크롤링 기초 정리

- 파이썬입문과 크롤링기초 부트캠프 강좌를 통해 익힌 내용 정리

> 본 기초 기술은 익히 안다는 가정 하에
> Selenium/Scrapy 기술 설명 예정이므로
> 해당 기술 이해가 없다면 파이썬입문과 크롤링기초 부트캠프 수강 필요!

크롤링 핵심 코드 패턴으로 이해하기

```
import requests
from bs4 import BeautifulSoup
res = requests.get('http://v.media.daum.net/v/20170615203441266')
soup = BeautifulSoup(res.content, 'html.parser')
mydata = soup.find('title')
print(mydata.get_text())
```

크롤링 핵심 코드 패턴으로 이해하기

```
import requests
from bs4 import BeautifulSoup
```
①

```
res = requests.get('http://v.media.daum.net/v/20170615203441266')
```
②

```
soup = BeautifulSoup(res.content, 'html.parser')
```
③

```
mydata = soup.find('title')
```
④

```
print(mydata.get_text())
```
⑤

① 라이브러리 임포트

```
import requests
from bs4 import BeautifulSoup
```

② 웹페이지 가져오기

```
res = requests.get('http://v.media.daum.net/v/20170615203441266')
```

③ 웹페이지 파싱하기

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

④ 필요한 데이터 추출하기

```
mydata = soup.find('title')
```

⑤ 추출한 데이터 활용하기

```
print(mydata.get_text())
```

필요 라이브러리

- requests
  - 웹페이지 가져오기 라이브러리
- bs4 (BeautifulSoup)
  - 웹페이지 분석(크롤링) 라이브러리

1. 라이브러리 임포트(import)

- 필요 라이브러리
  - requests
    - 웹페이지 가져오기 라이브러리
  - bs4 (BeautifulSoup)
    - 웹페이지 분석(크롤링) 라이브러리

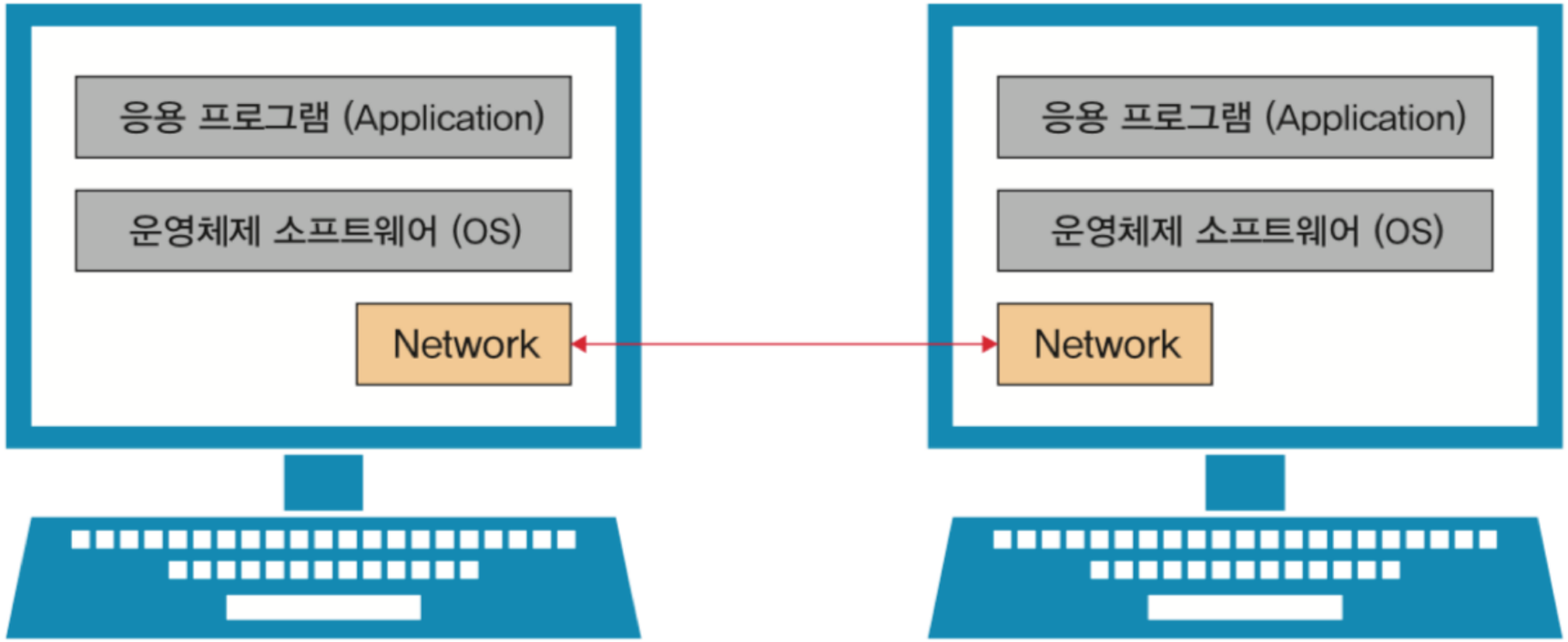
```
import requests
from bs4 import BeautifulSoup
```

