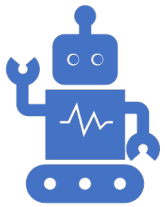
2020.11.07. ~ 2020.11.22.
경상대학교 407동 202호

Day 02



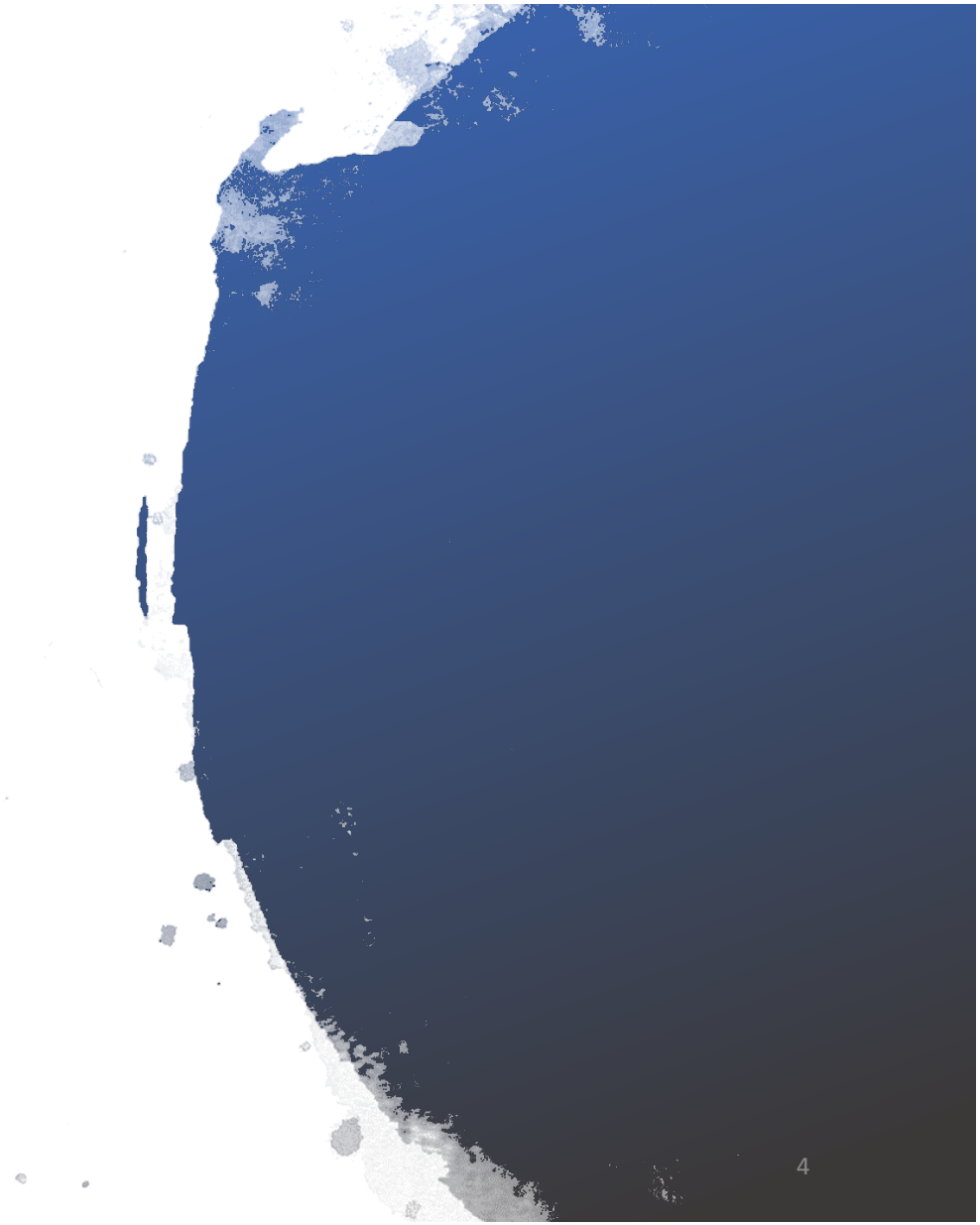
초보자를 위한 빅 데이터/ 머신러닝

Index

오늘의 할 일 요약

- Python 개발 환경을 설치한다.
- 웹 사이트 HTML 구조를 보고 파싱한다.
- CSV FILE을 이해하고 사용하는 법을 익힌다.
- 웹 사이트에 있는 Text를 저장하는 법을 익힌다.
- 웹 사이트에 있는 이미지 데이터를 저장하는 법을 익힌다.
- 멀티 프로세스을 사용해 병렬적으로 데이터를 연산하는 법을 익힌다.

프로젝트 생성



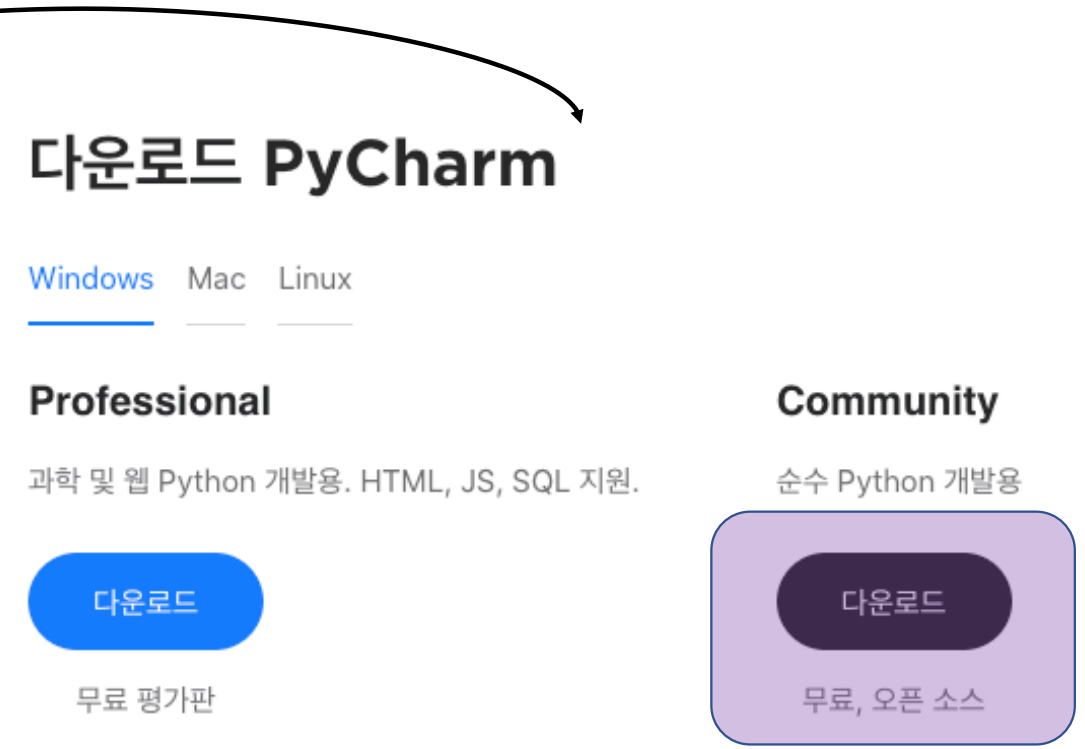
1. Pycharm Install

[Pycharm Download]

