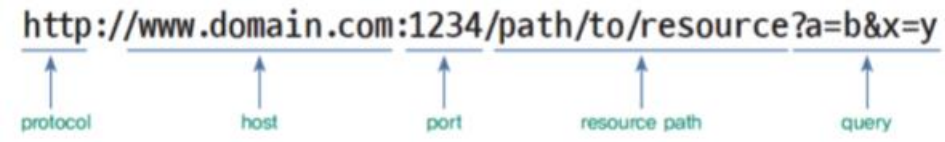
- markup language

-태그 : < >로 둘러싸여 있고, 그 안에 정보에 대한 의미를 작성. 의미가 끝나는 부분에 /사용하여 종료

-HTTP (Hypertext Transaction Protocol) : 인터넷에서 컴퓨터 간 정보를 주고받을 때 사용하는 일종의 약속 → 프로토콜(protocol)

웹의 동작 순서

- 웹브라우저를 시작하고 주소정보를 입력. URL(Uniform Resource Locator)



- 일반적으로 컴퓨터는 인터넷 프로토콜 주소(Internet Protocol address, IP address)를 가짐

- 도메인 네임 서버(Domain Name Server, DNS)를 운영해 도메인 네임과 IP주소를 연결

웹상의 정보를 추출하는 방법

- urllib 라이브러리를 사용 → HTTP / FTP를 이용해 데이터를 다운로드

- urllib는 URL을 다루는 모듈을 모아 놓은 패키지

→ 특히 urllib.request는 웹 사이트에 있는 데이터에 접근하는 기능을 제공

*** 설치 : pip install request, pip install beautifulsoup4**

함수	설명
urlretrieve(url, name)	URL주소의 파일을 다운로드
urlopen()	곧바로 파일을 저장하지 않고 메모리상에 load

1. urlopen() 은 binary 형태로 저장됨 (b'로 시작)
2. .read() 메서드 통해서 읽어줘야함
3. .decode('utf-8')을 통해 문자열로 바꿀 수 있음

cf.

<a class = >

a태그가 보통 링크 주소 href= 속성 주소로 이동

-20190701 스크래핑, 크롤링.ipynb

urllib.request를 사용해서 데이터 다운로드하기

urllib.request.urlretrieve()

In [40]:
```
import urllib.request #패키지 실행
url = "http://uta.pw/shodou/img/28/214.png" #다운로드할 파일 주소
savename = "test_download.png" #저장할 파일 이름
urllib.request.urlretrieve(url, savename) #파일 다운로드
```

Out[40]:
```
('test_download.png', <http.client.HTTPMessage at 0x217a70030b8>)
```

이 때 저장은 임시폴더에 저장됨

위치를 지정해서 저장하고 싶을 때 --> 주소도 입력

In [41]:
```
urllib.request.urlretrieve(url, "C:/Users/Affinity/Downloads/test_downloads.png")
```

Out[41]:
```
('C:/Users/Affinity/Downloads/test_downloads.png',
 <http.client.HTTPMessage at 0x217a70034e0>)
```

urllib.request.urlopen()

In [39]:
```
#메모리 상에 load 후 바이너리파일로 변환하여 파일을 저장

url = "http://uta.pw/shodou/img/28/214.png"
#파일 메모리에 불러오기
memory = urllib.request.urlopen(url).read() # url 리소스를 열로 read
#메모리에 있는 걸 파일로 저장
with open("C:/Users/Affinity/Downloads/test_downloadsN.png", mode = "wb") as f : #w :
    f.write(memory)
```

cf. with~ as문 --> .close()를 사용하지 않아도 됨

In [37]:
```
site="https://www.data.go.kr/dataset/fileDownload.do?atchFileId=FILE_000000001455071&f
urllib.request.urlretrieve(site, "C:/Users/Affinity/Downloads/ba20180629.csv")
```

Out[37]:
```
('C:/Users/Affinity/Downloads/ba20180629.csv',
 <http.client.HTTPMessage at 0x217a7003128>)
```

