

[BeautifulSoup 사용 - 1]

WWR - [<https://weworkremotely.com/>]

사이트의 정보를 읽어오는 파이썬 코드를 작성하는 영상으로 시작됨

코드

```
import requests
from bs4 import BeautifulSoup

url = "https://weworkremotely.com/categories/remote-full-stack-programming-jobs"

response = requests.get(url)

response.status_code # 200(성공), 400 같은 코드
```

```
import requests
from bs4 import BeautifulSoup

url = "https://weworkremotely.com/categories/remote-full-stack-programming-jobs"

response = requests.get(url)

response.status_code # 200(성공), 400 같은 코드
```

현재 WWR 사이트에서 얻고자 하는 정보는 <section class="jobs">안에 있는 정보다.

```
<section class="jobs" id="category-2"> == $0
```

이 정보를 받기 위해서는 BeautifulSoup라는 것을 활용한다.

코드

```
'''
BeautifulSoup : 지저분한 html 코드를 정리한다고 이해하기
'''

soup = BeautifulSoup(response.content,
                      "html.parser",
                      )
# BeautifulSoup(넘겨줄 데이터, 넘겨준 데이터 형식/종류)

# jobs = soup.find("section", id="category-2")
jobs = soup.find("section", class_="jobs").find("li")
'''

class는 Python언어로, class를 만들 때 사용하는 것으로 예약되어 있음
그래서 class_로 해줘야 우리가 원하는 html의 class 검색 가능

현재 위 코드는 html코드에서 section에 해당하는 태그의 class_가 "jobs"로 되어 있는
데이터들 중에서 li 태그의 데이터를 찾아 jobs 변수에 저장하라는 의미이다.
'''

print(jobs)
```

```
'''
BeautifulSoup : 지저분한 html 코드를 정리한다고 이해하기
'''

soup = BeautifulSoup(response.content,
                      "html.parser",
                      )
# BeautifulSoup(넘겨줄 데이터, 넘겨준 데이터 형식/종류)

# jobs = soup.find("section", id="category-2")
jobs = soup.find("section", class_="jobs").find("li")
'''

class는 Python언어로, class를 만들 때 사용하는 것으로 예약되어 있음
그래서 class_로 해줘야 우리가 원하는 html의 class 검색 가능

현재 위 코드는 html코드에서 section에 해당하는 태그의 class_가 "jobs"로 되어 있는
데이터들 중에서 li 태그의 데이터를 찾아 jobs 변수에 저장하라는 의미이다.
'''

print(jobs)
```

여러 가지 li중에서 제일 처음 li가 출력되어 선택됨

```
<section class="jobs" id="category-2"> == $0
  <article>
    <h2> ... </h2>
    <ul>
      <style> ... </style>
      <div id="bsa-zone_1724253015566-5_123456"></div>
      <script> ... </script>
      <section class="content-wrap" id="desktop-digest"> ... </section>
      <li id="one-signal-subscription-form" class="inline" style> ... </li>
      <script> ... </script>
      <li class="feature"> ... </li>
      <script> ... </script>
```

```
<li class="inline" id="one-signal-subscription-form" style="display: none;">Get instant
listings in your browser! <a class="inline" href="#" id="wwr-notification-button" style=
"display: none;">Subscribe to Notifications</a></li>
PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace>
```

<li> 데이터 하나가 아니라 여러개를 가져오기 위해서는 find가 아니라 find\_all()이라는 메소드 사용하기

|   |
|---|
| 코드  |
| <pre>jobs = soup.find("section",                   class_="jobs").find_all("li") # find_all()하면 li를 전부 찾음 print(jobs)</pre>   |
| <pre>jobs = soup.find("section",           class_="jobs").find_all("li") # find_all()하면 li를 전부 찾 print(jobs)</pre>  |
| <pre>"company"&gt;Nordhealth&lt;/span&gt;&lt;br&gt;&lt;span class="title"&gt;Senior Full-Stack Engineer&lt;/span&gt;&lt;div class="listing-date p ition-absolute"&gt;&lt;span class="listing-date__date"&gt;30d&lt;/span&gt;&lt;/div&gt;&lt;br&gt;&lt;span class="company"&gt;Full-Time&lt;/span&gt;&lt;sp &gt;&lt;/span&gt;&lt;span class="region company"&gt;Europe Only&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;, &lt;li&gt;&lt;a class="view-all" href="/"&gt;← Back to a jobs&lt;/a&gt;&lt;/li&gt;]</pre> |

2024. 10. 14. (월)

```
jobs = soup.find("section",
                 class_="jobs").find_all("li")[1:-1]
# print(jobs)

for job in jobs:
    title = job.find("span", class_="title")
    region = job.find("span", class_="region")

    print(title, "----", region)
```

```
<span class="title">Technical Lead</span> ---- <span class="region company">Europe Only</span>
<span class="title">Senior Full Stack Engineer</span> ---- <span class="region company">Americas Only</span>
<span class="title">Senior Full Stack Engineer (React Native)</span> ---- <span class="region company">North America Only/Latin America Only/Canada Only</span>
<span class="title">Software Engineer - BRAZIL</span> ---- <span class="region company">Anywhere in the World</span>
<span class="title">Senior Software Engineer</span> ---- <span class="region company">Americas Only</span>
<span class="title">Senior Independent Software Developer ($90-$170/hr)</span> ---- <span class="region company">Anywhere in the World</span>
<span class="title">Full-Stack Ruby on Rails Developer</span> ---- <span class="region company">Europe Only/UK Only/Africa Only</span>
```

위에서 얻은 결과값에서 <span>태그 내부의 텍스트만 얻고 싶을 경우  
아래와 같이 .text를 추가해주면 된다.

```
for job in jobs:
    title = job.find("span", class_="title").text
    region = job.find("span", class_="region").text

    print(title, "----", region)
```

```
Course Development Engineer (Full-Stack Software) ---- Europe Only
Technical Lead ---- Europe Only
Senior Full Stack Engineer ---- Americas Only
Senior Full Stack Engineer (React Native) ---- North America Only/Latin America Only/Canada Only
Software Engineer - BRAZIL ---- Anywhere in the World
Senior Software Engineer ---- Americas Only
Senior Independent Software Developer ($90-$170/hr) ---- Anywhere in the World
Full-Stack Ruby on Rails Developer ---- Europe Only/UK Only/Africa Only
```