Windows : install

Linux(Ubuntu) : install

macOS : install



Community 선택

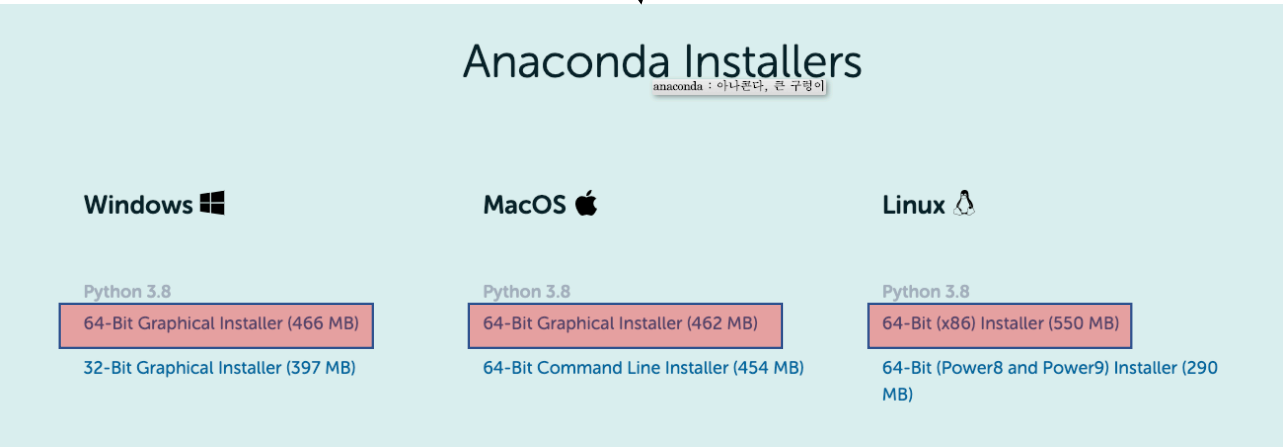
2. Anaconda3 Install

Anaconda3 Download

Windows : **install guide**

Linux(Ubuntu) : **install guide**

macOS : **install guide**



64Bit 다운로드

3. 데이터 크롤링 package 설치

데이터 크롤링을 위해 필요한 package는 다음과 같음

- requests : python에서 HTTP 요청을 보내는 모듈
- BeautifulSoup(bs4) : html을 python 객체 구조로 변환하여 Parsing을 진행하는 모듈
- tqdm : python에서 for 문에 반복되는 진행상황을 출력하는 모듈

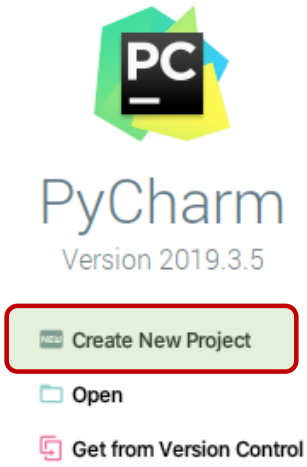
각 모듈은 터미널 혹은 CMD에서 다음과 같이 실행

```
pip install requests bs4 tqdm
```

혹은 pycharm 에서 설치가능 (뒷장에서 설명)