In [47]:
```
url = "http://api.aoikujira.com/ip/ini" #web 문서를 저장
savename = "C:/Users/Affinity/Downloads/hi.csv"
tmp=urllib.request.urlopen(url).read() #이진파일의 형태, text로 바꿔야함
print(tmp) # b'로 시작 --> 바이너리
print("₩n")
print(tmp.decode('utf-8'))# --> 문자열 형태로 반환하여 출력
```

```
b'[ip]₩nAPI_URI=http://api.aoikujira.com/ip/get.php₩nREMOTE_ADDR=115.88.249.138₩nREMO
TE_HOST=115.88.249.138₩nREMOTE_PORT=35598₩nHTTP_HOST=api.aoikujira.com₩nHTTP_USER_AGE
NT=Python-urllib/3.6₩nHTTP_ACCEPT_LANGUAGE=₩nHTTP_ACCEPT_CHARSET=₩nSERVER_PORT=80₩nFO
RMAT=ini₩n₩n'


[ip]
API_URI=http://api.aoikujira.com/ip/get.php
REMOTE_ADDR=115.88.249.138
REMOTE_HOST=115.88.249.138
REMOTE_PORT=35598
HTTP_HOST=api.aoikujira.com
HTTP_USER_AGENT=Python-urllib/3.6
HTTP_ACCEPT_LANGUAGE=
HTTP_ACCEPT_CHARSET=
SERVER_PORT=80
FORMAT=ini
```

urlopen --> decode로 text화

In [50]:
```
url = "https://www.fun-coding.org/crawl_basic2.html"
res = urllib.request.urlopen(url) # URL 리소스를 열기
data = res.read() # 바이너리 데이터로 읽어 들이기

#바이너리를 문자열로 변환(HTML소스 불러오기)
text = data.decode('utf-8')
print(text)
```

```
        <meta name='title' content='웹크롤링 기본:  크롤링(crawling) 이해 및 기본 -
잔재미코딩'>
        <meta name="description" content="잔재미코딩은 IT 교육 컨텐츠와 강의 전문
연구소입니다.">
        <meta name="keywords" content='웹크롤링 기본, 크롤링(crawling) 이해 및 기
본'>
        <meta name="author" content="Dave Lee">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="shortcut icon" href="style/images/favicon.png">
        <link href="style/css/bootstrap.css" rel="stylesheet">
        <link href="style/css/settings.css" rel="stylesheet">
        <link href="style/css/owl.carousel.css" rel="stylesheet">
        <link href="style/js/google-code-prettify/prettify.css" rel="stylesheet">
        <link href="style/js/fancybox/jquery.fancybox.css" rel="stylesheet" type="t
ext/css" media="all" />
        <link href="style/js/fancybox/helpers/jquery.fancybox-thumbs.css?v=1.0.2" r
el="stylesheet" type="text/css" />
        <link href="style.css" rel="stylesheet">
        <link href="style/css/color/blue.css" rel="stylesheet">
```