글자를 얻음



아래 사진처럼 class="company"로 되어 있는게 3개가 있다.

```
▼ <a href="/remote-jobs/lemon-io-senior-android-developer-start-as
  <span class="company">Lemon.io</span>
  ▶ <object> ... </object>
  <br>
  <span class="title">Senior Android Developer (start - ASAP)</span>
  ▶ <div class="position-absolute listing-date"> ... </div>
  <br>
  <span class="company">Full-Time</span> == $0
  <span>/</span>
  <span class="region company">Latin America Only/Europe Only/UK
</a>
```

```
for job in jobs:
    title = job.find("span", class_="title").text
    # company, position, _ = job.find_all("span", class_="company")
    # _(undersocre)이라는 특수문자를 사용해서 무시한다는 데이터인 것을 알려줌

    company, position, region = job.find_all("span", class_="company")

    company = company.text
    position = position.text
    region = region.text

    print(title, company, position, region, "-----")
```

span태그의 class이름이 company인 데이터들을 모두 찾아서 각각 company, position, region이라는 변수에 저장했다. (.text를 하는 이유는 필요한 텍스트 값만을 저장하기 위함)

실행하면 아래처럼 값이 정상 출력되는 것을 확인할 수 있음

```
Senior Android Developer (start - ASAP) Lemon.io Full-Time Latin America Only/Europe Only/UK Only/Canada
Only/Asia Only/Oceania Only -----
Senior iOS Developer (start - ASAP) Lemon.io Full-Time Latin America Only/Europe Only/UK Only/Canada Only
/Asia Only/Oceania Only -----
Lead Software Engineer Loop HQ Inc. Full-Time USA Only -----
Senior Full Stack Engineer - IMS Discogs Inc Full-Time USA Only -----
Full Stack Shopify App Developer We Work Remotely Full-Time Latin America Only -----
Senior React.js & PHP Developer Lemon.io Full-Time Latin America Only/Europe Only/UK Only/Canada Only/Asi
a Only/Oceania Only -----
Senior Full-Stack Web Developer (.NET/C#/MongoDB/Angular/Redux) with GDPR Experience Secure Privacy Full-
Time Anywhere in the World -----
Full Stack Developer Bootcamp - Job Guaranteed 🚀 Metana - Coding School Full-Time Anywhere in the World -
```

<li> 태그에 저장되어 있는 url 링크가 excel에서도 링크로 유지되어 있기를 원함

```
<li class="feature">
  <div class="highlight-bar"></div>
  <div class="tooltip--flag-logo">...</div>
  <a href="/remote-jobs/lemon-io-senior-ios-developer-start-asap">
</li>
```

```
url = job.find("a") # job에서 <a>태그를 찾아서 url에 저장
if url: # 만약에 url이 있으면 아래 실행, 없다면 실행 안됨
    url = url["href"]
```

```
url = job.find("a")["href"]
```

첫 번째 a태그 값또한 url로 받아옴

```
[{'title': 'Senior Android Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time', 'region': 'Latin America Only/Europe Only/UK Only/Canada Only/Asia Only/Oceania Only', 'url': '/company/lemon-io'}, {'title': 'Senior iOS Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time',
```

그리고 잘못된 값도 url로 받아옴

무슨 말이나 하면 아래 사진을 참고해서 설명하겠음

```
<li class="feature">
  <div class="highlight-bar"></div>
  <div class="tooltip--flag-logo"> == $0
    <a href="/company/lemon-io">
      <div class="flag-logo" style="background-image:url(https://we-wc
      =compress)" loading="lazy" alt="Lemon.io is hiring a remote Seni
    <span class="tooltiptext">
      "View Company Profile"
      ::after
    </span>
  </a>
</div>
  <a href="/remote-jobs/lemon-io-senior-android-developer-start-asap">
```

현재 li에는 2개의 <a>태그가 존재하는 상태임, 근데 내가 받아온 것은 그중에서 첫 번째 <a>값임  
class="tooltip--flag-logo"의 href값을 받아온 상태 -> 두 번째 <a>의 href를 받아오도록 설정 해야함.

전의 `job.find()`코드를 아래처럼 수정하면 “tooltip--flag-logo” 다음의 href 값을 url로 받아올 수 있게 됨

```
url = job.find("div", class_="tooltip--flag-logo").next_sibling("href")
```

이제 정상적으로 원하는 값을 받아올 수 있는걸 확인할 수 있음

```
[{'title': 'Senior Android Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time', 'region': 'Latin America Only/Europe Only/UK Only/Canada Only/Asia Only/Oceania Only', 'url': '/remote-jobs/lemon-io-senior-android-developer-start-asap'}, {'title': 'Senior iOS Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time', 'region': 'Latin America Only/Europe Only/UK Only/Canada Only/Asia Only/Oceania Only', 'url': '/remote-jobs/lemon-io-senior-ios-developer-start-asap'}]
```

# 위에서 추출한 데이터를 딕셔너리 형식으로 만들어서 저장.

```
job_data = {
    "title" : title,
    "company" : company.text,
    "position" : position.text,
    "region" : region.text,
    "url" : f"https://weworkremotely.com{url}"
}
all_jobs.append(job_data) # job_data를 all_jobs List에 저장
```

이제 얻은 url값을 WWR사이트의 주소인 “<https://weworkremotely.com>”을 추가해서 위 사진처럼 만들어주면 접속할 수 있는 링크 형식으로 저장되어 접속할 수 있게 된다.

```
[{'title': 'Senior Android Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time', 'region': 'Latin America Only/Europe Only/UK Only/Canada Only/Asia Only/Oceania Only', 'url': 'https://weworkremotely.com/remote-jobs/lemon-io-senior-android-developer-start-asap'}, {'title': 'Senior iOS Developer (start - ASAP)', 'company': 'Lemon.io', 'position': 'Full-Time', 'region': 'Latin America Only/Europe Only/UK Only/Canada Only/Asia Only/Oceania Only', 'url': 'https://weworkremotely.com/remote-jobs/lemon-io-senior-ios-developer-start-asap'}]
```

← Back to all jobs

See more Full-Stack Programming jobs →

POSTED OCT 11

## Remote Senior Android Developer (start – ASAP)

FULL-TIME FULL-STACK PROGRAMMING LATIN AMERICA ONLY  
EUROPE ONLY UK ONLY CANADA ONLY ASIA ONLY OCEANIA ONLY  
A/B TESTING GIT JAVA RESTFUL SERVICES/APIs UI/UX ENGINEER  
BACKEND FRONT END FULL STACK FULL TIME ENGLISH ANDROID  
KOTLIN \$75,000 - \$99,999 USD