2. 웹페이지 가져오기

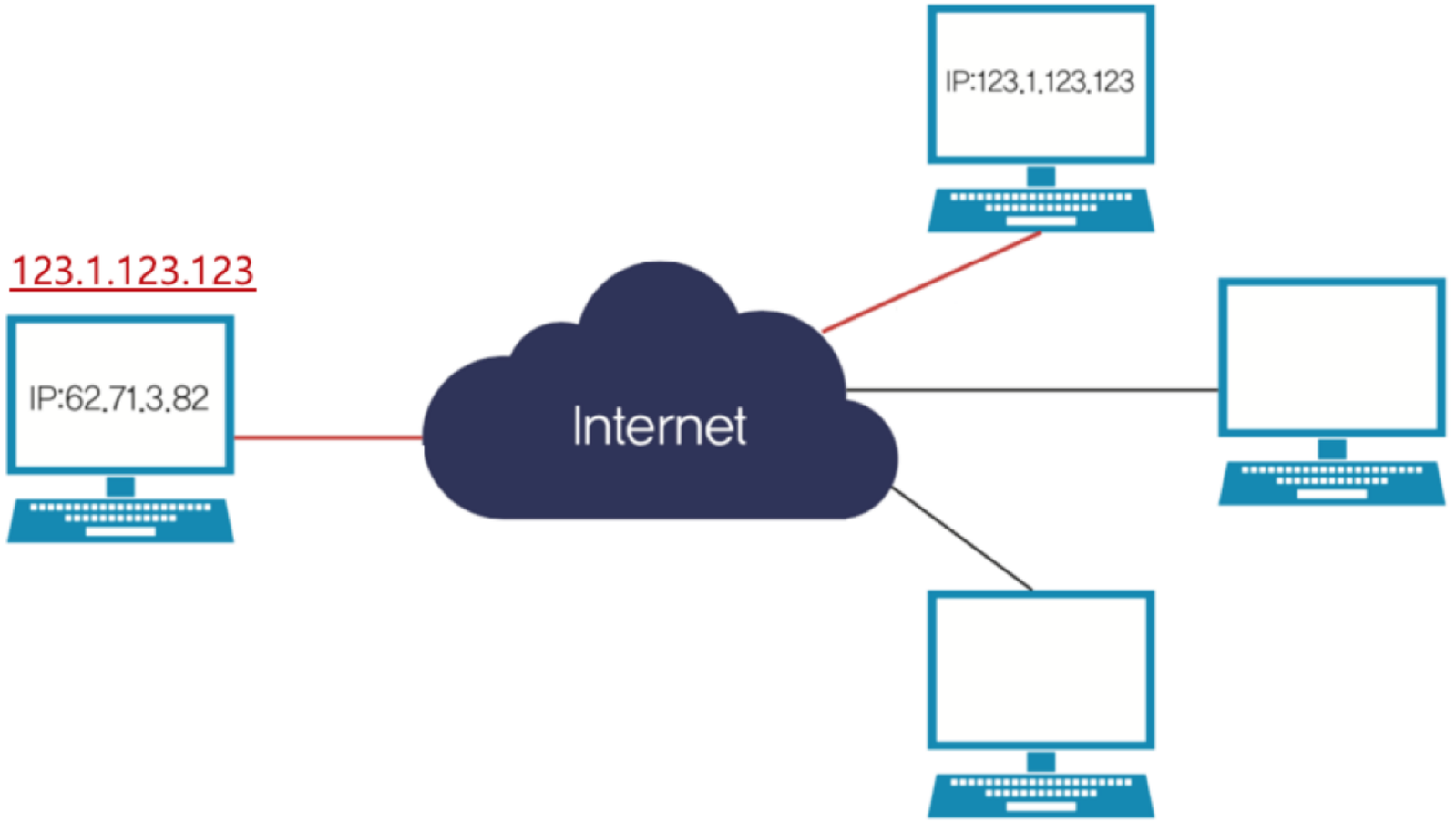
- res.content 확인해보기

```
res = requests.get('http://v.media.daum.net/v/20170615203441266')
```

