PYTHON을 이용한 COVID-19 확산현황 공공데이터 크롤러 개발

XEROS

프로그램 개요

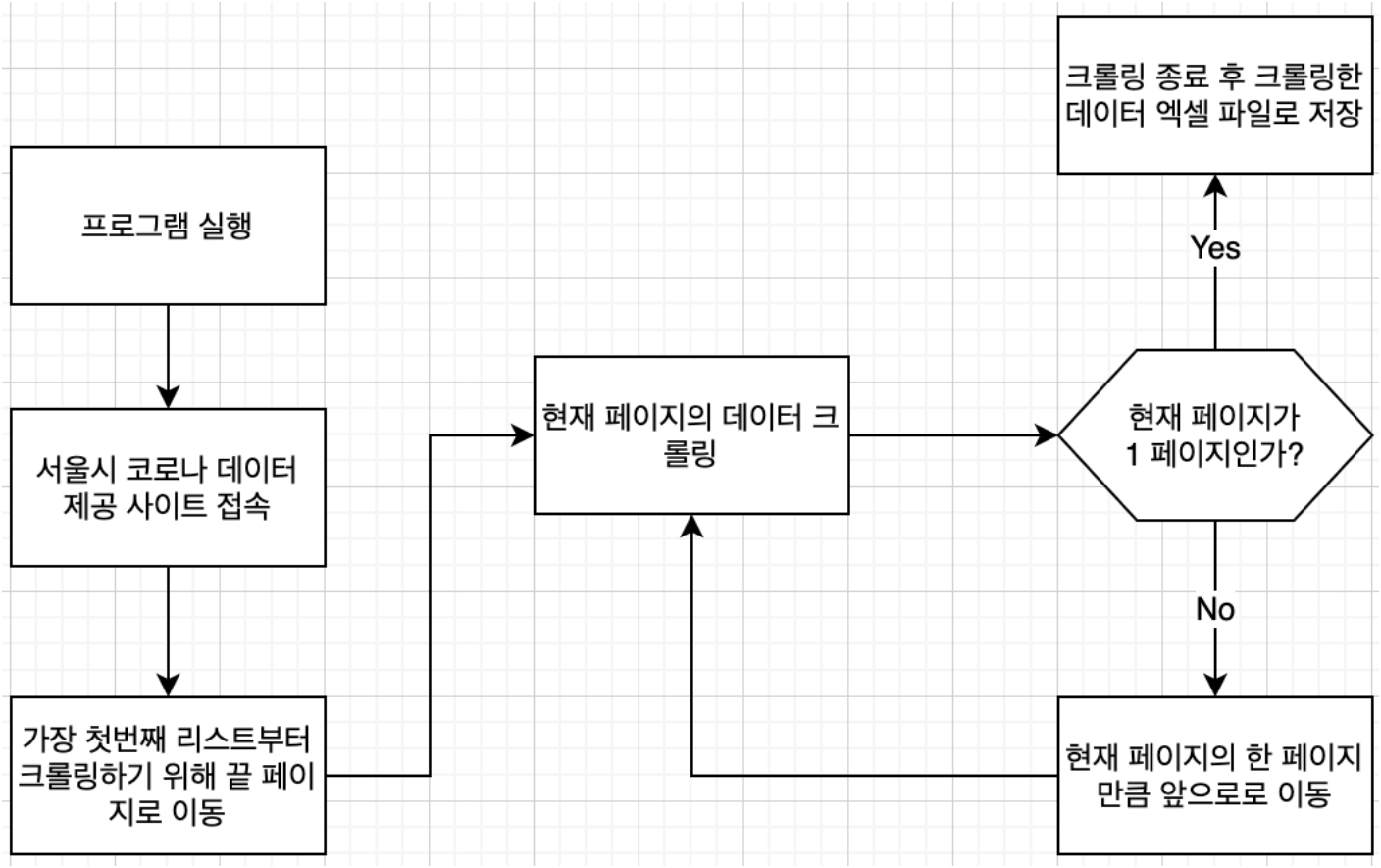
서울시 코로나 확산 현황 공공 데이터 (http://www.seoul.go.kr/coronaV/coronaStatus.do)를 이용하여 엑셀 데이터화하는 프로그램을 구현함

프로그래밍 언어는 Python3를 이용했으며, chrome 버전은 83버전을 이용함 또한 부가적으로 Selenium, bs4, openpyxl 3가지 외부 모듈을 이용해 개발하였음

Selenium 모듈은 웹의 데이터를 읽거나, 자바스크립트로 구현된 웹페이지에서 버튼 클릭등을 위해 사용하였으며, Selenium 모듈은 실행속도가 다소 느리기때문에 크롤러의 전반적인 크롤링 함수는 bs4를 이용해 파싱하는쪽으로 구성함

또한 Selenium과 bs4 모듈을 이용해 파싱한 데이터를 엑셀 파일로 저장하기 위해 openpyxl 모듈을 이용하였음

프로그램 프로세스 모식도



프로그램이 실행되면 먼저 공공데이터 사이트로 접속합니다.