Lemon.io

Top 100

📍 Kyiv, Ukraine  
🌐 Website  
📄 DEI Policy  
📄 Remote Policy  
JOBS POSTED: 67

지금까지의 코드를 정리

```
import requests
from bs4 import BeautifulSoup

url = "https://weworkremotely.com/categories/remote-full-stack-programming-jobs"

response = requests.get(url)

response.status_code # 200(성공), 400 같은 코드

# print(response.content)

'''
BeautifulSoup : 지저분한 html 코드를 정리한다고 이해하기
'''
soup = BeautifulSoup(response.content,
                      "html.parser",
                      )
# BeautifulSoup(넘겨줄 데이터, 넘겨준 데이터 형식/종류)

# jobs = soup.find("section", id="category-2")
jobs = soup.find("section", class_="jobs").find("li")
'''
class는 Python언어로, class를 만들 때 사용하는 것으로 예약되어 있음
그래서 class_로 해줘야 우리가 원하는 html의 class 검색 가능

현재 위 코드는 html코드에서 section에 해당하는 태그의 class_가 "jobs"로 되어 있는
데이터들 중에서 li 태그의 데이터를 찾아 jobs 변수에 저장하라는 의미이다.
'''

# print(jobs)
'''
출력하면 첫번째 li만 출력 됨
<li class="inline" id="one-signal-subscription-form" style="display: none;">Get
instant listings in your browser! <a class="inline" href="#" id="wvr-notification-button"
style="display: none;">Subscribe to Notifications</a></li>
'''

jobs = soup.find("section",
                 class_="jobs").find_all("li")[1:-1] # find_all()하면 li를 전부 찾음
# print(jobs)
```



```

# for job in jobs:
#     title = job.find("span", class_="title").text
#     region = job.find("span", class_="region").text
#     companies = job.find_all("span", class_="company")

#     print(title, "-----", region)

'''
letters = ["a", "b", "c"]
a = letters[0]; b = letters[1]; c = letters[2]

위의 것은 shortcut할 수 있음
a, b, c = letters

x, y = letters 하면 작동하지 않음
letters에 있는 값의 갯수와 변수 갯수가 같아야함
'''

all_jobs = []

for job in jobs:
    title = job.find("span", class_="title").text
    # company, position, _ = job.find_all("span", class_="company")
    # _(underscore)이라는 특수문자를 사용해서 무시한다는 데이터인 것을 알려줌

    company, position, region = job.find_all("span", class_="company")

    url = job.find("div", class_="tooltip--flag-logo").next_sibling["href"]

    # 위에서 추출한 데이터를 딕셔너리 형식으로 만들어서 저장.
    job_data = {
        "title" : title,
        "company" : company.text,
        "position" : position.text,
        "region" : region.text,
        "url" : f"https://weworkremotely.com{url}"
    }
    all_jobs.append(job_data) # job_data를 all_jobs List에 저장

print(all_jobs)

# Pagination / 2024.10.17.(Jueves)

```

```
'''
```

웹사이트의 page가 2개 이상 있다고 할때, 첫번째를 제외한 페이지(2번째 페이지)를 클릭하면 url에 page=2 추가 됨을 확인할 수 있다.

오늘은 이러한 페이지에 접근하는 것을 진행한다.

```
'''
```

웹 스크랩하려는 화면에서 넘어갈 수 있는 Page가 몇 개 있는지 확인하기 위해 다음과 같은 코드를 작성한다.

```
response = requests.get("https://weworkremotely.com/remote-full-time-jobs?page=1")
soup = BeautifulSoup(response.content, "html.parser")
buttons = soup.find("div", class_="pagination").find_all("span", class_="page")
```

```
# scrape_page 함수를 몇번이나 호출해야 하는지 알아내는 코드
```

```
# 맨 처음, 페이지 갯수를 알고 싶어하는 곳의 첫번째 페이지의 URL을 가져온다.
```

```
response = requests.get("https://weworkremotely.com/remote-full-time-jobs?page=1")
```

```
soup = BeautifulSoup(response.content, "html.parser")
```

```
buttons = soup.find("div", class_="pagination").find_all("span", class_="page")
```

결과는 아래처럼 잘 나온다.

```
[<span class="page current"> 1 </span>, <span class="page"><a href="/remote-full-time-jobs?page=2" rel="
ext">2</a></span>, <span class="page"><a href="/remote-full-time-jobs?page=3">3</a></span>, <span class=
page"><a href="/remote-full-time-jobs?page=4">4</a></span>]
```

```
PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders> █
```

```
response = requests.get("https://weworkremotely.com/remote-full-time-jobs?page=1")
```

```
soup = BeautifulSoup(response.content, "html.parser")
```

```
buttons = len(soup.find("div", class_="pagination").find_all("span", class_="page"))
```

```
# print(buttons) # 출력 결과는 4가 나옴
```

```
Scraping https://weworkremotely.com/remote-full-time-jobs?page=1
Scraping https://weworkremotely.com/remote-full-time-jobs?page=2
Scraping https://weworkremotely.com/remote-full-time-jobs?page=3
Scraping https://weworkremotely.com/remote-full-time-jobs?page=4
169
```

```

all_jobs = []

def scrape_page(url):
    print(f"Scrapping {url}...")

    response = requests.get(url)

    soup = BeautifulSoup(response.content,
                           "html.parser",
                           )

    jobs = soup.find("section", class_="jobs").find_all("li")[1:-1] # find_all()하면 li를 전부 찾음

    for job in jobs:
        title = job.find("span", class_="title").text
        # company, position, _ = job.find_all("span", class_="company")
        # _(underscore)이라는 특수문자를 사용해서 무시한다는 데이터인 것을 알려줌

        company, position, region = job.find_all("span", class_="company")

        url = job.find("div", class_="tooltip--flag-logo").next_sibling["href"]

        # 위에서 추출한 데이터를 딕셔너리 형식으로 만들어서 저장.
        job_data = {
            "title" : title,
            "company" : company.text,
            "position" : position.text,
            "region" : region.text,
            "url" : f"https://weworkremotely.com{url}"
        }
        all_jobs.append(job_data) # job_data를 all_jobs List에 저장

```

def scrape\_page()를 만들어서 이전 코드를 넣음. (url에서 페이지 원하는 값을 읽는 코드를 말함)

