

ML&DL

WITH PYTHON

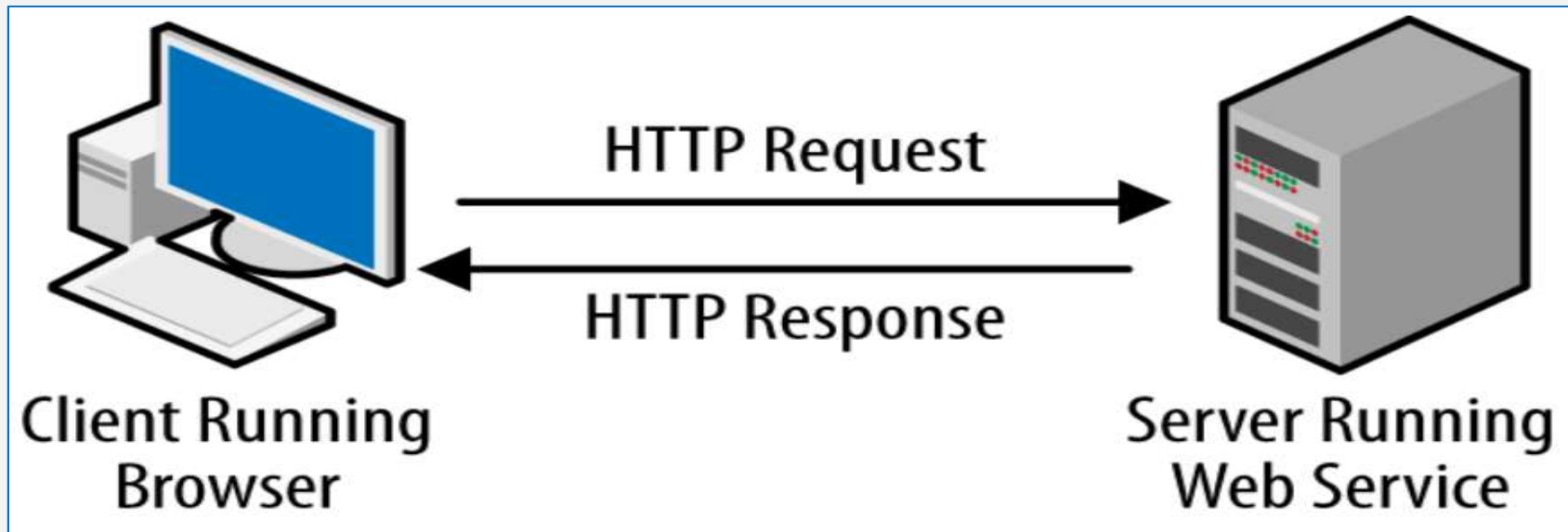
PART II

DATA 수집 및 추출

로그인 스크레이핑

LOGIN SCRAPING

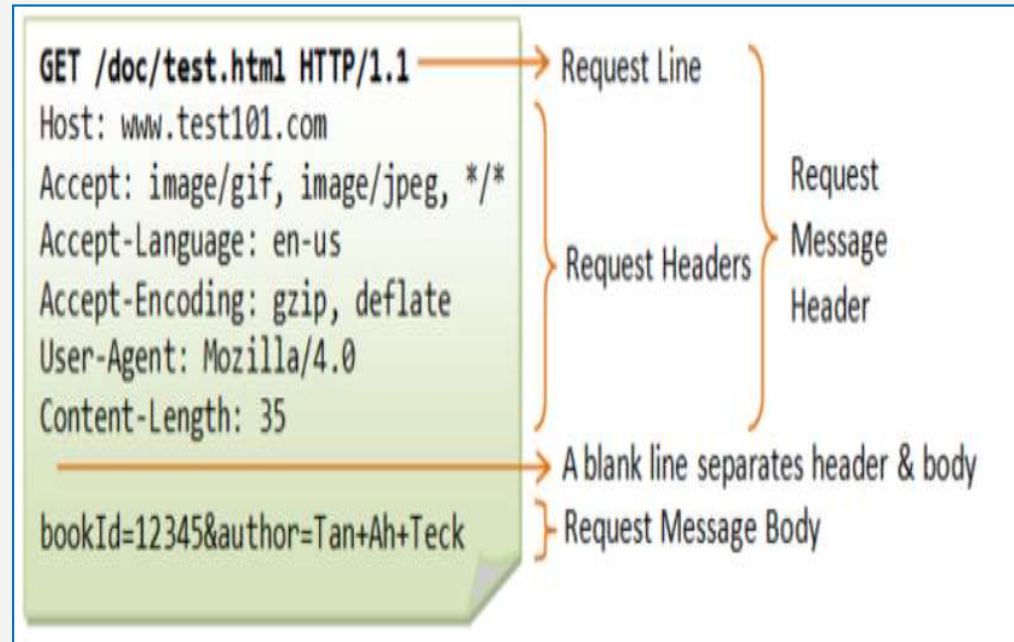
◆WEB – Server & Client



LOGIN SCRAPING

◆WEB – Server & Client

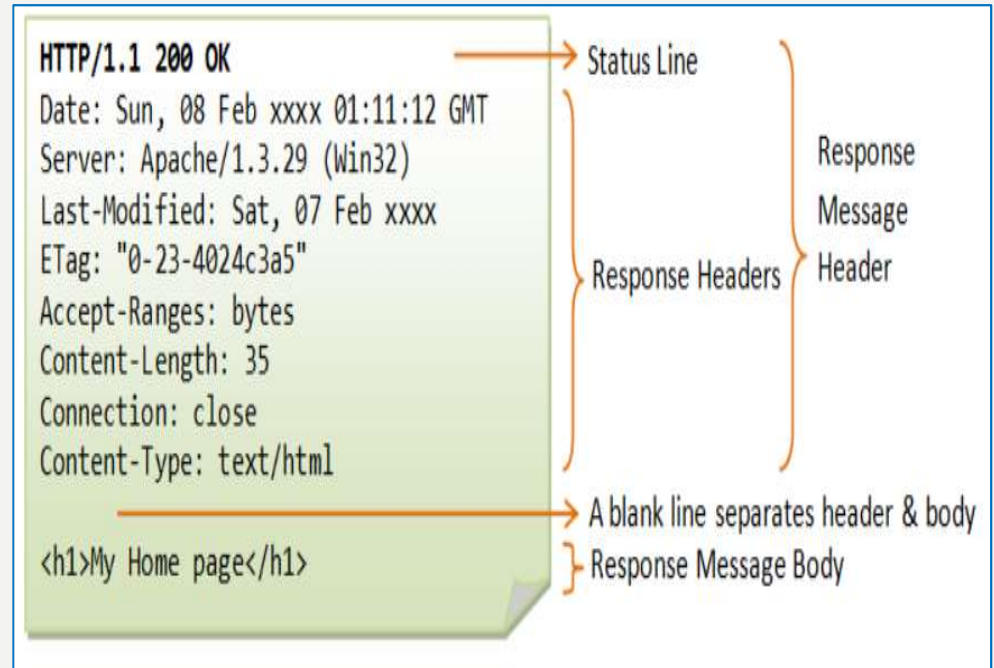
HTTP Request Format	
웹 서버에 데이터 요청 또는 전송할 때 보내는 패킷 (정보)	
GET/HTTP/1.1\r\n	요청 방식
Accept: */*\r\nUA-CPU: AMD64\r\nAccept-Encoding: gzip, deflate\r\nUser-Agent: Mozilla/5.0 (; rv:11.0\r\nHost: 192.168.35.242\r\nDNT: 1\r\nConnection: Keep-Alive\r\n	HEADER
\r\n	Empty Line



LOGIN SCRAPING

◆WEB – Server & Client

HTTP Response Format	
클라이언트 요청에 보내는 응답 데이터 형태	
HTTP/1.1 200 OK	STATUS LINE
Content-type: text/html -----	HEADER
Empty Line	
<!DOCTYPE html> <html> <head></head> <body></body> </html>	Client에게 보여 줄 Web Page



LOGIN SCRAPING

◆WEB – Server & Client

- HTTP METHOD

- 특정 URL에 요청 시 해당 URL에 원하는 요청 동작
 - GET → URL 정보 검색 요청
 - POST → 요청 자원 생성
 - PUT → 전체 수정 요청
 - PATCH → 일부 수정 요청
 - DELETE → 제거 요청

LOGIN SCRAPING

◆ WEB – HTTP 프로토콜

