복습

URI : 자원정보를 주소에 넣는 형태

id : html에서 절대 겹쳐서는 안됨

class : 여러번 사용 가능

json : 문법, 표기법

```
$(document).on("click","#rest_button_json_json",function() {
        $.ajax({ url         : "./form_rest_json_json" ,
                 method      : "POST",
                 data        : JSON.stringify({"userid":"hellodddd"}) ,
                 contentType : "application/json; charset=UTF-8",  //내 데이터는 글씨에요
                 dataType    : "json" , //결과는 json로 주세요
                 error       : function(aaa){ alert(aaa) },
                 success     : function(res){
                                    console.log ('test합니다.')
                                    console.log(res) //이미 객체
                                    console.log(res[0])     //str --> undefined
                                    console.log(res['msg']) //객체 --> 바로 꺼내기 가능

                                    json_str = JSON.stringify(res) // 브라우저에 따라 문제가 발생할 수
// 있기 때문에(객체 전달 안되는 경우)
                                                                   // 한번 글자 처리 후 객체로 변환
                                    console.log(json_str)
                                    console.log(json_str.slice(0,5)) //슬라이싱

                                    json_obj = JSON.parse(json_str) //
                                    console.log(json_obj)
                                }
        });
    });
```

Open API

1. 맵 웹에 뿌리기

1.1 flask_run

```
import pandas as pd
import numpy as np
import folium
from folium import plugins
import re
import googlemaps
import pprint


@app.route("/map")
def map():
    # ----------------------------------------------------
    # data load : DataFrame 작업
    dataset = pd.read_csv("./datasets/제주관광공사_여행장소_20220322.csv", encoding="cp949")
    df = dataset[dataset['장소상세설명']=='숙소']
    geo_list = []
    name_list = []
    for i in range(len(df))[:100]:
        lat   = df.iloc[i]['위도']
        lng   = df.iloc[i]['경도']
        sname = df.iloc[i]['장소명']
        # print(lat, lng, sname)
        geo_list.append((lat, lng))
        name_list.append(sname)
    # ----------------------------------------------------
    # folium map
    map = folium.Map(location=[33.41041350000001, 126.4913534], zoom_start=10,
                     tiles='OpenStreetMap')  # Stamen Terrain')
    plugins.MarkerCluster(geo_list, popups=name_list).add_to(map)
    #----------------------------------------------------
    # web browser에 보이기 위한 준비
    map.get_root().width = "800px" # 인터넷에서의 map 크기 설정
    map.get_root().height = "600px"
    html_str = map.get_root()._repr_html_() # map을 html로 바꿔줌
    # ----------------------------------------------------


    return render_template('result_map.html'
                           , KEY_MYDATA=html_str)

    # return render_template_string(
    #     """
    #         <!DOCTYPE html>
    #         <html>
```

```
    #             <head></head>
    #             <body>
    #                 <h1>Using an iframe</h1>
    #                 {{ iframe|safe }}
    #             </body>
    #         </html>
    #     """,
    #     iframe=iframe,
    # )


if __name__ == '__main__':
    app.debug = True
    app.run(host='0.0.0.0', port=7878)
```

1.2 result_map.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
결과페이지
<hr>
{{ KEY_MYDATA|safe }}  // < 와같은 기호가 있더라도 안전하다 실행해라
</body>
</html>
```

2. Selenium

2.1 selenium_naver.py

```
# pip install selenium
# download chrome driver

import pandas as pd
import time
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
```

```
from selenium.webdriver.common.keys import Keys


def mysearch(search_word="제주관광지") :
    #-------------------------------------------- 크롬 옵션 객체 생성
    # options = webdriver.ChromeOptions()
    # options.add_argument("window-size=1000x800") # 화면크기(전체화면)
    # user_agent = "Mozilla/5.0 (Windows NT 4.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/37.0.2049.0 Safari/537.36 "
    # options.add_argument('user-agent=' + user_agent)
    # options.add_argument('headless') # headless 모드 설정
    # options.add_argument("disable-gpu")
    # options.add_argument("disable-infobars")
    # options.add_argument("--disable-extensions")
    # options.add_argument("--mute-audio") #mute
    # options.add_argument('--blink-settings=imagesEnabled=false') #브라우저에서 이미지 로딩을 하지 않
습니다.
    # options.add_argument('incognito') #시크릿 모드의 브라우저가 실행됩니다.
    # options.add_argument("--start-maximized")
    # driver = webdriver.Chrome('./chromedriver_102.0.5005.27.exe', options=options)

    print("search_word", search_word)
    #-------------------------------------------- 크롬 드라이버 로드  110.0.5481.177
    # https://chromedriver.chromium.org/downloads
    # https://chromedriver.storage.googleapis.com/index.html?path=110.0.5481.77/
    # ---------------------------------------------
    driver = webdriver.Chrome('chromedriver_110.exe')
    driver.get("https://travel.naver.com/domestic/14/guide/tour?limit=12")

