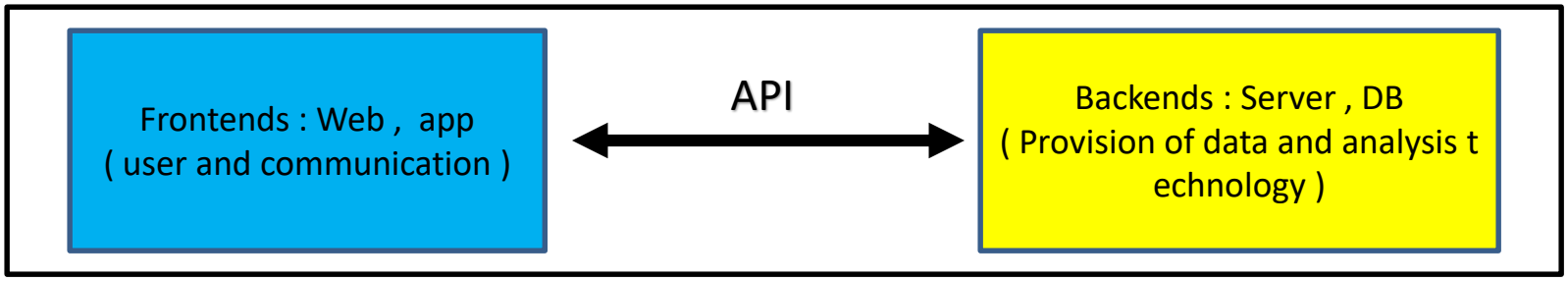
3. Advanced Web data scraping

- Open API
- Selenium
- User Agent String

Sogang University Department of Business Administration
Professor Myung Suk Kim

Open API

- API Structure : Frontends and backends interface role



- Open API

Data and analytics technology providers providing methods for using

- REST API way

Specify resources through URI (uniform Resource Identifier) and process actions through

- API operation

Request / Response structure _ _

Request : URL address , transmission method (GET, POST,…), request information (search , sorting, etc.)

Response : Provide

Open API

- Web Open API : Accessing data provided by web applications or web applications service
- Open API Providers: Web Applications and Web API
- Open APIs : Access APIs to collect data or communicate with web applications.

  Wrapper software (e.g. converting JSON format to R data) may be needed to seamlessly exchange data with web services.
- Open API data format: JSON, XML many. HTML, CSV, and binary files as well has exist
- REST (representation state transfer): As a representative API standard , it identifies the location of resources (data) by URL and defines a method for exchanging these resources .
- RESTful API: It is composed of three things.

  [1] Resource Identification (URL)