이후 데이터 첫 페이지로 이동합니다.

현재 페이지의 모든 데이터를 크롤링합니다.

이후 현재 페이지의 페이지 번호가 1페이지일 경우 모든 데이터를 크롤링했다고 판단하고 크롤러 프로그램을 종료, 수집한 데이터를 엑셀 파일로 저장합니다.

만약 페이지의 번호가 1페이지가 아닐 경우 한 페이지만큼 앞으로 이동 후 다시 페이지를 크롤링하며, 이 과정을 반복합니다.

기본 프로그램 구성 코드 설명

```
1   import os
2   import bs4
3   from openpyxl import Workbook
4   from selenium import webdriver
```

사용할 모듈을 import합니다.

os모듈은 프로젝트 폴더의 위치 반환, 저장될 엑셀파일의 경로, 여러 경로의 조합을 목적으로 사용합니다.

Bs4모듈은 selenium모듈의 느린 실행속도를 대체하기 위해 사용합니다. 대부분의 데이터는 이 bs4 모듈을 이용하여 크롤링합니다.

Openpyxl 모듈은 수집한 데이터를 엑셀파일로 내보내기 위해서 사용합니다.

Selenium 모듈은 페이지 이동 버튼 클릭 등 웹페이즈의 동적 영역 (자바스크립트등)을 읽어오거나 click 매서드를 사용하기 위해 이용합니다.

엑셀 기초 데이터 설정 및 데이터 리스트 초기화 코드 설명

```
6   # 엑셀 데이터 제어 초기설정
7   write_wb = Workbook()
8   write_ws = write_wb.active
9
10  write_ws["A1"] = "연번"
11  write_ws["B1"] = "환자"
12  write_ws["C1"] = "확진일"
13  write_ws["D1"] = "거주지"
14  write_ws["E1"] = "여행력"
15  write_ws["F1"] = "접촉력"
16  write_ws["G1"] = "조치사항"
17
18  # 데이터 폼 초기화
19  PR = []
20  DT = []
21  AR = []
22  IF = []
23  CT = []
24  AC = []
```

Write_wb 변수에 openpyxl 모듈의 Workbook()을 매칭합니다.

Write_ws 변수에 현재 활성화된 엑셀 sheet를 로드합니다. 앞으로 write_ws를 이용해 엑셀 데이터를 제어합니다.

엑셀 셀 A1, B1, C1, D1, E1, F1, G1에 데이터 타이틀을 지정하여 추가하도록 합니다.

또한 PR, DT, AR, IF, CT, AC 리스트형 변수들을 초기화해줍니다. 추후 이 리스트들에는 크롤링한 데이터들이 카테고리별로 append됩니다.

엑셀 기초 데이터 설정 및 데이터 리스트 초기화 코드 설명

```
26  # 현재 작업디렉토리를 BASE_DIR 변수에 저장
27  BASE_DIR = os.path.realpath(os.path.dirname(__file__))
28  # Selenium 모듈에서 사용할 chrome driver를 로드
29  try:
30      driver = webdriver.Chrome(os.path.join(BASE_DIR, "chromedriver.exe"))
31  except FileNotFoundError:
32      print("크롬 드라이버를 찾을 수 없습니다.")
```

BASE_DIR 변수에 프로젝트 폴더의 절대경로를 지정하도록 설정합니다. 이 과정을 거치는 이유는 프로그램이 실행되었을때 외부 리소스(엑셀 데이터, chrome driver)를 오류없이 제대로 로드하기 위함입니다.

이후 os.path.join 함수를 이용해 BASE_DIR과 “chromedriver.exe”의 경로를 합쳐, chrome driver를 로드합니다.

만약 이 과정에서 FileNotFoundError가 발생하게 될 경우(프로젝트 폴더에 chromedriver.exe가 존재하지 않을 경우) try ~ else절을 이용하여 예외처리함으로써 보다 유연한 에러를 발생시킵니다.