    #-------------------------------------------- 스크롤 다운
    endkey = 4  # 스크롤 다운 시 12개목록씩 추가  -- 총 12*4=48개
    while endkey:
        # driver.find_element_by_tag_name('body').send_keys(Keys.END)
        #ele_path = driver.find_element(By.CSS_SELECTOR, "# __next > div >
div.mainContainer_container__1GEbx > div > div > main > div:nth-child(2) >
div.guide_GuidePanel__3S6xd > div > div > button")
        ele_path = driver.find_element(By.XPATH, '//*
[@id="__next"]/div/div[2]/div/div/main/div[2]/div[2]/div/div/button')
        ## click button
        driver.execute_script("arguments[0].click();", ele_path); # 브라우저 화면 제어
        time.sleep(2) # 화면 움직이고 페이지 로드 기다리기
        endkey -= 1
    #-------------------------------------------- lxml
    # soup = BeautifulSoup(htmlstr, 'lxml')
    # video_list0 = soup.find('div', {'id': 'contents'})
    # print(video_list0)
    #-------------------------------------------- html.parser
    htmlstr = driver.page_source
    soup = BeautifulSoup(htmlstr, features="html.parser") #selenium을 통해 긁어온 정보를 파싱하기
    "#__next > div > div.mainContainer_container__1GEbx > div > div > main > div:nth-child(2) >
div.guide_GuidePanel__3S6xd > div > ul > li:nth-child(1)"
    div_list = soup.select("div#__next > div > div.mainContainer_container__1GEbx > div > div > main
> div:nth-child(2) > div.guide_GuidePanel__3S6xd > div > ul > li")
    #---------------------------------------------
```

```
    movie_list = []
    for div in div_list:
        url     = div.select_one("div > a").get('href')
        title   = div.select_one("div > a > b").text
        img = div.select_one("div > a > figure > img").get('src')
        print(title, url, img)
        movie_list.append([title, url, img])

    print(len(movie_list))
    df = pd.DataFrame(movie_list, columns=["title","url","img"])
    print(df.head())
    print(df.info())
    df.to_csv("youtebe_sel_res.csv", index=False)
    return movie_list

mysearch(search_word="제주관광지")
```

2.2 youtube_search.py

```
# https://github.com/alexmercerind/youtube-search-python
#Sync
# pip install youtube-search-python

import pandas as pd
from youtubesearchpython import VideosSearch
import json

#------------------------------------------------------------
def my_youtube_search(search_str='이누야샤', nrows=7) :
    videosSearch = VideosSearch(search_str, limit=nrows)
    json_res = videosSearch.result()

    #print(videosSearch.result())  ## [{},{},{}]
    #print(json.dumps(videosSearch.result(), sort_keys=True, indent=4))

    movie_list = json_res['result']
    print(json.dumps(movie_list, sort_keys=True, indent=4))

    tot_list = []
    for movie in movie_list:
        dict = {}
        # print(movie['thumbnails'][0]['url'])
        # print(movie['link'])
        # print(movie['title'])
        dict["title"]    = movie['title']
        try :
            dict["movie"] = movie['richThumbnail']['url']
```

```
        except :
            dict["movie"] = movie['thumbnails'][0]['url']

        dict["img"]      = movie['thumbnails'][0]['url']
        dict["duration"] = movie['duration']
        dict["url"]      = movie['link']
        dict["rdate"]    = movie['publishedTime']
        dict["cnt"]  = movie['viewCount']['text']
        tot_list.append(dict)
    return tot_list  #[{},{}]

tot_list = my_youtube_search('지브리 영화', 10)
df = pd.DataFrame(tot_list)
print(df.head())
# print(df.info())
```

2.3 selenium_youtube.py