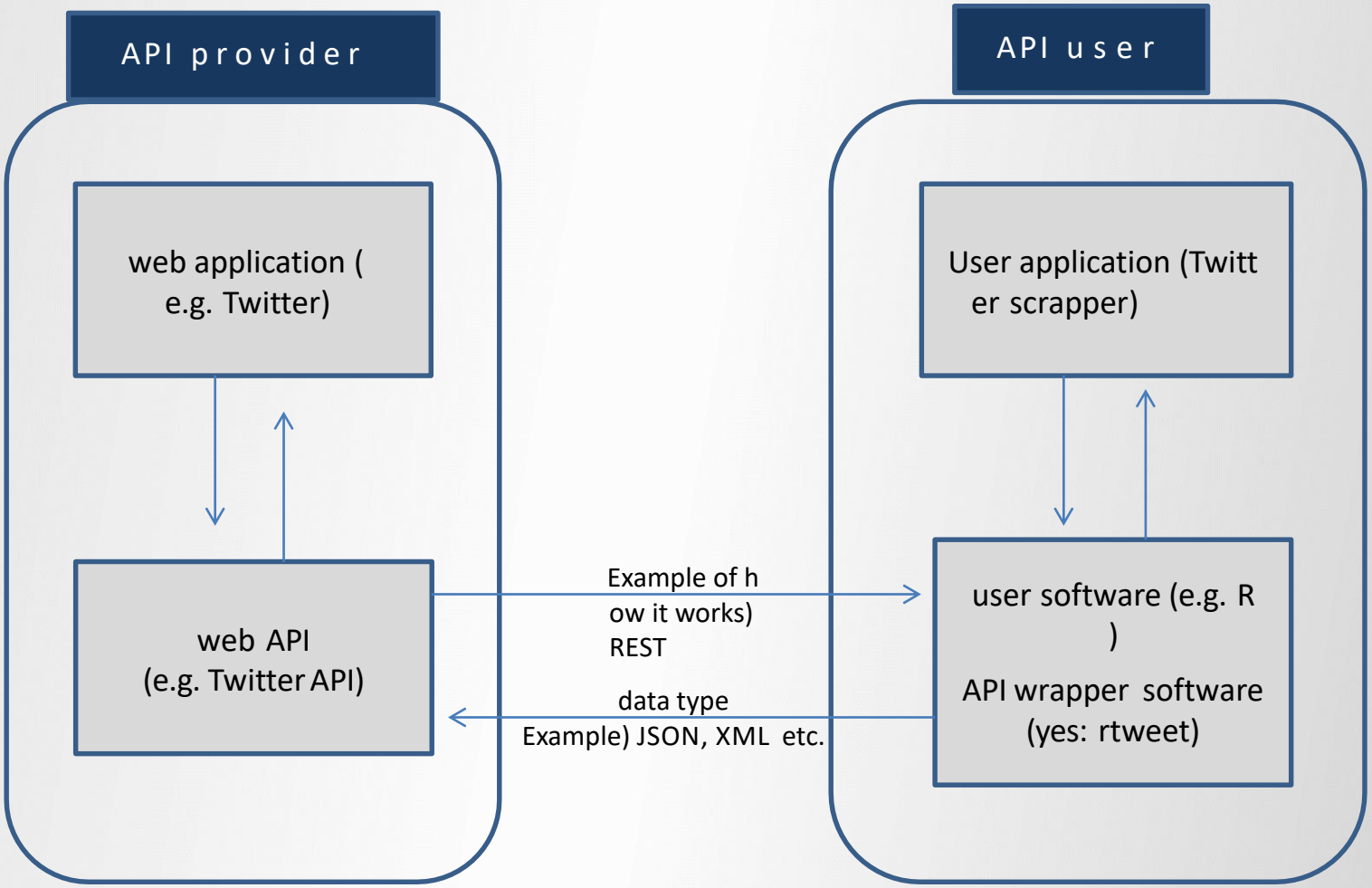
  [2] Resource representation (JSON, HTML,…)

  [3] Resource exchange method (HTTP method: GET (Query), POST (Create), PUT (Change), DELETE (Delete))

  Representation is a document (data) defined in a language such as HTML, XML, or JSON. For example , Twitter messages are represented in JSON, and the Twitter API uses JSON format. Twitter message processing the way defined.

Open API

- Web Open API



NAVER Open API

NAVER API

https://developers.naver.com Click [Service API] (fourth icon) in Stop and select [Search] <Use Open API Application>

❖ Write

Application name selection> [Web service setting] is selected in [Non-login open API service environment] URL comes out .
If there is no URL , enter http : //localhost and click

❖ Once registered, check the information on the

[0/25,000] is displayed in [Non -Login Open API Daily Usage] at the bottom . Up to 25,000 calls
100 extractions per cycle are possible . To request an excess of your daily allowance, apply using
Make a note of ID and Client secret .

❖ https://developers.naver.com You can check [API call examples] related to various Naver documents (blog , news , books , movies , etc.) _ _ _ _

❖ self app site Access

https://developers.naver.com/apps/#/myapps/ My ClientID /overview

Or, while logged in to your Naver account
https://developers.naver.com/apps/#/myapps

NAVER Open API

```
# Since the Naver search API example has the same calling method for blogs and specialized materials, only blog search i
s presented as a representative example .
# Naver Search Open API Example - Blog Search
import os
import sys
import urllib.request
client_id = "YOUR_CLIENT_ID"
client_secret = "YOUR_CLIENT_SECRET"
encText = urllib.parse.quote (" Specific word to search for " ) # Example : Sinchon

url = "https://openapi.naver.com/v1/search/ blog ?query =" + encText # json from blog result
# url = https://openapi.naver.com/v1/search/blog.xml?query= + encText # blog의 xml 결과

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)

recode = response.getcode ()
if ( rescode == 200 ):
    response_body = response.read ()
print( response_body.decode ('utf-8'))
else:
print("Error Code:" + recode )
```

Applications (Open API)

NAVER Open API

Python big data analysis based on data science (Lee Ji -young / Hanbit Academy)

[Naver News Crawling]

1. Specify search term : srcText = ' search term '
2. Search Naver News : getNaverSeach ()
   Url Configuration : url = base+node+srcText
   Url Access and search requests : urllib.request.urlopen ()

Receive request result as response Json : json.load ()