connectionless (비연결성)

- 클라이언트 Request에 Response 전송 후 연결 해제
- 인터넷 상 불특정 다수의 통신 환경 기반 설계
- 동일한 클라이언트 요청에 새로운 연결 시도/해제 과정 => 오버헤드 발생
- KeepAlive 속성으로 주기적인 클라이언트 상태 체크로 문제점 보완

stateless (무상태)

- 연결해제에 따른 클라이언트 상태 정보 유지 하지 않음
- 클라이언트에 대한 매번 인증 필요
- 쿠키** → 브라우저 단에서 클라이언트 요청시 식별 데이터 파일 부여 및 저장
- 세션** → 쿠키를 기반으로 사용자 정보를 서버측에 저장

LOGIN SCRAPING

◆ Requests 모듈

- <https://requests.readthedocs.io/en/latest/>



Requests: HTTP for Humans™

Release v2.22.0. ([Installation](#))

downloads 1G license Apache 2.0 wheel yes python 2.7 | 3.5 | 3.6 | 3.7

Requests is an elegant and simple HTTP library for Python, built for human beings.

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
```

Star 41,521

LOGIN SCRAPING

◆ Requests 모듈

- GET 요청

```
URL = "http://www.naver.com?num=1&data=good"
```

```
response = requests.get( URL )
```

```
URL = http://www.naver.com
```

```
dict_param = { "num"=1, "data"="good" }
```

```
response = requests.get( URL , params=dict_param )
```

LOGIN SCRAPING

◆ Requests 모듈

- POST 요청