```
# pip install selenium
# download chrome driver

import pandas as pd
import time
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By


from selenium.webdriver.common.keys import Keys
def mysearch(search_word="지브리 영화") :
    #---------------------------------------------- 크롬 옵션 객체 생성
    # options = webdriver.ChromeOptions()
    # options.add_argument("window-size=1000x800") # 화면크기(전체화면)
    # user_agent = "Mozilla/5.0 (Windows NT 4.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/37.0.2049.0 Safari/537.36 "
    # options.add_argument('user-agent=' + user_agent)
    # options.add_argument('headless') # headless 모드 설정
    # options.add_argument("disable-gpu")
    # options.add_argument("disable-infobars")
    # options.add_argument("--disable-extensions")
    # options.add_argument("--mute-audio") #mute
    # options.add_argument('--blink-settings=imagesEnabled=false') #브라우저에서 이미지 로딩을 하지 않
습니다.
    # options.add_argument('incognito') #시크릿 모드의 브라우저가 실행됩니다.
    # options.add_argument("--start-maximized")
    # driver = webdriver.Chrome('./chromedriver_102.0.5005.27.exe', options=options)
```

```
    print("search_word", search_word)
    #-------------------------------------------- 크롬 드라이버 로드  110.0.5481.177
    # https://chromedriver.chromium.org/downloads
    # https://chromedriver.storage.googleapis.com/index.html?path=110.0.5481.77/
    # --------------------------------------------
    driver = webdriver.Chrome('chromedriver_110.exe')
    driver.get("https://www.youtube.com/results?search_query="+search_word)

    #-------------------------------------------- 스크롤 다운
    endkey = 4  # 스크롤 다운 시 30개목록씩 추가  -- 총 30*4=120개
    while endkey:
        # driver.find_element_by_tag_name('body').send_keys(Keys.END)
        driver.find_element(By.TAG_NAME, "body").send_keys(Keys.END)

        time.sleep(1.5)
        endkey -= 1

    #-------------------------------------------- lxml
    # soup = BeautifulSoup(htmlstr, 'lxml')
    # video_list0 = soup.find('div', {'id': 'contents'})
    # print(video_list0)
    #-------------------------------------------- html.parser
    htmlstr = driver.page_source
    soup = BeautifulSoup(htmlstr, features="html.parser")
    div_list = soup.select("#contents > ytd-video-renderer")
    #--------------------------------------------

    movie_list = []
    for div in div_list:
        url     = div.select_one("div#dismissible > ytd-thumbnail > a#thumbnail").get('href')
        url     = "https://www.youtube.com" + url
        title   = div.select_one("a#video-title > yt-formatted-string").text
        try:
            rdate   = div.select_one("div#metadata-line > span:nth-child(2)").text
        except: # 예외 상황이 발생하면 밑의 코드 실행 -> 프로그램을 오류로 인해 중간에 중단하는 일 없이 끝까지 돌리기 위함
            rdate  = ""
        cnt_str = div.select_one("div#metadata-line > span:nth-child(1)").text
        cnt_str = cnt_str.replace("조회수 ","")[:-1]