GETDATA 함수 구성

```
35  def getData(pageSource):
36      # 새로 로드된 페이지 내용 로드
37      PR = pageSource.find_all("td", attrs={"data-tit":"환자 번호"})
38      # 확진일
39      DT = pageSource.find_all("td", attrs={"data-tit":"확진일"})
40      # 거주지
41      AR = pageSource.find_all("td", attrs={"data-tit":"거주지"})
42      # 여행력
43      IF = pageSource.find_all("td", attrs={"data-tit":"여행력"})
44      # 접촉력
45      CT = pageSource.find_all("td", attrs={"data-tit":"접촉력"})
46      # 조치사항
47      AC = pageSource.find_all("td", attrs={"data-tit":"조치사항"})
48
49      PR.reverse()
50      DT.reverse()
51      AR.reverse()
52      IF.reverse()
53      CT.reverse()
54      AC.reverse()
55
56      return PR, DT, AR, IF, CT, AC
```

조금 더 효율적인 프로그램 구성을 위해 데이터 파싱 전용 함수인 getData 함수를 만들어 main함수와 구별하였습니다.

getData함수에서는 pageSource(웹페이지 HTML Source)를 인자로 받아 필요한 데이터를 파싱하여 이전에 초기화해두었던 리스트형 변수인 PR, DT, AR, IF, CT, AC 리스트에 저장합니다.

이때 각 데이터들은 “td”태그를 가지며, custom tag로 data-tit를 가진 요소들을 파싱하여 구합니다.

또한 올림차순으로 정렬되어있는 데이터를 내림차순으로 전환하기 위해 reverse매서드를 이용해 정렬하고, 가공이 완료된 데이터들을 반환하며 함수를 종료합니다.

MAIN 함수 구성 1

```
def main():
    # 크롬 창 크기 설정
    driver.set_window_size(1024, 800)
    # 드라이버 로드 대기
    driver.implicitly_wait(1)
    # 서울 코로나 공공데이터 사이트 접속
    driver.get("http://www.seoul.go.kr/coronaV/coronaStatus.do#status_page_top")

    # 첫번째 페이지로 이동
    firstPageBtn = driver.find_element_by_css_selector('#DataTables_Table_0_paginate > span > a:nth-child(7)')
    driver.execute_script("arguments[0].click();", firstPageBtn)
```

Main함수는 실행되면 driver.set_window_size 함수를 통해 1024(x), 800(y)의 윈도우 크기를 가지고 실행됩니다.

또한 정상적으로 드라이버를 로드하기 위해 driver.implicitly_wait(1) 함수를 이용해 1초간 기다립니다.

위 과정이 완료되면 크롤러는 설정해둔 코로나 확산 공공데이터 사이트에 접속합니다.

그리고 보다 쉽게 데이터를 정렬하기 위해 맨 마지막(첫)페이지로 이동합니다.

MAIN 함수 구성 2

```
# 모든 페이지 크롤링
while True:
    # 환자 데이터 파싱
    pageSource = bs4.BeautifulSoup(driver.page_source, "html.parser")
    # 바로 위 pageSource를 getData함수에 넣어서 계산된 값을 data 변수에 넣겠다.
    data = getData(pageSource)

    # getData함수를 이용해 return 받은 데이터를 append
    for i in range(len(data[0])):
        PR.append(data[0][i])
        DT.append(data[1][i])
        AR.append(data[2][i])
        IF.append(data[3][i])
        CT.append(data[4][i])
        AC.append(data[5][i])

    # 모든 페이지를 크롤링했는지 체크
    if int(driver.find_elements_by_class_name("current")[0].text) == 1:
        break
    else:
        # 현재 페이지 크롤링 종료시 이전 페이지로 이동
        beforeBtn = driver.find_elements_by_xpath('//*[@id="DataTables_Table_0_previous"]')
        beforeBtn[0].click()
```

main함수에서는 이전에 선언해두었던 getData함수를 이용하여 크롤링 데이터를 전달받습니다.

여기서 pageSource는 인자값으로 홈페이지의 HTML 영역을 받습니다.

이제 반복문을 이용해 데이터를 다시한번 정렬, 선언해줍니다.

위 과정이 끝났다면, 현재 페이지가 1페이진지 체크하고 만약 1페이지라면 크롤러를 종료하고 아니라면 현재 페이지에서 한 페이지 앞으로 가도록 설정합니다.

MAIN 함수 구성 3

```
94      SN = len(PR)
95
96      # 엑셀 파일 제어
97      count = 2
98      for i in range(SN):
99          write_ws.cell(count, 1, i+1)
100         write_ws.cell(count, 2, PR[i].text)
101         write_ws.cell(count, 3, DT[i].text)
102         write_ws.cell(count, 4, AR[i].text)
103         write_ws.cell(count, 5, IF[i].text)
104         write_ws.cell(count, 6, CT[i].text)
105         write_ws.cell(count, 7, AC[i].text)
106         count += 1
107     write_wb.save(os.path.join(BASE_DIR, "result.xlsx"))
```

모든 데이터를 크롤링하고 무사히 94번 라인으로 이동하게되면, SN(연번) 데이터를 생성하게된다.

그리고 SN만큼 반복문을 실행하는데 바로 이부분이 크롤링한 데이터를 엑셀파일로 저장하는 부분이다.