



고용노동부



한국산업인력공단



국가인적자원개발컨소시엄

프로그래밍

Python 데이터 수집과 분석

2018.4.9 ~ 4.13

백명숙



과정개요

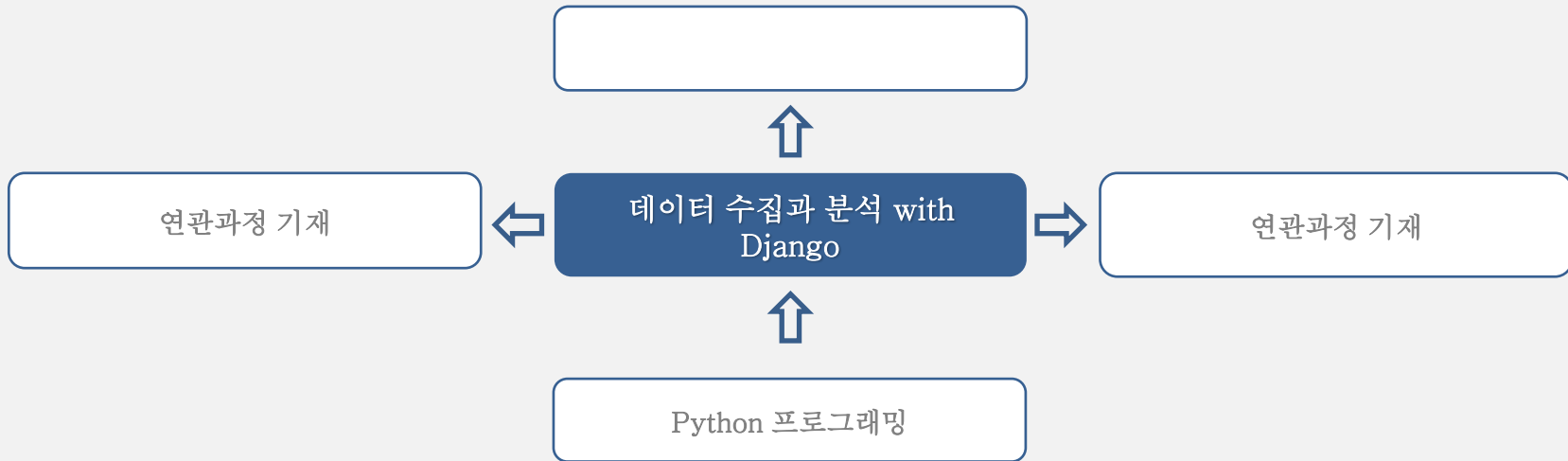
- 파이썬을 통해 웹의 데이터를 수집하고 가공하는 방법에 대해서 학습합니다.
- 웹 스크래핑과 크롤링 기법을 통해 웹에 존재하는 데이터를 추출하는 방법을 알아봅니다.
- 수집된 데이터를 분석하고, 기본적인 머신러닝에 대한 내용을 학습합니다

과정목표

- 파이썬으로 웹 상의 데이터를 스크래핑 할 수 있는 능력
- 기본적인 머신러닝의 방법을 이해할 수 있는 능력

러닝 맵

- 본 과정 이수와 관련 선수과정, 후수과정과 순서에는 관계없이 해당 과정을 이해하는데 도움이 되는 연관과정을 기재해 주십시오.
- 과정명 기재시 본 협회 컨소시엄 재직자 과정 150개 중에서 선·후수 과정, 연관 과정명을 기재하시면 됩니다.

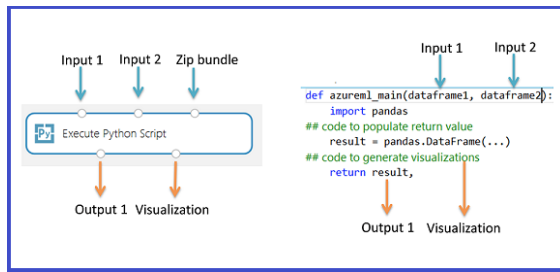


강사 프로필

성명	백명숙
소속 및 직함	휴클라우드 전임강사
주요 경력	일은시스템(제일은행 IT자회사), Sun Microsystems 교육서비스, 인터넛커머스코리아
강의/관심 분야	Java, Framework, Python, Data Science, Machine Learning, Deep Learning
자격/저서/대외활동	Java기반 오픈소스 프레임워크/NCS 개발위원

학습모듈(Learning Object) 및 목차

LO	커리큘럼
데이터 수집(웹스크래핑)	<ul style="list-style-type: none">- 개발환경 설정- Requests- BeautifulSoup- Selenium
데이터 분석 (머신러닝 입문)	<ul style="list-style-type: none">- Pandas를 이용한 데이터 분석- Load Dataset- Train (학습하기)- Visualize (시각화하기)- Predict (예측하기)



데이터수집과 분석 with Python



목차

- 01. 개발환경 설치
- 02. Web Scraping (Requests, BeautifulSoup, Selenium)
- 03. 데이터 분석과 시각화 – Machine Learning
 - : Predict survival on the Titanic

웹 스크래핑 : 파이썬으로 데이터 수집/처리 에서의 위치

- 데이터 수집
 - 오픈 API
 - 웹 스크래핑
- 데이터 가공
 - 데이터 랭글링 (Data Wrangling)
 - 데이터 클리닝
 - 데이터 분석이나 머신러닝(딥러닝)을 위한 데이터 전처리
- 데이터 분석
 - 통계적 분석
 - 머신러닝 (딥러닝)
- 데이터 저장
 - 관계형/NoSQL 데이터베이스
 - 엑셀파일, CSV/TSV, JSON, YAML
- 커뮤니케이션
 - 이메일, 메신저, slack
 - 데이터 시각화

01. 개발환경 구성

환경 구성하기

■ Anaconda (<https://www.continuum.io/anaconda-overview>)

- 대용량 데이터 처리, 예측 분석, 과학 계산을 파이썬 배포판입니다. Anaconda아나콘다는 NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook, 그리고 scikit-learn을 모두 포함합니다.
- macOS, 윈도우, 리눅스를 모두 지원하며 매우 편리한 기능을 제공하므로 파이썬 과학 패키지가 없는 사람에게 추천하는 배포판 입니다.

■ Miniconda 설치 - <http://conda.pydata.org/miniconda.html>

- Anaconda는 데이터 분석을 위한 오픈 소스 Python 플랫폼입니다. 단 한 번의 설치로 모든 환경 설정을 한 번에 끝내주기 때문에 사용하기에 아주 좋습니다.
- 한편 miniconda는 Anaconda의 핵심 부분만을 추출해서 재구성한 플랫폼으로, 고유한 패키지 관리 시스템인 conda와 기본적인 Python만을 포함하고 있기 때문에 설치 소요 시간이 더 짧습니다.

■ Conda 버전 업데이트

```
conda update conda
```

■ Python 버전 확인

```
python --version
```

Ipython과 Jupyter Notebook

■ Ipython 설치

```
pip install ipython
```

■ Jupyter Notebook 설치

```
pip install jupyter
```

■ Jupyter Notebook 실행방법

```
jupyter notebook
```

Jupyter Notebook이 실행되면 <http://localhost:8888>에서 실행된 웹 브라우저를 실행하여 해당 URL로 접속하면, Jupyter Notebook의 GUI 화면을 확인. 상단의 New 버튼을 클릭, 맨 하단의 Python 3 메뉴를 클릭하면, 새로운 notebook을 생성하고 편집이 가능하다.

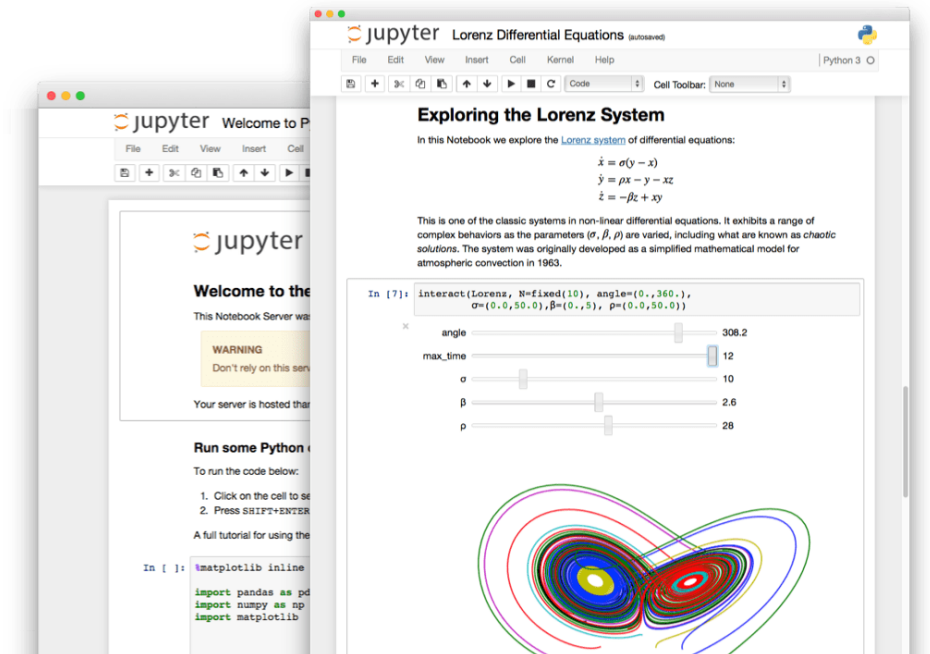
■ Pandas 설치

```
pip install jupyter
```

Jupyter notebook

- * IPython 에서 최근 Jupyter 로 이름이 변경
- * 기본 파이썬 셸(REPL) 에 몇 가지 강력한 기능을 추가한 것
- * 데이터 분석할 때 주로 많이 사용됨
- * 노트 형식의 주석(문서화)을 추가 할 수 있는 것이 특징
- * 웹브라우저에서 수행되어 시각화와 스크립트의 저장이 간편함

* <http://jupyter.org>



Web Scraping

: Requests, BeautifulSoup, Selenium

웹 데이터 수집

- * 웹 데이터

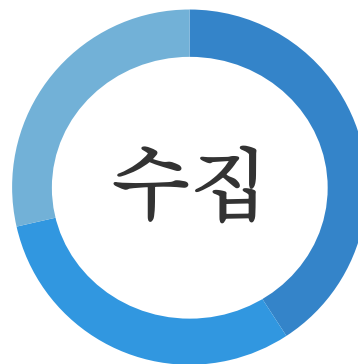
- EMC - Digital Universe (2020년에는 35ZB, 생성된 데이터의 95% 디지털화)

