

Python 심화/크롤링

Session 4

NEXT X LIKELION 박채원

| Session4

1. Class 개념

2. Python 가상환경

3. Python Crawling



1. Class 개념

Class가 뭘까?

```
fishbread1 = {  
    "flavor" : "팥",  
    "price" : 1000,  
    "amount" : 1,  
}  
  
fishbread2 = {  
    "flavor" : "슈크림",  
    "price" : 1200,  
    "amount" : 1,  
}  
  
fishbread3 = {  
    "flavor" : "치즈",  
    "price" : 1500,  
    "amount" : 1,  
}
```

손님이 팥, 슈크림, 치즈 블루베리를 각각
1개씩 주문했습니다. 주문 정보를 정리해볼까요?

1) 각 블루베리를 딕셔너리의 key-value 형태로 저장

1. Class 개념

Class가 뭘까?

```
def one_more_cream():
    fishbread2["amount"] += 1
    fishbread2["price"] += 1000

def one_more_cheese():
    fishbread3["amount"] += 1
    fishbread3["price"] += 1000
```

손님이 슈크림 봉어빵과 치즈 봉어빵을 1개씩
추가 주문하였습니다. 주문 정보를 업데이트해볼까요?

2) 각각의 함수를 만들어 주문 정보를 업데이트

3) python 실행해보기

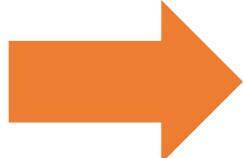
```
nunu@bagchaewon-ui-MacBookAir cstudy % python whyClass1.py
{'flavor': '팥', 'price': 1000, 'amount': 1}
{'flavor': '슈크림', 'price': 1200, 'amount': 1}
{'flavor': '치즈', 'price': 1500, 'amount': 1}
-----
추가 주문 후
{'flavor': '팥', 'price': 1000, 'amount': 1}
{'flavor': '슈크림', 'price': 2200, 'amount': 2}
{'flavor': '치즈', 'price': 2500, 'amount': 2}
```

1. Class 개념

Class가 뭘까?

이번엔 봉어빵 주문 정보를 조금 더 효율적으로 정리해볼까요?

```
fishbread1 = {  
    "flavor" : "팥",  
    "price" : 1000,  
    "amount" : 1,  
}  
  
fishbread2 = {  
    "flavor" : "슈크림",  
    "price" : 1200,  
    "amount" : 1,  
}  
  
fishbread3 = {  
    "flavor" : "치즈",  
    "price" : 1500,  
    "amount" : 1,  
}  
  
  
def one_more_cream():  
    fishbread2["amount"] += 1  
    fishbread2["price"] += 1000  
  
def one_more_cheese():  
    fishbread3["amount"] += 1  
    fishbread3["price"] += 1000
```



```
class Fishbread:  
    def __init__(self, flavor, price, amount):  
        self.flavor = flavor  
        self.price = price  
        self.amount = amount  
  
    def one_more(self):  
        self.price += 1000  
        self.amount += 1  
  
fishbread1 = Fishbread("팥", 1000, 1)  
fishbread2 = Fishbread("슈크림", 1200, 1)  
fishbread3 = Fishbread("치즈", 1500, 1)
```

클래스를 사용하면 반복적인 코드를 줄일 수 있음
같은 틀을 통해 쉽게 관리할 수 있음

1. Class 개념

Class가 뭘까?

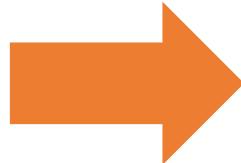
결국 Class는 봉어빵 틀이다.

봉어빵을 하나씩 빚어서 모양을 만드는 것보다 틀로 찍어냈을 때 훨씬 간단하다!

클래스



객체



1) 객체를 찍어내기 위해 Class가 필요하다

2) 객체들은 서로 영향을 주지 않으며, 고유한 성격을 가진다 (ex - 팥, 슈크림, 치즈)

1. Class 개념

Class가 뭘까?

데이터와 기능을 함께 묶는 방법을 제공

Python, C++, Java 등에서 주로 사용 (**객체지향 프로그래밍**)

이후 Django의 DB Model 작성 시 주로 사용

1. Class 개념

Class 문법 배우기



Class를 이용하여 계산기를 직접 만들어 봅시다

1. Class 실습

즐거운 실습 시간



0. 멋쟁이 사자처럼 풀더로 이동 (cd)
1. cd session4
2. code .
3. calculator.py 열기

1. Class 개념

Class 기본 배우기

코드를 따라쳐보세요!

```
1  class Calculator:
2      def __init__(self, name):
3          self.name = name
4          self.result = 0
5
6  calculator1 = Calculator("박채원")
7  calculator2 = Calculator("이영서")
8
9  print(calculator1.name)
10
11
```

__init__: 생성자 함수(멤버 변수를 정의)

self: 인스턴스 본인을 의미, 본인 스스로의 값에 접근 가능

1. Class 개념

Class 기본 배우기

Class명은 항상
대문자로 작성!

```
1 class Calculator:  
2     def __init__(self, name):  
3         self.name = name  
4         self.result = 0  
5  
6 calculator1 = Calculator("박채원")  
7 calculator2 = Calculator("이영서")  
8  
9 print(calculator1.name)  
10  
11
```

1. **Calculator**의 첫번째 매개변수에는 **self**가 포함되어 있음
2. **calculator1**은 **Calculator**의 인스턴스, 혹은 객체

