20220516

Selenium을 통해서 DOM요소 추출

메서드

- 여러개 중에 처음 요소를 추출
  - **find_element_by_id(id) : id 속성으로 요소를 하나 추출**
  - **find_element_by_name(name) : name속성으로 요소를 하나 추출**
  - **find_element_by_css_selector(query) : css선택자로 요소를 하나 추출**
  - **find_element_by_xpath(query) : Xpath를 지정해 요소를 하나 추출**
  - **fine_element_by_tag_name(name) : 태그 이름에 해당하는 요소를 하나 추출**
  - **find_element_by_link_text(text) : 링크 텍스트로 요소를 하나 추출**
  - **find_element_by_partial_link_text(text) : 링크의 자식 요소에 포함되어 있는 텍스트 요소를 하나 추출**
  - **find_element_by_class_name(name) : 클래스 이름에 해당하는 요소를 하나 추출**
- 모든 요소를 추출
  - **find_elements_by_css_selector(query) : css선택자로 모든 요소를 추출**
  - **find_element_by_xpath(query) : Xpath를 지정해 모든 요소를 추출**
  - **fine_element_by_tag_name(name) : 태그 이름에 해당하는 모든 요소를 추출**
  - **find_element_by_link_text(text) : 링크 텍스트로 모든 요소를 추출**
  - **find_element_by_partial_link_text(text) : 링크의 자식 요소에 포함되어 있는 모든 텍스트 요소를 추출**
- **NoSuchElementException 이라는 찾고자하는 요소가 없는 경우에 발생함.**

- 네이버 헤드라인 기사 크롤링 연습
  - 라이브러리 호출

```
from selenium import webdriver
```

  - 경로 및 드라이버 설정

```
path = "C:\\webdriver\\chromedriver"
driver = webdriver.Chrome(path)
```

  - URL 설정 및 드라이버로 이동

```
url = "https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=102"
driver.get(url)
```

  - 헤드라인 기사 찾기

```
elements = driver.find_elements_by_class_name("cluster_text_headline")
```

  - 출력

```
for element in elements:
    print(element.text)
```

- 다음 뉴스 헤드라인 크롤링

```
# 라이브러리 호출
from selenium import webdriver

# 드라이버 경로 및 드라이버 실행
path = "C:\\webdriver\\chromedriver"
driver = webdriver.Chrome(path)

# URL 요청
url = "https://news.daum.net/"
driver.get(url)

# 헤드라인 기사 선택자
ul = "body > div.container-doc > main > section > div > div.content-article > div.box_g.box_news_issue > ul > li"
sel = "body > div.container-doc > main > section > div > div.content-article > div.box_g.box_news_issue > ul > li:nth-child(1) > d
sel2 = "body > div.container-doc > main > section > div > div.content-article > div.box_g.box_news_issue > ul > li:nth-child(2) >

test = driver.find_elements_by_css_selector(ul)

li = "body > div.container-doc > main > section > div > div.content-article > div.box_g.box_news_issue > ul"
for i in range(1,len(test)+1):
    head = driver.find_element_by_css_selector(li+f"> li:nth-child({i}) > div > div > strong > a")
    print(head.text)
```

OpenWeatherMap의 API 접속 사용

서울, 도쿄, 뉴욕 날씨를 알아와 출력하기

```
# 라이브러리 설정
import requests
import json

# APIkey 지정...
api_key = "113a4bce75769be2c1564ddf3e9022e2"

# 날씨를 확인할 도시 시정
Cities = ["Seoul,KR" , "Tokyo,JP", "New York,US"]

api = "https://api.openweathermap.org/data/2.5/weather?q={City}&APPID={key}"

# 켈빈 온도를 섭씨로 변환
k2c = lambda k:k - 273.15

# 각 도시의 정보 추출
for name in Cities:
    ## API요청 URL구성
    url = api.format(City=name, key=api_key)

    ## API요청
    re = requests.get(url)

    ## 요청 결과를 Json으로
    data = json.loads(re.text)

    ## 출력 결과 보기
    print("+ 도시 = ", data["name"])
    print("| 날씨 = ", data["weather"][0]["description"])
    print("| 최고 기온 = ", round(k2c(data["main"]["temp_max"]),1))
    print("| 최저 기온 = ", round(k2c(data["main"]["temp_min"]),1))
    print("| 습도 = ", data["main"]["humidity"])
    print("| 기압 = ", data["main"]["pressure"])
    print("| 풍향 = ", data["wind"]["deg"])
    print("| 풍속 = ", data["wind"]["speed"])
    print()


## http://bulk.openweathermap.org/sample/city.list.json.gz
```