- * 데이터 수집의 3단계



1단계 : 대상선정

목적 데이터의 위치를 파악



2단계 : 수집

대상 위치에서 원하는 데이터를 수집



3단계 : 정리

수집된 데이터를 정리

웹 데이터 수집 - 자동화

- * 웹스크래핑과 크롤러

- 스크래핑 - 각각의 페이지에서 정보를 추출하는 행위
- 크롤러 - 자동으로 정보추출을 반복하는 프로그램

- * 반 자동화 프로그램

- 수 작업의 일부를 프로그래밍 지원하는 형태
- 1단계 : 수집할 페이지를 지정하여 프로그램 시작 → 수동
- 2단계 : 대상 페이지를 내려 받고 특정 데이터 추출 → 프로그램
- 3단계 : 수집한 데이터를 일정 형식으로 저장 → 수동 또는 프로그램

- * 완전 자동화 프로그램

- 반 자동화 프로그램의 모든 부분을 자동화 프로그램으로 작성하여 실행
- 스케줄링을 이용하여 순환/반복 기능을 가짐 - 크롤러
- 변화에 취약하다는 단점 존재

웹 데이터 수집 - 주의사항

* 수집 데이터의 처리와 저작권

- 웹 사이트의 정보는 기본적으로 저작물
- 정보를 읽어올 수 있다고 해서, 마음대로 활용할 수 있다는 것은 아닙니다.
저작권에 유의해 주세요.
- 2016년 제정된 저작권법 제 30조 : 정보 해석을 목적으로 저작물을 복제/변
안 가능

* 웹 사이트의 리소스 압박과 업무 방해

- 웹 사이트의 자원을 독점하게 되면 다른 사람이 웹 사이트를 이용할 수 없음
- 무한 크롤러 사용 시 업무방해 혐의 적용 가능

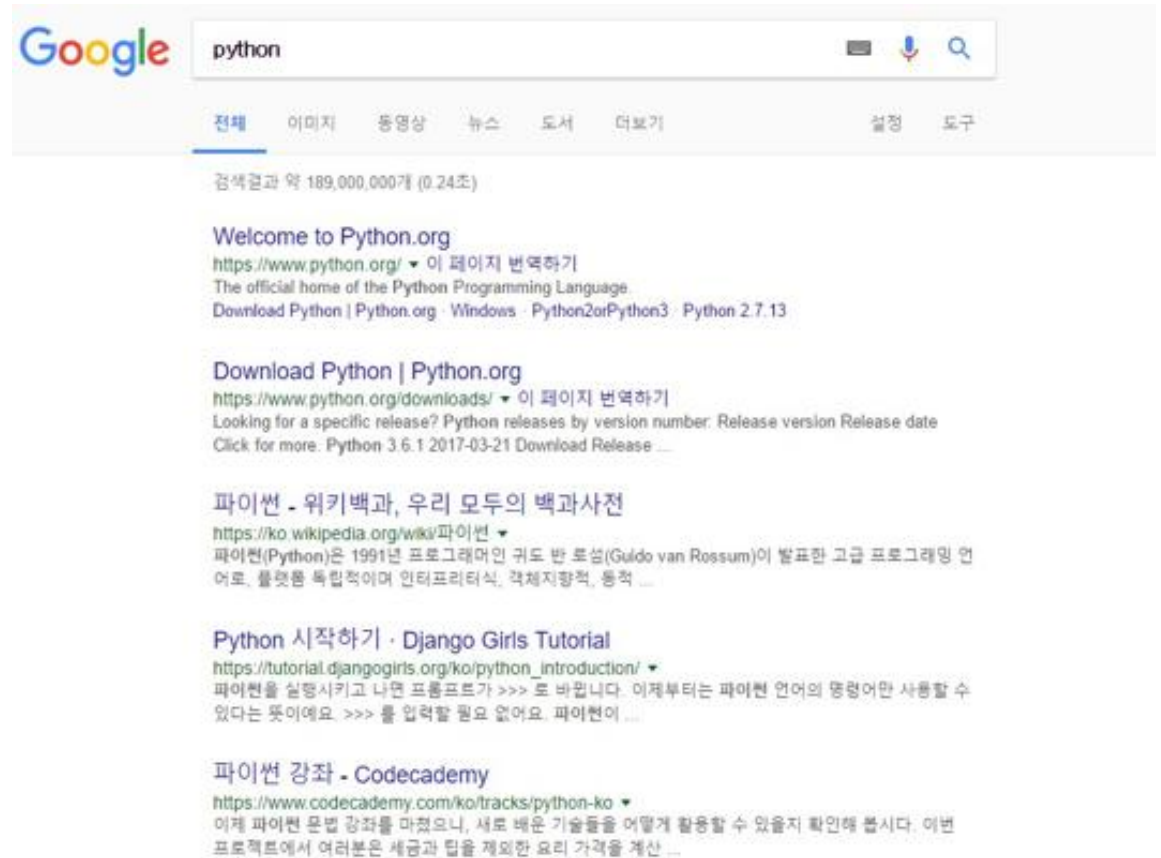
* 크롤러와 API

- 해당 사이트에서 API 지원 여부 확인

크롤링 사례 - 구글

구글 Google

수 많은 웹 사이트
를 크롤링하여 검
색서비스 제공



Google python

전체 이미지 동영상 뉴스 도서 더보기 설정 도구

검색결과 약 189,000,000개 (0.24초)

Welcome to Python.org
<https://www.python.org/> 이 페이지 번역하기
The official home of the Python Programming Language
[Download Python](#) | [Python.org](#) · [Windows](#) · [Python2orPython3](#) · [Python 2.7.13](#)

Download Python | Python.org
<https://www.python.org/downloads/> 이 페이지 번역하기
Looking for a specific release? Python releases by version number: Release version Release date
Click for more: [Python 3.6.1](#) [2017-03-21](#) [Download](#) [Release](#) ...

파이썬 - 위키백과, 우리 모두의 백과사전
<https://ko.wikipedia.org/wiki/파이썬> ▼
파이썬(Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 ...

Python 시작하기 - Django Girls Tutorial
https://tutorial.djangogirls.org/ko/python_introduction/ ▼
파이썬을 실행시키고 나면 프롬프트가 >>> 로 바뀝니다. 이제부터는 파이썬 언어의 문법어만 사용할 수 있다는 뜻이에요. >>> 를 입력할 필요 없어요. 파이썬어 ...

파이썬 강좌 - Codecademy
<https://www.codecademy.com/ko/tracks/python-ko> ▼
이제 파이썬 문법 강좌를 마쳤으니, 새로 배운 기술들을 어떻게 활용할 수 있을지 확인해 봅시다. 이번 프로젝트에서 여러분은 세금과 팁을 제외한 요리 가격을 계산 ...

크롤링 사례 - 쿠차

쿠차 COOCHA

각종 소셜커머스 사이트를 크롤링 하여 최저가 정보 제공

The screenshot displays the COOCHA website interface. At the top, there's a navigation bar with the COOCHA logo, a search bar, and a shopping cart icon. Below the navigation bar, a section titled '최저가 핫딜' (Lowest Price Hot Deal) features four product cards, each showing a different water bottle product with its price and a '최저가' (Lowest Price) badge.

Below the product cards, a detailed view for 'Jeju Sangdo Sangsu 2L [1개]' is shown. This view includes a product image, a star rating, and a table comparing prices across different retailers.

가장 낮은 가격		상품명 #	
소매점	판매가	비교가	추가 혜택
CU	830원	1,000원	최저가 할인
신세계	900원	1,000원	CU와 동일
CU	1,020원	1,000원	7% 할인
CU	1,020원	1,000원	7% 할인
11번가	1,100원	1,000원	CU와 동일
CU	1,210원	1,000원	CU와 동일
CU	1,200원	1,000원	7% 할인

크롤링 사례 - 지진발생 알림

각종 커뮤니티에서 지진에 관련된 글을 수집하여 지진 발생시 텔레그램으로 알림

지진희알림

7568 members

September 21

지진희알림

2016-09-21 11:53:55

규모: 3.7

경북 경주시 남서쪽 11Km 지역

👁 4642 11:54 AM

지진희알림

클리앙 지진관련 11개의 글

1. 울산지진
2. 현직 양산 지진이네요.
3. 포항에도 지진이 ㄸㄸ
4. 지진
5. 대구 지진인가요?
6. 지진?
7. 경주 지진이네요.
8. 대구 지진....
9. 또 지진이네요..
10. 경주 지진이네요
11. 울산 지진이 또!!

👁 4553 11:55 AM

Web 의 이해

- * 웹 페이지 Web Page
 - 웹 상의 문서
 - 우리가 보고 있는 웹 사이트들은 문서로 이루어져 있다.
 - 텍스트, 그림, 소리, 동영상 등을 표현 가능
 - 대부분 HTML 이라는 언어로 이루어져 있음