1. Class 개념

Class 기본 배우기

```
1  class Calculator:  
2      def __init__(self, name):  
3          self.name = name  
4          self.result = 0  
5  
6      def add(self, num):  
7          self.result += num  
8          return self.result  
9  
10     calculator1 = Calculator("박채원")  
11     calculator1.add(2)  
12     calculator1.add(3)  
13     print(calculator1.result)  
14
```

1. **add Method**를 추가해봅시다!
2. **result** 멤버 변수에 더한 결과를 저장합니다

1. Class 실습

즐거운 실습 시간



실습 요구사항

필요한 속성 (멤버 변수)

- 현재 계산기 값 (기본값 0)
- 이름 (누구 계산기인지)
- 나이

필요한 기능 정리 (메소드)

- 덧셈 (add)
- 뺄셈 (sub)
- 곱셈 (mul)
- 나눗셈 (div)

1. Class 실습

정답 확인

```
class Calculator:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.result = 0
    def add(self, num):
        self.result += num
        return self.result
    def sub(self, num):
        self.result -= num
        return self.result
    def mul(self, num):
        self.result *= num
        return self.result
    def div(self, num):
        self.result /= num
        return self.result
```

```
# step1: 계산기 멤버 변수 정의
calculator1 = Calculator("박재원", 23)
calculator2 = Calculator("권규리", 22)
print(calculator1.name)

# step2: 계산기 기능 만들기
print(calculator1.result)
calculator1.add(3)
print(calculator1.result)
calculator1.sub(4)
print(calculator1.result)
calculator1.mul(4)
print(calculator1.result)
calculator1.div(2)
print(calculator1.result)
```

1. Class 실습

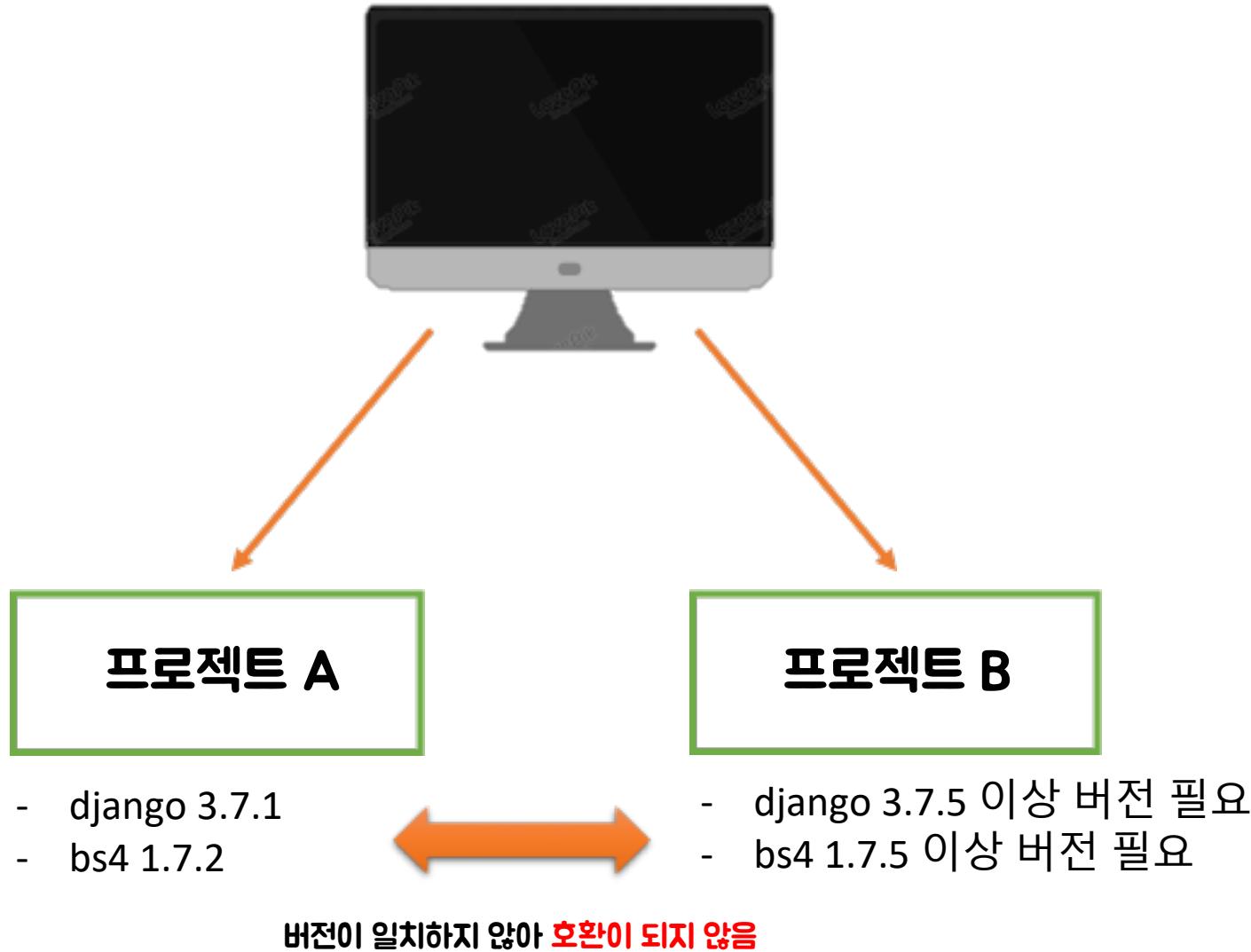
정답 확인

```
def div(self, num):
    if (num == 0) :
        print("0으로 나눌 수 없습니다")
        return None
    self.result /= num
    return self.result
```