```
URL = http://www.naver.com
```

```
dict_param = { "num"=1, "data"="good" }
```

```
response = requests.post( URL , params=dict_param )
```

LOGIN SCRAPING

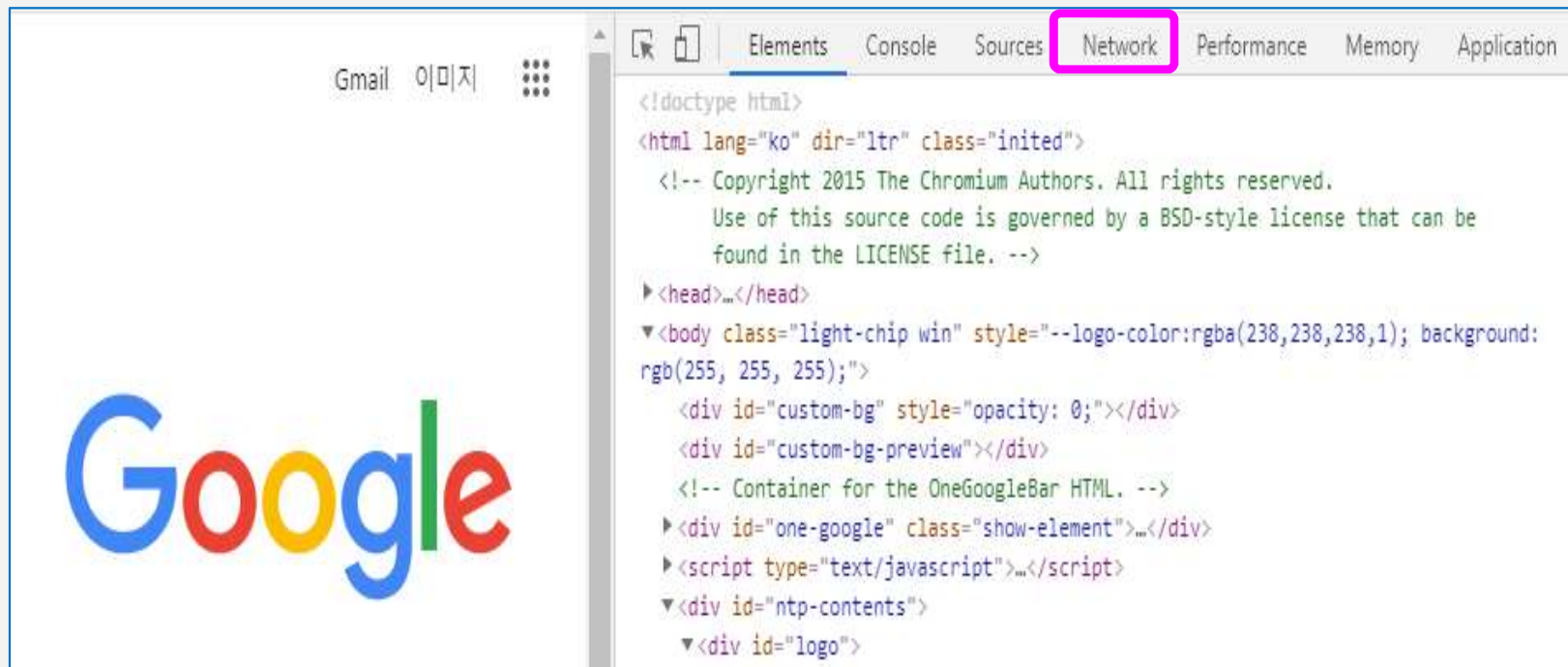
◆ WEB 네트워크 상태

- (1) WEB 페이지 열기
- (2) 원하는 지점 + 오른쪽 클릭
- (3) 검사 메뉴 클릭 ➔ Network 탭 클릭

LOGIN SCRAPING

◆ WEB 네트워크 상태

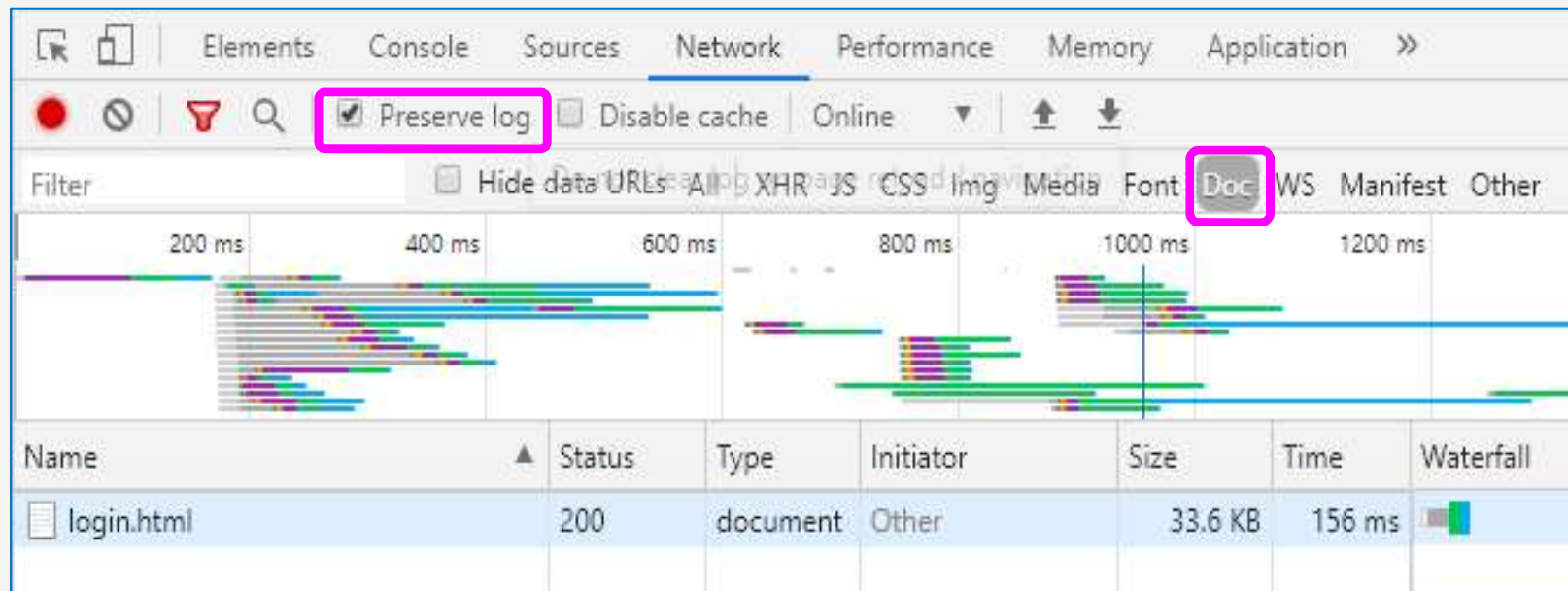
- 페이지 이동 정보 내용 확인



LOGIN SCRAPING

◆ WEB 네트워크 상태

- 페이지 이동 정보 내용 확인



LOGIN SCRAPING

◆로그인 웹 페이지 데이터 추출