3. Organize response data and store them in a list : getPostData ()
4. Save the list as a JSON file : json.dumps ()

```
##########################################

import os
import sys
import urllib.request
import datetime
import time
import json

client_id = "YOUR_CLIENT_ID"
client_secret = "YOUR_CLIENT_SECRET"
```

Web analysis application (Open API)

```
#[CODE 1]
def getRequestUrl ( url ):
    req = urllib.request.Request(url)
    req.add_header("X-Naver-Client-Id", client_id)
    req.add_header("X-Naver-Client-Secret", client_secret)

    try:
        response = urllib.request.urlopen(req)
        if response.getcode() == 200:
            print ("[%s] Url Request Success" % datetime.datetime.now())
            return response.read().decode('utf-8')
    except Exception as e:
        print(e)
        print("[%s] Error for URL : %s" % (datetime.datetime.now(), url))
        return None
```

Web analysis application (Open API)

```
#[CODE 2]
def getNaverSearch(node, srcText, start, display):
    base = "https://openapi.naver.com/v1/search"
    node = "/%s.json" % node
    parameters = "?query=%s&start=%s&display=%s" % (urllib.parse.quote(srcText), start, display)

    url = base + node + parameters
    responseDecode = getRequestUrl(url)    #[CODE 1]

    if (responseDecode == None):
        return None
    else:
        return json.loads(responseDecode)

#[CODE 3]
def getPostData(post, jsonResult, cnt):
    title = post['title']
    description = post['description']
    org_link = post['originallink']
    link = post['link']

    pDate = datetime.datetime.strptime(post['pubDate'],  '%a, %d %b %Y %H:%M:%S +0900')
    pDate = pDate.strftime('%Y-%m-%d %H:%M:%S')

    jsonResult.append({'cnt':cnt, 'title':title, 'description': description, 'org_link':org_link,    'link': link,    'pDate':pDate})
    return
```

Web analysis application (Open API)

```
#[CODE 0]=>news search word (Ex: Sogang When you enter AI MBA), crawling starts , and when completed, it is saved to
def main():
    node = 'news'    # what to crawl
    srcText = input(' Please enter a search term : ')
     cnt = 0
     jsonResult = []

     jsonResponse = getNaverSearch ( node , srcText , 1, 100) #[CODE 2]
     total = jsonResponse ['total']

     while (( jsonResponse != None) and ( jsonResponse ['display'] != 0)):
         for post in jsonResponse['items']:
             cnt += 1
             getPostData(post, jsonResult, cnt)  #[CODE 3]
         start = jsonResponse['start'] + jsonResponse['display']
         jsonResponse = getNaverSearch(node, srcText, start, 100)  #[CODE 2]
     print(' Total search : %d items ' %total)

     with open(' %s_ naver _% s.json ' % ( srcText , node) , 'w' , encoding='utf8') as outfile :
         jsonFile = json.dumps(jsonResult,   indent=4, sort_keys=True,   ensure_ascii=False)
         outfile.write(jsonFile)
     print("가져온 데이터 : %d 건" %(cnt))
     print ('%s_naver_%s.json SAVED' % (srcText, node))

if __name__ == '__main__': #Execute the command below the if statement only when executing without importing module .
main()
https://medium.com/@chullino/if-name-main-%EC%9D%80-%EC%99%9C-%ED%95%84%EC%9A%94%ED%95%A0%EA%B9%8C-bc48cba7f720
```

Selenium

- Selenium

Create a virtual web browser to (1) click on a web page to navigate to another web page and (2) web browser Scroll down to remotely extract the contents at the bottom of the web Executed (resolving the part that cannot be processed with the requests module)
https://www.geeksforgeeks.org/selenium-python-introduction-and-installation/?ref=lbp
https://www.browserstack.com/guide/selenium-webdriver-tutorial

- How to select

https://greeksharifa.github.io/references/2020/10/30/python-selenium-usage/
https://selenium-python.readthedocs.io/locating-elements.html

- References :

https://stackoverflow.com/questions/32391303/how-to-scroll-to-the-end-of-the-page-using-selenium-in-python
https://jhleeeme.github.io/scrolling-in-selenium/
https://www.browserstack.com/guide/run-selenium-tests-using-selenium-chromedriver

Web analytics application (Colab + selenium)