0으로 나눠질 경우 예외 처리 추가

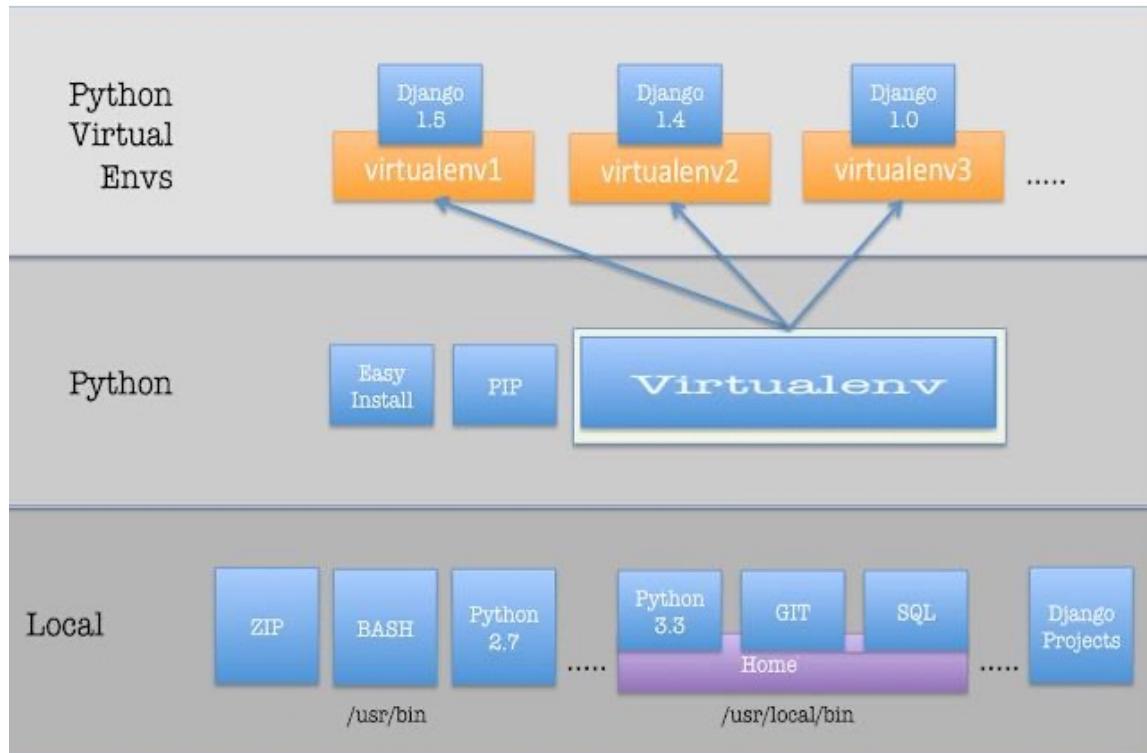
2. 파이썬 가상환경

가상환경이란?



2. 파이썬 가상환경

가상환경은 독립된 파이썬 개발 프로젝트 환경을 만드는 것



가상 환경

독립된 공간 -> **프로젝트별로 개발환경 구축**

통일된 라이브러리 버전을 사용하므로 협업 시에
버전이 충돌 날 일이 없음

가상 환경은 **독립적**이어서 서로 다른 가상 환경에
설치된 모듈들의 영향을 받지 않음

2. 파이썬 가상환경

파이썬 가상환경 종류

venv : Python 3.3 버전 이후부터 기본 모듈에 포함됨

virtualenv : Python 2 버전부터 사용해오던 가상환경 라이브러리, Python 3에서도 사용 가능

conda : Anaconda Python을 설치했을 시 사용할 수 있는 모듈

pipenv : 입문자가 사용하기에 좋은 가상환경

2. 파이썬 가상환경

pipenv



패키지 관리를 자동으로 해준다

처음에는 일단 pipenv로 시작하자!

2. 파이썬 가상환경

pip 설치/버전 확인

pip 설치 (컴퓨터 전역 설치)

Windows

1. curl <https://bootstrap.pypa.io/get-pip.py> -o get-pip.py.
2. python get-pip.py.

Mac

1. sudo apt-get install python3-pip

pip 버전 확인

pip --version

2. 파이썬 가상환경

pipenv

pipenv 설치

`pip install pipenv` (Windows)
`pip3 install pipenv` (Mac)

pipenv 버전 확인

`pipenv --version`

2. 파이썬 가상환경

pipenv

0. 멋쟁이 사자처럼 session4 폴더로 이동 (cd)

1. pipenv shell (꼭 해당 프로젝트 최상단 위치에서 생성할 것!)