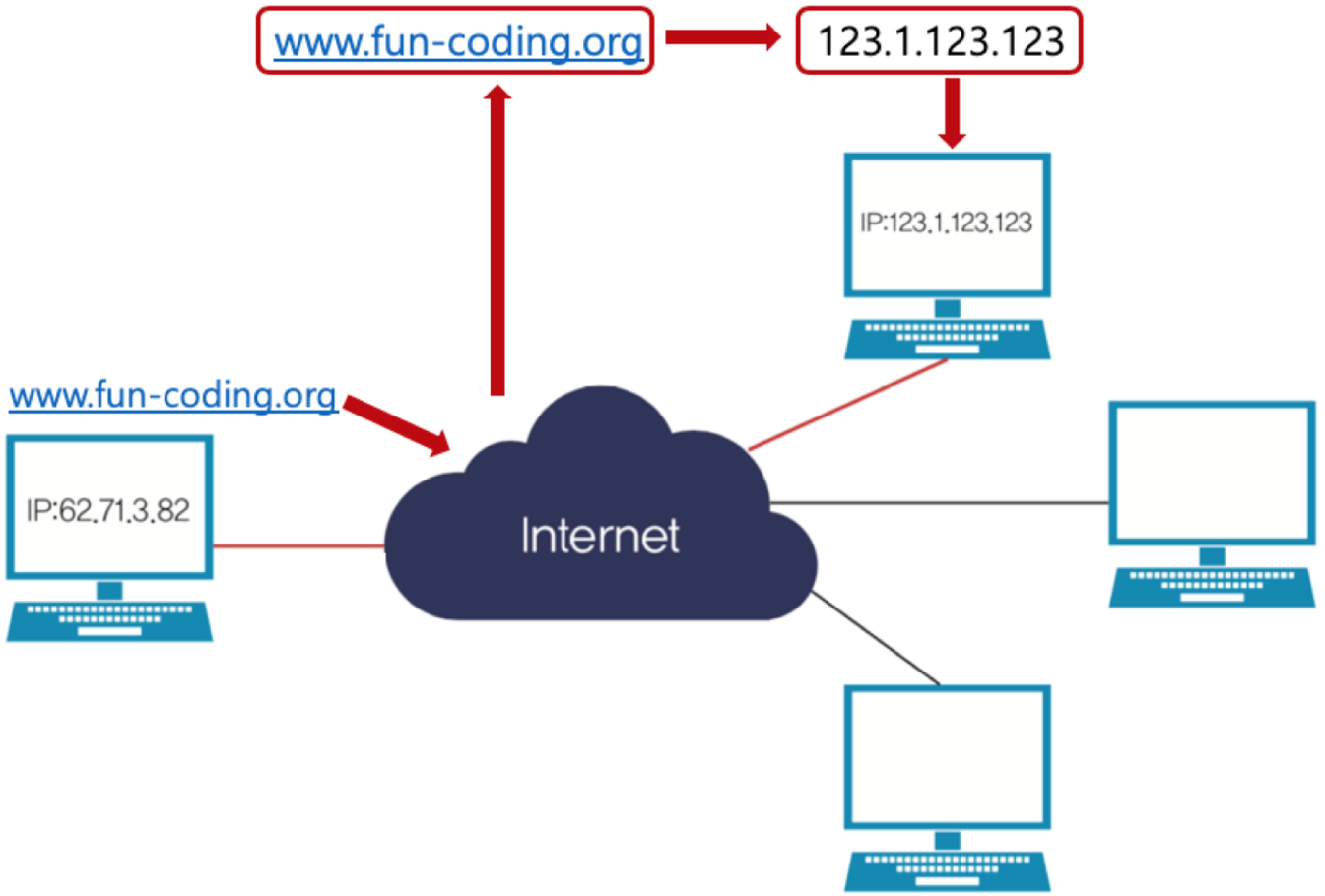
이미 알고 있지만 - 웹 기본 구조 이해1



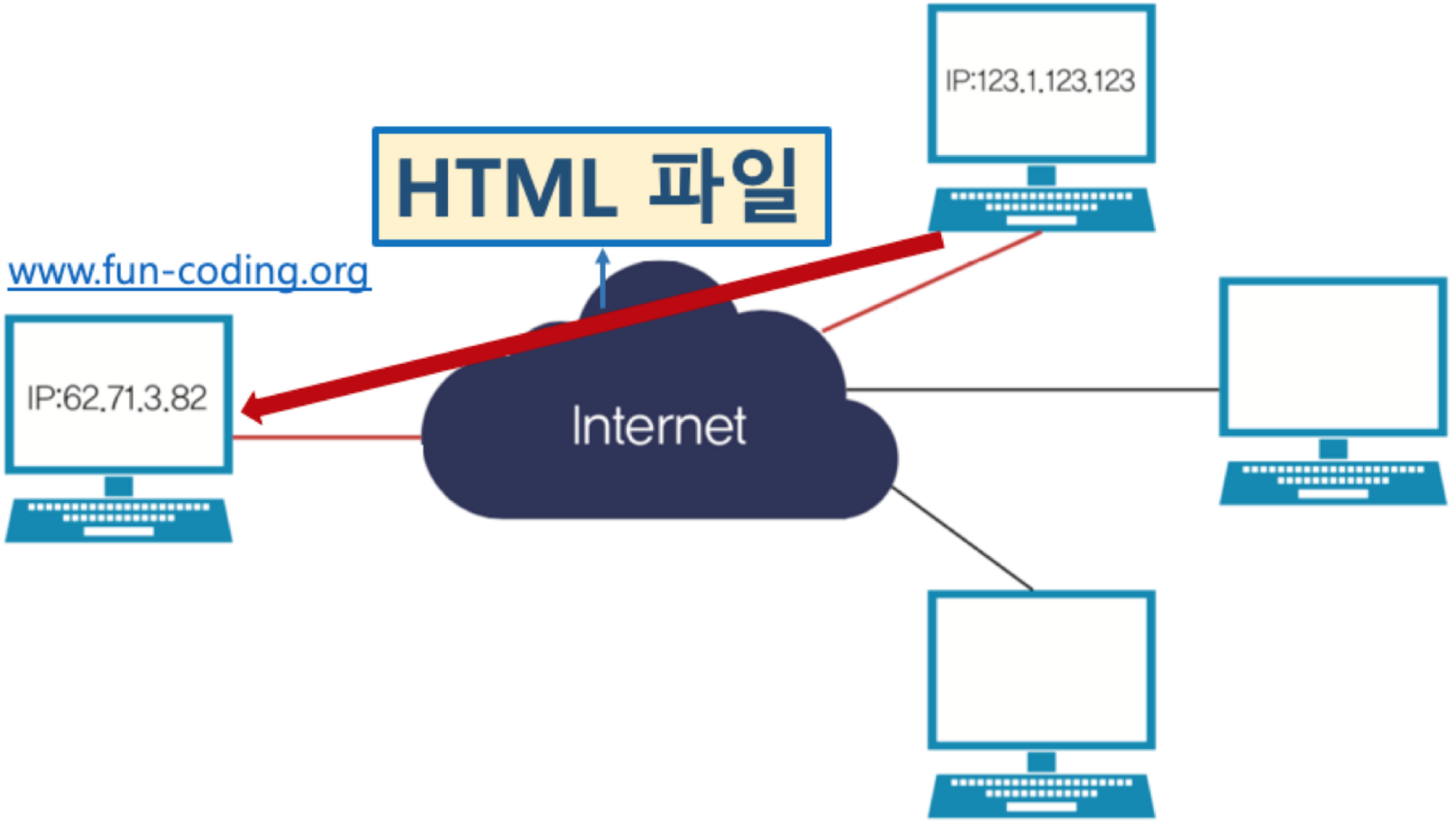
이미 알고 있지만 - 웹 기본 구조 이해2