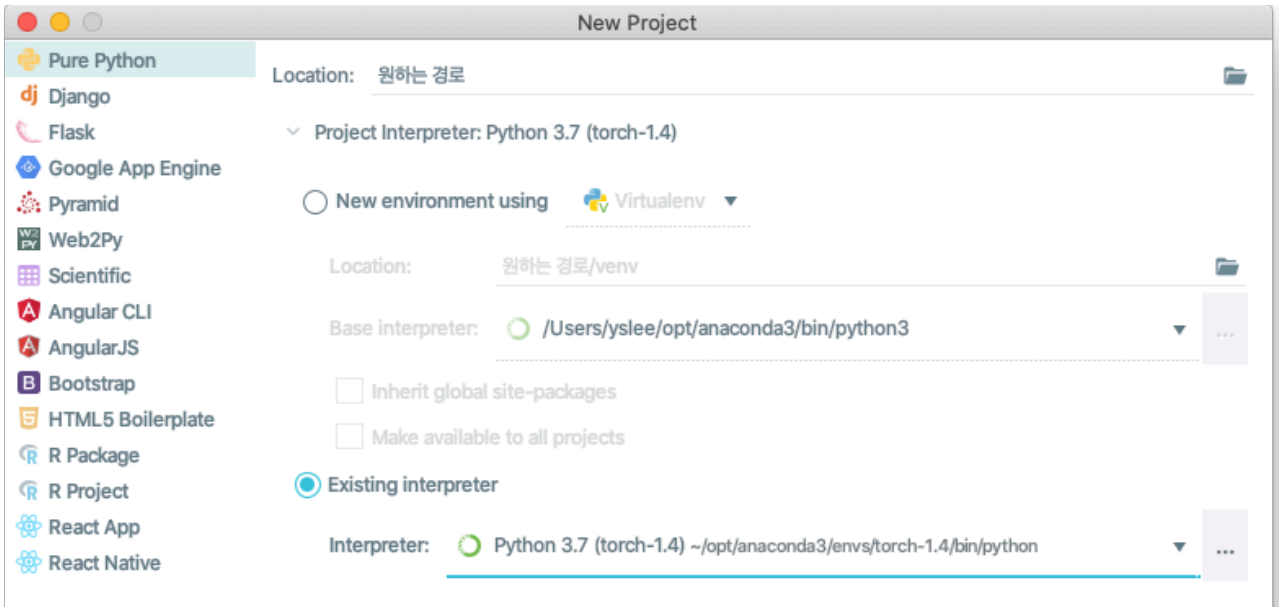
4. Pycharm Project 생성

1



Pycharm 실행 후
Create New Project

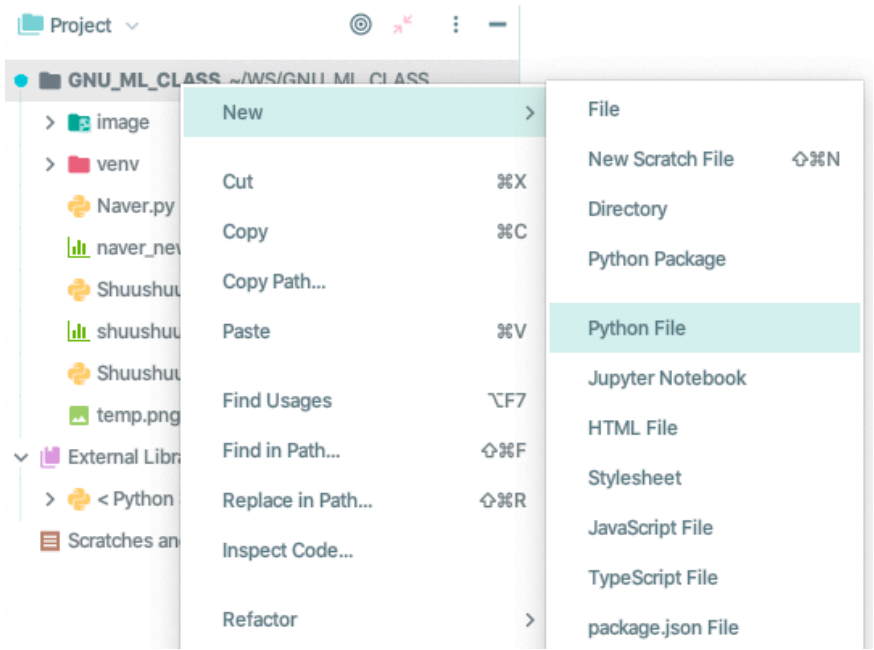
2



- 원하는 경로로 프로젝트를 생성하고
- 원하는 파이썬 버전을 선택

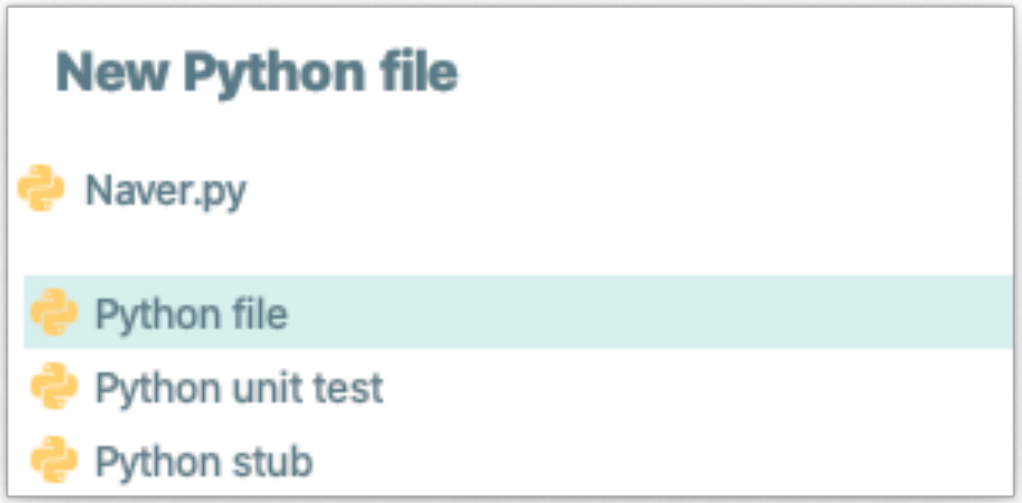
5. Pycharm file 생성

1



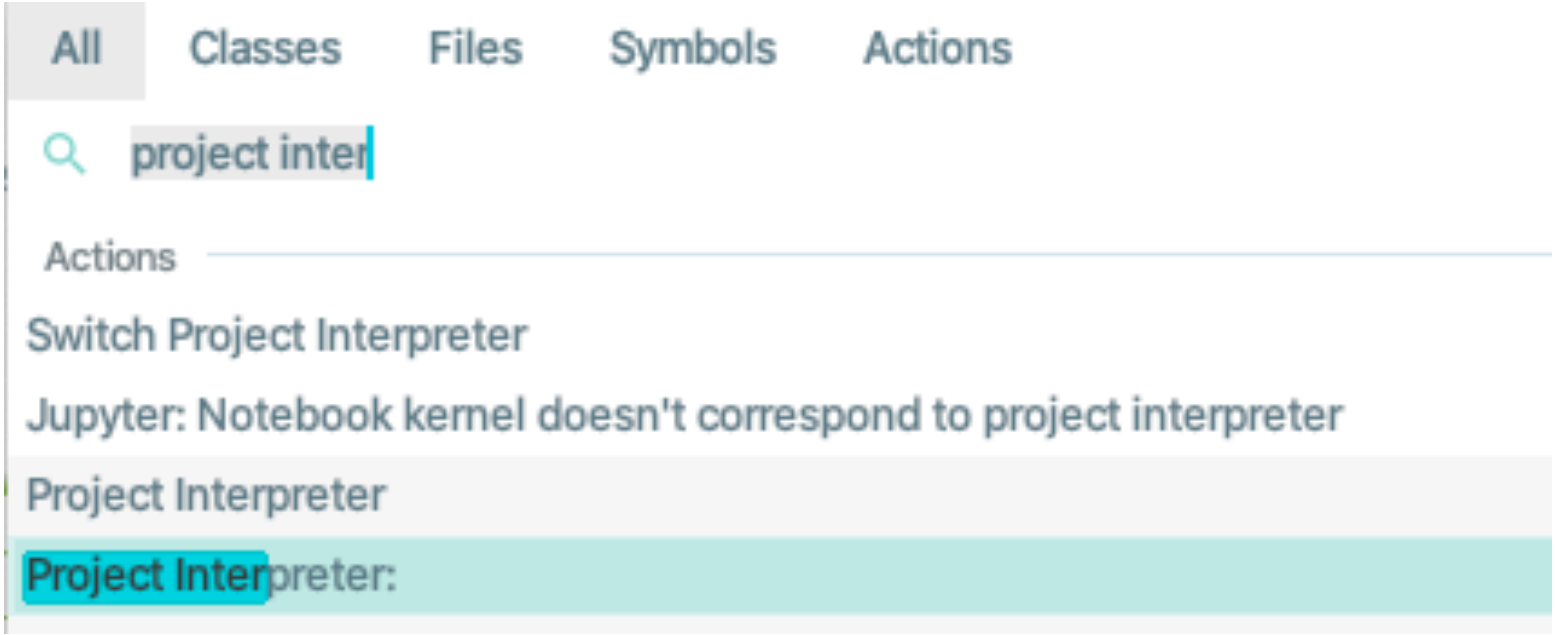
Project 생성 폴더 우 클릭
NEW -> Python File 클릭

2



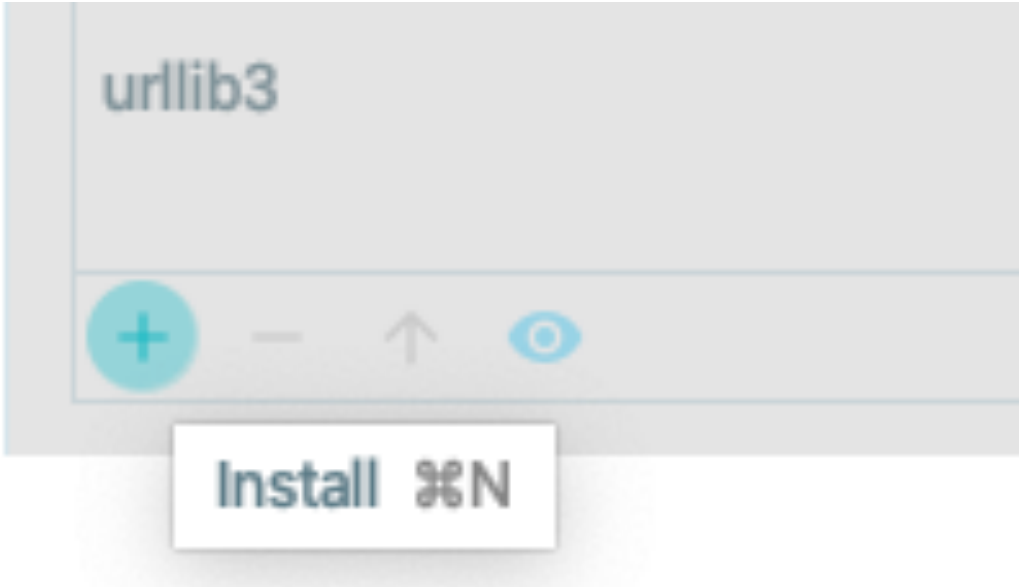
원하는 파일 이름을 입력 후 앤터
여기서는 ”Naver.py " 이름으로 파이썬 파일을 생성