2. pipenv install requests

3. pipenv install bs4

2. 파이썬 가상환경

pipenv

pipenv 명령어 정리

pipenv shell - 가상환경 생성 및 시작

exit - 가상환경 끄기

pipenv install 패키지명 - 해당 패키지 설치

pipenv uninstall 패키지명 - 해당 패키지 제거

| 쉬는 시간



| 3. 크롤링

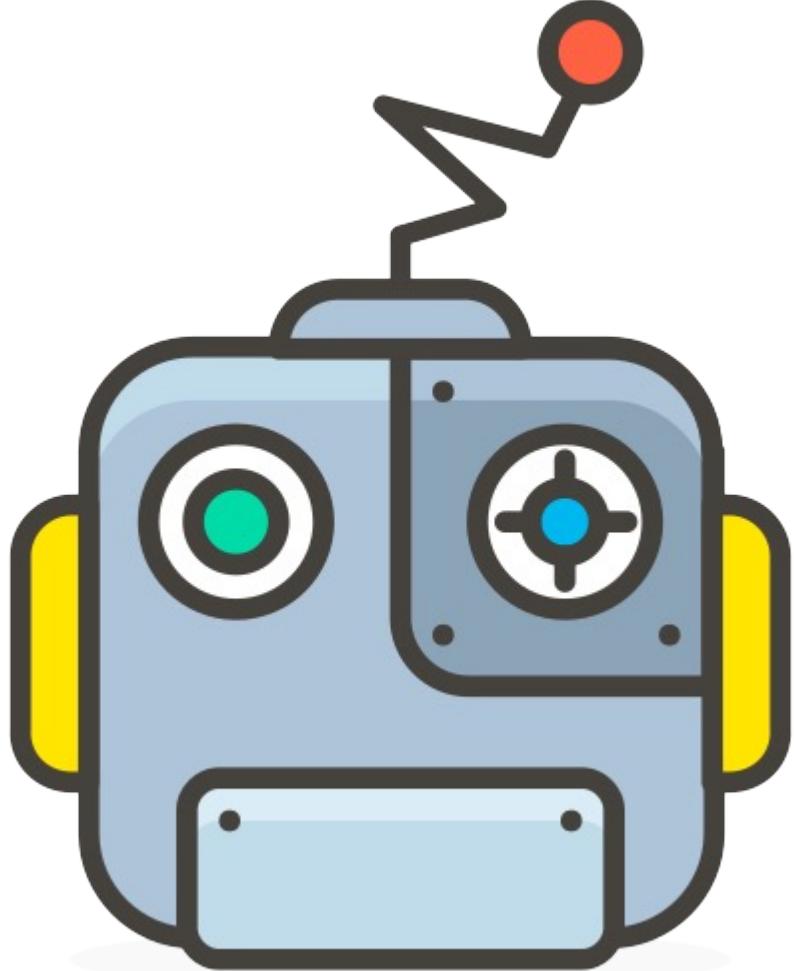
인터넷상에 존재하는 웹 문서들을 추적하여 필요한 정보를 수집하는 기법

HTML 페이지를 가져와서, HTML/CSS 등을 파싱하고, 필요한 데이터만 추출하는 기법

무분별하게 해당 웹사이트에서 데이터를 가져와서 상업적으로 이용하면 안됨

3. 크롤링

robots.txt



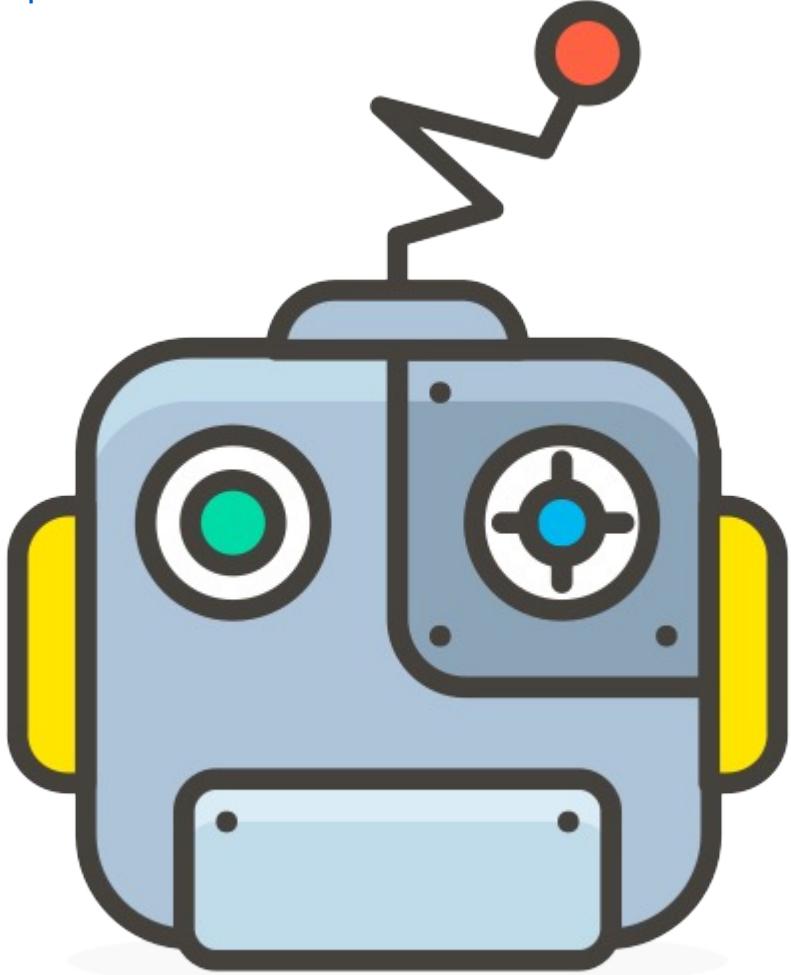
크롤링으로 수집한 데이터로 이익을 취하면 문제가 될 수 있음

각 사이트에서는 크롤러가 요청을 해도 되거나,
해서는 안되는 사항들을 **robots.txt**를 통하여 확인할 수 있음!