1. Selenium setup (in Colab)

```
# https://nariyoo.com/python-how-to-run-selenium-in-google-colab/
# Set up for running selenium in Google Colab
## You don't need to run this code if you do it in Jupyter notebook, or other local Python setting
## if you receive error signs when you run the following commands, then execute the command line by line

%%shell
!sudo apt -y update
!sudo apt install -y wget curl unzip
!wget http://archive.ubuntu.com/ubuntu/pool/main/libu/libu2f-host/libu2f-udev_1.1.4-1_all.deb
!dpkg -i libu2f-udev_1.1.4-1_all.deb
!wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
!dpkg -i google-chrome-stable_current_amd64.deb
CHROME_DRIVER_VERSION='curl -sS chromedriver.storage.googleapis.com/LATEST_RELEASE'
!wget -N https://chromedriver.storage.googleapis.com/$CHROME_DRIVER_VERSION/chromedriver_linux64.zip -P /tmp/
!unzip -o /tmp/chromedriver_linux64.zip -d /tmp/
!chmod +x /tmp/chromedriver
!mv /tmp/chromedriver /usr/local/bin/chromedriver
!pip install selenium
```

Web analytics application (Colab + selenium)

```
!pip install chromedriver-autoinstaller
import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')
import time
import pandas as pd
from selenium import webdriver
import chromedriver_autoinstaller  # setup chrome options
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless') # ensure GUI is off
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')  # set path to chromedriver as per your configuration
chromedriver_autoinstaller.install()  # set the target URL
url = "put-url-here-to-scrape"  # set up the webdriver

driver = webdriver.Chrome(options=chrome_options)
#########################################################

from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

from bs4 import BeautifulSoup as bs
from urllib.request import urlopen
from urllib.parse import quote_plus
from selenium.webdriver.common.keys import Keys
import time
```

Web analytics application (Colab + selenium)

2. Search Naver on Google

```
url ="https://www.google.com/"
driver = webdriver.Chrome(options=chrome_options)
driver .get ( url ) # selenium Use

from selenium.webdriver.common.by import By
# Go to naver using google search
driver.find_element (By.NAME, "q"). send_keys (" Naver ") # naver from google search input
driver.find_element (By.NAME, "q"). send_keys ( Keys.ENTER )
driver.find_element ( By.PARTIAL_LINK_TEXT , "NAVER").click()
# naver Click the search button ( results vary depending on whether this button is executed or not )
print( driver. title )
print( driver. current_url )
driver.close ()
```

3. Save

```
url = "https://www.naver.com/"
driver = webdriver.Chrome(options=chrome_options)
driver .get ( url ) # selenium Use

Preprocessing command to save files to google drive ( set path )
from google.colab import drive
drive.mount ('/content/ gdrive ')
# Specify screen size: driver.set_window_size (320, 600)
# Specify the file name and save path
driver.save_screenshot ('/content/ gdrive /My Drive/ Colab Notebooks/ Textmining /Website.png')
driver.quit ()
```

Web analytics application (Colab + selenium)

4.1 NAVER News : Link News (using selenium)

```
driver = webdriver.Chrome(options=chrome_options)
#### Extract
import requests; import re
from bs4 import BeautifulSoup
url = "https://news.naver.com/"
req = requests.get ( url )
html = req. text
soup = BeautifulSoup (html, ' html.parser ')

############## News title and url by media Extraction (80 cases )
titles = soup.select (" div.cjs_journal_wrap a")
title_list = []
url_list = []
for title in titles:
    text1 = title.text
    text2 = re.sub('₩t|₩r|₩n|₩.|//|₩{|₩}|', '', text1)
    title_list.append(text2)
    url_list.append(title["href"])

############## news information building (combine title, url)
news_list = zip(title_list, url_list)
news_info_list = []
for news in news_list:
    news_dict = {
"title": news[0],
" url ": news[1]
}
    news_info_list.append ( news_dict )
```

Web analytics application (Colab + selenium)

```
############# Extract linked news content ( takes about 2 minutes )
news_stories =[]
for url_i in url_list :
    driver .get ( url_i ) # using selenium
html = driver. page_source
    soup_i = BeautifulSoup (html, " html.parser ")
string0 = soup_i.select (" div#dic_area ")
strings = " ".join(map( str , string0)) # convert list to string
   string1 = re.sub ('<script.*?>.*?</script>', '', strings, 0, re.I|re.S ) # Remove content inside script
   string2 = re.sub ('<.+?>', '', string1, 0, re.I|re.S ) # remove tags and comments
   storys = re.sub ('₩t|₩r|₩n|₩.|//|₩{|₩}|', '', string2) # Delete other unnecessary symbols
   news_stories.append ( stories )

print( news_stories )

############## news information building(combine title, content)
news_list = zip(title_list, news_stories)
news_info_list = []
for news in news_list:
    news_dict = {
        "title": news[0],
        "content": news[1]
    }
    news_info_list.append(news_dict)

print(news_info_list)
```