서버와 클라이언트

Client

- 서비스를 요청하는 프로그램

Server

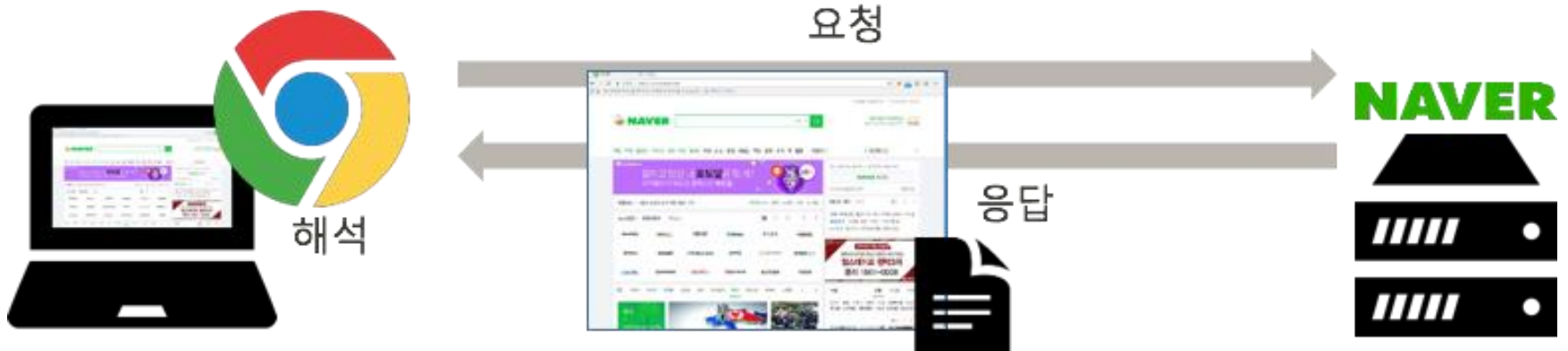
- 요청에 대해 응답을 해주는 프로그램



Web 의 이해

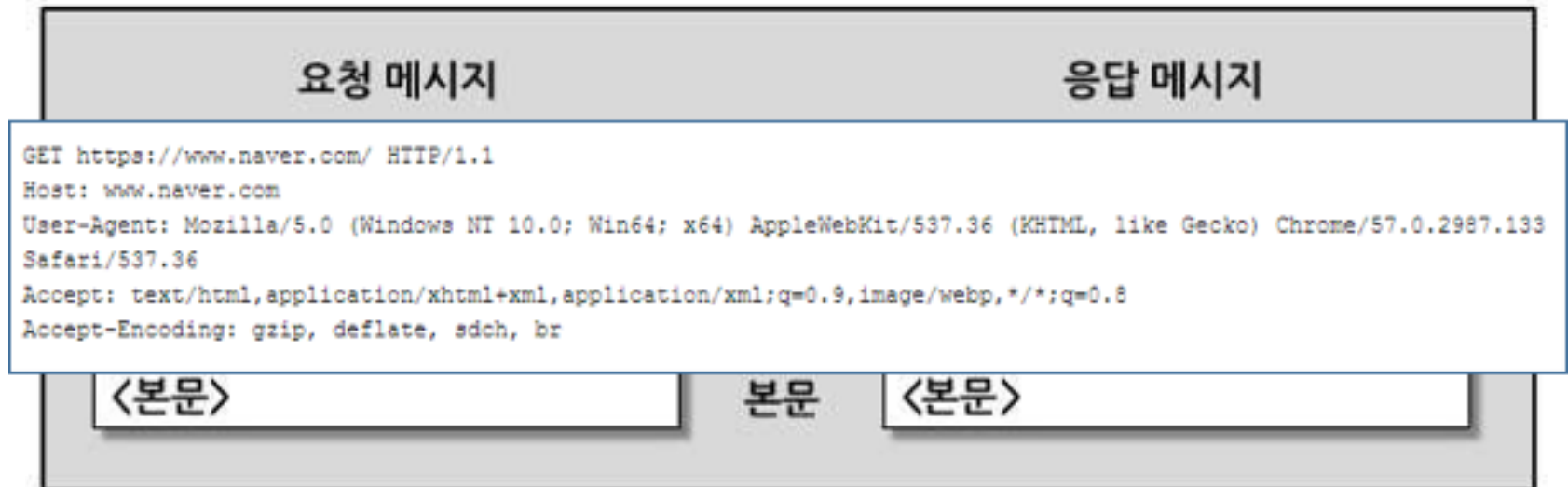
* 웹 페이지 해석 순서

1. 클라이언트가 서버에게 contents를 요청한다.
2. 서버는 요청 받은 contents를 클라이언트에게 건네준다.
3. 브라우저는 서버에게 받은 HTML을 해석하여 화면에 보여준다



Web 의 이해

- * HTTP - Hyper Text Transfer Protocol
 - 서버와 클라이언트 사이에서 정보를 주고 받기 위한 규약
 - 시작줄, 헤더(Header), 본문(Body) 으로 이루어져 있음
 - 9개의 메소드가 존재하지만 주로 GET과 POST만 쓰인다.



HTTP

GET

- Body 없이 Header만으로 전송된다.
- 링크 / 북마크 가 가능하다.
- 요청에 길이 제한이 있다.

/test/demo_form.php?name1=value1&name2=value2

- URL의 ? 뒤에 쿼리 문자열이 올 수 있다.
- 쿼리 문자열은 key와 value를 가지고 있으며, 각 쿼리는 & 로 구분한다.

POST

- Body에 query data가 들어간다.
- 링크 / 북마크가 불가능하다.
- 데이터 길이에 제한이 없다.
- URL을 가지지 않으므로 주로 중요한 데이터를 다룰 때 사용한다.

HTTP Client 모듈 - Python

- * urllib

- Python built-in module
- 간편하게 HTTP request를 보낼 수 있음
- 로그인 및 세션을 유지하기가 번거로움

- * Requests [#doc](#)

- 간편하게 HTTP request를 보낼 수 있음
- 세션을 유지하기가 용이함
- python2 / python3 완벽 지원
- 코드가 간결하고 documentation이 잘 되어 있음

- * Selenium [#doc](#) : 웹브라우저 자동화 tool

- javascript/css 지원, 기존 GUI 브라우저 자동화 라이브러리
- 사람이 웹서핑 하는 것과 동일한 환경, 대신에 리소스를 많이 사용함
- 웹브라우저에서 HTML에 명시된 이미지/CSS/JavaScript를 모두 자동 다운로드/적용

requests 설치

```
pip install requests
```



Requests: HTTP for Humans

Release v2.18.4. ([Installation](#))

license Apache 2.0 wheel yes python 2.6, 2.7, 3.4, 3.5, 3.6 codecov 88% Say Thanks! 🐻

Requests is the only *Non-GMO* HTTP library for Python, safe for human consumption.

- 파이썬에서는 기본 라이브러리로 urllib이 제공되지만, 이보다 간결한 코드로 다양한 HTTP요청을 할 수 있는 최고의 라이브러리
- JavaScript 처리가 필요한 경우에는 selenium을 고려할 순 있겠습니다. 하지만 이 경우에도 requests 적용이 가능할 수도 있습니다. 크롤링 할 페이지에 대해 다각도로 검토가 필요합니다.
- 크롤링 시에 웹 요청에 requests를 쓸 수 있다면, 가장 효율적으로 처리 가능

requests: GET 요청

* 단순 GET 요청

```
import requests  
  
response = requests.get('http://news.naver.com/main/home.nhn')
```

* GET 요청 시에 커스텀 헤더 지정

```
request_headers = {  
    'User-Agent': ('Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 '  
    '(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36'),  
    'Referer': 'http://news.naver.com/main/home.nhn', # 뉴스홈  
}  
  
response = requests.get('http://news.naver.com/main/main.nhn', headers=request_headers)
```

requests 라이브러리에서의 기본 *User-Agent* 값은 '*python-requests*/버전'입니다.
서버에 따라 *User-Agent* 값으로 응답 거부 여부를 결정 하기도 합니다.

requests: GET 요청

- * GET 요청 시에 GET인자(? 뒤에 붙이는 urlencoded) 지정

- params 인자로 dict 지정 : 동일 Key의 인자를 다수 지정 불가

```
get_params = {'mode': 'LSD', 'mid': 'shm', 'sid1': '105'} # IT/과학 탭을 위한 GET인자
response = requests.get('http://news.naver.com/main/main.nhn', params=get_params)
```

- params 인자로 (key, value) 형식의 tuple 지정 : 동일 Key의 인자를 다수 지정 가능

```
get_params = [('mode', 'LSD'), ('mid', 'shm'), ('sid1', '105')]
response = requests.get('http://news.naver.com/main/main.nhn', params=get_params)
get_params = (('k1', 'v1'), ('k1', 'v3'), ('k2', 'v2'))
response = requests.get('http://httpbin.org/get', params=get_params)
```

requests: GET 응답

- * 상태코드

```
>>> response.status_code    #int
```

```
>>> response.ok    # status_code가 200이상 400미만의 값인지 여부 (bool)
```

- * 응답 헤더

dict 타입이 아니라 requests.structures.CaseInsensitiveDict 타입

Key문자열은 대소문자를 가리지 않습니다.

각 헤더의 값은 헤더이름을 Key로 접근 하여 획득

```
>>> response.headers
```

```
>>> response.headers['Content-Type'] # Key문자열 대소문자에 상관없이 접근  
'text/html; charset=UTF-8'
```

```
>>> response.headers['content-type']  
'text/html; charset=UTF-8'
```

```
>>> response.encoding  
'UTF-8'
```

requests: GET 응답

- 응답 Body

`bytes_data = response.content # 응답 Raw 데이터 (bytes)`

`str_data = response.text # response.encoding으로 디코딩하여 유니코드 변환`

- 이미지 데이터일 경우에는 `.content`만 사용

`with open('flower.jpg', 'wb') as f:`

`f.write(response.content)`

- 문자열 데이터일 경우에는 `.text`를 사용

`html = response.text`

`html = response.content.decode('utf8') # 혹은 .content 필드를 직접 디코딩`

- json 포맷의 응답일 경우

`json.loads(응답문자열)`을 통해 직접 Deserialize를 수행 혹은 `.json()`함수를 통해 Deserialize 수행

```
import json
```

```
obj = json.loads(response.text)
```

```
obj = response.json() # 위와 동일
```

requests: GET 응답

- * 한글 인코딩

charset 정보가 없을 경우, 먼저 *utf8*로 디코딩을 시도하고

*UnicodeDecodeError*가

발생할 경우, *iso-8859-1 (latin-1)*로 디코딩을 수행. 이때 한글이 깨진 것처럼 보여집니다.

이때는 다음과 같이 직접 인코딩을 지정한 후에 `.text`에 접근해주세요.

```
>>> response.encoding
```

```
'iso-8859-1'
```

```
>>> response.encoding = 'euc-kr'
```

```
>>> html = response.text
```

- * 혹은 `.content`를 직접 디코딩 할 수도 있습니다.