<https://www.naver.com/robots.txt>

3. 크롤링

<https://www.naver.com/robots.txt>



User-agent: *

Disallow: /

Allow : /\$

네이버 메인페이지 이외에서는 크롤링을 금지

www.naver.com/만 허용

3. 크롤링

Beautifulsoup4 문법

beautifulsoup4 -HTML과 XML 문서를 파싱하기 위한 파이썬 패키지 (<https://www.crummy.com/software/BeautifulSoup/bs4/doc>)

find 함수

```
soup.find('p')
soup.find('div',{'class':'클래스 네임명'})
soup.find('img',{'id':'아이디 네임명'})

soup.find_all('div')
```

select 함수

```
soup.select('p')
soup.select('.클래스 네임명')
soup.select('상위태그명 > 하위태그명 > 하위태그명')
soup.select('상위태그명.클래스명 > 하위태그명.클래스명')
soup.select('#아이디명')
soup.select('#아이디명 > 태그명.클래스명')
soup.select('태그명[속성1=값1]')
```

3. 크롤링 실습 전

함께 실습을 해봅시다!

〈크롤링 전체적인 Flow〉

1. 웹 사이트 url에 get요청
2. find나 select 함수를 이용해서 원하는 HTML Element 가져오기
3. 원하는 데이터 결과 값 형태로 가공하기

3. 크롤링 실습 전

함께 실습을 해봅시다!

스마트 스토어에서 직접 크롤링 실습을 시작해볼까요?

(<https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=%EB%85%A5%EC%8A%A4%ED%8A%B8>)

3. 크롤링 실습 전

함께 실습을 해봅시다!

Query String

`https://search.shopping.naver.com/search/all.nhn?변수1=값1&변수2=값2`

- get 요청을 보낼 때, 주로 query에 변수를 담아 정보를 전송함
- Url 주소 뒤에 ?로 시작
- 변수와 값의 쌍으로 구성 (각 쌍은 &로 구분)

3. 크롤링 실습 전

함께 실습을 해봅시다!

Query String

<https://search.shopping.naver.com/search/all?pagingIndex=2&pagingSize=80&query=%EB%8A%A8%EC%9E%91%ED%8B%8C>

-2번째 페이지

-한 페이지에는 80개의 검색 결과

-검색어는 반숙란

3. 크롤링 실습 전

함께 실습을 해봅시다!

session4 폴더 안에 egg.py를 열어주세요!

3. 크롤링 실습

해당 url 연결

해당 주소에 요청 보내기

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text,"html.parser")
print(egg_soup)
```

3. 크롤링 실습

상품 리스트 출력

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text,"html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
print(egg_list)
```

3. 크롤링 실습

첫번째 상품의 이름 찾기

ul > li > .basicList_inner__eY_mq > .basicList_title__3P9Q7 > a

▼ 네이버 랭킹순 · 낮은 가격순 · 높은 가격순 · 드로이수 · 리뷰 막으수 · 리뷰 죽으수

[a.basicList_link__1MaTN 294px × 14px](#)
Role link

[행복담기] 편의점 촉촉한 반숙란 찐 반숙계란 등의반숙란 20구

광고 ① 10,900원

식품 > 축산 > 알류 > 훈제란

식품품질인증 : HACCP | 형태 : 반숙 | 달걀크기 : 대란 | 색상별 : 갈색

1.43초마다 한개씩 팔리는 촉촉 탱글한 반숙란 !

등록일 2021.08. · ❤️ 쪽하기 281 · 💬 신고하기



```
▼ <li class="basicList_item__2XT81 ad">
  ▼ <div class="basicList_inner__eY_mq">
    ▶ <div class="basicList_img_area__a3NRA">...</div>
    ▼ <div class="basicList_info_area__17Xyo">
      ▶ <div class="basicList_title__3P9Q7">...</div> = $0
      ▶ <div class="basicList_price_area__1UXXR">...</div>
      ▶ <div class="basicList_depth__2QIie">...</div>
      ▶ <div class="basicList_desc__2-tko basicList_max__boWiv">
        ...</div>
      ▶ <div class="basicList_etc_box__1Jzg6">...</div>
      </div>
      ▶ <div class="basicList_mall_area__lIA7R">...</div>
    </div>
  </li>
```

3. 크롤링 실습

첫번째 상품의 이름 찾기

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text, "html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
title = egg_list[0].find("div", {"class": "basicList_title__3P9Q7"}).find("a").string
print(title)
```

3. 크롤링 실습

첫번째 상품의 가격 찾기

ul > li > .basicList_inner__eY_mq > .basicList_price_area__1UXXR > .price_num__2WUXn



[행복담기] 59px × 17px

광고 ⓘ 10,900원

식품 > 축산 > 알류 > 훈제란

식품품질인증 : HACCP | 형태 : 반숙 | 달걀크기 : 대란 | 색상별 : 갈색로

1.43초마다 한개씩 팔리는 촉촉 탱글한 반숙란 !

등록일 2021.08. · ❤️ 찜하기 281 · 💬 신고하기