### 지금까지의 코드 정리

# Pagination / 2024.10.17.(Jueves)

```

import requests
from bs4 import BeautifulSoup

all_jobs = []

def scrape_page(url):
    print(f"Scrapping {url}...")

    response = requests.get(url)

    soup = BeautifulSoup(response.content,
                           "html.parser",
                           )

```

```
jobs = soup.find("section", class_="jobs").find_all("li")[1:-1] # find_all()하면 li를 전부 찾음
```

```
for job in jobs:
```

```
    title = job.find("span", class_="title").text
```

```
    # company, position, _ = job.find_all("span", class_="company")
```

```
    # _(underscore)이라는 특수문자를 사용해서 무시한다는 데이터인 것을 알려줌
```

```
    company, position, region = job.find_all("span", class_="company")
```

```
    url = job.find("div", class_="tooltip--flag-logo").next_sibling["href"]
```

```
    # 위에서 추출한 데이터를 딕셔너리 형식으로 만들어서 저장.
```

```
    job_data = {
```

```
        "title" : title,
```

```
        "company" : company.text,
```

```
        "position" : position.text,
```

```
        "region" : region.text,
```

```
        "url" : f"https://weworkremotely.com{url}"
```

```
    }
```

```
    all_jobs.append(job_data) # job_data를 all_jobs List에 저장
```

```
def get_pages(url):
```

```
    # scrape_page 함수를 몇번이나 호출해야 하는지 알아내는 코드
```

```
    # 맨 처음, 페이지 갯수를 알고 싶어하는 곳의 첫번째 페이지의 URL을 가져온다.
```

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

```
    soup = BeautifulSoup(response.content, "html.parser")
```

```
    return len(soup.find("div", class_="pagination").find_all("span", class_="page"))
```

```
total_pages = get_pages("https://weworkremotely.com/remote-full-time-jobs?page=1")
```

```
# pagination div의 갯수에 따라 유동적인 for_loop -> range를 통해서 진행
```

```
for x in range(total_pages):
```

```
    url = f"https://weworkremotely.com/remote-full-time-jobs?page={x+1}"
```

```
    scrape_page(url)
```

```
...
```



```
range를 통해서 response에서 얻은 페이지 갯수에 따라 유동적으로 반복 횟수가 변하는 for문
url = f"https://weworkremotely.com/remote-full-time-jobs?page={x+1}"을 통해서
페이지 url 완성
```

```
하지만, 이러한 형식은 url에서 page=() 형식으로 되어 있을 때 가능
거의 대부분의 웹사이트는 페이지를 query argument로 처리함
'''

print(len(all_jobs))
```

---

### [Code Challenge]

remote 웹사이트에서 request.get(url) 해서 status code를 보면 접근 차단을 당한다.

remote 웹사이트에 가서 <검사>-<Network>-<remote-flutter-jobs>-<Request Header> 순서로 찾아간다.

Request Header : 우리 브라우저가 서버에 보내는 정보

Request Header에서 User-Agent를 찾는다.

User-Agent : 헤더 / 브라우저가 서버로 전송하는 정보라 할 수 있음

이 정보를 복사해서 사용하면 “쏘지마세요, 저는 일반인입니다.” 같은 느낌으로 말할 때 “아군이군” 하는 느낌이  
라 할 수 있다.

그러나 어떻게 해서도 403 에러가 발생함

그래서 나머지 강의를 듣고 난 다음에 해결해보기로 결정!

일단 해결 하신분 코드를 살짝 가져옴

위 사진의 주인공의 코드

```
from bs4 import BeautifulSoup
import csv
from playwright.sync_api import sync_playwright
import time

class remoteok_scraping:

    def __init__(self, keyword):
        self.keyword=keyword
        self.jobs_list = []
```

def whole\_page\_scraping(self): # 강의 시점과 다르게 동적 환경으로 바뀌어서 playwright를 써야함.

<https://gist.github.com/Srable666/d4890d3c2f66139470ce9733766cbdc4>

print(f'Whole {self.keyword} page scraping.') # 진행상황 브리핑

현재 시점에 remoteok가 동적 페이지로 바뀐 후라서, 결국 몇강까지 듣고서야

완성했네요. 더 적용하고 싶은 아이디어는 있는데, 아직 실력이 부족해서 지금은 이걸로

만족하겠습니다.

p.sync\_playwright().start()

browser=p.chromium.launch()

page=browser.new\_page()

ghvnddl123 3주일 전

import requests

# User-Agent를 추가적인 헤더 설정(해당 페이지가 헤더 설정 없으면 접근을 막음)

page.set\_extra\_http\_headers({

Job\_List=[]

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36',

def \_init\_(self,url):

self.Accept-Language: 'en-US,en;q=0.9'

self.response = requests.get(url, headers={'User-Agent': 'Mozilla/5.0

(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

}) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0'))

soup = BeautifulSoup(self.response.content, "html.parser")

jobs = soup.find("table", id = "jobsboard").find\_all("tr", class\_ = "job")

page.goto(f'https://remoteok.com/remote={self.keyword}-jobs')

time.sleep(3) # 혹시 모르니 첫 로딩 대기시간 설정

# 스크롤을 더이상 내릴 수 없을 때까지 내리는 코드도 짜봤는데, python이나 golang이 경우 1년 이상 지난 게시물까지 전부 로딩되는걸 확인하고 아래 코드로 최종 결정.

for i in range(4): # 스크롤 4번만 내리면 내 컴퓨터 기준 120개 정도의 게시물이 로딩됨.

page.keyboard.down('End')

time.sleep(5)

content = page.content() # 전체 로딩 완료 후 페이지 내용 가져오기

browser.close()

p.stop()

self.content=content

print(f'Whole {self.keyword} page scraping finish.') # 진행상황 브리핑

def job\_list\_output(self):

print(f'{self.keyword} job list making...') # 진행상황 브리핑

soup=BeautifulSoup(self.content, 'html.parser')

jobsboard=soup.find('table', id='jobsboard')