- *

```
>>> html = response.content.decode('euc-kr')
```


requests: POST 요청

- * 단순 POST 요청

```
response = requests.post('http://httpbin.org/post')
```

- * 단순 POST 요청시 커스텀 헤더, GET 인자 지정

```
request_headers = {  
    'User-Agent': ('Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 '  
    '(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36'),  
    'Referer': 'http://httpbin.org',  
}  
  
get_params = {'k1': 'v1', 'k2': 'v2'}  
  
response = requests.post('http://httpbin.org/post', headers=request_headers,  
params=get_params)
```

- * data인자로 dict지정 : 동일 Key의 인자를 다수 지정 불가

```
data = {'k1': 'v1', 'k2': 'v2'}  
  
response = requests.post('http://httpbin.org/post', data=data)
```

- * data인자로 (key, value) 형식의 tuple 지정 : 동일 Key의 인자를 다수 지정 가능

```
data = (('k1', 'v1'), ('k1', 'v3'), ('k2', 'v2'))  
  
response = requests.post('http://httpbin.org/post', data=data)
```

requests: POST 요청

- * JSON POST 요청

- JSON 인코딩

```
import json
```

```
json_data = {'k1': 'v2', 'k2': [1, 2, 3], 'name': 'Django'}
```

- json포맷 문자열로 변환한 후, data인자로 지정

```
json_string = json.dumps(json_data, ensure_ascii=False)
```

```
response = requests.post('http://httpbin.org/post', data=json_string)
```

- 객체를 json인자로 지정하면, 내부적으로 json.dumps 처리

```
response = requests.post('http://httpbin.org/post', json=json_data)
```

requests: POST 요청

- * 파일 업로드 요청

multipart/form-data 인코딩

```
files = {
```

```
    'photo1': open('f1.jpg', 'rb'), # 데이터만 전송
```

```
    'photo2': open('f2.jpg', 'rb'),
```

```
    'photo3': ('f3.jpg', open('f3.jpg', 'rb'), 'image/jpeg', {'Expires': '0'}),
```

```
}
```

```
post_params = {'k1': 'v1'}
```

```
response = requests.post('http://httpbin.org/post', files=files,  
data=post_params)
```

파싱 (parsing)

- 가공되지 않은 문자열에서 필요한 부분을 추출하여 의미있는 (구조화된) 데이터로 만드는 과정

HTML DOM

- * HTML DOM = HTML 엘리먼트(태그) 로 구성된 Tree

```
<!DOCTYPE html>
<html>
  <head>
    <title>타이틀</title>
  </head>
  <body>
    <h1>제일 큰 제목</h1>
    <div>파이썬 웹 스크래핑</div>
  </body>
</html>
```

html > body > div
> 는 앞선 엘리먼트의 자식 (child)

웹상에서 특정 문자열 정보를 가져오려면?

- * 방법1: 정규 표현식을 활용
 - 가장 빠른 처리가 가능하나, 정규 표현식 Rule을 만드는 것이 많이 번거롭고 복잡합니다.
 - 때에 따라 필요할 수도 있습니다.
- * 방법2: HTML Parser 라이브러리를 활용
 - DOM Tree을 탐색하는 방식으로 적용이 쉽습니다.
 - ex) BeautifulSoup4, lxml

BeautifulSoup

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

```
pip install beautifulsoup4
```

- * 파싱을 도와주는 강력한 python 라이브러리
- * HTML/XML Parser : HTML/XML문자열에서 원하는 태그 정보를 추출합니다.
- * 정규식을 작성할 필요 없이 tag, id, class 등의 이름으로 쉽게 파싱 가능
- * 쉽고 간결하며, documentation이 매우 잘 되어 있음

BeautifulSoup 코드

```
from bs4 import BeautifulSoup

html = '''
<ol>
  <li>NEVER - 국민의 아들</li>
  <li>SIGNAL - TWICE</li>
  <li>LONELY - 씨스타</li>
  <li>I LUV IT - PSY</li>
  <li>New Face - PSY</li>
</ol>
'''

soup = BeautifulSoup(html, 'html.parser')
for tag in soup.select('li'):
    print(tag.text)
```


BeautifulSoup Parser

- BeautifulSoup4 내장 파서

```
soup = BeautifulSoup(파싱할문자열, 'html.parser')
```

- lxml HTML 파서 사용 (외부 C 라이브러리)
 - html.parser 보다 좀 더 유연하고, 빠른 처리
 - 설치 : pip3 install lxml
 - soup = BeautifulSoup(파싱할문자열, 'lxml')

- Tag를 찾는 2가지 방법

1. find를 통해 태그 하나씩 찾기
2. 태그 관계를 지정하여 찾기 (CSS Selector 사용)

BeautifulSoup : find()

* find() 메서드 사용

- find ('tag_name') : 태그 이름으로 엘리먼트 찾기
- find ('css_class_name') : CSS 클래스 명으로 엘리먼트 찾기

```
soup.find('title')  
soup.find('.className')
```

```
import requests  
  
url = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"  
html = requests.get(url).text  
# BeautifulSoup으로 분석하기  
soup = BeautifulSoup(html, 'html.parser')  
# 원하는 데이터 추출하기  
title = soup.find("title").string  
wf = soup.find("wf").string  
print(title, wf)
```

BeautifulSoup : find_all, select()

* find_all() 메소드 사용

- find_all('tag_name') : 태그 이름으로 엘리먼트 찾기
- find_all('css_class_name') : CSS 클래스 명으로 엘리먼트 찾기

```
soup.find_all('a')  
soup.find_all('.className')
```

* select() 메소드 사용

- select('css_선택터') : CSS 선택터 문법으로 엘리먼트 찾기 (배열로 반환)
- select_one('css_선택터') : 하나의 엘리먼트만 반환

```
soup.select('#myId > div.className > a')  
soup.select_one('#myId > div.className > a')
```

BeautifulSoup 는 CSS 선택터 문법 중 :nth-child 를 지원하지 않음

BeautifulSoup : attrs()

- * attrs 속성 사용

- a.attrs :
- 해당 엘리먼트에서 속성 추출하기
- <a> 태그의 모든 속성을 dict 타입으로 반환

```
>>> a = soup.find_all('a')
>>> type(a.attrs)
<class 'dict'>
>>> a['href']
'a.html'
```

BeautifulSoup : find(), find_all()

예시)멜론 TOP100 차트

```
import requests

from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101
    Firefox/57.0',
}

html = requests.get('http://www.melon.com/chart/index.htm',headers=headers).text
soup = BeautifulSoup(html, 'html.parser')
tag_list = []
for tr_tag in soup.find(id='tb_list').find_all('tr'):
    tag = tr_tag.find(class_='wrap_song_info')
    if tag:
        tag_sub_list = tag.find_all(href=lambda value: (value and 'playSong' in value))
        tag_list.extend(tag_sub_list)
for idx, tag in enumerate(tag_list, 1):
    print(idx, tag.text)
```

BeautifulSoup : CSS Selector 사용

- * CSS Selector를 통한 Tag 찾기 지원
 - tag name : "tag_name"
 - tag id : "#tag_id"
 - tag class names : ".tag_class"
- * CSS Selector를 통한 Tag 찾기 지원
 - * : 모든 Tag
 - tag : 해당 모든 Tag
 - Tag1 > Tag2 : Tag1 의 직계인 모든 Tag2
 - Tag1 Tag2 : Tag1 의 자손인 모든 Tag2 (직계임이 요구되지 않음)
 - Tag1, Tag2 : Tag1이거나 Tag2인 모든 Tag
 - tag[attr] : attr속성이 정의된 모든 Tag
 - tag[attr="bar"] : attr속성이 "bar"문자열과 일치하는 모든 Tag
 - tag[attr*="bar"] : attr속성이 "bar"문자열과 부분 매칭되는 모든 Tag
 - tag[attr^="bar"] : attr속성이 "bar"문자열로 시작하는 모든 Tag
 - tag[attr\$="bar"] : attr속성이 "bar"문자열로 끝나는 모든 Tag

BeautifulSoup : CSS Selector 사용

- * CSS Selector를 통한 Tag 찾기 지원
 - `tag#tag_id` : id가 `tag_id`인 모든 Tag
 - `tag.tag_class` : 클래스명 중에 `tag_class`가 포함된 모든 Tag
 - `tag#tag_id.tag_cls1.tag_cls2`
: id가 `tag_id` 이고, 클래스명 중에 `tag_cls1`와 `tag_cls2`가 모두 포함된 Tag
 - `tag.tag_cls1.tag_cls2`
: 클래스명 중에 `tag_cls1`와 `tag_cls2`가 모든 포함된 모든 Tag
 - `tag.tag_cls1 .tag_cls2`
: 클래스명 중에 `tag_cls1`이 포함된 Tag의 자식 중에 (직계가 아니어도 OK), 클래스명에 `tag_cls2`가 포함된 모든 Tag
- * CSS Selector를 지정할 때에는
 - 패턴을 너무 타이트하게 지정하시면, HTML 마크업이 조금만 변경되어도 태그를 찾을 수 없게 됩니다.

BeautifulSoup : CSS Selector를 사용하여 태그 찾아가기

```
<ul id="bible">
  <li id="ge">비밀물고기</li>
  <li id="ex">나의첫사회생활</li>
  <li id="le">Still Me</li>
  <li id="nu">화염과 분노 : 도널드 트럼프의 백악관 뒷이야기</li>
  <li id="de">매일 좋을 수만은 없는 여행을 한다 </li>
</ul>
```

```
<div id="main-goods" role="page">
  <h1>과일과 야채</h1>
  <ul id="fr-list">
    <li class="red green" data-lo="ko">사과</li>
    <li class="purple" data-lo="us">포도</li>
    <li class="yellow" data-lo="us">레몬</li>
    <li class="yellow" data-lo="ko">오렌지</li>
  </ul>
  <ul id="ve-list">
    <li class="white green" data-lo="ko">무</li>
    <li class="red green" data-lo="us">파프리카</li>
    <li class="black" data-lo="ko">가지</li>
    <li class="black" data-lo="us">아보카도</li>
    <li class="white" data-lo="cn">연근</li>
  </ul>
</div>
```


BeautifulSoup : CSS Selector를 사용하여 태그 찾아가기

예시)멜론 TOP100 차트

```
import requests

from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101
    Firefox/57.0',
}

html = requests.get('http://www.melon.com/chart/index.htm',headers=headers).text
soup = BeautifulSoup(html, 'html.parser')
tag_list = soup.select('#tb_list tr .wrap_song_info a[href*=playSong]')
for idx, tag in enumerate(tag_list, 1):
    print(idx, tag.text)
```

이미지 파일 다운로드

- * `res = requests.get(이미지 파일 경로)`
 - `res.content` : 이미지파일을 바이너리로 받아옴
 - 해당 바이너리를 파일로 저장