```
# 모듈 로딩 -----  
import requests  
from bs4 import BeautifulSoup  
  
# 변수 선언 -----  
# 아이디와 비밀번호 지정  
USER = "XXXX"  
PASS = "XXXXXX"  
login_info = { "m_id": USER, "m_passwd": PASS }
```

LOGIN SCRAPING

◆로그인 웹 페이지 데이터 추출

```
# 모듈 객체 생성 및 선언 -----  
# 세션 시작  
session = requests.session()  
  
# 로그인 페이지 접근  
url_login = "https://www.hanbit.co.kr/member/login_proc.php"  
res = session.post(url_login, data=login_info)  
res.raise_for_status()  
  
# 마이페이지 접근  
url_mypage = "http://www.hanbit.co.kr/myhanbit/myhanbit.html"  
res = session.get(url_mypage)  
res.raise_for_status()
```


LOGIN SCRAPING

◆로그인 웹 페이지 데이터 추출

```
# Web Page에서 데이터 추출 -----  
soup = BeautifulSoup(res.text, "html.parser")  
#print(soup)  
  
mileage = soup.select_one("dl.mileage_section1 span").get_text()  
ecoin = soup.select_one(".mileage_section2 span").get_text()  
print("마일리지: " + mileage)  
print("이코인: " + ecoin)
```

브라우저 스크레이핑

BROWSER SCRAPING

◆브라우저 원격 제어 모듈

selenium

- 자동화 방지 사이트 제어
- 웹 브라우저 원격 제어 및 테스트 모듈
- <http://www.seleniumhq.org/>
- selenium 모듈 설치
 - PyCharm에서 설치
 - pip install selenium 또는 conda install selenium



BROWSER SCRAPING

◆브라우저 제어 모듈

- WebDriver
 - 디바이스에 설치된 브라우저 제어
 - 브라우저 업데이트마다 새로운 드라이버 설치 필요
 - 브라우저별 드라이버 제공

BROWSER SCRAPING

◆브라우저 제어 모듈

- Firefox WebDriver 설치

- Firefox 브라우저 설치
- URL : <https://github.com/mozilla/geckodriver/releases>

 geckodriver-v0.26.0-linux32.tar.gz	2.22 MB
 geckodriver-v0.26.0-linux64.tar.gz	2.28 MB
 geckodriver-v0.26.0-macos.tar.gz	1.91 MB
 geckodriver-v0.26.0-win32.zip	1.37 MB
 geckodriver-v0.26.0-win64.zip	1.46 MB
 Source code (zip)	
 Source code (tar.gz)	

BROWSER SCRAPING

◆브라우저 제어 모듈

- Chrom WebDriver 설치
 - 현재 사용 크롬 브라우저 버전 체크
 - URL 주소창 : chrome://version 입력



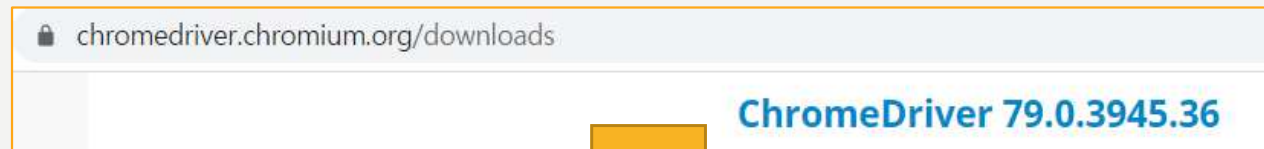
BROWSER SCRAPING

◆브라우저 제어 모듈






- Chrom WebDriver 설치

- 다운로드 사이트 접속

- <https://chromedriver.chromium.org/downloads>



Index of /79.0.3945.16/

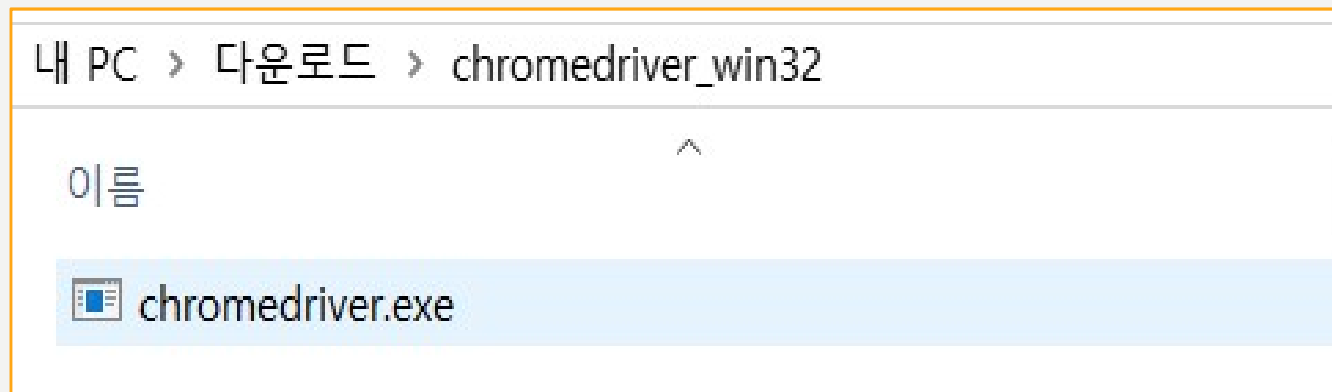
	Name	Last modified	Size	ETag
	Parent Directory	-	-	-
	chromedriver_linux64.zip	2019-10-30 16:10:56	4.65MB	105ccad6c939b96600d3963bd758a6c9
	chromedriver_mac64.zip	2019-10-30 16:10:59	6.59MB	39b7db6c05a92850833f6826ecbd2dcd
	chromedriver_win32.zip	2019-10-30 16:11:00	4.06MB	6c36e6f2c1cb7ec0f8cb4201a7fb03e1
	notes.txt	2019-10-30 16:11:04	0.00MB	eb3d9792539dcd95b12d890e8cfa7ee8

BROWSER SCRAPING

◆브라우저 제어 모듈

- Chrom WebDriver 설치

- zip파일 압축해제



- 경로는 프로그래밍에 사용됨!

BROWSER SCRAPING

◆브라우저 제어 예제