```
▼ <li class="basicList_item__2XT81 ad">
  ▼ <div class="basicList_inner__eY_mq">
    ▶ <div class="basicList_img_area__a3NRA">...</div>
    ▼ <div class="basicList_info_area__17Xyo">
      ▶ <div class="basicList_title__3P9Q7">...</div>
      ▼ <div class="basicList_price_area__1UXXR">...</div>
        <button type="button" class="ad_ad_stk__12U34">광고</button>
      ▼ <strong class="basicList_price__2r23_">
        ▼ <span>
          <span class="price_num__2WUXn">10,900원</span> = $0
        </span>
      </strong>
    </div>
    ▶ <div class="basicList_depth__2QIie">...</div>
    ▶ <div class="basicList_desc__2-tko basicList_max__boWiv">...
    </div>
    ▶ <div class="basicList_etc_box__1Jzg6">...</div>
    </div>
    ▶ <div class="basicList_mall_area__lIA7R">...</div>
  </div>
</li>
```

3. 크롤링 실습

첫번째 상품의 가격 찾기

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text,"html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
price = egg_list[0].find("div", {"class": "basicList_price_area__1UXXR"}).find("span", {"class": "price_num__2WUXn"}).text
print(price)
```

3. 크롤링 실습

텍스트를 추출하는 방법

```
<td>some text</td>
<td></td>
<td><p>more text</p></td>
<td>even <p>more text</p></td>
```

① string

```
some text
None
more text
None
```

- 태그 하위에 문자열을 객체화
- 문자열이 없으면 None

② text

```
some text
more text
even more text
```

- 하위 자식태그의 텍스트까지 문자열로 반환

③ 그 외

- strip : 공백 없애기
- split : 문자열을 배열로 나누기
- replace : 특정 문자열을 교체

3. 크롤링 실습

데이터를 저장해 봅시다

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text, "html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
title = egg_list[0].find("div", {"class": "basicList_title__3P9Q7"}).find("a").string
price = egg_list[0].find("div", {"class": "basicList_price_area__1UXXR"}).find("span", {"class": "price_num__2WUXn"}).text

result = {
    'title': title,
    'price': price,
}

print(result)
```

3. 크롤링 실습

노트북의 상세 정보를 배열로 담아 result에 추가하기

The screenshot shows a product detail page for soft-boiled eggs. At the top, there's a navigation bar with various links like '전체' (4,817), '가격비교' (31), '네이버페이' (2,504), '백화점/홈쇼핑' (286), '장보기' (15), '쇼핑윈도' (23), and '해외직구' (2,311). Below the navigation, there are sorting options ('네이버 랭킹순', '낮은 가격순', '높은 가격순', '등록일순', '리뷰 많은순', '리뷰 좋은순') and filters ('쇼핑몰선택', '상품타입(전체)', '40개씩 보기').

The main content area features a large image of three soft-boiled eggs. The product title is '[행복담기] 편의점 촉촉한 반숙란 찐 반숙계란 동의반숙란 20구'. A yellow callout box highlights the price '광고 ① 10,900원' and the CSS selector 'div.basicList_detail_box__3ta3h 538px × 20px'. Below the title, there's a detailed description: '식품품질인증 : HACCP | 형태 : 반숙 | 달걀크기 : 대란 | 색상별 : 갈색란 | 수정여부 : 무정란 | 개수 : 20구'. A note below states '1.43초마다 한개씩 팔리는 촉촉 탱글한 반숙란!'. At the bottom, it says '등록일 2021.08.' and has social sharing buttons for '찜하기 282' and '신고하기'.

On the right side, there's a sidebar for '행복담기 스토어' which includes a logo for '빕파워', payment methods ('N Pay + 포인트 327원'), shipping information ('배송비 3,000원 | 오늘출발'), and a purchase button ('할인 구매정보').

Hint) find_all과 for문을 이용.

3. 크롤링 실습

노트북의 상세 정보 추가

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text, "html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
title = egg_list[0].find("div", {"class": "basicList_title__3P9Q7"}).find("a").string
price = egg_list[0].find("div", {"class": "basicList_price_area__1UXXR"}).find("span", {"class": "price_num__2WUXn"}).text
detail_lists = egg_list[0].find("div", {"class": "basicList_detail_box__3ta3h"}).find_all("a")

detail = []
for detail_list in detail_lists:
    detail.append([detail_list.text])

result = {
    'title': title,
    'price': price,
    'detail' : detail,
}

print(result)
```

3. 크롤링 실습

반복문을 돌려서 모든 데이터를 저장해 봅시다

```
import requests
from bs4 import BeautifulSoup

# 우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
# get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text,"html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})
result = []

for egg in egg_list:
    title = egg.find("div", {"class": "basicList_title__3P9Q7"}).find("a").string
    price = egg.find("div", {"class": "basicList_price_area__1UXXR"}).find("span", {"class": "price_num__2WUXn"}).text
    detail_lists = egg.find("div", {"class": "basicList_detail_box__3ta3h"}).find_all("a")

    detail = []
    for detail_list in detail_lists:
        detail.append(detail_list.text)

    egg_info = {
        'title': title,
        'price': price,
        'detail' : detail,
    }
    result.append([egg_info])

print(result)
```

3. 크롤링 실습

코드가 길어지니, 기능별로 파일을 나누어서 저장해봅시다

1. `main.py` - url 연결하는 파일
2. `note_book.py` - 주어진 데이터를 추출하고 가공하는 파일

3. 크롤링 실습

1. main.py - url 연결하는 파일

from 파일명 import 함수명

```
import requests
from bs4 import BeautifulSoup
from egg import extract_info

#우리가 정보를 얻고 싶어 하는 URL
EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex=1&pagingSize=80&query=반숙란'
#get 요청을 통해 해당 페이지 정보를 저장
egg_html = requests.get(EGG_URL)
# bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
egg_soup = BeautifulSoup(egg_html.text,"html.parser")

egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})

print(extract_info(egg_list))
```