cf. 이진데이터 프린트

In [52]:
```
print(data)
```

```
b'₩n    <!DOCTYPE html>₩n    <html>₩n    <head>₩n        <meta charset="utf-8">₩n
<title>₩xec₩x9b₩xb9₩xed₩x81₩xac₩xeb₩xa1₩xa4₩xeb₩xa7₩x81 ₩xea₩xb8₩xb0₩xeb₩xb3₩xb8:
₩xed₩x81₩xac₩xeb₩xa1₩xa4₩xeb₩xa7₩x81(crawling) ₩xec₩x9d₩xb4₩xed₩x95₩xb4 ₩xeb₩xb0₩x8
f ₩xea₩xb8₩xb0₩xeb₩xb3₩xb8 - ₩xec₩x9e₩x94₩xec₩x9e₩xac₩xeb₩xaf₩xb8₩xec₩xbd₩x94₩xeb₩x
94₩xa9</title>₩n        <meta name=₩'title₩' content=₩'₩xec₩x9b₩xb9₩xed₩x81₩xac₩xeb
₩xa1₩xa4₩xeb₩xa7₩x81 ₩xea₩xb8₩xb0₩xeb₩xb3₩xb8:  ₩xed₩x81₩xac₩xeb₩xa1₩xa4₩xeb₩xa7₩x8
1(crawling) ₩xec₩x9d₩xb4₩xed₩x95₩xb4 ₩xeb₩xb0₩x8f ₩xea₩xb8₩xb0₩xeb₩xb3₩xb8 - ₩xec₩x
9e₩x94₩xec₩x9e₩xac₩xeb₩xaf₩xb8₩xec₩xbd₩x94₩xeb₩x94₩xa9₩'>₩n        <meta name="desc
ription" content="₩xec₩x9e₩x94₩xec₩x9e₩xac₩xeb₩xaf₩xb8₩xec₩xbd₩x94₩xeb₩x94₩xa9₩xec
₩x9d₩x80 IT ₩xea₩xb5₩x90₩xec₩x9c₩xa1 ₩xec₩xbb₩xa8₩xed₩x85₩x90₩xec₩xb8₩xa0₩xec₩x99₩x
80 ₩xea₩xb0₩x95₩xec₩x9d₩x98 ₩xec₩xa0₩x84₩xeb₩xac₩xb8 ₩xec₩x97₩xb0₩xea₩xb5₩xac₩xec₩x
86₩x8c₩xec₩x9e₩x85₩xeb₩x8b₩x88₩xeb₩x8b₩xa4.">₩n        <meta name="keywords" conten
t=₩'₩xec₩x9b₩xb9₩xed₩x81₩xac₩xeb₩xa1₩xa4₩xeb₩xa7₩x81 ₩xea₩xb8₩xb0₩xeb₩xb3₩xb8, ₩xed
₩x81₩xac₩xeb₩xa1₩xa4₩xeb₩xa7₩x81(crawling) ₩xec₩x9d₩xb4₩xed₩x95₩xb4 ₩xeb₩xb0₩x8f ₩x
ea₩xb8₩xb0₩xeb₩xb3₩xb8₩'>₩n        <meta name="author" content="Dave Lee">₩n
<meta name="viewport" content="width=device-width, initial-scale=1.0">₩n        <li
nk rel="shortcut icon" href="style/images/favicon.png">₩n        <link href="style/
css/bootstrap.css" rel="stylesheet">₩n        <link href="style/css/settings.css" r
el="stylesheet">₩n        <link href="style/css/owl.carousel.css" rel="stylesheet">
₩n        <link href="style/js/google-code-prettify/prettify.css" rel="stylesheet">
```

매개변수 추가해 요청 전송

https://www.data.go.kr/search/index.do?
index=DATA&query=&currentPage=1&countPerPage=10

쿼리 ~ 출력 옵션

index = 탭 이름

query = 검색어

currentPage = 출력되는 페이지

counterPerPage = 출력되는 개수

ex. 1개만

urllib.parse.urlencode()

parse + urlencode를 통해 query값 형태--> key값 = value값 형태로 반환됨

In [73]:
```
import urllib.request
import urllib.parse

API = "https://www.data.go.kr/search/index.do"  # 데이터 기본 주소, 물음표 이전

#매개변수(딕셔너리형)를 URL 인코딩
value = {"currentPage" : "3"}
params = urllib.parse.urlencode(value)

params
```

Out[73]: 'currentPage=3'

urllib.parse.urlencode() 참고

옵션 값에 들어갈 것들을 한글로 그냥 쓰면 안 됨,,! ex. params = "query=관광"

한글로 썼을 때 반드시 urllib.parse.urlencode()를 통해 아스키처리를 해야 함.

일반적으로는(숫자, 영어) 그냥 '옵션=값' 형태로 문자열 입력해도 가능

In [75]:
```
url = API + "?" + params
#API에 ?까지 쓰고 결합해도 됨(only for query)


data = urllib.request.urlopen(url).read() # 바이너리 형태로 읽어들임
text = data.decode('utf-8') #텍스트 형태로 읽기
print(text)
```

```
<!DOCTYPE HTML>
<html lang="ko">
        <head>



<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="keywords" content="정부공공API, 파일데이터, openapi, 표준데이터, file D
ata" />
```

query 값에 들어갈 내용을 직접 입력해서 여러 개의 결과를 반환하는 프로그램 만들기 (while)