```
# 모듈 로딩 -----  
from selenium import webdriver  
  
# 크롬 브라우저 제어 드라이버 생성  
driver=webdriver.Chrome('C:/WebDriver/chromedriver.exe')  
  
# 크롬 브라우저 드라이버를 활용한 웹 페이지 제어  
# URL 열기  
driver.get('https://www.naver.com')  
  
# WEB 페이지의 특정 요소에 접근  
# //*[@id="query"]  
driver.find_element_by_xpath( '//*[@id="query"]' ).send_keys('안녕하세요.')
```

BROWSER SCRAPING

◆ 브라우저 제어 예제

```
# 모듈 로딩 -----  
from selenium import webdriver  
  
# 크롬 브라우저 제어 드라이버 생성  
brower=webdriver.Chrome('C:/WebDriver/chromedriver.exe')  
  
# 크롬 브라우저 드라이버를 활용한 웹 페이지 제어  
# 웹 사이트 가져오기  
brower.get('https://www.naver.com')  
# 웹 사이트의 특정 요소에 접근 후 값 전송  
brower.find_element_by_xpath('//*[@id="query"]').send_keys('안녕하세요.')  
# 3초 대기  
brower.implicitly_wait(3)  
# 웹 사이트의 특정 요소에 접근 후 동작 제어  
brower.find_lament_by_xpath('//*[@id="search_btn"]').click()
```

BROWSER SCRAPING

◆ 브라우저 제어 예제