3. 크롤링 실습

2. egg.py - 주어진 데이터를 추출하고 가공하는 파일

```
def extract_info(egg_list):
    result = []

    for egg in egg_list:
        title = egg.find("div", {"class": "basicList_title_3P9Q7"}).find("a").string
        price = egg.find("div", {"class": "basicList_price_area_1UXXR"}).find("span", {"class": "price_num_2WUXn"}).text
        detail_lists = egg.find("div", {"class": "basicList_detail_box_3ta3h"}).find_all("a")

        detail = []
        for detail_list in detail_lists:
            detail.append(detail_list.text)

        egg_info = {
            'title': title,
            'price': price,
            'detail' : detail,
        }
        result.append(egg_info)

    return result
```

3. 크롤링 실습

실습 시간

- 1 ~10페이지 전체 상품 목록 저장 후 print 하기
2. main.py만 수정하기

Hint) for 문 사용, f-string을 이용해 page 변수 넣기

3. 크롤링 실습

실습 정답

```
import requests
from bs4 import BeautifulSoup
from egg import extract_info

final_result = []

for page in range(1,11):
    #우리가 정보를 얻고 싶어 하는 URL
    EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex={page}&pagingSize=80&query=반숙란'
    #get 요청을 통해 해당 페이지 정보를 저장
    egg_html = requests.get(EGG_URL)
    # bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
    egg_soup = BeautifulSoup(egg_html.text,"html.parser")

    egg_list_box = egg_soup.find("ul", {"class": "list_basis"})
    egg_list = egg_list_box.find_all("li", {"class": "basicList_item__2XT81"})

    final_result += extract_info(egg_list)

print(final_result)
```

3. 크롤링 실습

거의 다 왔습니다ㅠㅠ 좀만 더 힘내요ㅠㅠㅠ



지금까지 저장한 데이터를 CSV 파일에 저장해봅시다!

3. 크롤링 실습

CSV (comma-separated values)

1. 몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일
2. 엑셀 프로그램으로 열기 가능!
3. 이후, DB에 직접 넣을 수 있는 파일 형식

3. 크롤링 실습

```
import requests
from bs4 import BeautifulSoup
from egg import extract_info
import csv

file = open('egg.csv', mode='w', newline='')
# 'w'는 쓰기 모드, newline=''은 띄어쓰기 없이 데이터 입력
writer = csv.writer(file)
writer.writerow(["title","price","detail"])

final_result = []
for page in range(1,11):
    #우리가 정보를 얻고 싶어 하는 URL
    EGG_URL = f'https://search.shopping.naver.com/search/all?pagingIndex={page}&pagingSize=80&query=반숙란'
    #get 요청을 통해 해당 페이지 정보를 저장
    egg_html = requests.get(EGG_URL)
    # bs4 라이브러리를 통해 불러온 html을 우리가 원하는 형태로 파싱
    egg_soup = BeautifulSoup(egg_html.text,"html.parser")

    egg_list_box = egg_soup.find("ul", {"class":"list_basis"})
    egg_list = egg_list_box.find_all("li", {"class":"basicList_item__2XT81"})

    final_result += extract_info(egg_list)

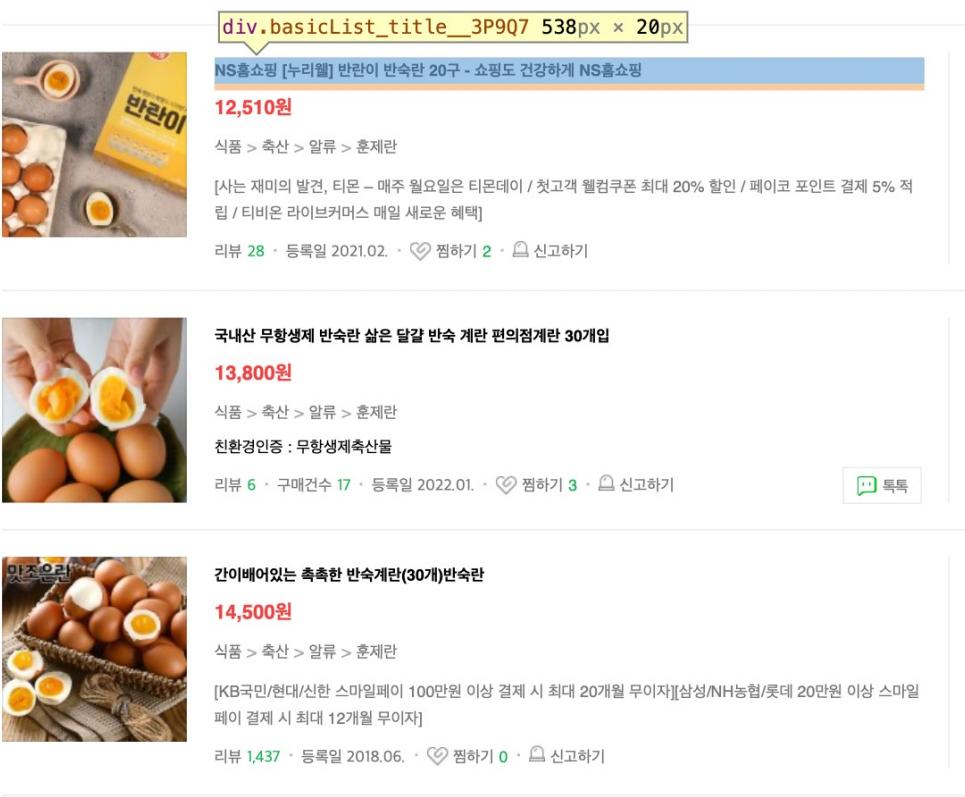
for result in final_result:
    row=[]
    row.append(result['title'])
    row.append(result['price'])
    row.append(result['detail'])
    writer.writerow(row)
print(final_result)
```