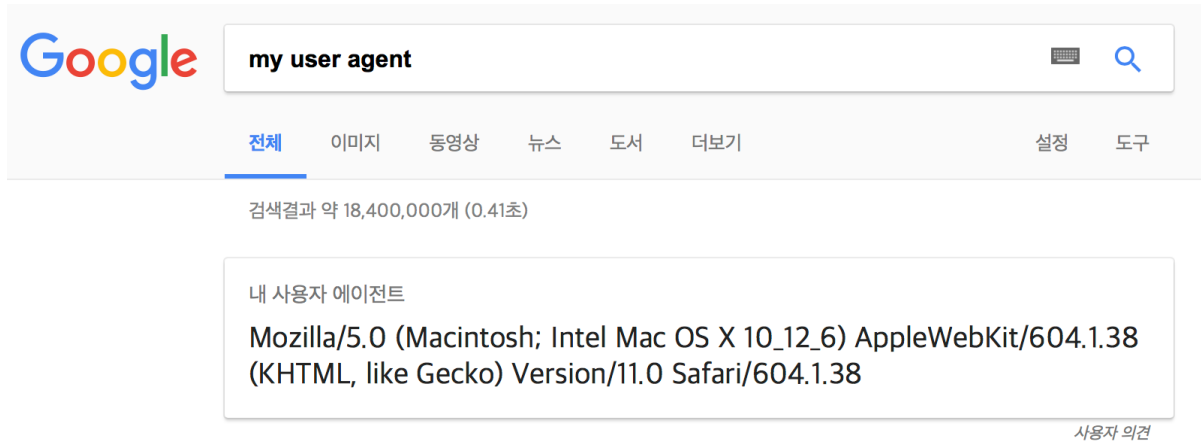
```
import requests

res = requests.get(
    'http://www.jetbrains.com/idea/img/screenshots/idea_overview_5_1.png')

# print(res.content)

with open('img.png', 'wb') as f:
    f.write(res.content)
```

헤더를 수정하세요



- * user-agent 헤더 수정
 - 기본 urllib 모듈을 사용했을 때 user-agent
Python-urllib/3.4

브라우저에서 요청하는 것 같이 보이기!!

헤더 수정 - requests

- * requests 를 사용해 헤더를 별도로 설정하여 요청

```
import requests

# 헤더 설정을 위해서 session 객체 생성
session = requests.Session()

# User-Agent 설정
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh . . . safari/604.1.38'
}

# 세션을 통해 get 요청을 시도
res = session.get('http://www.naver.com', headers=headers)

# 응답 받은 HTML 을 출력
print(res.text)
```

헤더 수정 - requests

예시)네이버 웹툰

```
import os
import requests
# URL 소스 :
http://comic.naver.com/webtoon/detail.nhn?titleId=119874&no=1015&weekday=tue
image_urls = [

'http://imgcomic.naver.net/webtoon/119874/1015/20170528204207_6e9df8e618b97520233bbb35e7d4eaaf_IMAG01_1.jpg',

'http://imgcomic.naver.net/webtoon/119874/1015/20170528204207_6e9df8e618b97520233bbb35e7d4eaaf_IMAG01_2.jpg',

'http://imgcomic.naver.net/webtoon/119874/1015/20170528204207_6e9df8e618b97520233bbb35e7d4eaaf_IMAG01_3.jpg',
]
for image_url in image_urls:
    response = requests.get(image_url)
    image_data = response.content
    filename = os.path.basename(image_url)
    with open(filename, 'wb') as f:
        print('writing to {} ({} bytes)'.format(filename, len(image_data)))
        f.write(image_data)
```

사람처럼 보이기 위한 체크리스트

- * 기계가 웹 서핑을 하는 것 같은 패턴을 보이면 IP가 차단될 수 있습니다.
- * 자바스크립트가 실행되기 전의 페이지는 아무것도 없이 비어 있을 수 있습니다.
- * 폼을 전송하거나 POST 요청을 보낼 때는 서버에서 기대하는 모든 데이터를 보내야 합니다.
 - 크롬 개발자 툴에서 network 탭을 통해 요청되는 정보를 확인
 - 폼의 hidden 필드를 확인
- * 쿠키가 함께 전송되는 지 확인
- * 403 Forbidden 에러를 받는다면 IP가 차단되었을 가능성도 있습니다.
 - 새로운 IP로 요청을 시도하거나, 가까운 카페에 가서 스크래핑을 수행하세요
- * 사이트를 너무 빨리 이동하지 마세요
 - 페이지 이동 시 지연시간을 추가
- * 헤더를 바꾸세요

Selenium

- * 브라우저를 조종하여 데이터를 얻는 방법

- Selenium
- 브라우저를 직접 띄우기 때문에 css나 image 등 모든 데이터를 다운 받음
- 속도가 느리다.
- 동적 페이지도 크롤링이 가능하다. (javascript 실행 가능)

- * HTTP request를 날려서 데이터를 얻는 방법

- requests, scrapy
- 속도가 빠르다.
- Javascript 실행이 불가능함 -> Web page에 대한 사전 분석이 필요

Selenium

- * 웹 브라우저 자동화 tool
- * Java, C#, Perl , PHP, Python , Ruby 등 다양한 언어 지원
- * 직접 브라우저를 실행하여 python code로 mouse click, keyboard input 등의 event를 발생시킴
- * 실제 브라우저로 실행한 것과 동일한 값을 얻을 수 있음
- * 속도가 많이 느리다.



Selenium

* Selenium 특징

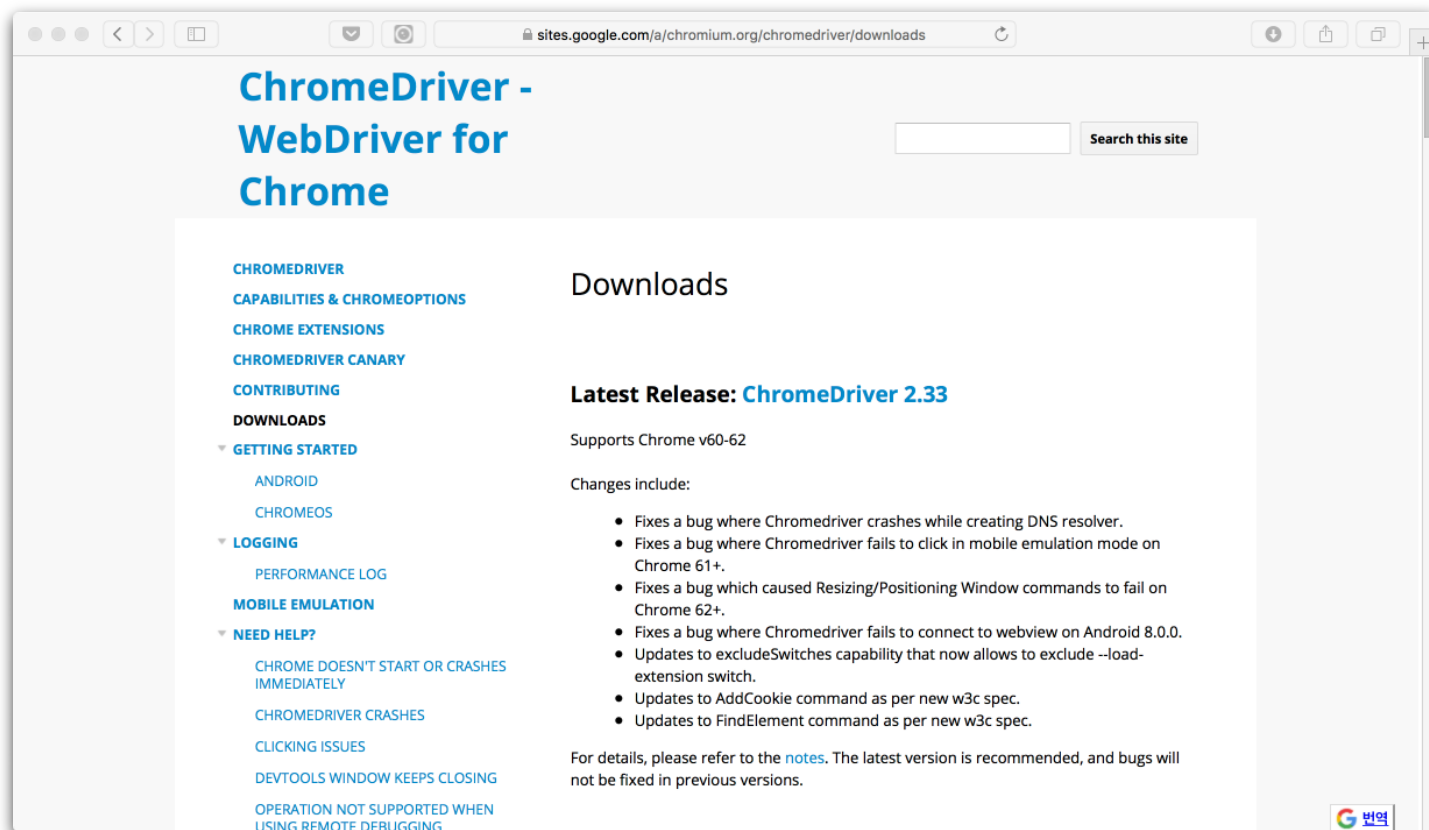
- 주로 웹앱을 테스트하는데 이용하는 프레임워크
- webdriver 라는 API를 통해 운영체제에 설치된 Chrome 등의 브라우저를 제어
- 브라우저를 직접 동작 시키기 때문에 JavaScript 등 비동적으로 혹은 뒤늦게 로드되는 컨텐츠들을 가져올 수 있음
- 즉, 눈에 보이는 모든 컨텐츠를 다 가져올 수 있음
- 비교) requests.text는 브라우저 소스 보기와 같이 이후에 변화된 HTML은 제어 할 수 없다.