In [91]:
```
import sys
import urllib.request as req
import urllib.parse as parse

text = []
API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
# while문 바깥에 있는 게 효율적

while (True) :
    regionNumber = input("USAGE : download-forecast-arvg : ")
#반복구문 종료 조건
    if (regionNumber.upper() == "EXIT") :
        break
    elif int(regionNumber) not in [108,109,105,131,133,146,156,134,159,184] :
        continue


    values = {"stnId" : regionNumber}
    params = parse.urlencode(values)
    url = API + "?" + params
    print("URL = ", url)

    data = req.urlopen(url).read()
    text.append(data.decode('utf-8'))
```

```
USAGE : download-forecast-arvg : 146
URL =  http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=146
USAGE : download-forecast-arvg : 59
USAGE : download-forecast-arvg : 159
URL =  http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=159
USAGE : download-forecast-arvg : exit
```

for문으로 옵션 값을 리스트로 지정해 데이터 가져오기

In [94]:
```
text = []
API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
for i in [108,109,105,131,133,146,156,134,159,184] :
    values = {"stnId" : i}
    params = parse.urlencode(values)
    url = API + "?" + params
    data = req.urlopen(url).read()
    text.append(data.decode('utf-8'))
```

BeautifulSoup 로 스크래핑하기

-스크래핑 : 웹사이트에서 데이터를 추출하여 원하는 정보를 얻어내는 것

-HTML 과 XML 분석을 해주는 라이브러리

-자체로 다운로드 기능은 없음

markup parser	설명
html.parser	기본옵션으로 빠르지만 유연하지 못함 (단순한 html 문서에서 사용)
lxml	매우 빠르고 유연
xml	xml 파일에만 사용
html5lib	매우 느리지만 유연 (구조가 복잡한 html 문서에 사용)

Beautiful Soup로 스크래핑하기

In [124]:
```
from bs4 import BeautifulSoup

aaa='''
<html><head> 너무신난당!!!!!!!!!!!!!!!</head>
<body>
<h1>저는 금요일에</h1>
<p>방콕에 갑니다~!</p>

<p>기대가 됩니다!~</p>
</body></html>'''

soup = BeautifulSoup(aaa, "html.parser") # Beautiful 인스턴스 생성
```

.prettify() 메서드를 통해 들여쓰기 확인 가능 --> 상위태그 / 하위태그 구분 쉬움

In [108]:
```
print(soup.prettify())
```

```
<html>
 <head>
  너무신난당!!!!!!!!!!!!!!!
 </head>
 <body>
  <h1>
   저는 금요일에
  </h1>
  <p>
   방콕에 갑니다~!
  </p>
  <p>
   기대가 됩니다!~
  </p>
 </body>
</html>
```

BeautifulSoup()으로 parsing 한 instance는 상위태그부터 하위태그 순으로 작성하면 됨

In [116]:
```
#원하는 부분 추출
h = soup.html.head
h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling



p2
```

Out[116]:
```
<p>기대가 됩니다!~</p>
```

--> next_sibling : 태그 종류가 달라도 관계 없음. 다음에 위치한 형제값을 데려옴

1st 첫 번째 p태그 다음 공백 줄바꿈 문자

2nd 두 번째 p 태그

In [117]:
```
soup.body
```

Out[117]:
```
<body>
<h1>저는 금요일에</h1>
<p>방콕에 갑니다~!</p>
<p>기대가 됩니다!~</p>
</body>
```

--> head, body는 하나씩 밖에 없기 때문에 html 생략해줘도 무방

.string : 앞 뒤 <> 태그 형식 제외하고 문자열만 가져오기

In [115]:
```
print("h = {}".format(h.string))
print(h1.string)
print(p1.string)
print(p2.string)
```

```
h =  너무신난당!!!!!!!!!!!!!!!
저는 금요일에
방콕에 갑니다~!
기대가 됩니다!~
```

추가

.next_sibling , .previous_sibling

.next_sibling ~ .previous_sibling~ 태그가 써진 형태 살펴보기 필수 ~

구조가 줄바꿈인지 아닌지에 따라 다른 결과

In [143]:
```
soup2='''<body>
<h1>저는 금요일에</h1><p>방콕에 갑니다~!</p><p>기대가 됩니다!~</p>
</body>'''
soup22 = BeautifulSoup(soup2, 'html.parser')
print(soup22.body.p.previous_sibling)
print(soup22.body.p.next_sibling)
```

```
<h1>저는 금요일에</h1>
<p>기대가 됩니다!~</p>
```

띄어쓰기 안한 경우 한 번 next_ or previous_sibling 은 ₩n 이 아닌 그 전 값!