3. 크롤링 실습

egg

| title | price | detail |
|---|---------|---|
| 홍생농장 반숙란40구 촉촉한 부드러운 반숙계란 | 19,900원 | [‘형태 : 반숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’, ‘사육환경 : 개선케이지’] |
| [행복담기] 편의점 촉촉한 반숙란 찐 반숙계란 동의반숙란 20구 | 10,900원 | [‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’, ‘개수 : 20구’] |
| 짜지않고 촉촉한 저염 반숙란 40구 국내산 반숙계란 | 23,900원 | [‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’, ‘사육환경 : 기존케이지’] |
| 짜지않고 촉촉한 저염 반숙란 20구 국내산 반숙계란 | 10,900원 | [‘친환경인증 : 무항생제축산물’, ‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘달걀크기 : 중란’, ‘색상별 : 갈색란’] |
| [호유란] 30개입, 짜지않고 촉촉한 반숙계란 반숙란 | 15,900원 | [‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’] |
| [Kurly's] 동물복지 반숙란 6구 | 4,700원 | [] |
| 무항생제 반숙란30구 계란반숙 구운란 군계란 찜질방 훈제 삶은달걀 | 12,900원 | [‘형태 : 반숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘사육환경 : 축사내 평사’] |
| 꼬꼬 반숙란 한판(30구) / 반숙계란 영양간식 | 13,900원 | [] |
| 계림농장 무항생제 구운계란 구운란 훈제란 반숙란 30구 | 9,900원 | [‘친환경인증 : 무항생제축산물’, ‘형태 : 완숙’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’] |
| 홍생농장 반숙란 20구 - 홍생농장 반숙란 훈제란 구운란 20구 | 14,500원 | [] |
| [호유란] [호유란] 20개입 짜지않고 촉촉한 반숙계란 반숙란 | 13,800원 | [] |
| 친환경 구운란 반숙란 20구 30구 40구 60구 120구 150구 골라담기! HACCP인증 | 9,200원 | [] |
| 반숙란 10개입 500g | 5,980원 | [] |
| 자연애찬 반숙란3입 | 1,980원 | [] |
| 고소한 영양간식 안심반숙란30구 | 14,870원 | [] |
| 반숙란 10개입 500g | 5,980원 | [] |
| HACCP 촉촉한 반숙계란 30구 반숙란 | 13,900원 | [] |
| 계림농장 7일간 숙성을 거친 반숙란 30구 | 11,800원 | [‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘달걀크기 : 중란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’, ‘사육환경 : 방사’] |
| [미니언즈] 반숙란 반숙계란 8구 + 구운란 구운계란 8구 + 계란케이스 2개 | 16,900원 | [‘식품품질인증 : HACCP’, ‘달걀크기 : 대란’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’, ‘개수 : 8구구’] |
| 고소한 영양간식 안심반숙란20구 | 11,870원 | [] |
| [에코백 증정] 미니언즈 저염 촉촉 반숙란 40구+에코백 | 23,900원 | [‘친환경인증 : 무항생제축산물’, ‘식품품질인증 : HACCP’, ‘형태 : 반숙’, ‘색상별 : 갈색란’, ‘수정여부 : 무정란’] |
| 무항생제 HACCP 인증 구운계란 구운란 반숙란 | 8,700원 | [] |
| 가능 오메가 반숙란 20구 | 25,100원 | [] |
| 가능 오메가 반숙란 20구 - UnKnown | 22,880원 | [] |
| 가능 오메가 반숙란 20구 | 25,100원 | [] |
| [온도씨]하루한알 쫀깃한 구운란 vs 촉촉한 반숙란 | 12,870원 | [] |
| 구운란 대란 20구 + 반숙란 20구 / 구운계란 반숙계란 | 19,640원 | [] |
| 구운란(중란) 20구 + 반숙란 20구 / 구운계란 반숙계란 | 20,490원 | [] |
| 촉촉한 반숙란 20구 / 대란 HACCP 반숙계란 | 12,250원 | [] |

3. 크롤링 마무리



가급적 구조를 분석한 후, 사용하는 것을 추천

| 3. 크롤링 마무리

bs4의 한계

1. 웹 사이트의 구조가 바뀌면 코드를 다시 작성해야 함
2. 동적으로 만들어진 웹 사이트 코드를 읽지 못함

3. 크롤링 마무리

bs4의 한계

과거 - HTML Page가 모두 완성되어서 Response로 내려옴 (Server-Side-Rendering)

bs4 사용x

현재 - HTML Page가 Javascript를 이용하여 동적으로 완성됨 (Client-Side-Rendering)

=> Selenium이 필요 (사람이 손으로 동작하는거에 따라서 정보를 수집)

| 과제 1

1. 자기가 좋아하는 요일의 웹툰 정보 크롤링 하기

(<https://comic.naver.com/webtoon/weekdayList?week=mon>)

2. 웹툰 제목, 작가 이름, 평점 정보 추출

3. 파일을 두개로 나누어서 작성하기

4. 최종 결과는 csv 파일 형태로 저장!

| 수고하셨습니다~!