Web analytics application (Colab + selenium)

4.2 Extraction of headline news titles in economy section of NAVER News

```
driver = webdriver.Chrome(options=chrome_options)
#### Extract
from bs4 import BeautifulSoup
import requests
import re

# Economy section
url = "https://news.naver.com/main/ main.naver?mode = LSD&mid =shm&sid1=101"
driver.get ( url )

html = driver .page_source
type(html)
soup = BeautifulSoup (html, ' html.parser ')
type (soup)

############# headline news (12) title and url extraction
titles = soup.select(".sh_text>a")
title_list = []
url_list = []
for title in titles:
text1 = title. text
text2 = re.sub ('₩n', '', text1)
    title_list.append (text2)
    url_list.append (title[" href "])

title_list
url_list
```

Web analytics application (Colab + selenium)

웹 분석 응용 (Colab + selenium)

4.3 Extraction of headline news titles and linked contents of 3 sections of NAVER News

```
driver = webdriver.Chrome(options=chrome_options)
#### NAVER News extraction
from bs4 import BeautifulSoup
import requests
import re

######## NAVER News: Politics(정치), Economy(경제), Social(사회) Section URL
news_site = "https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1="
url_section=['101', '102', '103']

title_list_all = []
url_list_all = []
news_stories_all =[]

for url_i in url_section:
  driver.get(news_site + url_i)  # Section-based news list
  html1 = driver.page_source
  soup1 = BeautifulSoup(html1, 'html.parser')
  titles1 = soup1.select(".sh_text>a")
  title_list1 = []
  url_list1 = []

  for title1 in titles1:
    text1 = title1.text
    text2 = re.sub('₩₩|₩n', '', text1)
    title_list1.append(text2)
    url_list1.append(title["href"])
```

웹 분석 응용 (Colab + selenium)

```
    news_stories =[]

    for url_j in url_list1:
      driver.get(url_j)
      html_j = driver.page_source
      soup_j = BeautifulSoup(html_j, "html.parser")
      string0 = soup_j.select("div#newsct_article")   # news contents in the linked articles
      strings = " ".join(map(str, string0))   # list into string
      string1 = re.sub('<script.*?>.*?</script>', '', strings, 0, re.I|re.S)   # remove contents in script
      string2 = re.sub('<.+?>', '', string1, 0, re.I|re.S)  # remove tags and annotations
      storys = re.sub('₩t|₩r|₩n|₩.|//|₩{|₩}|', '', string2)  # remove unnecessary symbols
      news_stories.append(storys)

  title_list_all.append(title_list1)
  url_list_all.append(url_list1)
  news_stories_all.append(news_stories)

############### New information (title, content)
news_list = zip(title_list_all, news_stories_all)
news_info_list = []
for news in news_list:
    news_dict = {
        "title": news[0],
        "content": news[1]
    }
    news_info_list.append(news_dict)

print(news_info_list)
news_info_list[0]['title']
```

Web analytics application (Colab + selenium)

5. Amazon product review (updating page) data extraction (service unavailable site)

```
driver = webdriver.Chrome(options=chrome_options)
from bs4 import BeautifulSoup
import requests
import re

#https://www.amazon.com/Fire-HD-10-tablet/product-reviews/B08BX7FV5L/ref=cm_cr_getr_d_paging_btm_prev_1?ie=UTF8&reviewerType=all_reviews&pageNumber=1

url = "https://www.amazon.com/Fire-HD-10-tablet/product-reviews/B08BX7FV5L/ref=cm_cr_getr_d_paging_btm_prev_1?ie=UTF8&reviewerType=all_reviews&pageNumber="
num = list(map( str , range(1,4))) # print 1, 2, 3 as strings

author_set = []
score_set = []
review_set = []
help_set = []

####### Extract information by page using for statement
for i in num:
  driver.get(url+i)      # selenium 이용
  html = driver.page_source
  soup = BeautifulSoup(html, 'html.parser')
  author = soup.select("span.a-profile-name")
  score = soup.select(".review-rating > span.a-icon-alt")
# If there is an image , only .a-icon-alt can be used for repeated extraction , so .review-rating must also be used .
  review = soup.select (" span.review -text > span")
help = soup.select (".a-color-tertiary")
# You can use only one attribute value that can distinguish the element among several attribute values in the class attribute .
```