.find() 메서드

In [160]:
```
# —>
soup3='''<html>
<head>
너무신난당!!!!!!!!!!!!!!!
</head>
<body>
<h1>
저는 금요일에
</h1>
<p id ='ex1'>
방콕에 갑니다~!
</p>
<p id = 'ex2'>
기대가 됩니다!~
</p>
</body>
</html>'''

#같은 p
```

같은 p태그여도 속성 (ex:id)에 따라서 구분 가능함

In [161]:
```
soup33 = BeautifulSoup(soup3, 'html.parser')
P2 = soup33.find("p", id = "ex2")

print(P2)
```

```
<p id="ex2">
기대가 됩니다!~
</p>
```

파일 불러오기

In [167]:
```
with open('C:/Users/Affinity/Downloads/modeule04/ch01/HTML_Exam.html', 'r', encoding =
    soup = BeautifulSoup(html1, 'html.parser')
```

In [170]:
```
print(soup.prettify())
```

```
 </title>
</head>
<body>
 <div>
  <p>
   a
  </p>
  <p>
   b
  </p>
  <p>
   c
  </p>
 </div>
 <div class="ex_class">
  <p>
   d
  </p>
  <p>
   e
```

가장 처음에 나오는 p, div를 하고 싶다면 상위 태그는 사용하지 않아도 됨

(특정 태그를 뽑으려면 생략 안하는 방법이 더 바람직)

In [188]:
```
Tag = soup.div
print(Tag)
```

```
<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>
```

In [186]:
```
soup.p
```

Out[186]:
```
<p>a</p>
```

In [185]:
```
soup.div.next_sibling.next_sibling
```

Out[185]:
```
<div class="ex_class">
<p>d</p>
<p>e</p>
<p>f</p>
</div>
```

해당하는 태그 모두 찾기 : .find_all() 메서드 ¶

find_all은 리스트를 객체로 돌려줌, beautifulsoup의 instance가 아님!

따라서 [ ] 색인을 사용

In [176]:
```
soup.find_all('div')
```

Out[176]:
```
[<div>
 <p>a</p>
 <p>b</p>
 <p>c</p>
 </div>, <div class="ex_class">
 <p>d</p>
 <p>e</p>
 <p>f</p>
 </div>, <div id="ex_id">
 <p>g</p>
 <p>h</p>
 <p>i</p>
 </div>]
```

In [197]:
```
soup.find_all('div')[2]
```

Out[197]:
```
<div id="ex_id">
<p>g</p>
<p>h</p>
<p>i</p>
</div>
```

★ class 속성 find로 찾기 ★

class는 예약어여서 .find()에서 class = 로 찾을 수 없음.

https://godoftyping.wordpress.com/2017/06/24/python-beautifulsoup/

1st : find에서 class_ 라고 작성하기

In [192]:
```
soup.find(class_='ex_class') # or find_all
```

Out[192]:
```
<div class="ex_class">
<p>d</p>
<p>e</p>
<p>f</p>
</div>
```

2nd : { } dictionary 활용하기

In [191]:
```
soup.find('', {'class' : 'ex_class'}) #or find_all
```

Out[191]:
```
<div class="ex_class">
<p>d</p>
<p>e</p>
<p>f</p>
</div>
```

네이버

In [214]:
```
from bs4 import BeautifulSoup
from urllib.request import urlopen

soup = BeautifulSoup(urlopen('http://www.naver.com'), 'html.parser')

a = soup.find_all('a', class_ = 'ah_da')

a
```

Out[214]:
```
[<a class="ah_da" data-clk="lve.kwdhistory" href="http://datalab.naver.com/keyword/
realtimeDetail.naver?datetime=2019-07-01T15:40:00&amp;query=%EC%8B%9C%EB%AA%AC%EC%8
A%A4+%EC%9B%A8%EB%94%A9+%ED%94%84%EB%A1%9C%EB%AA%A8%EC%85%98&amp;where=main">
 <span class="blind">데이터랩 그래프 보기</span>
 <span class="ah_ico_datagraph"></span>
 </a>,
 <a class="ah_da" data-clk="lve.kwdhistory" href="http://datalab.naver.com/keyword/
realtimeDetail.naver?datetime=2019-07-01T15:40:00&amp;query=%EC%86%A1%EC%A4%91%EA%B
8%B0+%ED%83%88%EB%AA%A8%EC%82%AC%EC%A7%84&amp;where=main">
 <span class="blind">데이터랩 그래프 보기</span>
 <span class="ah_ico_datagraph"></span>
 </a>,
 <a class="ah_da" data-clk="lve.kwdhistory" href="http://datalab.naver.com/keyword/
realtimeDetail.naver?datetime=2019-07-01T15:40:00&amp;query=%EC%9C%A0%ED%95%9C%EC%9
6%91%ED%96%89&amp;where=main">
 <span class="blind">데이터랩 그래프 보기</span>
 <span class="ah_ico_datagraph"></span>
 </a>,
 <a class="ah_da" data-clk="lve.kwdhistory" href="http://datalab.naver.com/keyword/
```