6. python Package 설치(pycharm) : 1/3



1. Shift key를 두번 누르면 나오는 검색창에서 project interpreter 검색
2. project interpreter 클릭

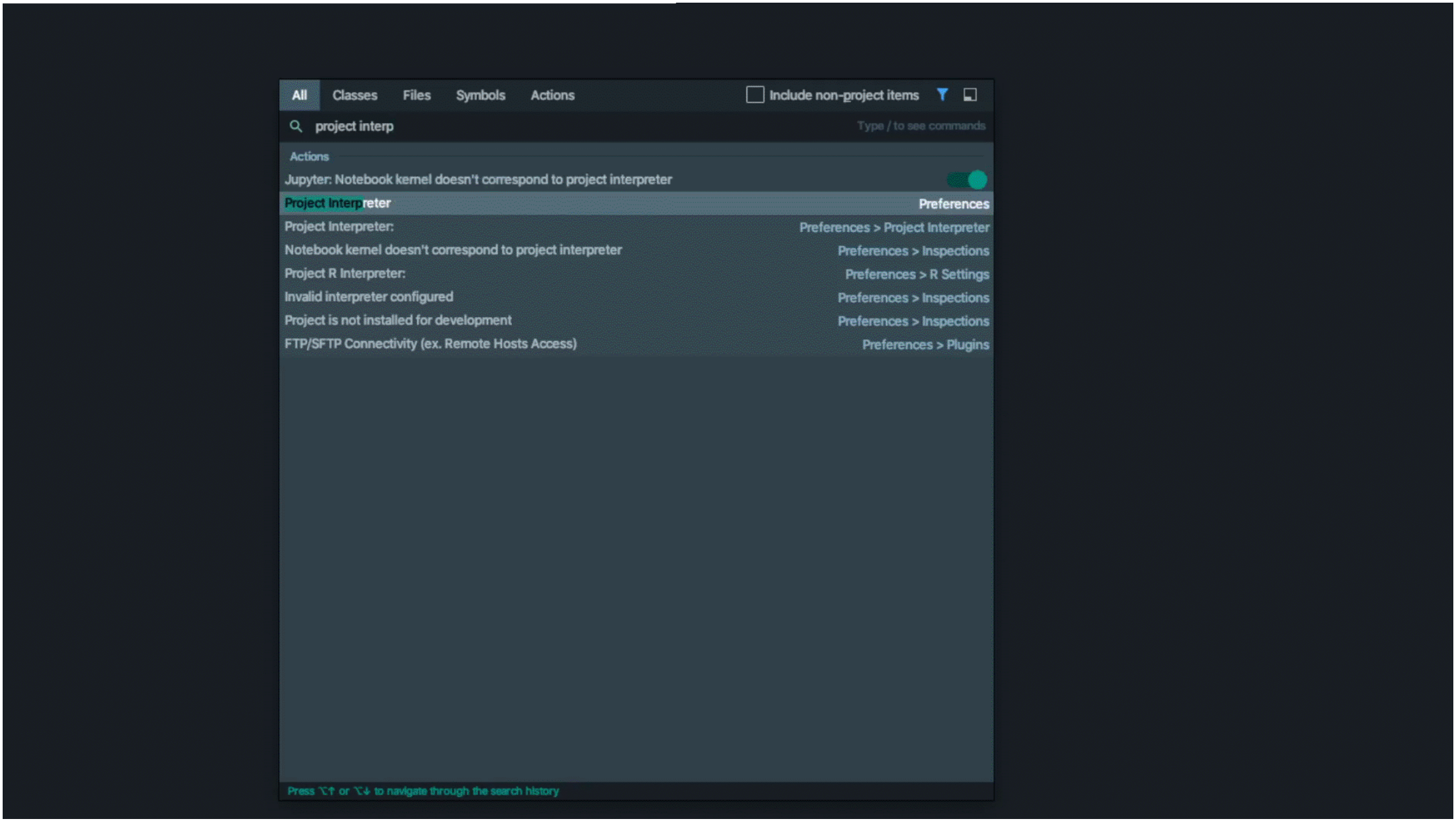
6. python Package 설치(pycharm) : 2/3



1. project interpreter 창에서 왼쪽 하단의 + 기호를 클릭

6. python Package 설치(pycharm) : 3/3

1. Available Packages 창에서 필요한 Package를 검색 후 왼쪽 하단의 INSTALL PACKAGE 클릭
2. 필요한 Package를 설치 했다면 창을 닫기


All
Classes
Files
Symbols
Actions
Include non-project items
project interp
Type / to see commands
Actions
Jupyter: Notebook kernel doesn't correspond to project interpreter
Project Interpreter
Preferences
Project Interpreter:
Preferences > Project Interpreter
Notebook kernel doesn't correspond to project interpreter
Preferences > Inspections
Project R Interpreter:
Preferences > R Settings
Invalid interpreter configured
Preferences > Inspections
Project is not installed for development
Preferences > Inspections
FTP/SFTP Connectivity (ex. Remote Hosts Access)
Preferences > Plugins
Press ⌥↑ or ⌥↓ to navigate through the search history

TEXT 기반 웹 크롤링

프로젝트
최상위 경로

- Naver.py : 작성 및 실행할 파이썬 코드
- naver_news.csv : 네이버 기사 데이터가 저장될 CSV file

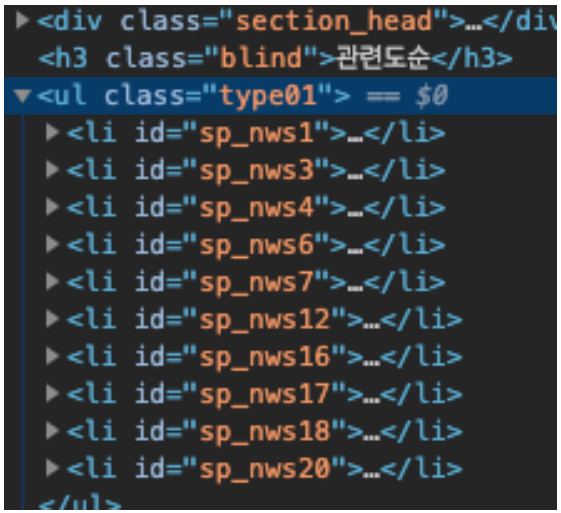
Naver News HTML 구조


```
▼<ul class="type01">
  ▼<li id="sp_nws1">
    ▶<div class="thumb">…</div>
    ▼<dl>
      ▼<dt>
        ▶<a href="http://www.hellot.net/new_hellot/magazine/magazine_read.html?code=202&sub=004&idx=53787" target="_blank" class="_sp_each_url _sp_each_title" onclick="return goOtherCR(this, 'a=nws*b.tit&r=1&i=88166537_00000000000000000021034&g=5417.0000021034&u='+urlencode(this.href));" title="ST, 머신러닝 간소화 위한 STM32 상태 모니터링 기능팩 출시">…</a> == $0
      </dt>
      ▶<dd class="txt_inline">…</dd>
      ▶<dd>…</dd>
      ▶<dd>…</dd>
    </dl>
    ::after
  </li>
  ▶<li id="sp_nws3">…</li>
```

Naver News HTML 구조

네이버 뉴스의 경우 ' news mynews section _prs_nws’ 이름의 class로 보관됨

각 뉴스의 경우 news mynews section _prs_nws -> type01 -> sp_nws{#}의 계층 구조를 가지고 있으며
sp_nws{#}의 리스트로 각 기사들이 보여짐



1. Python Package import

Naver.py

```
import requests # http 요청을 위한 모듈
import bs4 # html을 파싱하기 위한 모듈
import csv # csv data를 생성하기 위한 모듈
from tqdm import tqdm # 반복 문 진행 상황을 출력하기 위한 모듈
```

```
import requests
# “requests” 라는 이름의 모듈을 불러옴

from tqdm import tqdm
# “tqdm” 라는 모듈에서 ”tqdm” 라는 모듈을 불러옴
```

2. 설정 변수 생성

Naver.py

```
headers = {'User-Agent': 'Mozilla/5.0'}
base_url = 'https://search.naver.com/search.naver'
naver_news_search =
'https://search.naver.com/search.naver?where=news&query={}'
file_path = 'naver_news.csv'
count = 1
max_count = 100000
```

base_url: 네이버 뉴스 메인 주소
naver_news_search: 네이버 뉴스 검색에 사용할 주소
file_path: 크롤링한 데이터를 저장할 csv file 경로
count: 저장한 기사 개수
max_count: 최대 저장 가능한 기사 개수

2. 설정 변수 생성 : string format

Naver.py

```
naver_news_search =
'https://search.naver.com/search.naver?where=news&query={}'
```

naver_news_search 문자열 내부 {}의 경우 naver_news_search.format(값)을 사용해 원하는 값을 {} 내부에 삽입 할 수 있음

이를 통해 원하는 검색 키워드를 넣어 크롤링을 진행 할 수 있음

3. CSV file 생성 및 초기화

Naver.py

```
with open(file_path, 'w') as f:
    csv_file_w = csv.writer(f)
    csv_file_w.writerow(['title', 'url'])
```

CVS file을 생성하고 헤더를 추가하는 과정

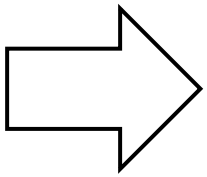
file_path('naver_news.csv')로 파일을 생성
csv.writer.writerow()로 'title', 'url' 데이터를 추가

csv file에서 첫 데이터는 각 데이터 열을 나타내는 헤더

3. CSV file 생성 및 초기화 : CSV File

CSV: Comma Separated Values 의 약자로 데이터 열을 쉼표(,)로 구분한 텍스트 데이터 파일

id	name	age
1	임진악	25
2	홍길동	21
3	김준호	26



```
id,name,age
1,임진악,25
2,홍길동,21
3,김준호,26
```