).find\_all('td', class\_='company position company\_and\_position')[1:]

for job in jobsboard:

```

url=f'https://remoteok.com{job.find('a')['href']}'
title=job.find('h2', itemprop='title').text.strip()
location_check = job.find('div', class_='location') # 이렇게만 가져오면 location_tooltip,
location_tooltip-set 까지 같이 가져오짐. 필터링 필요.
if location_check is None: # 내용이 없어서 None 으로 불러와진 경우 편집
    location='정보 없음'
elif 'tooltip' in location_check.get('class', []): # 'tooltip'이 class 속성에 포함되어 있는지
확인 후 편집
    location='정보 없음'
elif 'tooltip-set' in location_check.get('class', []): # 'tooltip-set'이 class 속성에 포함되
어 있는지 확인 후 편집
    location='정보 없음'
else:
    location=location_check.text.strip()

job_data={
    'title': title,
    'location': location,
    "url": f'https://remoteok.com{url}'
}
self.jobs_list.append(job_data)

file=open(f'remoteok_{self.keyword}_jobs_list.csv', 'w', encoding='utf-8-sig')
writer=csv.writer(file)
writer.writerow(
    [
        'Title',
        'Location',
        'Link'
    ]
)

for job in self.jobs_list:
    writer.writerow(job.values())
file.close()

print(f'{self.keyword} job list making finish.') # 진행상황 브리핑
print(f'정리된 {self.keyword} job 수는 {len(self.jobs_list)} 입니다. 자세한 내용은 csv 파일을 확
인해주세요.') # 결과 확인용 메시지

keywords=[
    'flutter',
    'python',

```

```
'golang',  
]  
  
for i in range(len(keywords)):  
    keyword=remoteok_scraping(keywords[i])  
    keyword.whole_page_scraping()  
    keyword.job_list_output()
```

## [Playwright]

이전까지의 request는 정적 페이지를 웹 스크랩하는데 적합했다.  
그러나 이제 동적으로 하려고 한다면 playwright를 사용한다고 한다.

그래서 설치는 아래와 같다.

```
pip install pytest-playwright  
playwright install
```

설치하는데 뭔가 진짜 웃기고 귀엽게 생김

```
PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders> playwright install  
Downloading Chromium 129.0.6668.29 (playwright build v1134) from https://playwright.azureedge.net/builds/  
chromium/1134/chromium-win64.zip  
139 MiB [=====] 100% 0.0s  
Chromium 129.0.6668.29 (playwright build v1134) downloaded to C:\Users\jgy09\AppData\Local\ms-playwright\  
chromium-1134  
Downloading FFMPEG playwright build v1010 from https://playwright.azureedge.net/builds/ffmpeg/1010/ffmpeg  
-win64.zip  
1.3 MiB [=====] 100% 0.0s  
FFMPEG playwright build v1010 downloaded to C:\Users\jgy09\AppData\Local\ms-playwright\ffmpeg-1010  
Downloading Firefox 130.0 (playwright build v1463) from https://playwright.azureedge.net/builds/firefox/1  
463/firefox-win64.zip  
84.6 MiB [==== ] 17% 28.1s
```

playwright를 사용하여 스크린샷 찍기

# 이전까지의 request는 정적인 페이지에 적합했으나 이제 동적(스크롤 움직임 등)을 해야하는  
# 사이트에는 playwright라는 것을 사용해서 할 듯함

```
from playwright.sync_api import sync_playwright  
  
p = sync_playwright().start() # playwright가 초기화 됨  
  
browser = p.chromium.launch(headless=False) # 브라우저 : 크롬  
  
page = browser.new_page() # 브라우저에 새로운 탭 생성  
  
page.goto("https://www.wanted.co.kr")
```



```
page.screenshot(path="screenshot.png") # 페이지 스크린샷 하기 (path에 그냥 이름만 적어주면 저 이름으로 저장됨)
```

```
# 저장 위치는 실행한 위치(PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders\Python_Web>)라면
```

```
# PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders\Python_Web>에 저장됨
```

...

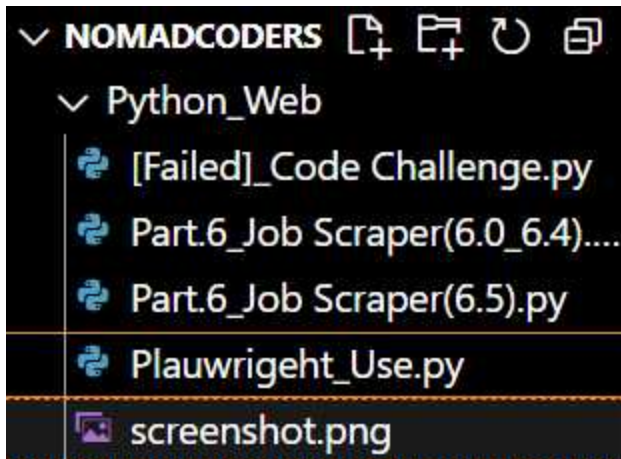
현재 위 코드를 실행해도 브라우저 스크린 샷은 생성되어도 해당 브라우저를 볼 수 없는 상태임  
-> headless mode 하고 한다.