기상청

In [247]:
```
from bs4 import BeautifulSoup
import urllib.request as req

URL = 'http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp'
soup = BeautifulSoup(req.urlopen(URL), 'html.parser')

#첫 번째 title과 첫 번째 wf 찾기
title = soup.find('title').string
wf = soup.find('wf').string
print(title,'₩n', wf)
```

```
기상청 육상 중기예보
 장마전선의 영향으로 6~7일에 남부지방과 제주도에 비가 오겠습니다. <br />그 밖의 날은 고기압의 영향으로 맑은 날이 많겠습니다.<br />기온은 평년(최저기온: 18~22℃, 최고기온: 25~30℃) 보다 4~5일에는 조금 높겠고, 그 밖의 날은 비슷하겠습니다.<br />강수량은 평년(5~18mm)보다 남부지방과 제주도는 많겠고, 중부지방은 적겠습니다.<br /><br />* 한편, 장마전선은 6일부터 제주도남쪽먼바다에서 다시 북상할 것으로 예상되나, 북태평양고기압의 확장 정도에 따라 장마전선의 위치와 강수 영역이 달라질 수 있으니, 앞으로 발표되는 기상정보를 참고하기 바랍니다.<br />* 내륙지역을 중심으로 낮 기온이 30도 이상 올라 덥겠으니, 보건, 축산 등 폭염피해에 유의하기 바랍니다.
```

CSS 선택자 (cascading style sheets)

-CSS 는 웹문서의 전반적인 스타일을 미리 저장해 둔 스타일 시트로, 문서 전체의 일관성을 유지할 수 있고 세세한 스타일 지정의 필요를 줄어들게 함

-CSS 선택자를 지정해서 원하는 요소를 추출

- copy selector 를 통해 형식 가져오기

-클래스 / id 의 경우 띄어쓰기가 있는 속성은 공백 앞부분만 써줘도 검색 가능하다.

메소드	설명
soup.select_one(<선택자>)	CSS 선택자로 요소 하나를 추출
soup.select(<선택자>)	CSS 선택자로 요소 여러 개를 리스트로 추출

CS 선택자

.select_one() / .select()

"상위 태그 > 하위태그"

id : #의 형태 /// class : .의 형태

In [249]:
```
with open('C:/Users/Affinity/Downloads/module04/ch01/CSS_Exam.html', encoding = 'utf-8
    soup = BeautifulSoup(html11, 'html.parser')
```

In [252]:
```
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8"/>
  <meta content="CSS, HTML" name="keywords"/>
  <meta content="width=device-width, initial-scale=1.0" name="viewport"/>
  <title>
   CSS 선택자
  </title>
 </head>
 <body>
  <div id="meigen">
   <h1>
    파이썬 프로그램
   </h1>
   <ul class="items">
    <li>
     <a href="/Python/Basics">
      Python 기초
```

하나만 선택

```
In [250]: soup.select_one("div#meigen > h1").string
Out[250]: '파이썬 프로그램'
```

전체 선택 (리스트로 반환)

```
In [266]: li_list = soup.select('div#meigen > ul.items > li')

In [267]: li_list
Out[267]: [<li><a href="/Python/Basics">Python 기초</a></li>,
 <li><a href="/Python/Gui">GUI 프로그래밍</a></li>,
 <li><a href="/Python/Data">Python 데이타</a></li>,
 <li><a href="/Python/Django">Django 기초</a></li>,
 <li><a href="/Python/Applications">Python 활용</a></li>,
 <li><a href="/Python/Tips">Python 팁</a></li>,
 <li><a href="/Home/Contact">Contact</a></li>,
 <li><a href="javascript:showSearch()"><i aria-hidden="true" class="fa fa-search"></i
>검색</a></li>]

In [268]: [i.string for i in li_list]
Out[268]: ['Python 기초',
 'GUI 프로그래밍',
 'Python 데이타',
 'Django 기초',
 'Python 활용',
 'Python 팁',
 'Contact',
 None]
```

cf.