Selenium - webdriver

* Chrome WebDriver

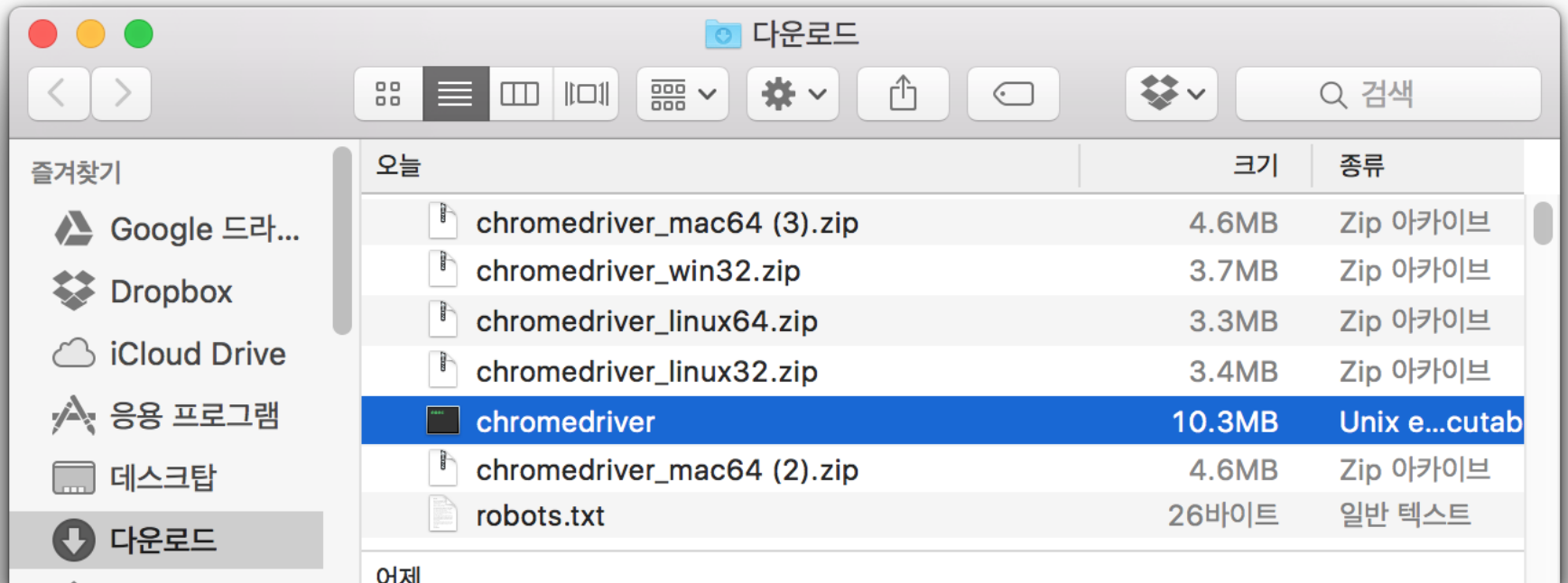
- 크롬 웹드라이버 설치 시 로컬에 크롬 브라우저 반드시 설치 되어 있어야 함
- 크롬 드라이버 다운로드

<https://sites.google.com/a/chromium.org/chromedriver/downloads>



Selenium - webdriver

- * zip 파일을 받고 압축을 풀면 chromedriver .exe라는 파일이 저장됨
- * Selenium 객체를 지정할 때 크롬 드라이버의 위치가 필요



크롬이 60버전 이상되어야 함 (chrome://chrome)

Selenium - 사이트 브라우징

- * 먼저 webdriver 를 import

```
from selenium import webdriver
```

- * webdriver 객체 만들기

```
from selenium import webdriver

# chromedriver 의 경로를 지정하여 웹드라이버 객체 생성
driver = webdriver.Chrome('./chromedriver')

# 암묵적으로 웹 자원 로드를 위해 최대 3초까지 기다린다.
driver.implicitly_wait(3)

# url 에 접근
driver.get('http://www.naver.com')
```

Selenium - 주요 메소드

- * URL 접근

- `get('http://url.com')`

- * 페이지의 단일 element 에 접근하는 API

- `find_element_by_name('HTML_name')`
- `find_element_by_id('HTML_id')`
- `find_element_by_xpath('/html/body/some/xpath')`

- * 페이지의 여러 elements 에 접근하는 API

- `find_elements_by_css_selector('#css > div.selector')`
- `find_elements_by_class_name('some_class_name')`
- `find_elements_by_tag_name('h1')`

Selenium : find_element_by_css_selector

- * driver 를 통해 여러 방법으로 DOM을 찾을 수 있지만, CSS Selector가 더 편리

```
input_id = driver.find_element_by_css_selector('#id')
input_pw = driver.find_element_by_css_selector('#pw')
login_button = driver.find_element_by_css_selector(
    '#frmNIDLogin > fieldset > span > input[type="submit"]'
)
```

- * 폼 관련 Selector : input[type="submit"]
 - input 태그 중 type 속성이 “submit” 인 элемент

Selenium - 네이버 로그인 하기

- * 네이버 로그인은 프론트 단에서 JS를 이용하여 처리
 - requests 는 동적으로 생성된 콘텐츠를 처리 할 수 없음

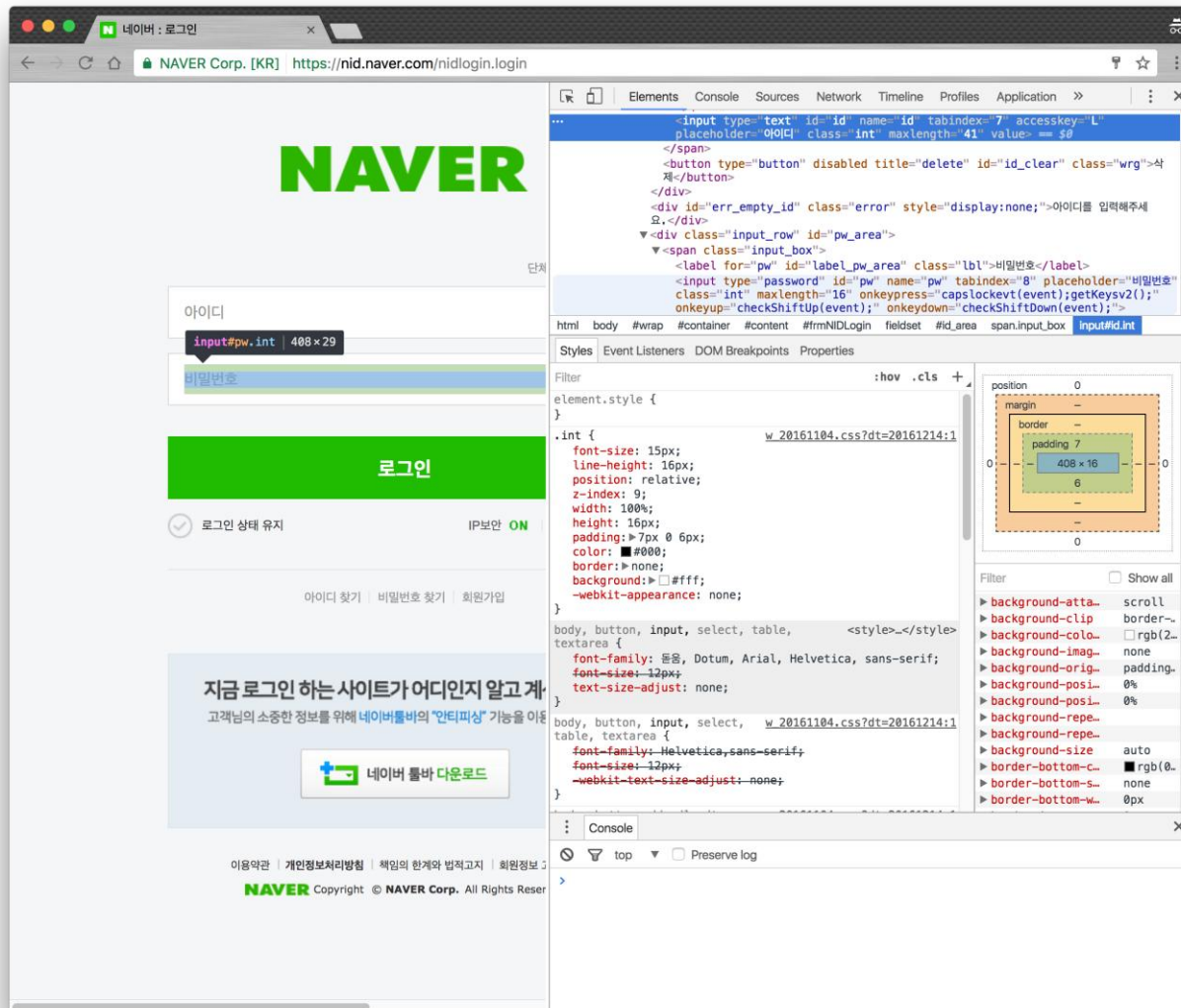
Selenium 으로 처리!!

```
from selenium import webdriver

driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
# url 에 접근
driver.get('https://nid.www.naver.com/nidlogin.login')
```

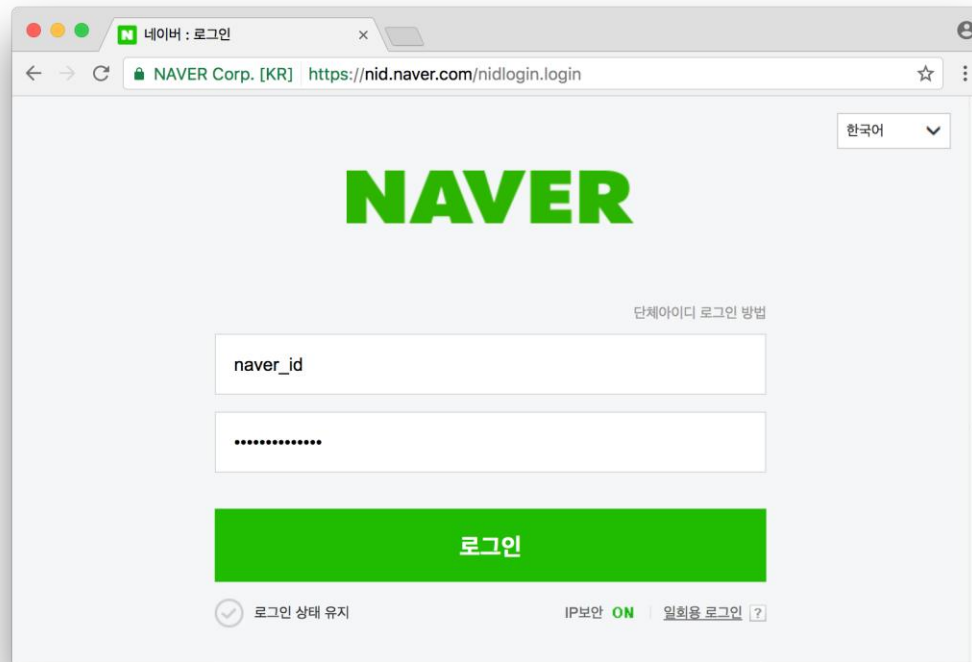
Selenium - 네이버 로그인

- 아이디를 입력 받는 부분 : name : id
- 패스워드 입력 받는 부분 : name : pw



Selenium - 네이버 로그인

```
driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
driver.get('https://nid.www.naver.com/nidlogin.login')
# 아이디/비밀번호를 입력한다.
driver.find_element_by_name('id').send_keys('naver_id')
driver.find_element_by_name('pw').send_keys('mypassword1234')
```



Selenium - 네이버 로그인

```
driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
driver.get('https://nid.www.naver.com/nidlogin.login')
# 아이디/비밀번호를 입력한다.
driver.find_element_by_name('id').send_keys('naver_id')
driver.find_element_by_name('pw').send_keys('mypassword1234')
# 로그인 버튼 누르기
driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/input').click()
```

- 아이디/패스워드 값이 입력된 것을 확인 후
- 로그인 버튼을 눌러 실제로 로그인 되는 것을 확인해 보자.
- 로그인 성공을 확인할 수 있다.