데이터 테이블

CSV FILE text data

왼쪽과 같은 데이터 테이블을 CSV 텍스트로 표현하면 오른쪽과 같이 나타낼 수 있음

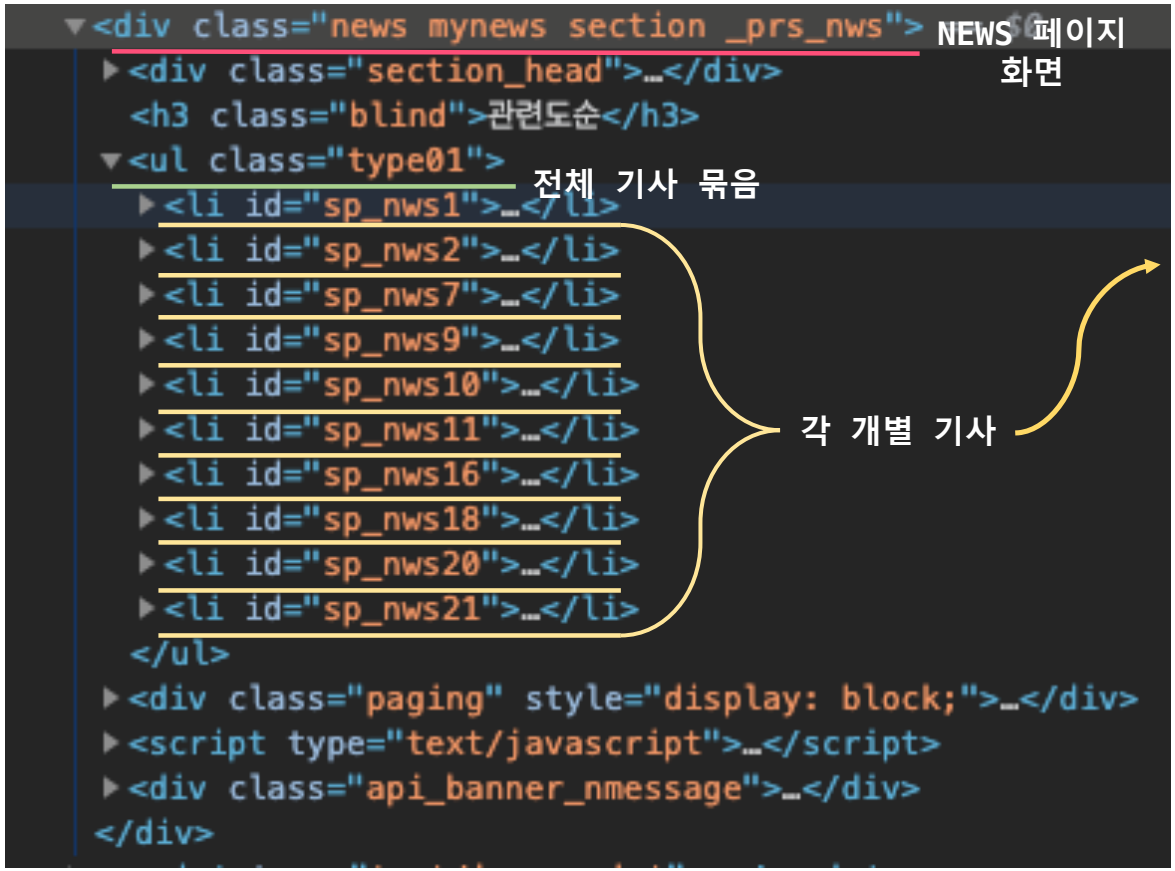
3. CSV file 생성 및 초기화 : python file open

```
open(file_path, 'w')
```

Python에서 파일을 사용 할때 open(파일경로, 파일열기모드)을 사용함

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일을 만들고 내용을 쓸 때 사용
a	추가모드 – 파일의 마지막에 새로운 내용을 추가 시킬 때 사용

4. 뉴스 크롤링 : Naver News HTML 구조



4. 뉴스 크롤링 : for loop 1/3 Start (Tag 분석)

Naver.py

```
raw = requests.get(url=naver_news_search.format('머신러닝'), headers=headers)

for itr in tqdm(range(1000)):
    soup = bs4.BeautifulSoup(raw.text, 'html.parser')
    news_titles = soup.select('.news .type01 li dt a[title]')
    #이어짐
```

```
soup = bs4.BeautifulSoup(raw.text, 'html.parser')
# html을 BS를 통해 파싱

news_titles = soup.select('.news .type01 li dt a[title]')
# 뉴스페이지의 html 구조에 맞도록 html 데이터를 파싱
'.news .type01 li dt a[title]' : 각각 news(class), type01(class), li(tag), dt(tag),
a[title](속성)으로 검색하여 가져온다.
```

4. 뉴스 크롤링 : for loop 2/3 (Text 추출)

Naver.py

```
with open(file_path, 'a') as f:
    csv_file_w = csv.writer(f)

    for i, title in enumerate(news_titles):
        if count > max_count: exit(-1)
        data = [title['title'].replace(',', ' '), title['href']]
        csv_file_w.writerow(data)
        count += 1
#for 문 이어짐
```

앞서 생성한 csv 파일을 추가모드(a)로 열어줌.
파싱한 결과인 news_titles를 반복하여 기사 제목('title')과 제목에 달려있는 기사의 링크('href')를 csv에 저장

Csv에 데이터저장은 열 순서를 고려해 각 열 값을 list로 만들어
csv.writer(file).writerow(list)로 저장한다.

수집하는 기사 개수의 최대값 여부를 확인하고 반복을 계속 진행한다.

4. 뉴스 크롤링 : for loop 3/3 END (페이지 이동)

Naver.py

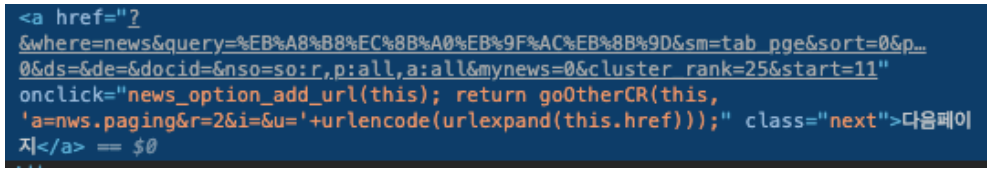
```
next_link = soup.find(class_='next')
next_link = next_link.attrs['href']

raw = requests.get(url=base_url + next_link, headers=headers)
```

다음 기사 탭의 데이터를 가지고 오는 과정

soup.find(class_='next')
네이버뉴스의 다음페이지(url)는 next라는 이름의 class에 저장되어 있다. 최초 가지고 온 html에서 find 함수를 사용해 'next'라는 이름을 가진 class를 가지고 온다.

next_link.attrs['href']
이후 해당 class의 속성 중 'href'에 저장된 url을 가지고 온다.

마지막으로 루프를 반복하기 전 raw에 새로운 url로 요청을 보내 다음페이지의 html 정보를 가지고 온다.