        try :
            img = div.select_one("img#img").get('src')
        except:
```

```
            img =
"
PgjIyPFxcUZGRlGRkaNjY0fHx8WFhbr6+tsbGweHh5cXFycnJx0dHT/4ODX19c1NTWAgICrq6sSEhIKCgr/6Ojv7+//mZm+vr6Xl
5fLy8v/YmL/Ly//Fhb/k5P/ICD/09NSUlKFhYU+Pj6lpaVkZGT/vr7/p6f/i4v/d3f/aGj/UVH/QUH/r6//8/P/wsL/SUn/hob/V
lb/eHj/wMDg4ODnNabOAAAFwklEQVR4nO2aa3+iOBSHI0Gk3lAHFWrVaadVsbftdjq76/f/YJtzEvBK5ebM7P7+zyukEPKQy8kJF
QIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4L/I9cPj083zy+vXP95u7+9qe9zd3b59f
H19+XHz9Phw/atrWoT3b7e1HNw/X9qyMVXMqyvvMY+e5im5uc6c+pFK/ZCjK4bStuWiMsE/8wvWat/iu68kYX44dDw887yGPODq6
JIrx7KcXlWC10UEa7UHc3sUWJYlO3w8keq4PTtnGFp7nFCp1vC1mOGbuX1DVvaUj+dUd+n/bobFBGu1d3O/pyrpNvmw3ybbbcw9sS
FfhsB0dBccDrlLDh6KG300BI6Xo6Er2Wpbv6p574mbZVCwcfjNE/7KGheYZ4sYUMLVVVUM+pCEZjLM9t+lSj075Y6WGz0UNX0wBP
L3IjTBD0kw6Z9GGKZGlUsOCE02t9hGXsHJMy41VGzrnYkXMzzP8OF3/+7OGd3EJM1VXbyT0iDRzDof1KgwPSsmwnjji7nT9r8+Pz
7gEX5qmG6p6hbzW8gctFcnb64a+YjpSDKgjR3w02TOkU6MlVX9AR/0dw5laQjhR/KDOcqhKDRYZR3pMSvW/CPHjjGEcLkSo61qX8
dwxkB7HgpbUE2tfep7HcXJg01FnzzBwPc+1yFCqI3u9NRzS1OXIpX7MWOoQ44SrXA2Zbiiu/8pmuG5x21G8b1H9mjKJ5vaIDdvxS
mDgxZPR1rBlmT7Ar4ijDRuue/o96WlMzLlUdnTXOQTfPzEU4ntKH2aSBINmmHYkoraecXhKtaQlk9oVMrQcT3V0FuJ1II0BSy7aT
pZ1U2ZDIZ7SDeNLREdyxbotLbR0dR1ofFrusrjhqiNmdKdDS3M/DkpkmsxnGUhbdyfVr/+dZviQFKIe6liC+5D6RT2Jeysp85lih
pImJNsyg5tmbF46UWFOjszqrKEQX1ICytaQWi2klqQW0wkGzYhR29Sz2DjkaEFjnMvoOaZD8BoqbTFUzFCIf04Ox61hQz3U7asq2
w2TYASUbEwDI1bCUF+20Y3JA3IefhJIixoKcfupIVdtZR7M75hUWZwTqxKG1Dk5yMrY0A9zrA2zGp5rQ7HQuRAPD5pZdwxpdi1hy
Lcqw23f9/OsfrMZnh+Huh4cMszoq8wwMu9LG1L5udb3WQyzzKX6vRqJfqWG411Db+n7/jTIZ1hFPFS4bNhK2vMyhpYXhmFgVWmYb
U0T19wbXNowIYdhFetSYQIDh4iLGrrJDmQlhplzi1iGtS5p6M4mHUN2wQrywwPD6MLRIjflc/wDw8N4WDLiG8N6EvFzU36f5sAwW
cmYXbh5Nau2raHV645GOQxL77UdGnJETnamuHZVrLxX8cq7Lh3Py7HyLr9femhYT9JCThRpJzUyXe20Iae2Vophkj3RnZyTbZLNk
oyU3vM+NNTVSxLFXtyYlLX623C9NdTJYMe8kW1+2PbNnSyWJE2cKB5/rvqEooa7wWLPcEzHrUW0oA4W0oDUizp71NU7VAeGtAmpu
mlzFXh7ho5cLPgjDnd5bmBnFY04Ocs1qZb89nRsKDyuH2+omA1ibjrLawVNSmQPDKd6i8l15Li920tXrqNTFp0MNm0ulV9HmGuzr
eT3Q2NIbzY0hhuzmah6ZqhDc1+vudpXYuQeGYphS4tENGF6sWE474bxeWYRf5Xz8mxEEeW+ARtDGQSBNIais5ah+h3K7sScWUo7s
KVat0bqQjZc8h1sOFnRSVf157YdSBp0K3W7mkBnMrRDmfTIWcClyG6eFQ1T6jv+aerz8Xg83+lLnel4Okm93FcXny5m2tjtkL4qx
c+/r/8b/i/GRfh//z8NAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMCv5V8LFXt2dpAA1
AAAAABJRU5ErkJggg=="
        # if "만" in cnt_str:
        #   print(cnt_str, float(cnt_str[:-1])*10000, url, img, title)
        # elif "천" in cnt_str:
        #   print(cnt_str, float(cnt_str[:-1])*1000, url, img, title)
        # else :
        #   print(cnt_str, url, img, title)
        print(title, cnt_str, url, img, rdate[:-2])
        movie_list.append([url, title, rdate, cnt_str, img])

    print(len(movie_list))
    df = pd.DataFrame(movie_list, columns=["url","title","rdate","cnt_str","img"])
    print(df.head())
    print(df.info())
    df.to_csv("youtebe_sel_res.csv", index=False)
    return movie_list

mysearch(search_word="지브리 영화")
```