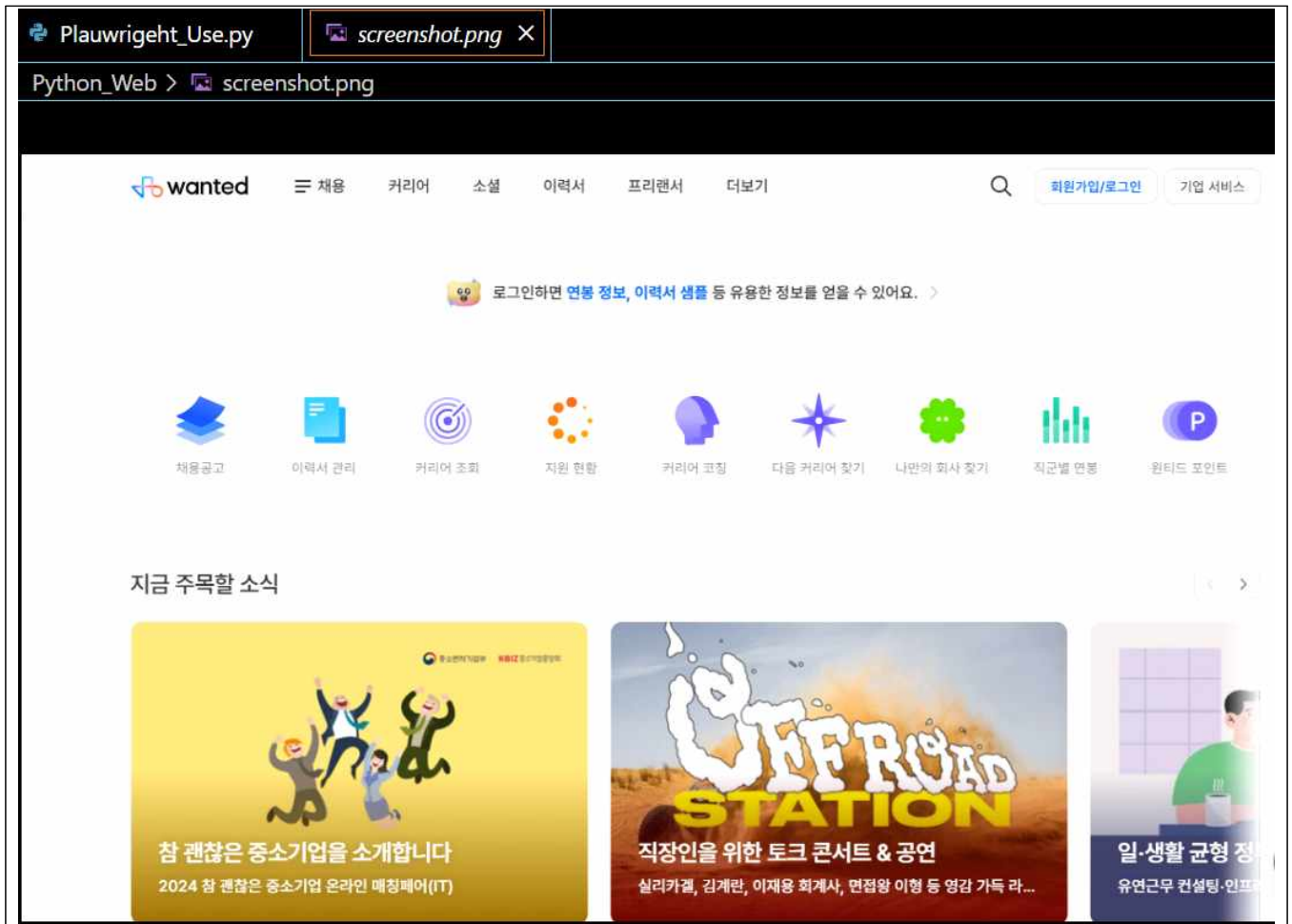
playwright는 기본적으로 브라우저를 보여주지 않기 때문에 여러분이 볼 수 없다고,  
그저 작업을 시작하고 우리가 원하는걸 실행 함. playwright는 우리에게 브라우저 on/off를  
하라 하지 않음 -> headless mode

```
browser = p.chromium.launch(headless=True)가 기본형태라 그렇다고 함  
이것을 False로 하면 화면에 보여짐
```

...

실행 결과





[Keyword Arguments - p.chromium.launch(headless=False)의 headless를 설명하기 위함]

```
def plus(a, b, c, d, e):  
    return a+b  
  
# plus(1, 2) # 1과 2는 positional argument  
...  
왜 positional이냐면 인자값의 위치가 영향을 가지기 때문임  
plus(1, 2)에서 1은 첫번째 위치에 있으니 a로, 2는 두번째 있어서 b로 인식됨  
  
하지만 인자가 많아질 경우에는 넣어주는 인자값이 무엇인지 헷갈릴 수 있음  
그래서 positional argument는 인자가 2개 있을때 사용하며 인자값이 많을 경우에는 사용 X  
...  
  
plus(a=1, b=2, c=False, d=[1, 2], e="hello")
```

```
'''
```

같이 메소드의 이름을 직접적으로 명시해주면 헛갈리지 않고 사용할 수 있으며,  
위치와 상관 없이 아래 처럼 작성할 수 있다.

```
'''
```

```
plus(d=[1, 2], e="hello", a=1, b=2, c=False)
```

```
'''
```

앞서 plus(1, 2)는 순서가 중요했지만 이제 plus 메소드의 인자 값의 이름을 명시해서  
값을 주었으므로 순서에 상관없어졌다.

우리는 이것을 Keyword Arguments라고 부른다.

앞서 browser = p.chromium.launch(headless=True)의  
headless는 keyword argument이다.

R언어의 rep처럼 times를 적어주는 것과 적어주지 않는 것의 차이? 라고 보면 편안하다.

rep(c(1, 2), 4)를 하면 (1, 2)를 총 4번 반복해주며

1 1 1 1 2 2 2 2 형식으로 하고 싶으면

rep(c(1, 2), each=4)로 해주어서 인자값을 지정해주는 것과 비슷하다.

<R언어로 든 예시>

```
rep(c(1,2), 4) # 결과값 : 1 2 1 2 1 2 1 2 / 그냥 4를 적으면 times로 인식됨
```

```
rep(c(1,2), times=4) # 결과값 : 1 2 1 2 1 2 1 2
```

```
rep(c(1,2), each=4) # 결과값 : 1 1 1 1 2 2 2 2
```

```
rep(c(1, 2), 4, 4) # 결과값 : 1 2 1 2
```

```
rep(c(1, 2), 4, , 4) # 결과값 : 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2
```

```
rep(c(1, 2), length.out=6, each=2, times=3) # 결과값 : 1 1 2 2 1 1
```

인자이름을 적어주면 헛갈리지 않고 명령어가 명령을 수행함을 확인할 수 있음

이 처럼 keyword argument를 사용하면 순서 상관 없이 실행 가능하며, 인자가 많아도 헛갈리지 않음

단, 기억할 사항은 keyword argument를 사용했다면 positional argument를 사용할 수 없다는 것

아래 작성된 코드가 keyword로 시작하면 positional을 사용할 수 없음을 잘 보여줌

```
'''
```

```
# plus(e=1, c=2, b=False, d=[1, 2], "hello") # 마지막 "hello"부분에서 에러 발생함
```

```
'''
```

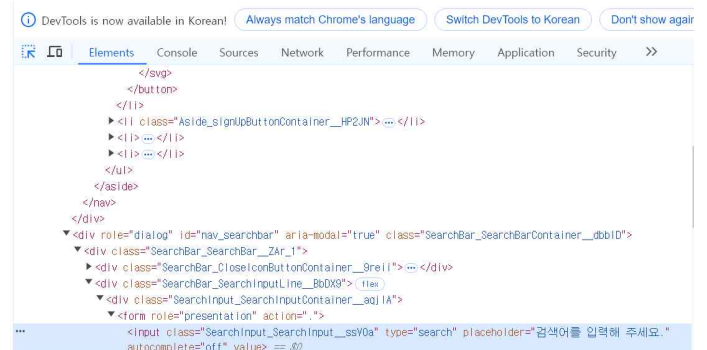
하지만 positional argument로 시작했고 뒤에 만약 keyword argument를 사용했다면 정상작동