Selenium - 네이버 로그인

```
driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
driver.get('https://nid.www.naver.com/nidlogin.login')
driver.find_element_by_name('id').send_keys('naver_id')
driver.find_element_by_name('pw').send_keys('mypassword1234')
driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/input').click()
```

네이버 페이지 들어가기

```
driver.get('https://order.pay.naver.com/home')
```

해당 페이지의 소스를 가져온다.

```
html = driver.page_source
```

BeautifulSoup 객체를 생성 필요한 정보를 파싱할 수 있다.

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

```
notices = soup.select('div.p_inr > div.p_info > a > span')
```

**로그인 후 로그인에 필요한 페이지로 이동해서 필요한 내용을
스크래핑!**

Selenium - BeautifulSoup 과 같이 사용하기

- * driver.page_source 를 사용하여 현재 렌더링 된 페이지 소스를 모두 가져옴

```
from selenium import webdriver

driver = webdriver.Chrome('./chromedriver')
driver.implicitly_wait(3)
# url 에 접근
driver.get('http://www.naver.com')
# 현재 페이지의 소스를 가져온다.
html = driver.page_source
# BeautifulSoup 객체를 생성
soup = BeautifulSoup(html, 'lxml')

...

# 브라우저 닫기
driver.quit()
```

크롬 - Headless 모드 사용하기

```
from selenium import webdriver

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('window-size=1920x1080')

driver = webdriver.Chrome('./chromedriver', chrome_options=options)
driver.implicitly_wait(3)
# url 에 접근
driver.get('http://www.naver.com')
# 현재 화면 스크린샷으로 저장
driver.get_screenshot_as_file('main-page.png')

# ...이하 동일
```

웹 API

- * 웹 API – 오픈 API 또는 API 라고 함
 - 어떤 사이트의 기능을 외부로 공개하는 것
 - 일반적으로 HTTP 통신을 사용
 - 요청의 결과로 주로 XML 이나 JSON 형태로 데이터를 응답함
 - 최근에는 JSON 방식의 응답을 하는 API가 빠르게 늘어나고 있음
 - 유용한 형식으로 정리된 데이터를 제공 받을 수 있음
- * API 동작 방식
 - 브라우저나 HTTP client 툴을 사용해 요청을 보냄
<http://api.github.com/users/vega2k>
 - 응답은 JSON 형태로 반환

JSON 포맷

* 데이터 교환의 표준 포맷 - JSON

JSON은 키/값 쌍으로 된 컬렉션 형태
파이썬 딕셔너리 리스트와 유사함

List

[..]

Dictionary

{'key': 'value', 'key': 'value'}

```
[
  {
    "price": "3.00",
    "name": "Omelet",
    "desc": "Yummy"
  },
  {
    "price": "5.75",
    "name": "Burrito",
    "desc": "Breakfast Burrito"
  },
  {
    "price": "4.50",
    "name": "Waffles",
    "desc": "Belgian waffles with syrup"
  }
]
```

```
{
  "login": "soongon",
  "id": 1142362,
  "avatar_url": "https://avatars3.githubusercontent.com/u/1142362?v=3",
  "gravatar_id": "",
  "url": "https://api.github.com/users/soongon",
  "html_url": "https://github.com/soongon",
  "followers_url": "https://api.github.com/users/soongon/followers",
  "following_url": "https://api.github.com/users/soongon/following{/other_user}",
  "gists_url": "https://api.github.com/users/soongon/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/soongon/starred{/owner}/{repo}",
  "subscriptions_url": "https://api.github.com/users/soongon/subscriptions",
  "organizations_url": "https://api.github.com/users/soongon/orgs",
  "repos_url": "https://api.github.com/users/soongon/repos",
  "events_url": "https://api.github.com/users/soongon/events{/privacy}",
  "received_events_url": "https://api.github.com/users/soongon/received_events",
  "type": "User",
  "site_admin": false,
  "name": "soongon",
  "company": null,
  "blog": null,
  "location": "seoul, korea",
  "email": "soongon@gmail.com",
  "hireable": null,
  "bio": null,
  "public_repos": 30,
  "public_gists": 3,
  "followers": 22,
  "following": 0,
  "created_at": "2011-10-21T04:05:16Z",
  "updated_at": "2017-02-20T06:56:50Z"
}
```

오픈 API 활용 : 국내 오픈 API 사용 사이트

- * API 스토어 : www.apistore.co.kr

- * 네이버 개발자 센터와 다음 개발자 센터

 - developers.naver.com

 - developers.daum.net

- * 쇼핑 정보

 - 다나와 : <http://api.danawa.com/main/index.html>

 - 옥션 : <http://developer.auction.co.kr/>

- * 주소 전환

 - 행정자치부 :

 - <https://www.iuso.go.kr/addrlink/devAddrLinkRequestWrite.do?returnFn=write&cntcMenu=URL>

 - 우체국 : <https://biz.epost.go.kr/ui/index.jsp>



연습문제 :

- 네이버 번역 API 사용 실습

1. <http://developers.naver.com> 사이트로 이동 (네이버 로그인 필요)
2. 네이버 API 사용을 위해 애플리케이션 등록 수행
3. NMT(Neural Machine Translation) 서비스 선택

```
import urllib.request
client_id = "cdJJWjwbSql1v23_5m9"
client_secret = "Bs0kvhbAx1"

encText = urllib.parse.quote("Yesterday all my troubles seemed so far away")
data = "source=en&target=ko&text=" + encText
url = "https://openapi.naver.com/v1/papago/n2mt"
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request, data=data.encode("utf-8"))
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
    print(response_body)
else:
    print("Error Code:" + rescode)
```

데이터 분석과 시각화

:Predict survival on the Titanic

Predict survival on the Titanic

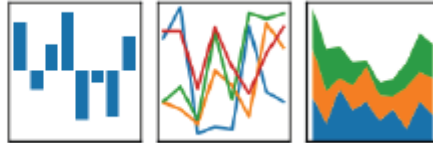
- * 0.Pandas
- * 1.Load Dataset
- * 2.Preprocessing
- * 3.Train
- * 4.Visualize
- * 5.Predict
- * 6.Submit

Pandas

- pandas(<http://pandas.pydata.org/>)는 데이터 처리와 분석을 위한 파이썬 라이브러리입니다. R의 data.frame을 본떠서 설계한 DataFrame이라는 데이터 구조를 기반으로 만들어졌습니다.
- 간단하게 말하면 pandas의 DataFrame은 엑셀의 스프레드시트와 비슷한 테이블 형태라고 할 수 있습니다. pandas는 이 테이블을 수정하고 조작하는 다양한 기능을 제공합니다. 특히, SQL처럼 테이블에 쿼리나 조인을 수행할 수 있습니다.
- 전체 배열의 원소가 동일한 타입이어야 하는 NumPy와는 달리 pandas는 각 열의 타입이 달라도 됩니다(예를 들면 정수, 날짜, 부동산숫점, 문자열).
- SQL, 엑셀 파일, CSV 파일 같은 다양한 파일과 데이터베이스에서 데이터를 읽어 들일 수 있는 것이 pandas가 제공하는 또 하나의 유용한 기능입니다.

pandas

$$y_{it} = \beta^i x_{it} + \mu_i + \epsilon_{it}$$



- [10 Minutes to pandas](#)라는 판다스의 공식 튜토리얼을 읽어보시는 것을 추천 드립니다.

Pandas

* 파일 읽어오기

```
import pandas as pd

# train.csv 파일을 읽어옵니다. 여기서 PassengerId라는 컬럼을 인덱스(index)로 지정
# 변수에 할당한 결과값은 데이터프레임(DataFrame)
train = pd.read_csv("data/train.csv", index_col="PassengerId")

# train 변수에 할당된 데이터의 행렬 사이즈를 출력합니다. 출력은 (row, column) 으로 표시됩니다.
print(train.shape)

# 이후 .head()로 train 데이터프레임의 전체가 아닌 상위 5개를 띄웁니다.
train.head()

# 인덱스(index)를 가져옵니다. 여기서 index는 PassengerId와 동일합니다.
train.index

# 컬럼(columns)을 가져옵니다.
train.columns
```

Pandas

* 행렬 : 열(column) 가져오기

```
train["Survived"].head()
train[["Sex", "Pclass", "Survived"]].head()
columns = ["Sex", "Pclass", "Survived"]
train[columns].head()
```

* 행렬 : 행(row) 가져오기

```
train.loc[1]
train.loc[1:7]
train.loc[[1, 3, 7, 13]]

passenger_ids = [1, 3, 7, 13]
train.loc[passenger_ids]
```

* 행렬 동시에 가져오기

```
train.loc[1, "Sex"]
train.loc[1, ["Pclass", "Sex", "Survived"]]
train.loc[[1, 3, 7, 13], "Sex"]
train.loc[1:7, "Sex"]
train.loc[[1, 3, 7, 13], ["Sex", "Pclass", "Survived"]]
```