```
# 모듈 로딩 -----  
from selenium import webdriver  
  
# 변수 선언 -----  
url = "http://www.naver.com/"  
  
# 웹 브라우저 접근 후 제어 -----  
driver=webdriver.Chrome('C:/WebDriver/chromedriver.exe')  
  
driver.implicitly_wait(3)           # 3초 대기  
driver.get(url)                     # URL 읽기  
driver.save_screenshot("Website.png") # 화면을 캡처 & 저장  
driver.quit()                       # 브라우저 종료
```

웹 API 스크레이핑

WEB API SCRAPING

◆ WEB OPEN API

공공기관, IT기업, 검색 사이트, 통신사 등에서 지원하는 서비스를 활용할 수 있는 API 공개

반드시 회원가입 & 사용 키 필요

코드에 키 입력 필수

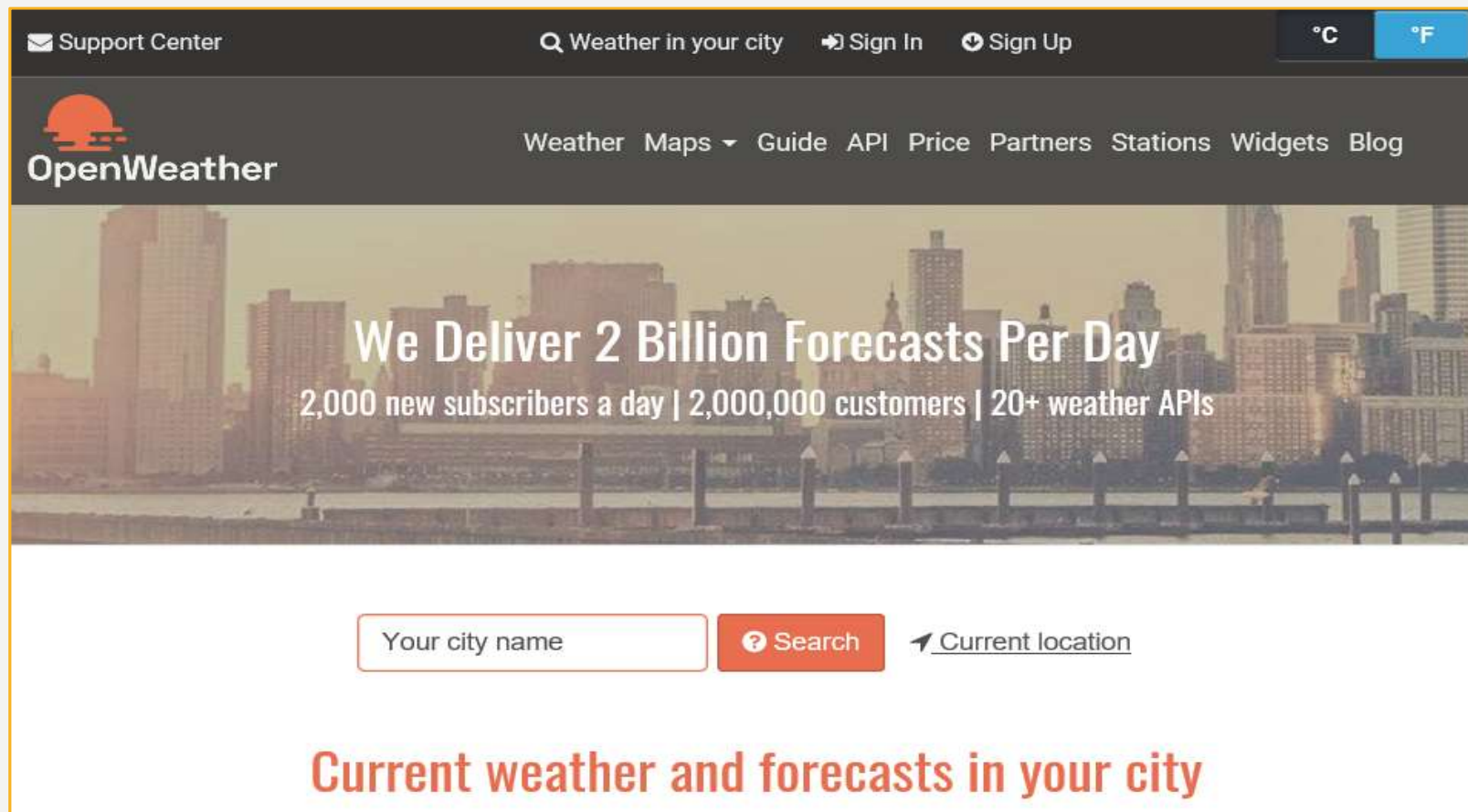
완전 무료 / 1일 제한 사용 횟수 이상 과금

WEB API SCRAPING

◆ WEB OPEN API

- OpenWeather => 전 세계 도시의 기상 정보 제공 사이트

<https://openweathermap.org/>



WEB API SCRAPING

◆ WEB OPEN API

- OpenWeather => 전 세계 도시의 기상 정보 제공 사이트

The screenshot displays the OpenWeather website's API management interface. At the top, there's a navigation bar with links like 'Support Center', 'Weather', 'Maps', 'API', 'Price', 'Partners', 'Stations', 'Widgets', 'News', and 'About'. A search bar and user information ('Hello shk', 'Logout') are also present. A green notice box states 'Signed in successfully.' Below this, a horizontal menu includes 'New Products', 'Services', 'API keys' (highlighted with a red box), 'Billing plans', 'Payments', 'Block logs', 'Store', and 'My profile'. The 'API keys' section is active, showing a message: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from...'. Below this, a table lists API keys with columns 'Key' and 'Name'. The first entry shows a key starting with '2bde6dcb' followed by a redacted portion, and the name 'Default'. There are icons for editing and deleting the key.

Key	Name
2bde6dcb [REDACTED]	Default

WEB API SCRAPING

◆ API

```
# 모듈 로딩 -----
import requests
import json

# 변수 선언 -----
apikey = '2bde6dcb08e5150c1fcdXXXXXXXXXXXXXXXXXXXXXXX'
cities = ["Seoul,KR", "Tokyo,JP", "New York,US"]
api = "http://api.openweathermap.org/data/2.5/weather?q={city}&APPID={key}"
print('api=>', api)

# 켈빈 온도-> 섭씨 온도 변환 익명함수 변수 -----
k2c = lambda k: k - 273.15

# 각 도시의 정보 추출 -----
for name in cities:
    # API URL 구성
    url = api.format(city=name, key=apikey)
    print('url=>', url)
```


WEB API SCRAPING

◆ API

```
# API에 요청을 보내 데이터 추출하기
r = requests.get(url)