이미 알고 있지만 - 웹 기본 구조 이해3



이미 알고 있지만 - 웹 기본 구조 이해4



HTML 파일 확인해보기

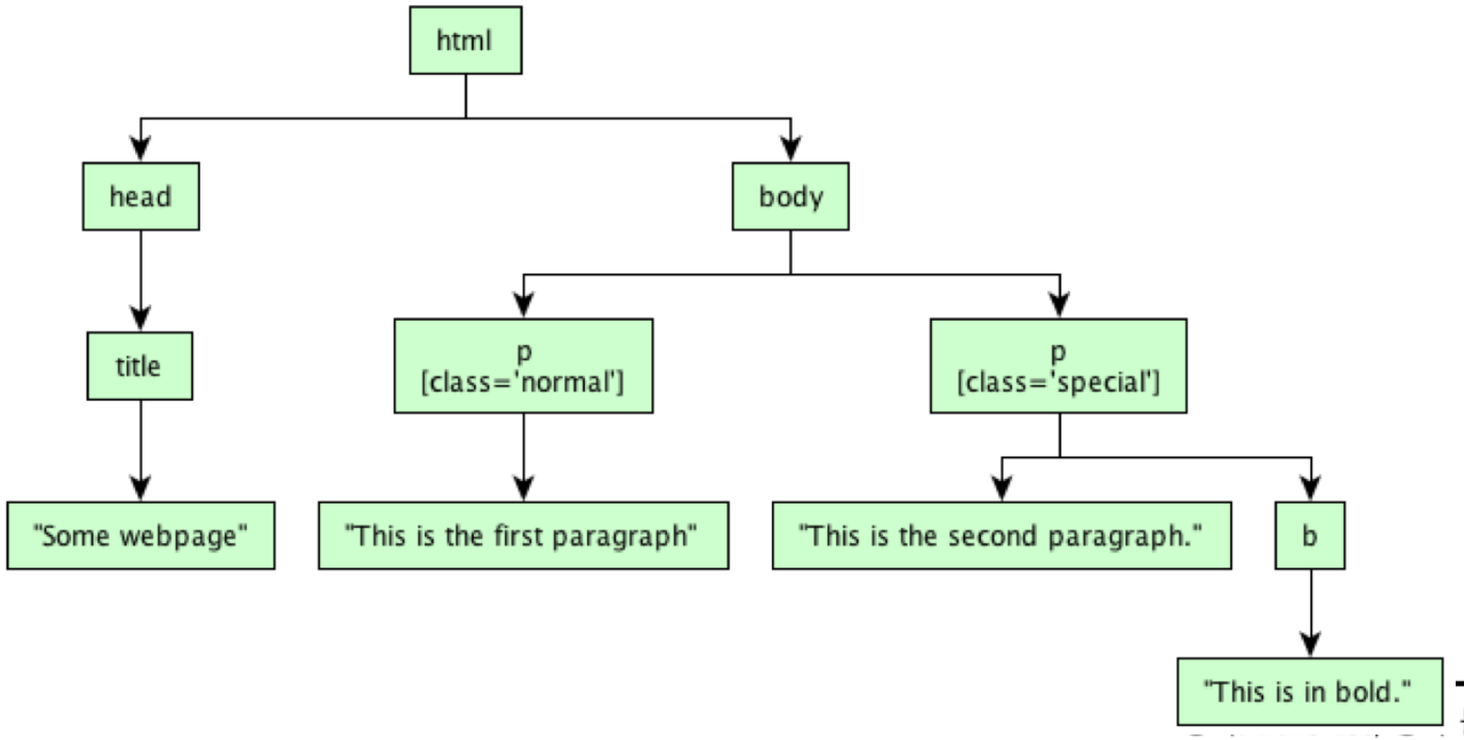
1. 웹브라우저로 확인하기
    - 오른쪽 클릭 + 페이지 소스 보기
2. 라이브러리로 확인하기

```
import requests
res = requests.get('http://v.media.daum.net/v/20170615203441266')
res.content # 변수명만 써도 쥬피터 노트북에서는 내용이 출력됩니다
```