Web analytics application (Colab + selenium)

```
####### Extract information by page using for statement
  author_list = []
  for author_i in author:
    author_list.append(author_i.text)

  score_list = []
  for score_i in score:
    score_list.append(score_i.text)

  review_list = []
  for review_i in review:
review1 = review_i.text
    review_list.append ( re.sub ('₩t|₩r|₩n', '', review1))

  help_list = []
for help_i in help:
help1 = help_i.text
    help_list.append ( re.sub ('₩t|₩r|₩n', '', help1))

del author_list [0:2] # Remove the first 2 reviews of each page as they are repeated
  del score_list [0:2] # Remove the first 2 scores on each page as they are repeated
  del help_list [0:2] # first 2 help on each page are repeated so remove them

  author_set.extend ( author_list ) # combine multiple lists into one list
  score_set.extend ( score_list )
   review_set.extend ( review_list )
   help_set.extend ( help_list )
print( i )
```

Web analytics application (Colab + selenium)

```
len ( author_set )
len ( score_set )
len ( review_set )
len ( help_set )

############## Write review information (title, content bundle )
info_list = zip(author_set, score_set, review_set, help_set)
info_info_list = []
for info in info_list:
    info_dict = {
        "author": info[0],
        "score": info[1],
        "review": info[2],
        "help": info[3]
    }
    info_info_list.append(info_dict)

print( info_info_list )
print( info_info_list [3])
```

Web analytics application (Colab + selenium)

6. Naver image keyword search and screen scroll down (https://wikidocs.net/61216)

```
driver = webdriver.Chrome(options=chrome_options)
key = ' AI '
url = 'https://search.naver.com/ search.naver?sm = tab_sug.top&where = image&query ='
url_new = url+key
driver .get ( url_new ) # via selenium

from selenium.webdriver.common.by import By
# Determine the page size to scroll down : range(n)
body = driver.find_element ( By.CSS_SELECTOR , 'body')
for i in range(20):
    body.send_keys ( Keys.PAGE_DOWN )
    time. sleep (1)

# Collect image links
imgs = driver.find_elements(By.CSS_SELECTOR, 'img._image')
result = []
for img in imgs:
    if 'http' in img.get_attribute('src'):
        result.append(img.get_attribute('src'))
print(result)

driver.close()
len (result) # check the change in len (result) while increasing the size of n in range(n)
```

Web analytics application (selenium)

```
# save to folder
# Preprocessing command to save file in colab ( set path )
from google.colab import drive
drive.mount ('/content/ gdrive ')

Specify URL , file name and save path
folder_name = "/content/ gdrive /My Drive/ Colab Notebooks/ Textmining /download/" # pre-create a folder called dow
nload in colab
num = list(map( str , range(1,len(result)+1)))
len (result)

# Download and save the file
import urllib.request

index = []
for link in result:
  savename = folder_name + num [ len (index)]
  index. append (link)
  urllib.request.urlretrieve (link, savename )
```

Web analytics application (user-agent/ Colab)