다음 페이지 HTML

다음 페이지 화면

4. 뉴스 크롤링 : 저장된 CSV file

```
title,url
슈퍼브에이아이  머신러닝 데이터 플랫폼 '스위트' 출시,http://www.irobotnews.com/news/articleView.html?idxno=21977
ST  머신러닝 간소화 위한 STM32 상태 모니터링 기능팩 출시,http://www.hellot.net/new_hellot/magazine/magazine_read.html?code=202&sub=004&idx=53787
코오롱베니트  'SAS 자동화 머신러닝 활용 전략' 웨비나 개최,http://www.ddaily.co.kr/news/article.html?no=200037
클라우드 머신러닝 플랫폼 선택 기준 12가지,http://www.itworld.co.kr/news/160710
머신러닝 기반 5년 후 암 생존율 예측모델 개발,http://yna.kr/AKR20200803089300017?did=1195m
아마존 머신러닝 대학  'AI 온라인 강좌' 누구나 학습할 수 있도록 개방,http://www.aitimes.kr/news/articleView.html?idxno=17365
ST  머신러닝 신속 구현 SW 기능팩 출시,http://www.epnc.co.kr/news/articleView.html?idxno=101957
이글루시큐리티  머신러닝 정확성 높이는 특허 취득,http://www.datanet.co.kr/news/articleView.html?idxno=149178
[누구나 개발자 #1] ③Azure에서 머신러닝을 한다는 것,http://it.chosun.com/site/data/html_dir/2020/08/02/2020080200103.html
'아내의 맛' 이필모  러닝머신 중 "나 돌아 갈래~" 눈물 뚝뚝,https://sports.donga.com/article/all/20200804/102287017/1
삶의 질 정보 활용한 머신러닝  폐암 생존 예측 정확도 높여,http://www.docdocdoc.co.kr/news/articleView.html?idxno=2001449
AI와 머신러닝으로 자율 운영 데이터센터를 구현하는 방법,http://www.itworld.co.kr/news/160715
폐암 치료 후 머신러닝 기반 '사망예측 모델' 개발,http://www.safetimes.co.kr/news/articleView.html?idxno=84058
""""알고리즘 재검토가 필요한 시점""··· 머신러닝 최신 성공사례 5선",http://www.ciokorea.com/news/160745
바리스타를 딥러닝 하다···현대카드  90명 로봇 동료 현장 배치 '시험대',http://www.etnews.com/20200805000223
과학기술 머신러닝 데이터 425만건 구축한다,http://www.fnnews.com/news/202007240944335435
```

저장된 naver_news 데이터

이미지 크롤링

프로젝트
최상위 경로

- ShuuShuu.py : 작성 및 실행할 파이썬 코드
- Shuushuu_image_metadata.csv : ShuuShuu 크롤링 데이터가 저장될 CSV file
- image : ShuuShuu 이미지가 저장될 폴더

이미지 크롤링

< e-shuushuu.net >

이미지 + 메타데이터 정보를 같이 수집

url 구조: https://e-shuushuu.net/image/1035262/
url 구조: https://e-shuushuu.net/image/id/

baseurl/image/id 로 간단한 구조를 가지고 있음
id는 1~ 104000 까지 있음

메타데이터는 meta라는 이름의 class에 저장되어 있음

1. Python Package import

ShuuShuu.py

```
import requests # http 요청을 위한 모듈
import urllib.request # url을 통해 image를 받기 위한 모듈
import bs4 # html을 파싱하기 위한 모듈
import csv # csv data를 생성하기 위한 모듈
from tqdm import tqdm # 반복 문 진행 상황을 출력하기 위한 모듈
import os # 파일 이름 변경을 사용하기 위한 모듈
```

```
import requests
# “requests” 라는 이름의 모듈을 불러옴

from tqdm import tqdm
# “tqdm” 라는 모듈에서 ”tqdm” 라는 모듈을 불러옴
```

2. 설정 변수 생성

ShuuShuu.py

```
headers = {'User-Agent': 'Mozilla/5.0'}
base_url = 'https://e-shuushuu.net'
search_url = base_url + '/image/{}/'
file_path = 'shuushuu_image_metadata.csv'
image_save_dir = 'image/
```

base_url: shuushuu 메인 주소
search_url: shuushuu 이미지 검색 주소, {}에 id값을 format으로 추가해 크롤링 진행
file_path: 크롤링한 데이터를 저장할 csv file 경로
image_save_dir: 이미지 저장 경로

3. CSV file 초기화

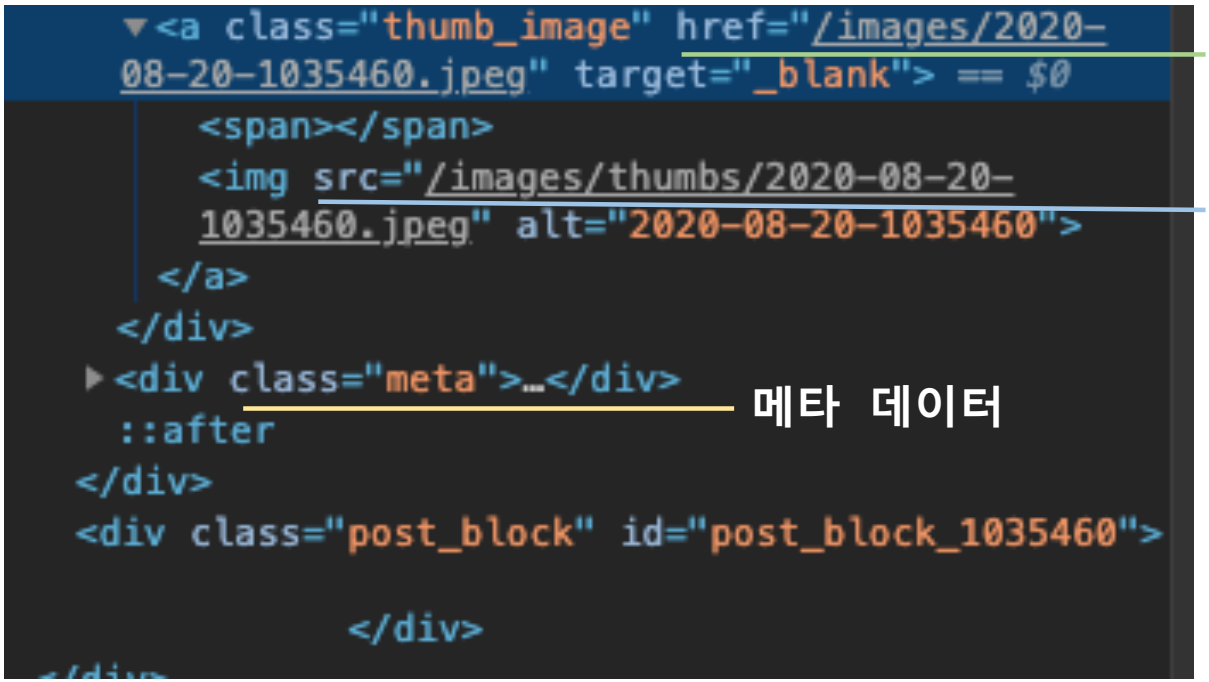
ShuuShuu.py

```
def csv_init(path, header):
   if not os.path.exists(image_save_dir):
      os.makedirs(image_save_dir)
   if os.path.exists(path): return -1
   with open(path, 'w') as f:
      csv_file_w = csv.writer(f)
      csv_file_w.writerow(header)
```

CSV file 초기화

```
if os.path.exists(path): return -1
# 이미 csv file이 있으면 건너뜀

#없으면 생성
```

4. 이미지 및 메타 데이터 크롤링 : shuushuu HTML



4. 이미지 및 메타 데이터 크롤링 : 함수 1/7 Start

ShuuShuu.py

```
def image_crawling(image_id):
    raw = requests.get(url=search_url.format(image_id),
    headers=headers)
    soup = bs4.BeautifulSoup(raw.text, 'html.parser')
    image_url = soup.find(class_='thumb_image’)
     # 이어짐
```

raw: 이미지 주소로 응답 받은 html
soup.find(class_='thumb_image’)
# html 객체에서 'thumb_image’이름을 가진 class를 가져옴

4. 이미지 및 메타 데이터 크롤링 : 함수 2/7

ShuuShuu.py

```
if image_url is None: return -1

try:
   image_url = base_url + image_url.attrs['href']
except KeyError:
   return -1

# 이어짐
```

```
base_url + image_url.attrs['href']
# 기본 url과 이미지 url의 링크를 합쳐 완전한 url을 생성
```

특정 id의 경우 서버에서 이미지가 삭제되어 있는 경우가 있어 예외처리

4. 이미지 및 메타 데이터 크롤링 : 함수 3/7

ShuuShuu.py

```
metadata = soup.find(class_='meta')

metadata_key = metadata.findAll('dt')
metadata_val = metadata.findAll('dd’)

# 이어짐
```

Shuushuu 이미지 보드에 메타데이터를 가져오기 위해 'meta’이름의 class를 가져옴
메타데이터의 경우 속성 이름은 dt, 속성 값은 dd 테그에 저장되어 있어 각각 속성과 값을 가져옴