1. font Awesome(폰트 아이콘)의 경우

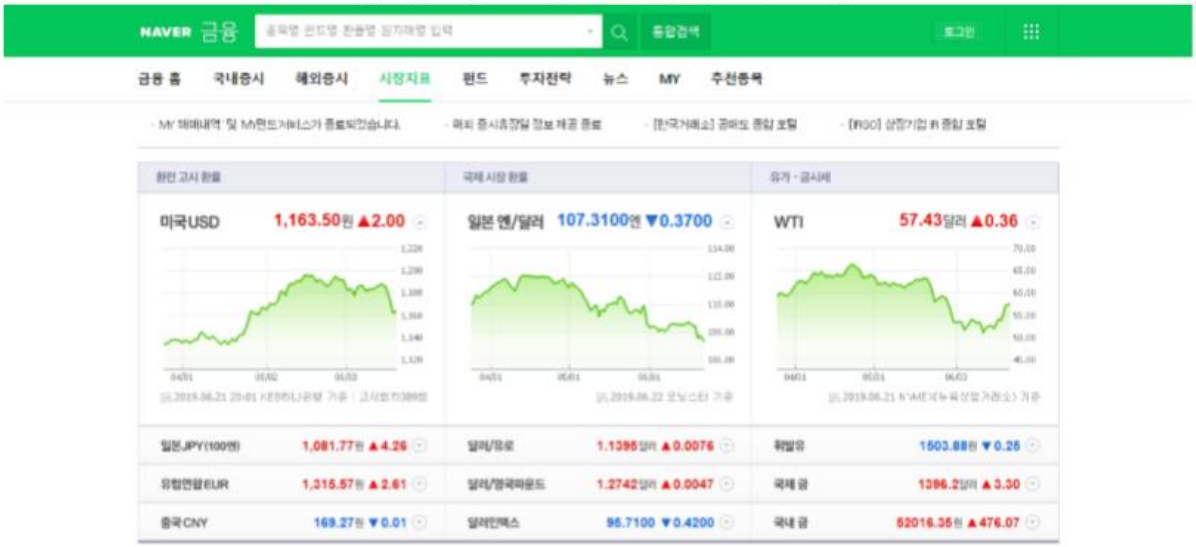
```
<i class="" title="아이콘 설명" aria-hidden="true"></i>
//안에 내용은 읽지 않고, 스크린리더에서만 읽을 수 있는 아이콘 설명이 들어간다. (title
 / aria-hidden 속성 중요)
```

  + aria-hidden 속성 → 숨기기

❖ 네이버 금융에서 환율 정보 추출

- 네이버 금융의 시장 지표 페이지

https://finance.naver.com/marketindex/



개발자 도구-> 오른쪽 마우스 -> copy selector 를 선택한 후 공통 분모 묶기

> 는 바로 직계! 하위!(자식) 넉넉하게 검색할 때에는 띄어쓰기만 사용

개발자 도구에서 copy selector 선택

In [296]:
```
url = "https://finance.naver.com/marketindex/"
res = req.urlopen(url)

soup = BeautifulSoup(res, 'html.parser')

#CSS 쿼리로 추출하기 ~ selector로 복사
```

In [282]:
```
soup.select('div.head_info > span.value')
```

Out[282]:
```
[<span class="value">1,159.00</span>,
 <span class="value">1,069.43</span>,
 <span class="value">1,312.68</span>,
 <span class="value">169.32</span>,
 <span class="value">107.8900</span>,
 <span class="value">1.1389</span>,
 <span class="value">1.2695</span>,
 <span class="value">95.6600</span>,
 <span class="value">58.47</span>,
 <span class="value">1496.64</span>,
 <span class="value">1409.7</span>,
 <span class="value">51653.96</span>]
```

In [295]:
```
soup.select('h3.h_lst > span.blind')
```

Out[295]:
```
[<span class="blind">미국 USD</span>,
 <span class="blind">일본 JPY(100엔)</span>,
 <span class="blind">유럽연합 EUR</span>,
 <span class="blind">중국 CNY</span>,
 <span class="blind">일본 엔/달러</span>,
 <span class="blind">달러/유로</span>,
 <span class="blind">달러/영국파운드</span>,
 <span class="blind">달러인덱스</span>,
 <span class="blind">WTI</span>,
 <span class="blind">휘발유</span>,
 <span class="blind">국제 금</span>,
 <span class="blind">국내 금</span>]
```