- 오픈 API 사이트
  - **https://apistore.co.kr/api/apiList.do**
  - **https://developers.naver.com/main/ 네이버 개발자 센터**
  - **https://developers.kakao.com/ 카카오 개발자 센터**
  - **https://data.go.kr/ 공공데이터포털**

- 기상 데이터 받아오기

```
import requests

url = 'http://apis.data.go.kr/1360000/WthrChartInfoService/getSurfaceChart'
params ={'serviceKey' : 'oafC5Zc4+cIQiw8Lu090+2+0ggaH2nlC1RfBnwhmRGSxTx5BWYgMGwJWg4XFvH1d8tAQDnjIIY0DGK9Aa3RKcQ==', 'pageNo' : '1', 'n

response = requests.get(url, params=params)

print(response.content)
```

다양한 데이터 형식

1. XML(Extensible Markup Language)
   a. 범용적인 데이터 형식으로 W3School 에서 만듬
   계층구조로 데이터를 표현할 수 있다는 특징
   b. 기본 구조 : <요소 속성=”속성값”>내용</요소>
   c. ex)

```
<products type="전자제품">
      <product id="S001" price="45000">SD 카드 128GB </product>
      <product id="S002" price="32000">마우스</product>
</products>
```

- XML 다운로드

```
# xml 다운로드
url = "https://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108"
savename = "forecast.xml"
if not os.path.exists("data/" + savename):
    req.urlretrieve(url,"data/"+savename)

# BeautifulSoup으로 분석

xml = open("data/"+savename,"r",encoding="utf-8")
soup = BeautifulSoup(xml, "html.parser")


# 각 지역별 정보 확인
info = {} ## 지역 정보를 저장할 공간

for location in soup.find_all("location"):
    name = location.find('city').string
    weather = location.find('wf').string
    if not (weather in info):
        info[weather] = []
    info[weather].append(name)

# 각 지역별 날씨를 구분해서 출력
for weather in info.keys():
    print("+", weather)
    for name in info[weather]:
        print("| - ",name)
```

JSON의 구조

- **https://json.org/json-ko.html**

YAML 분석

- 들여쓰기를 사용해서 계층 구조를 표현하는 것이 특징인 데이터 형식
- XML보다 간단하고, 거의 JSON과 비슷하다.
- 파이썬에서 YAML을 다루기 위해서는 PyYAML이라는 모듈을 설치해야 한다.

```
pip install PyYAML
```

- **YAML 사용하기 예제**

```
yaml_str = """
Date: 2022-05-16
PriceList:
    -
        tem_id: 1000
        name: Banana
        color: yellow
        price: 1000
    -
        item_id: 1001
        name: Orange
        color: orange
        price: 1400
    -
        item_id: 1002
        name: Apple
        color: red
        price: 2000
"""

## yaml 분석
data = yaml.full_load(yaml_str)
data2 = yaml.safe_load(yaml_str) ### load() 보안상 이유로 경고가 발생

# 이름 가격 출력
for item in data2['PriceList']:
    print(item['name'],item['price'])
```

- 파이썬 데이터를 yaml데이터로 출력

```
customer = [
    {"name":"InSeong","age":"24","gender":"man"},
    {"name":"gilja","age":"22","gender":"woman"},
    {"name":"sunshin","age":"31","gender":"man"},
    {"name":"bangown","age":"23","gender":"man"}
]

# yaml데이터로 변환
yaml_str1 = yaml.dump(customer)
print(yaml_str1)

# yaml데이터를 파이썬 데이터로 변환
data = yaml.safe_load(yaml_str1)

for name in data:
    print(name['name'])
```

- YAML의 기본은 배열, 해시, 스칼라(문자열, 숫자, 블리언)
  배열을 나타낼 때는 각 행의 앞은 하이픈(-)을 붙인다.
  하이픈 뒤에는 공백이 필요하다.