R의 rep(c(1, 2), length.out=6, each=2, times=3) 명령어 처럼

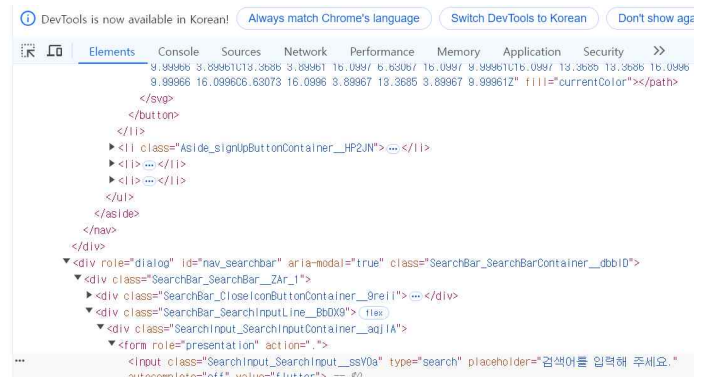
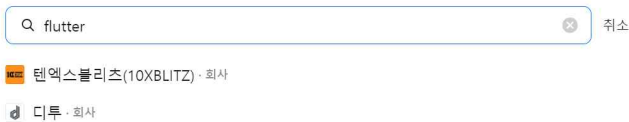
맨 처음 시작은 positional로 했어도 나머지는 keyword로 작성함.

```
'''
```

```
plus(1, c=2, b=False, d=[1, 2], e="hello") # 정상적으로 작동함. 문제 없음
```



위 사진의 검색창에 “flutter”를 입력하면 input의 value에 값이 추가되는 것을 확인할 수 있음



위 상황을 이용해서 input을 찾은 다음 fill을 통해서 원하는 글자를 채워 주면 됨

```
page.get_by_placeholder("검색어를 입력해 주세요.").fill("flutter")
# 위 명령어가 실행되면 html에서 placeholder 값이 "검색어를 입력해 주세요."를 찾아온 다음
# flutter이라는 값을 넣어준다.
```

```
time.sleep(5) # 대기함
```

```
page.keyboard.down("Enter") # Enter을 누름
```

```
time.sleep(5) # 대기함
```

```
page.click("a#search_tab_position") # <a>의 id가 search_tab_position를 찾아서 클릭함
```

```
time.sleep(5) # 대기함
```

```
for x in range(4): # 4번 정도 스크롤을 아래로 내리는 이벤트가 발생함
    page.keyboard.down("End")
```



```
time.sleep(5)
print(page.content())
print(page.content())를 통해서 수 많은 content 값을 얻을 수 있음
```

```
43 for x in range(4): # 4번 정도 스크롤을 아래로 내리는 이벤트가 발생함
44     page.keyboard.down("End")
45
46     time.sleep(5)
47
48     print(page.content())
49
50
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS Python + - [ ] [ ] ... ^ x

```
= "join-ad-interest-group" data-tagging-id="AW-850861095" data-load-time="1729259537749" height="0" width="0" src="https://td.doubleclick.net/td/rul/850861095?random=1729259537746&cv=11&fst=1729259537746&fmt=3&bg=ffffff&guid=ON&async=1&gtm=45be4ah0v9182482305z8810020603za201zb810020603&mp;gcd=13131313111&dma=0&tag_exp=101533421~101686685~101823847&u_w=1280&u_h=720&url=https%3A%2F%2Fwww.wanted.co.kr%2Fsearch%3Fquery%3Dflutter%26tab%3Dposition&hn=www.googleadservices.com&frm=0&tiba=flutter%20%7C%20%EC%9B%90%ED%8B%B0%EB%93%9C%20%ED%86%B5%ED%95%A9%EA%B2%80%EC%83%89&mp;npa=0&pscdl=noapi&uid=306278810.1729259513&uaa=x86&uab=64&uafvl=Chromium%3B129.0.
```

csv 파일을 열기! + 쓰기!

```
file = open("jobs.csv", "w", encoding="utf-8")
writer = csv.writer(file)
```

...

open해서 jobs.csv 파일을 여는데, 이때 해당 파일이 없으면 파일을 생성한 다음 열어준다. 그리고 열린 파일의 포인트? 정보는 file이라는 변수에 저장됨. 현재 열린 파일은 쓰기 형식으로 열렸다는 것을 확인할 수 있다.

writer에 csv.writer(file)을 해주었는데, 이것은 csv 파일에 행을 추가할 수 있게 해줌. -> 데이터 쓰기!

...

```
writer.writerow(
    [
        "Title",
        "company_Name",
        "reward",
        "Link"
    ]
)
```

```
for job in jobs_db:
    writer.writerow(job.values())
```

| 1  | title,company_Name,reward,Link   |
|----|--|
| 2  |  |
| 3  | React Typescript / Flutter 개발자,에이로직,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/226170">https://www.wanted.co.kr/wd/226170</a> |
| 4  |  |
| 5  | 모바일(Flutter) 개발자,심버스,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/249300">https://www.wanted.co.kr/wd/249300</a>                |
| 6  |  |
| 7  | Flutter 개발자 (5년 이상),알서포트(Rsupport),합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/240100">https://www.wanted.co.kr/wd/240100</a>  |
| 8  |  |
| 9  | Flutter/AOS 앱 개발자,에임(AIM),합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/2448">https://www.wanted.co.kr/wd/2448</a>               |
| 10 |  |
| 11 | Flutter 앱 개발자 (3년 이상),에픽코퍼레이션,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/243987">https://www.wanted.co.kr/wd/243987</a>       |
| 12 |  |
| 13 | 신규 플랫폼 Flutter 개발자(3년 이상),포캐스크이엔엠,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/243231">https://www.wanted.co.kr/wd/243231</a>   |
| 14 |  |
| 15 | Flutter 개발자 (FE),위이,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/243102">https://www.wanted.co.kr/wd/243102</a>                 |
| 16 |  |
| 17 | [계약직] Flutter / node.js 개발자,인절미,합격보상금 100만원, <a href="https://www.wanted.co.kr/wd/238403">https://www.wanted.co.kr/wd/238403</a>     |
| 18 |  |
| 19 |  |

위의 playwright를 사용해서 저번에 실패했던 CodeChallenge를 다시 시도했다!

[1단계] : playwright를 통해서 동적으로 페이지를 읽고 안의 내용을 가져올 수 있는지 확인하기!