Pandas

* Boolean Mask

```
train[train["Sex"] == "male"].head()
train[train["Fare"] > 20].head()
train[train["Embarked"].isin(["Q", "S"])].head()
train[train["Age"].isnull()].head()
train[train["Age"].notnull()].head()
train[~train["Age"].isnull()].head()
train[(train["Age"].isnull() | (train["Fare"].isnull()))].head()
train[(train["Age"].isnull() & (train["Fare"].isnull()))]
```

* 기본 연산

```
print(train["Fare"].mean())
print(train["Age"].max())
print(train["Age"].min())
```

* 컬럼 추가 & 수정

```
train["DataCategory"] = "Titanic"
train["Id"] = range(0, 891)
train["FamilySize"] = train["SibSp"] + train["Parch"] + 1
train[["SibSp", "Parch", "FamilySize"]].head()
train["Nationality_FR"] = train["Embarked"] == "C"
train["Nationality_UK"] = train["Embarked"].isin(["S", "Q"])
```

Pandas

* 컬럼 추가 & 수정

```
train.loc[train["Embarked"] == "C", "Nationality"] = "France"
train.loc[train["Embarked"].isin(["S", "Q"]), "Nationality"] = "England"

train["Fare_Cheap"] = train["Fare"] < 30
train["Fare_Medium"] = (train["Fare"] >= 30) & (train["Fare"] < 100)
train["Fare_Expensive"] = train["Fare"] >= 100

train.loc[train["Fare"] < 30, "FareType"] = "Cheap"
train.loc[(train["Fare"] >= 30) & (train["Fare"] < 100), "FareType"] = "Med"
train.loc[train["Fare"] >= 100, "FareType"] = "Expensive"

mean_age = train["Age"].mean()
train.loc[train["Age"].isnull(), "Age"] = mean_age
```


머신러닝이란?

- 머신러닝은 한 마디로

“ 데이터를 이용해서 명시적으로 정의되지 않은 패턴을 컴퓨터로 학습하여 결과를 만들어 내는 학문분야 ”

- 머신러닝의 분류

1. 지도학습(supervised learning), 지도러닝, 교사 학습

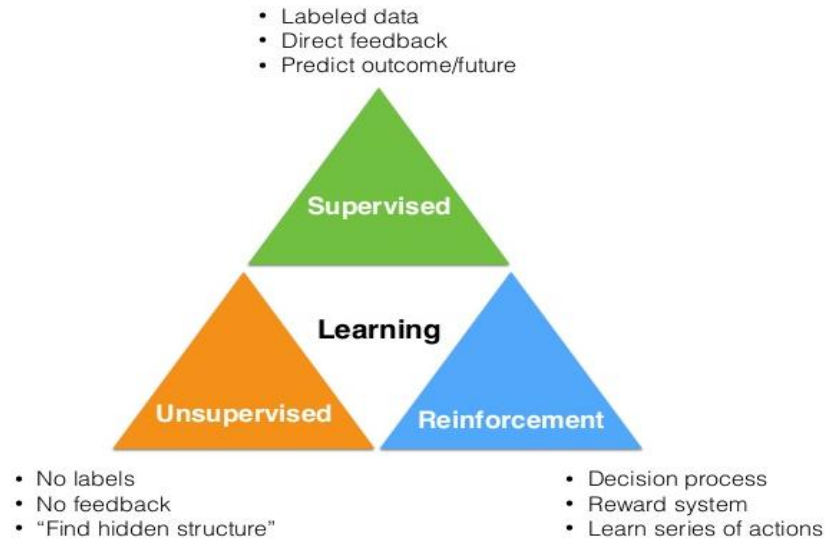
지도학습은 주어진 데이터와 레이블(정답)을 이용해서 **미지의 상태나 값을 예측하는** 학습

방법이다. 대부분의 머신러닝은 지도학습에 해당함.

2. 비지도 학습(unsupervised learning), 자율러닝, 비교사 학습

비지도학습은 데이터와 주어진 레이블 간의 관계를 구하는 것이 아니라 **데이터 자체에서 유용한 패턴을 찾아내는** 학습방법이다.

머신러닝 분류와 알고리즘 종류



지도학습(supervised)	비지도학습(unsupervised)
회귀(Regression) <ul style="list-style-type: none"> - Linear regression - Decision Tree - Random Forests - Neural Networks(딥러닝) 	<ul style="list-style-type: none"> - Clustering - K means - PCA(Principal component analysis) - Density estimation - Expectation maximization - Pazen window - DBSCAN
분류(classification) <ul style="list-style-type: none"> - Naïve-Bayes - K-NearestNeighbors(KNN) - Support Vector Machine(SVM) 	

Predict survival on the Titanic

* 1. Load Dataset

```
import pandas as pd
train = pd.read_csv("data/train.csv", index_col="PassengerId")
print(train.shape)
print(train.info())
train.head()
test = pd.read_csv("data/test.csv", index_col="PassengerId")
print(test.shape)
test.head()
```

* 2. Preprocessing

- preprocessing은 의사결정트리를 만들기 전에 데이터를 처리하는 과정입니다. preprocessing 하는 이유는 의사결정트리가 수치적으로 계산되기 때문에 데이터의 값을 수치적으로 변경해줘야 하기 때문입니다.

```
#Encode Sex
train.loc[train["Sex"] == "male", "Sex"] = 0
train.loc[train["Sex"] == "female", "Sex"] = 1
test.loc[test["Sex"] == "male", "Sex"] = 0
test.loc[test["Sex"] == "female", "Sex"] = 1
```

```
#Fill in missing fare
test.loc[test["Fare"].isnull(), "Fare"] = 0
test[test["Fare"].isnull()]
```

Predict survival on the Titanic

* 2. Preprocessing (one-hot encoding)

```
# Encode Embarked
train["Embarked_C"] = train["Embarked"] == "C"
train["Embarked_S"] = train["Embarked"] == "S"
train["Embarked_Q"] = train["Embarked"] == "Q"

test["Embarked_C"] = test["Embarked"] == "C"
test["Embarked_S"] = test["Embarked"] == "S"
test["Embarked_Q"] = test["Embarked"] == "Q"
```

* 3. Train

- * train은 특정 알고리즘을 선택하여 , Machine에게 featurizing한 데이터를 학습시키는 과정을 말합니다.

```
feature_names = ["Pclass", "Sex", "Fare",
                  "Embarked_C", "Embarked_S", "Embarked_Q"]
X_train = train[feature_names]
print(X_train.shape)
X_train.head()

X_test = test[feature_names]
print(X_test.shape)
X_test.head()
```

Predict survival on the Titanic

* 3. Train

```
label_name = "Survived"
# train dataframe의 Survived 컬럼을 가져옵니다.
y_train = train[label_name]

print(y_train.shape)
y_train.head()

# 의사결정트리를 만들기 위해서 scikit-learn 패키지의 tree 모듈중 DecisionTreeClassifier를 가지고 온다.
from sklearn.tree import DecisionTreeClassifier

# 의사결정트리의 최대 깊이를 5로 설정하고 model에 할당합니다.
model = DecisionTreeClassifier(max_depth=5)

# fit은 학습하는 함수
# X_train(feature)과 y_train(label)으로 학습을 합니다.
model.fit(X_train, y_train)
```

Predict survival on the Titanic

* 4. Visualize

시각화를 위해서는 Graphviz라는 툴의 설치가 필요합니다. 설치 방법은 다음과 같습니다.

1. 아나콘다에서 설치하기(아나콘다로 파이썬을 설치한 분들은 이 방법을 이용해주세요)

1) 아나콘다 네비게이터(Anaconda Navigator)를 실행합니다.

2) 좌측 환경(Enviromments) 탭을 클릭합니다.

3) 우측 콤보박스에 Installed로 되어있는 것을 Not Installed로 변경합니다.

4) 이후 검색창에 graphviz라고 치면 graphviz와 python-graphviz라는 패키지가 보입니다.

5) 위의 두 패키지를 설치해주시면 됩니다.

2. 직접 다운받아 설치하기(아나콘다로 파이썬을 설치하지 않은 분들인 이 방법을 이용해주세요)

1) <http://www.graphviz.org/download/> 에서 운영체제(Windows, MacOS)에 맞는 설치 파일을 다운받아주세요. (ex: graphviz-2.38.msi)

2) 다운받은 파일로 설치해주세요.

3) 주피터 노트북에서 !pip install graphviz 를 실행해주세요.

```
from sklearn.tree import export_graphviz
import graphviz
export_graphviz(model,
                 feature_names=feature_names,
                 class_names=["Perish", "Survived"],
                 out_file="decision-tree.dot")
with open("decision-tree.dot") as f:
    dot_graph = f.read()
graphviz.Source(dot_graph)
```

Predict survival on the Titanic

* 5. Predict

```
# predict는 예측하는 함수 입니다.  
# 만들어진 의사결정트리를 활용하여 테스트 데이터를 예측해서 predictions 변수에 할당합니다.  
predictions = model.predict(X_test)  
  
print(predictions.shape)  
predictions[0:10]
```

* 6. Submit (Kaggle 에 제출하여 Ranking 확인)

```
# 제출하기 위해서 제공된 gender_submission.csv 데이터를 불러옵니다.  
submit = pd.read_csv("data/gender_submission.csv", index_col="PassengerId")  
  
print(submit.shape)  
submit.head()  
  
# submit dataframe의 Survived 컬럼을 predictions(예측한 값)으로 수정합니다.  
submit["Survived"] = predictions  
  
print(submit.shape)  
submit.head()  
  
# submit dataframe을 csv파일로 저장합니다.  
submit.to_csv("data/decision-tree.csv")
```

Reference site

- * [Mukesh ChapagainTitanic Solution: A Beginner's Guide](https://www.kaggle.com/chapagain/titanic-solution-a-beginner-s-guide?scriptVersionId=1473689)

<https://www.kaggle.com/chapagain/titanic-solution-a-beginner-s-guide?scriptVersionId=1473689>

- * [How to score 0.8134 in Titanic Kaggle Challenge](https://ahmedbesbes.com/how-to-score-08134-in-titanic-kaggle-challenge.html)

<https://ahmedbesbes.com/how-to-score-08134-in-titanic-kaggle-challenge.html>

- * [Titanic: factors to survive](https://olegleyz.github.io/titanic_factors.html)

https://olegleyz.github.io/titanic_factors.html

- * [Titanic Survivors Dataset and Data Wrangling](http://www.codeastar.com/data-wrangling/)

<http://www.codeastar.com/data-wrangling/>

수고하셨습니다.