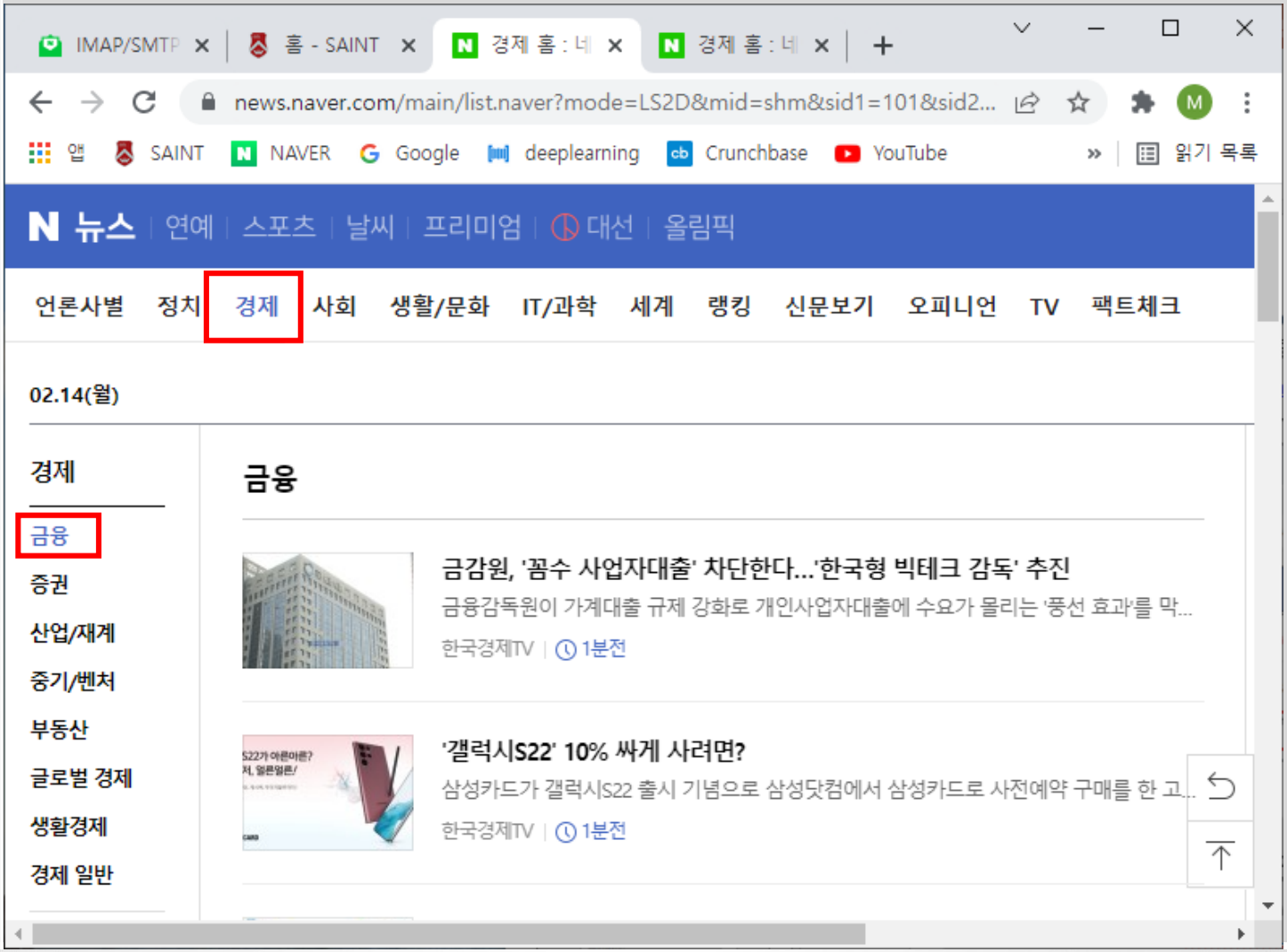
7. Naver News Economy - Top news titles and urls in the field of finance Extraction (using user-agent information)

 # If crawling is blocked on the server , try to solve the problem using User-Agent information (Check at www.useragentstring.com)

```
import requests
from bs4 import BeautifulSoup
import bs4.element
import datetime

def get_soup_obj ( url ):
head = {'User-Agent': ' Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit /537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36 '} # Retrieve User-Agent information from your PC Enter the value obtained by
res = requests.get ( url , headers = head)
soup = BeautifulSoup ( res.text , ' lxml ')
return soup

area_url = "https://news.naver.com/main/list.naver?mode=LS2D&mid=shm&sid1=101&sid2=259"
soup = get_soup_obj(area_url)

titles = soup.find('ul', class_='type06_headline').find_all("li", limit=10)
title_list = []
url_list = []
for title in titles:
    try:
       text1 = title.img.attrs.get('alt')
    except:
text1 = title.a.text
    title_list.append (text1)
text2 = title.a.attrs.get (' href ')
    url_list.append (text2)
```

Web analytics application (user-agent)

Naver News Economy - Extraction of top ranking news in the financial field (using user-agent information)



Web analytics application (PyCharm + selenium)

1.teamblind website crawling (crawling all the way down the screen)

[1] Crawl

```
from selenium import webdriver
from bs4 import BeautifulSoup
driver = webdriver.Chrome ( r"D :₩ selenium₩chromedriver.exe " )

url = "https://www.teamblind.com/us/topics/%EC%A7%80%EB%A6%84%C2%B7%EC%87%BC%ED%95%91"
driver.get ( url )

### ################### If you scroll down the mouse , you can read up to that amount of data
##################### Without mouse scroll down only 52 outputs
######## OK
html1 = driver.page_source
soup = BeautifulSoup (html1, ' html.parser ' )
title_list = soup.select ( " div.tit > h3 > a" )
len ( title_list )
title_list [90].text
```