4. 이미지 및 메타 데이터 크롤링 : 함수 4/7

ShuuShuu.py

```
meta = {}
for k, v in zip(metadata_key, metadata_val):
   k = k.text.replace('\t', '').replace('\n',
   '').replace(':', '').lower()
   v = v.text.replace('\n', '').replace('\t', '')
   meta[k] = v
# 이어짐
```

```
for k, v in zip(metadata_key, metadata_val)
# 속성 이름, 속성 값을 zip으로 묶어 한번에 반복
```

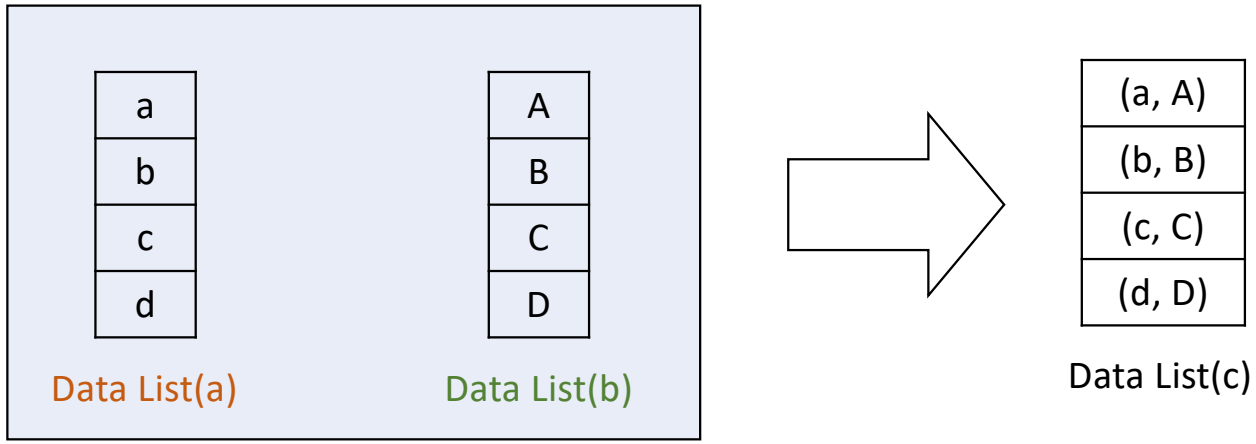
각 값의 불필요한 띄어쓰기와 문자를 replace로 제거하여 meta 딕셔너리 변수에 각각 키와 값으로 저장

4. 이미지 및 메타 데이터 크롤링 : 메타데이터 값

- 수집하는 메타데이터(Text)의 경우 문자열 내부에 탭과 줄 변경이 있음
- 크롤링을 위해 탭(\t),줄 변경('\n'),':' 와 같은 문자를 제거해야 함

메타데이터 HTML

4. 이미지 및 메타 데이터 크롤링 : python zip



list_c = zip(list_a, list_b)

zip() 은 동일한 개수로 이루어진 자료 형을 순서대로 묶어 주는 역할

4. 이미지 및 메타 데이터 크롤링 : 함수 5/7

ShuuShuu.py

```
try:
   source = meta['source']
except KeyError:
   source = ''

try:
   tags = meta['tags']
except KeyError:
   tags = ’’
# 이어짐
```

메타데이터 값이 없는 경우가 있으므로 수집하는 두 데이터(source, tags)에 대한 예외처리를 진행

4. 이미지 및 메타 데이터 크롤링 : 수집 가능 메타데이터

수집 가능한 데이터는 여러 항목이 있음
많은 경우 메타데이터가 누락되어 있기 때문에 예외처리(`try`)를 진행해야 함

Submitted By:	anonymous_object
Submitted On:	April 6th, 2005 1:41 AM
Filename:	2005-04-05-1.jpeg
Original Filename:	1108103081378.jpg
File size:	107.4 kB
Dimensions:	430x900 (0.387 MPixel)
Favorites:	90
Tags:	"4koma" "black hair" "chibi" "happy" "long hair" "red hair" "short hair" "smile" "surprised" "^_^"
Source:	"Habanero-tan"
Characters:	"Habanero-neesan" "Habanero-tan"
Old Characters:	Habanero-tan, Habanero-neesan
Artist:	"Shigatake"
Image Rating:	7.6

수집 가능한 메타데이터

4. 이미지 및 메타 데이터 크롤링 : 함수 6/7

ShuuShuu.py

```
image_path = os.path.join(image_save_dir, meta['filename'])
urllib.request.urlretrieve(image_url, image_path)

data = [image_id, image_url, tags, source, image_path]
# 이어짐
```

```
data = [image_id, image_url, tags, source]
# 생성한 csv 헤더 열의 순서를 고려하여 csv 행 데이터를 생성

os.path.join(image_save_dir, meta['filename’])
# 앞서 설정한 저장 경로로 이미지 저장 경로를 생성
# image_path = image/filename

urllib.request.urlretrieve(image_url, image_path)
# 이미지를 저장 후 append를 사용해 이미지 저장 경로를 행 데이터에 추가
```

4. 이미지 및 메타 데이터 크롤링 : 함수 7/7 END

ShuuShuu.py

```
with open(file_path, 'a') as f:
    csv_file_w = csv.writer(f)
    csv_file_w.writerow(data)
```

csv file을 추가 모드(a)로 열어 생성한 행 데이터를 추가(writerow(data))

	id	image_url	tags	source	image_path
NEW	1	https://~~~	”a” “b” “c”	“evangelion”	~~~/image/*.png

5. Main 함수

ShuuShuu.py

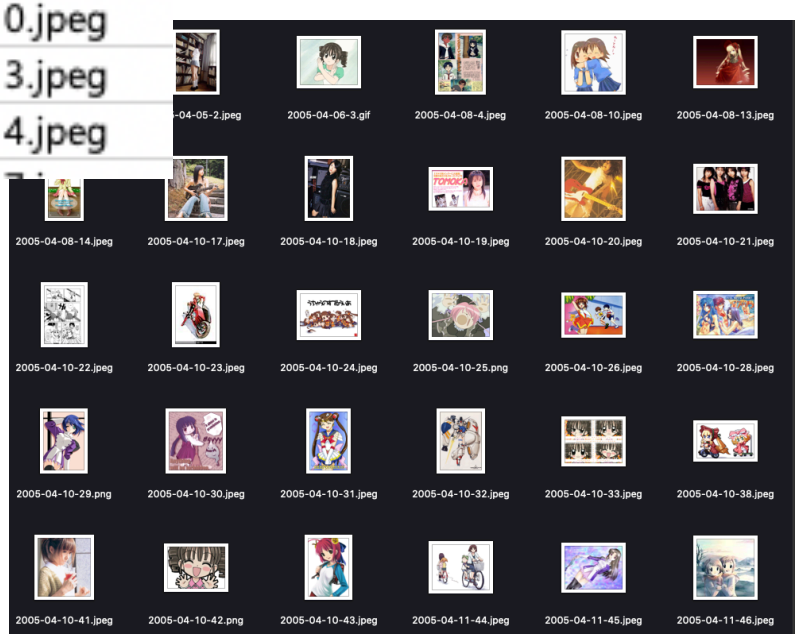
```python
if __name__ == '__main__':
    header = ['id', 'image_url', 'tags', 'source', 'image_path']
    csv_init(file_path, header)

    for image_id in tqdm(range(1, 100)):
        image_crawling(image_id)
```

csv file을 초기화 하고
반복 문을 사용 1에서 100까지의 id를 입력으로 image_crawling 함수를 실행한다.
만들어질 csv 데이터는 아래와 같음

id	image_url	tags	source	image_path
1	https://~~~	”a” “b” “c”	“evangelion”	~~~/image/*.png
N	https://~~~	”a” “b”	“ghost in the shell”	~~~/image/*.jpeg