# 결과를 JSON 형식으로 변환
data = json.loads(r.text)
print('data=>', data)

# 결과 출력
print("+ 도시 =", data["name"])
print("| 날씨 =", data["weather"][0]["description"])
print("| 최저 기온 =", k2c(data["main"]["temp_min"]))
print("| 최고 기온 =", k2c(data["main"]["temp_max"]))
print("| 습도 =", data["main"]["humidity"])
print("| 기압 =", data["main"]["pressure"])
print("| 풍향 =", data["wind"]["deg"])
print("| 풍속 =", data["wind"]["speed"])
print("")
```

정기적인 스크롤링

PERIODIC SCRAPING

◆ Python => EXE 변환

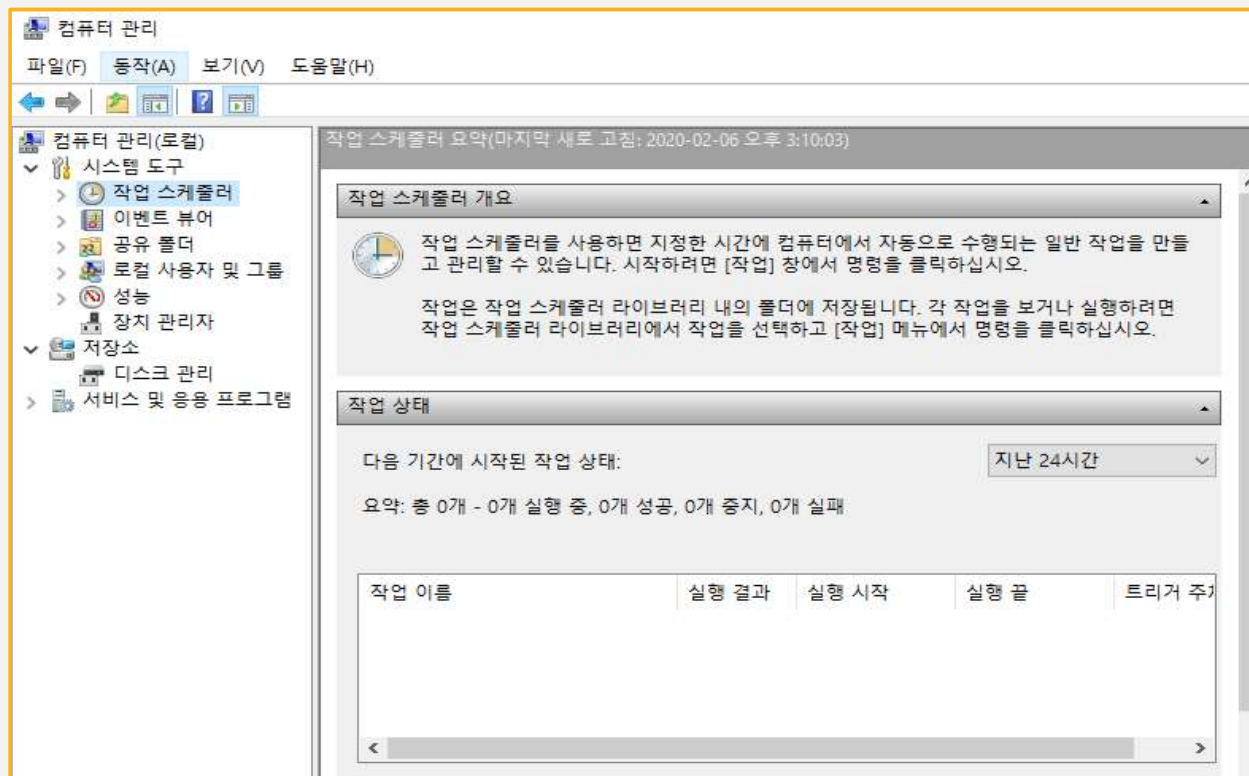
[Window CMD창]

- pyinstaller 설치 → `pip install pyinstall`
- py -> exe 변환 → `pyinstall --noconsole -onefile test.py`

PERIODIC SCRAPING

◆ Window 작업스케줄러

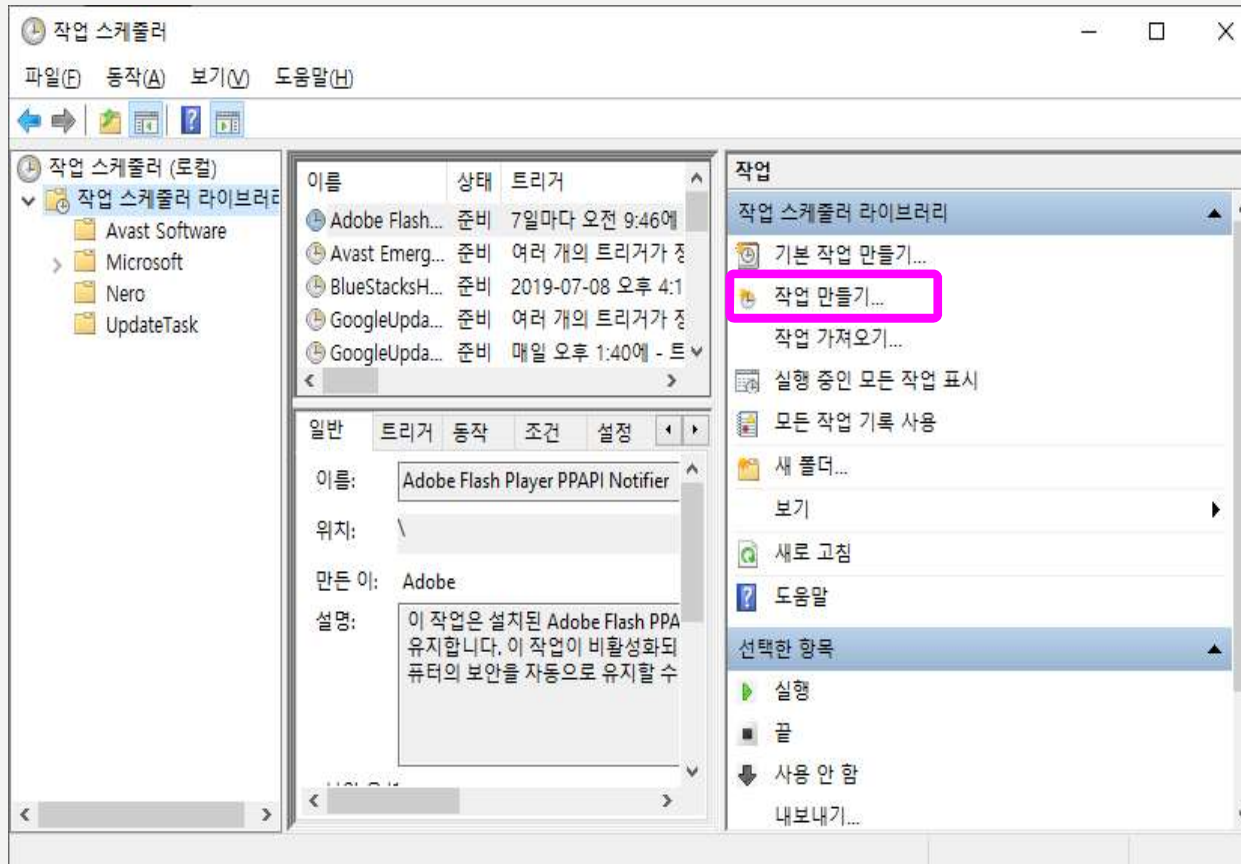
- CMD : taskchd.msc 입력



PERIODIC SCRAPING

◆ Window 작업스케줄러

- 작업 -> 작업 만들기 클릭



PERIODIC SCRAPING

◆ Window 작업스케줄러

- 일반탭 -> 이름

The screenshot shows the '새 작업 만들기' (New Task Wizard) window in Windows Task Scheduler. The '일반' (General) tab is selected. The '이름(M):' (Name) field contains 'REAL_USD'. The '위치:' (Location) field contains '₩'. The '만든 이:' (Author) field contains 'DESKTOP-1C43FOJ₩알엔유'. The '설명(D):' (Description) field is empty. Under the '보안 옵션' (Security options) section, the '작업을 실행할 때 사용할 사용자 계정:' (User account to use when running the task) is set to 'DESKTOP-1C43FOJ₩알엔유'. The '사용자 또는 그룹 변경(U)...' button is visible. The '사용자가 로그인할 때만 실행(R)' (Run only when user is logged on) radio button is selected. The '가장 높은 수준의 권한으로 실행(I)' (Run with highest privileges) checkbox is checked. The '구성 대상(C):' (Targeted configuration) dropdown is set to 'Windows Vista™, Windows Server™ 2008'. The '확인' (OK) and '취소' (Cancel) buttons are at the bottom right.