- 예시(배열)
  - - banana
  - - kiwi
  - - mango
- 예시(중첩 배열), 들여쓰기 다음에 바로 빈 요소를 사용함.
  - - Yellow
    -
      - Banana
      - Orange
    - Red
    -
      - Apple
      - StrawBerry
- 해시표현(”<키>:<값>”)
  - 해시표현시에도 들여쓰기를 사용하여 계층구조를 표현
  - name: Gurum
    age: 4
    color: brown
- YAML의 주석은 “#”
- YAML에서 여러줄 문자열을 저장하는 방법
  - multi-line: |
    I like Banana
    I like Mango
    I like Orange
- YAML은 앵커와 별칭 기능을 제공
  - “&<이름>” 형태로 변수를 선언, “*<이름>” 형태로 참조

```
import yaml
# 문자열로 yaml을 정의
yaml_str = """
color_def:
    - &color1 "#FF0000"
    - &color2 "#00FF00"
    - &color3 "#00FFFF"

# 색 설정 (별칭 테스트)
color:
     title: *color1
     body: *color2
     link: *color3
"""

# YAML 데이터로 분석
data = yaml.safe_load(yaml_str)

# 별칭 확인 테스트
print("title = ", data["color"]["title"])
print("body = ", data["color"]["body"])
print("link = ", data["color"]["link"])
```

Excel 파일 읽어오기

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b7b7ccfa-e8a7-4cb4-9281-14e51caf4bc4/stat_100701.xlsx

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a1d1e393-cbef-4a03-8975-7a49fa31964d/stat_100701_re.xlsx

```
import openpyxl

# 엑셀 파일 열기
filename = "stat_100701.xlsx"
book = openpyxl.load_workbook("data/"+filename)

# 맨 앞에 시트(sheet) 추출
sheet = book.worksheets[0]

# 시트의 각 행을 순서대로 추출
data = []
for row in sheet.rows:
    data.append([
        row[0].value,
        row[7].value
    ])

# 필요없는 줄(헤더, 연도, 계) 제거하기
del data[0:5]
del data[-4:]
#data

# 데이터를 인구 순서로 정렬
data = sorted(data, key=lambda x:int(x[1].replace(',','')))
data

# 하위 5위까지 출력
for i, a in enumerate(data):
    if(i >= 5):
        break
    print(i+1, a[0], int(a[1].replace(',','')))
```

```
import openpyxl

# 엑셀 파일 열기
filename = "stat_100701_re.xlsx"
book = openpyxl.load_workbook("data/"+filename)

# 맨 앞에 시트(sheet) 추출
sheet = book.worksheets[0]

# 시트의 각 행을 순서대로 추출
data = []
for row in sheet.rows:
    data.append([
        row[0].value,
        row[4].value
    ])

# 필요없는 줄(헤더, 연도, 계) 제거하기
del data[0:5]
del data[-4:]
#data

# 데이터를 인구 순서로 정렬
data = sorted(data, key=lambda x:x[1])
data

# 하위 5위까지 출력
for i, a in enumerate(data):
    if(i >= 5):
        break
    print(i+1, a[0], a[1]
```

Excel 데이터 쓰기

```
import openpyxl
```

```
# 엑셀파일 열기
filename = "stat_100701_re.xlsx"
book = openpyxl.load_workbook("data/"+filename)

# 활성화된 시트 추출
sheet = book.active
#sheet

# 서울을 제외한 인구를 구해서 쓰기
for i in range(0,4):
    total = int(sheet[str(chr(i+66))+"5"].value)
    seoul = int(sheet[str(chr(i+66))+"6"].value)
    output = total - seoul
    print(total, seoul)
    print("서울 제외 인구 = ", output)

    # 쓰기
    sheet[str(chr(i+66))+"24"] = output

    cell = sheet[str(chr(i+66))+"24"]

    # 폰트 색상 변경
    cell.font = openpyxl.styles.Font(size=14, color="FF0000")
    cell.number_format = cell.number_format
sheet[str(chr(65))+"24"] = "서울제외인구"

# 엑셀파일 저장하기
filename = "population.xlsx"
book.save("data/"+filename)
```