Web analytics application (PyCharm + selenium)

[2] Scrape as far as possible of the webpage for a specified period of time

```
from selenium import webdriver
from bs4 import BeautifulSoup
import datetime
import time
driver = webdriver.Chrome(r"D:₩selenium₩chromedriver.exe")

url = "https://www.teamblind.com/kr/topics/%EC%A7%80%EB%A6%84%C2%B7%EC%87%BC%ED%95%91"
driver.get(url)
##############################################################
Output after auto scrolling down for
30 seconds def doScrollDown ( whileSeconds ):
start = datetime.datetime.now ()
end = start + datetime.timedelta ( seconds = whileSeconds )
    while True :
        driver.execute_script ( ' window.scrollTo (0, document.body.scrollHeight );' )
        time. sleep ( 1 )
        if datetime.datetime.now () > end:
            break
doScrollDown ( 30 )
######################### OK
html1 = driver.page_source
soup = BeautifulSoup (html1, ' html.parser ' )
title_list = soup.select ( " div.tit > h3 > a" )
len ( title_list )
title_list [200].text
```

Reference : https://hello-bryan.tistory.com/194

Web analytics application (PyCharm + selenium)

[3] Web page Scraping by designating to the end point (continued)

```
From selenium import webdriver
from bs4 import BeautifulSoup
import time
driver = webdriver.Chrome ( r"D :₩ selenium₩chromedriver.exe " )

url = "https://www.teamblind.com/kr/topics/%EC%A7%80%EB%A6%84%C2%B7%EC%87%BC%ED%95%91"
driver.get(url)

################################
SCROLL_PAUSE_SEC = 1
last_height = driver.execute_script ( "return document.body.scrollHeight " ) # Get the scroll height
while True :
    driver.execute_script ( " window.scrollTo (0, document.body.scrollHeight );" ) # Scroll down to the end ( mouse movem
ent effect )
    time.sleep (SCROLL_PAUSE_SEC) # wait 1 second
    new_height = driver.execute_script ( "return document.body.scrollHeight " ) # Get scroll height back after scrolling do
wn
    if new_height == last_height :
        break
    last_height = new_height

######################### 확인
html1 = driver.page_source
soup = BeautifulSoup(html1, 'html.parser')
title_list = soup.select("div.tit > h3 > a")
len(title_list)
title_list[200].text
```

Web analytics application (PyCharm + selenium)

2. Coffee Bean crawling

Python big data analysis based on data science (Lee Ji -young / Hanbit Academy)

1) Find store information : www.coffeebeankorea.com > store > Find a store > Local search > See the lower left mark when selecting Seoul
2) Call javascript:storeLocal2(' Seoul ') after checking display
3) HTML Source Check (Ctrl+U): No list of stores searched => Need to use
4) If you return to the homepage and hover the mouse over [View Details] , javascript:storePop2('372') etc. come out

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import datetime

from selenium import webdriver
import time

#[CODE 1]
def CoffeeBean_store(result):
    CoffeeBean_URL = "https://www.coffeebeankorea.com/store/store.asp"
    wd =webdriver.Chrome(r"D:₩selenium₩chromedriver.exe")

    for i in range(1, 370):  #매장 수 만큼 반복
      wd.get(CoffeeBean_URL)
        time.sleep (1) # wait 1 second while connecting to webpage
      try:
```

Web analytics application (PyCharm + selenium)

```
#[CODE 1] ( continued )
            wd.execute_script ("storePop2(%d)" % i )
            time.sleep (1) # wait 1 second while script runs
          html = wd. page_source
            soupCB = BeautifulSoup(html, 'html.parser')
            store_name_h2 = soupCB.select("div.store_txt > h2")
            store_name = store_name_h2[0].string
            print(store_name)  #매장 이름 출력하기
          store_info = soupCB.select("div.store_txt > table.store_table > tbody > tr > td")
            store_address_list = list(store_info[2])
            store_address = store_address_list[0]
            store_phone = store_info[3].string
            result.append([store_name]+[store_address]+[store_phone])
        except:
            continue
    return

#[CODE 0]
def main():
    result = []
    print('CoffeeBean store crawling >>>>>>>>>>>>>>>>>>>>>>>>>>>>')
    CoffeeBean_store(result)  #[CODE 1]

    CB_tbl = pd.DataFrame (result, columns=('store', ' address','phone '))
    CB_tbl.to_csv ('D:/CoffeeBean.csv', encoding='cp949', mode='w', index=True)

if __name__ == '__main__':
main()
```