새 작업 만들기

일반 트리거 동작 조건 설정

이름(M): REAL_USD

위치: ₩

만든 이: DESKTOP-1C43FOJ₩알엔유

설명(D):

보안 옵션

작업을 실행할 때 사용할 사용자 계정:

DESKTOP-1C43FOJ₩알엔유 사용자 또는 그룹 변경(U)...

☒ 사용자가 로그인할 때만 실행(R)

☐ 사용자의 로그인 여부에 관계없이 실행(W)

☐ 암호를 저장하지 않습니다. 이 작업에서는 로컬 컴퓨터 리소스에만 액세스할 수 있습니다(P).

☒ 가장 높은 수준의 권한으로 실행(I)

☐ 숨김(E) 구성 대상(C): Windows Vista™, Windows Server™ 2008

확인 취소

PERIODIC SCRAPING

◆ Window 작업스케줄러

- 트리거탭 -> [새로 만들기] 버튼

새 트리거 만들기

작업 시작(G): 예약 상태

설정

☒ 한 번(N) 시작(S): 2020-02-06 오후 4:10:00 ☐ 표준 시간대 간 동기화(Z)

☐ 매일(D)

☐ 매주(W)

☐ 매월(M)

고급 설정

☐ 작업이 지연되는 최대 시간(임의 지연)(K): 1 시간

☒ 작업 반복 간격(P): 10 분 기간(F): 1 일

☐ 반복 기간이 종료될 때 실행 중인 모든 작업 중지(I)

☐ 다음 기간 이상 실행되는 작업 중지(L): 3 일

☐ 만료(X): 2021-02-06 오후 4:03:03 ☐ 표준 시간대 간 동기화(E)

☒ 사용(B)

확인 취소

새 작업 만들기

일반 트리거 동작 조건 설정

작업을 만들 때 작업을 트리거할 조건을 지정할 수 있습니다.

트리거	자세히	상태
한 번	2020-02-06 오후 4:10에 - 트리거된 후 1 일 기간 동안 10 분...	사용

새로 만들기(N)... 편집(E)... 삭제(D)

확인 취소

PERIODIC SCRAPING

◆ Window 작업스케줄러

- 동작탭 -> [새로 만들기] 버튼

새 동작 만들기

실행할 작업을 지정해야 합니다.

동작(I): 프로그램 시작

설정

프로그램/스크립트(P):
%PycharmProjects#W1D4#src#dist#everyday-dollar.exe

인수 추가(옵션)(A):

시작 위치(옵션)(I):

확인 취소

새 동작 만들기

실행할 작업을 지정해야 합니다.

동작(I): 프로그램 시작

설정

프로그램/스크립트(P):
D:#Anaconda3#envs#PY373#python.exe

인수 추가(옵션)(A):
#src#everyday-dollar.py

시작 위치(옵션)(I):
armProjects#W1D4#src

확인 취소