이미지 크롤링 수집 데이터

id	image_url	tags	source	image_path
1	https://e-shui	"4koma" "bla	"Habanero-ta	image/2005-04-05-1.jpeg
2	https://e-shui	"book" "brown hair" "cospl		image/2005-04-05-2.jpeg
3	https://e-shui	"animated" "t	"Full Moon w	image/2005-04-06-3.gif
4	https://e-shui	"ahoge" "blac	"Cardcaptor S	image/2005-04-08-4.jpeg
10	https://e-shui	"blush" "brow	"Azumanga D	image/2005-04-08-10.jpeg
13	https://e-shui	"blonde hair"	"Rozen Maide	image/2005-04-08-13.jpeg
14	https://e-shui	"black hair" "	"Full Moon w	image/2005-04-08-14.jpeg



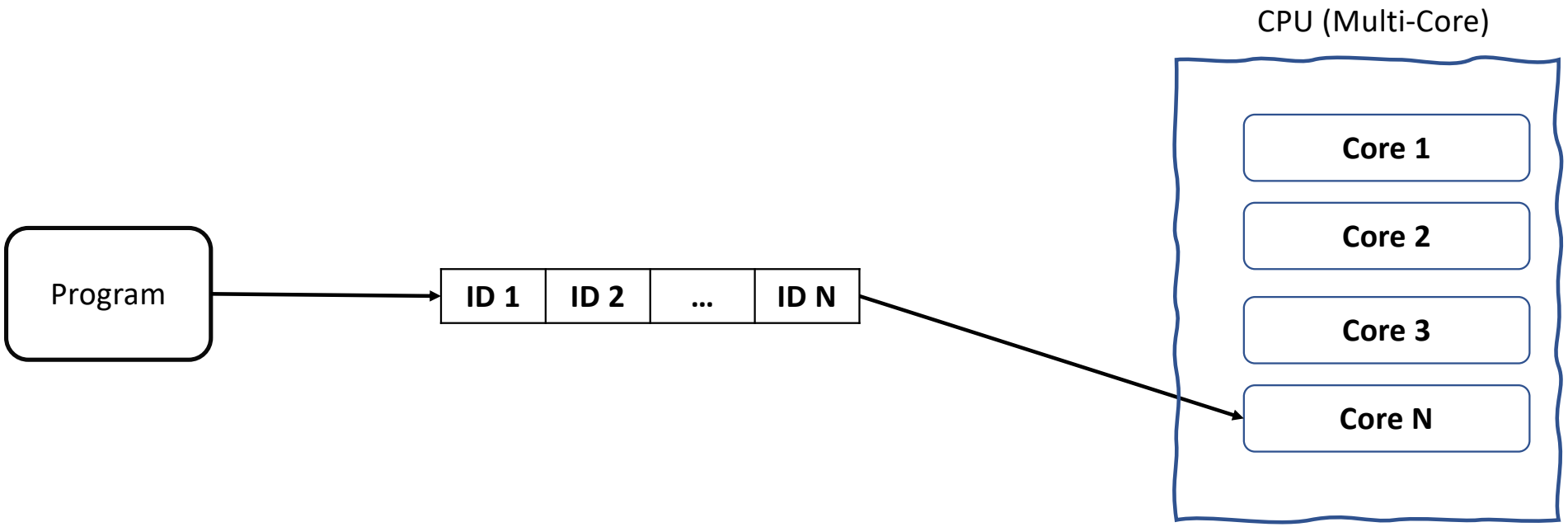
저장된 shuushuu csv 데이터 및 이미지 데이터

멀티 프로세싱 크롤러

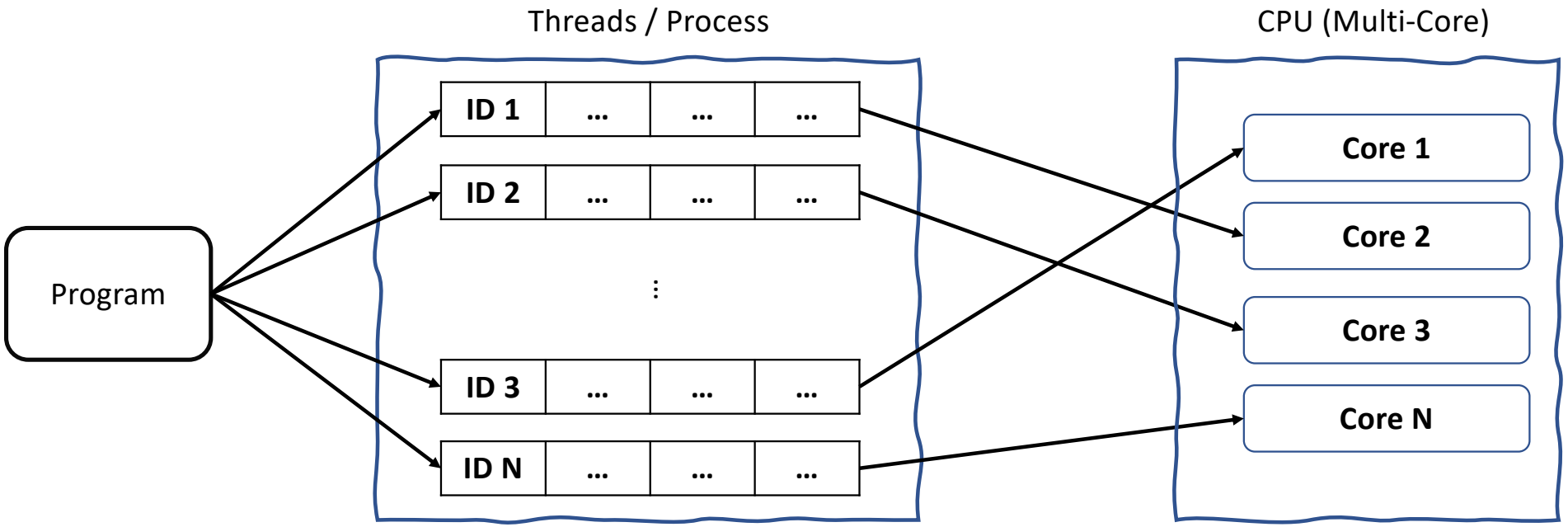
프로젝트
최상위 경로

- ShuuShuu.py : 작성 및 실행할 파이썬 코드
- Shuushuu_image_metadata.csv : ShuuShuu 크롤링 데이터가 저장될 CSV file
- image : ShuuShuu 이미지가 저장될 폴더

싱글 프로세싱 크롤러



멀티 프로세싱 크롤러


Threads / Process
CPU (Multi-Core)
Program
ID 1
ID 2
ID 3
ID N
...
Core 1
Core 2
Core 3
Core N

1. Python Package import

ShuuShuu.py

```
from multiprocessing import Pool
import requests
import urllib.request
import random
import time
import bs4
import csv
from tqdm import tqdm
import os
```

```
from multiprocessing import Pool
# 멀티 프로세싱을 하기위한 모듈
```

2. Main 함수

ShuuShuu.py

```
if __name__ == '__main__':
    header = ['id', 'image_url', 'tags', 'source', 'image_path']
    csv_init(file_path, header)

    with Pool(4) as pool:
        for _ in tqdm(pool.imap_unordered(image_crawling, range(1, 100)),
        total=100):
            pass
```

```
with Pool(4) as pool
# 멀티 프로세싱 풀을 생성 Pool(최대 병렬처리 개수)

pool.imap_unordered(image_crawling, range(1, 100)
# 풀에서 작업할 크롤링 함수(image_crawling) 및 데이터(range(1, 100), ID값)을 매핑
```

3. 싱글 / 멀티 프로세싱 크롤러

싱글

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+
50123	yslee	24	0	5129M	29872	?	3.1	0.1	0:00.31

멀티

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+
50049	yslee	24	0	5123M	21580	?	1.8	0.1	0:00.17
50051	yslee	24	0	5019M	19104	?	1.5	0.1	0:00.15
50048	yslee	24	0	5252M	21192	?	1.1	0.1	0:00.19
50050	yslee	24	0	5013M	20308	?	1.1	0.1	0:00.17
50045	yslee	24	0	4843M	22620	?	0.1	0.1	0:00.19