In [365]:
```
#제목
h2 = soup.select_one('div.title > h2').string
print('****',h2,'****')

#값, 제목
value_list = soup.select('div.head_info > span.value')

title_list = soup.select('h3.h_lst > span.blind')



for i in range(len(title_list)) :
    print("{0}₩t₩t: {1}".format(title_list[i].string, value_list[i].string))
```

```
**** 환전 고시 환율 ****
미국 USD                : 1,159.00
일본 JPY(100엔)         : 1,069.43
유럽연합 EUR            : 1,312.68
중국 CNY                : 169.32
일본 엔/달러            : 107.8900
달러/유로               : 1.1389
달러/영국파운드         : 1.2695
달러인덱스              : 95.6600
WTI             : 58.47
휘발유          : 1496.64
국제 금         : 1409.7
국내 금         : 51724.25
```

190702 화

❖ CSS 선택자로 지정할 수 있는 서식

▪ 기본서식

서식	설명
*	모든 요소 선택
<요소 이름>	요소 이름 기반으로 선택
.<클래스 이름>	클래스 이름 기반으로 선택
#<id 이름>	id 속성 기반으로 선택

▪ 선택자들의 관계를 지정하는 서식

서식	설명
<선택자>, <선택자>	쉼표로 구분된 여러 개의 선택자를 모두 선택
<선택자> <선택자>	앞 선택자의 후손 중 뒤 선택자에 해당하는 것을 모두 선택
<선택자> > <선택자>	앞 선택자의 자손 중 뒤 선택자에 해당하는 것을 모두 선택
<선택자> + <선택자>	같은 계층에서 바로 뒤에 있는 요소를 선택
<선택자1> ~ <선택자2>	선택자 1 부터 선택자 2 까지의 요소를 모두 선택

> : 직계(자식)--명시적으로 바로 하위태그임을 표시

: 후손,자손-- 하위태그의 하위태그도 포함 (공백)

▪ 선택자 속성을 기반으로 지정하는 서식

서식	설명
<요소>[<속성>]	해당 속성을 가진 요소를 선택
<요소>[<속성>=<값>]	해당 속성의 값이 지정한 값과 같은 요소를 선택
<요소>[<속성>~=<값>]	해당 속성의 값이 지정한 값을 단어로 포함하고 있다면 선택
<요소>[<속성>|=<값>]	해당 속성의 값으로 시작하면 선택
<요소>[<속성>^=<값>]	해당 속성의 값이 지정한 값으로 시작하면 선택
<요소>[<속성>$=<값>]	해당 속성의 값이 지정한 값으로 끝나면 선책
<요소>[<속성>*=<값>]	해당 속성의 값이 지정한 값을 포함하고 있다면 선택

▪ 위치 또는 상태를 지정하는 서식

서식	설명
<요소>:root	루트 요소
<요소>:nth-child(n)	n번째 자식요소
<요소>:nth-last-child(n)	뒤에서 n번째 자식요소
<요소>:nth-of-type(n)	n번째 해당 종류의 요소

빨간네모는 패턴

#nth-child(n) : 전체 태그 중 n 번째 자식

#nth-of-type(n) : 해당 타입 태그의 n 번째 -- 중간에 다른 데이터가 섞여있으면 nth-of-type 과 다른 값

#이 때 n 은 1 부터 count (파이썬과 다름)

▪ 위치 또는 상태를 지정하는 서식

서식	설명
<요소>:first-child	첫 번째 자식요소
<요소>:last-child	마지막 자식 요소
<요소>:first-of-type	첫 번째 해당 종류의 요소
<요소>:last-of-type	마지막 해당 종류의 요소
<요소>:only-child	자식으로 유일한 요소
<요소>:only-of-type	자식으로 유일한 종류의 요소
<요소>:empty	내용이 없는 요소
<요소>:lang(code)	특정 언어로 code를 지정한 요소
<요소>:not(s)	s 이외의 요소
<요소>:anabled	활성화된 UI 요소
<요소>:disabled	비활성화된 UI 요소
<요소>:checked	체크돼 있는 UI 요소