3. 웹페이지 파싱하기

- 파싱(parsing) 이란?
    - 문자열의 의미 분석

```
<html>
  <head>
    <title>Some webpage</title>
  </head>
  <body>
    <p class="normal">This is the first paragraph</p>
    <p class="special">This is the second paragraph. <b>This is in bold.</b></p>
  </body>
</html>
```



3. 웹페이지 파싱하기

> 이것을 어떻게 일일이 코드로 만들까? -> BeautifulSoup 라이브러리가 있습니다.

- soup 에 HTML 파일을 파싱한 정보가 들어감!

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

4. 필요한 데이터 추출하기

> 이 부분이 크롤링 핵심!

- soup.find() 함수로 원하는 부분을 지정하면 됨
- 변수.get_text() 함수로 추출한 부분을 가져올 수 있음

```
mydata = soup.find('title')
print(mydata.get_text())
```

> 이를 위해 HTML 언어로 어떻게 웹페이지를 만드는지, 기본 내용을 이해할 필요가 있음!

4. 필요한 데이터 추출하기

1. 태그와 속성으로 선택하기

```
crawling_data = soup.find('h1')
crawling_data = soup.find(id='title')
crawling_data = soup.find('p', class_='cssstyle')
crawling_data = soup.find('p', attrs = {'align': 'center'})
```

4. 필요한 데이터 추출하기

2. CSS Selector로 선택하기

```
crawling_data = soup.select('html > title')
crawling_data = soup.select('div.article_view')
crawling_data = soup.select('#harmonyContainer')
crawling_data = soup.select('div#mArticle')
```

5. 추출한 데이터 활용하기

> 필요한 데이터를 변수에 넣으면 이후 활용은 프로그래밍 영역

```
print(mydata.get_text())
```

다양한 크롤링 기술

- 유용한 데이터를 크롤링할 수 있는 Open API
- 로그인이 필요한 웹페이지 크롤링 기법
- 크롤링 데이터 후처리
    - 문자열 함수, 정규 표현식
    - JSON, XML 데이터 포멧

업무 자동화 기술

- 크롤링 데이터 기반, 엑셀 보고서 만들기
- 크롤링 데이터 기반, 구글 쉬트 보고서 만들기

패턴 총정리

> 다음 코드를 우선 그대로 쓰고, 두 부분만 수정!

```
import requests
from bs4 import BeautifulSoup

res = requests.get('http://v.media.daum.net/v/20170615203441266')

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

mydata = soup.find('title')
print(mydata.get_text())
```

① 크롤링할 페이지 주소 넣기

② 필요한 데이터 추출하는 코드 넣기

추출한 데이터를 변수에 넣은 후, 원하는 프로그래밍