```

from playwright.sync_api import sync_playwright
import time
from bs4 import BeautifulSoup
import csv

def keywordJob(keyWordList):
    # 동적으로 읽어야 읽히는 상황
    for word in keywords:
        p = sync_playwright().start() # playwright 시작!

        browser = p.chromium.launch(headless=False) # 브라우저 : 크롬

        page = browser.new_page() # 브라우저에 새로운 탭 생성

        page.goto(f"https://remoteok.com/remote-{word}-jobs") # keywords에 있는 직업을 검색!

        time.sleep(5)

        content = page.content()

```

```
p.stop()
```

```
print(content)
```

```
keywords = [  
    'flutter',  
    'python',  
    'golang'  
]
```

```
keywordJob(keywords)
```

값을 잘 읽어오는 것을 확인할 수 있음!

사이트>

Remote Flutter Jobs with great

remoteok.com/remote-flutter-jobs

remote | Flutter Jobs

find a remote  
work from anywhere

Q engineer|

Hiring for a Remote Flutter position? [Post a job](#) on the

```

<div class="END-OF-FILE"></div>

<div class="runtime" style="">
    255ms
</div>
<div class="END-OF-FILE"></div><div class="tooltip-ui" style="display: none;"></div><div><div class="gr
ay: block; transition: right 0.3s; bottom: 14px; right: -186px; box-shadow: gray 0px 0px 5px; border-ra
sentation" name="a-5fchms5041tf" frameborder="0" scrolling="no" sandbox="allow-forms allow-popups allow
er-activation" src="https://www.google.com/recaptcha/api2/anchor?ar=1&k=6Ld_BecUAAAAAxxzQwBa6NY3-m5
cb=uh424xvxfnm"></iframe></div><div class="greaptcha-error"></div><textarea id="g-recaptcha-response-
rgb(193, 193, 193); margin: 10px 25px; padding: 0px; resize: none; display: none;"></textarea></div><if
AAAAAxxzQwBa6NY3-m5RDmjVY_ScSww&onload=recaptchaCallback" async="" defer=""></script></html>

```

[2단계] : 읽은 값들에서 필요로 한 데이터만을 뽑아내기

```

from playwright.sync_api import sync_playwright
import time
from bs4 import BeautifulSoup
import re
import csv

def keywordJob(keyWord):
    # 동적으로 읽어야 읽히는 상황
    p = sync_playwright().start() # playwright 시작!

    browser = p.chromium.launch(headless=False) # 브라우저 : 크롬

    page = browser.new_page() # 브라우저에 새로운 탭 생성

    page.goto(f"https://remoteok.com/remote-{keyWord}-jobs") # keywords에 있는 직업을 검색!

    time.sleep(5)

    content = page.content()

    p.stop()

    return content

# keywords = [
#     'flutter',
#     'python',
#     'golang'
# ]

```



```

keywords = ['flutter']

for key in keywords:
    jobs_content = keywordJob(key) # keywords의 검색 결과를 content에 저장

soup = BeautifulSoup(jobs_content, "html.parser")

jobsboard=soup.find('table', id='jobsboard'
                    ).find_all('td', class_='company position company_and_position')[1:]

all_jobs = []

for job in jobsboard:
    link = f"https://remoteok.com{job.find('a')['href']}"
    title = job.find("h2", itemprop="title").text.strip()
    company = job.find("h3", itemprop="name").text.strip()

    location_check = job.find('div', class_='location') # 이렇게만 가져오면 location_tooltip,
    location_tooltip-set 까지 같이 가져오짐. 필터링 필요.
    if location_check is None: # 내용이 없어서 None 으로 불리워진 경우 편집
        location='정보 없음'
    elif 'tooltip' in location_check.get('class', []): # 'tooltip'이 class 속성에 포함되어 있는지 확인 후
    편집
        location='정보 없음'
    elif 'tooltip-set' in location_check.get('class', []): # 'tooltip-set'이 class 속성에 포함되어 있는지
    확인 후 편집
        location='정보 없음'
    else:
        location=location_check.text.strip()

    pay = job.find("div", class_="location tooltip", title=re.compile("No salary data published"))

    if pay:
        pay = pay.get_text(strip=True) # 텍스트만 가져오고 양쪽 공백을 제거
        print(pay) # 💵 $60k - $130k*
    else:
        pya = "None" # 값이 없을 경우!

    job = {
        "title" : title,
        "company" : company,

```

```

"location" : location,
"pay" : pay,
"link" : link
}
all_jobs.append(job)
print(all_jobs)

```

잘 된다!!

```

{'title': 'Kotlin Multiplatform Engineer', 'company': 'Trust Wallet', 'location': '정보 없음', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-kotlin-multiplatform-engineer-trust-wallet-955235'}, {'title': 'Kotlin Multiplatform Engineer', 'company': 'Trust Wallet', 'location': '정보 없음', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-kotlin-multiplatform-engineer-trust-wallet-889551'}, {'title': 'Senior Mobile Engineer', 'company': 'Fujitsu Launchpad', 'location': '정보 없음', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-mobile-engineer-fujitsu-launchpad-870219'}, {'title': 'Senior Engineering Manager Mobile Bengaluru India', 'company': 'Degreed', 'location': 'IN India', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-engineering-manager-mobile-bengaluru-india-degreed-819691'}, {'title': 'SR Desenvolvimento Flutter', 'company': 'Illa', 'location': 'BR Brazil', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-sr-desenvolvimento-flutter-111a-886242'}, {'title': 'Pl Desenvolvimento Flutter', 'company': 'Illa', 'location': 'BR Brazil', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-pl-desenvolvimento-flutter-111a-886241'}, {'title': 'Sr Flutter Engineer', 'company': 'Illa', 'location': 'BR Brazil', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-sr-flutter-engineer-111a-693865'}, {'title': 'Senior Mobile Engineer', 'company': 'Glooko', 'location': 'HR Croatia', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-mobile-engineer-glooko-643158'}, {'title': 'Senior Mobile Engineer Flutter', 'company': 'Transcarent', 'location': 'US United States', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-mobile-engineer-flutter-transcarent-643158'}, {'title': 'Senior Flutter Engineer', 'company': 'XMTP', 'location': '정보 없음', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-flutter-engineer-xmtp-638496'}, {'title': 'Senior Software Developer I', 'company': 'Spring Health', 'location': 'US United States', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-software-developer-i-spring-health-15620'}, {'title': 'Senior Flutter Developer', 'company': 'Proxify', 'location': 'Worldwide', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/remote-senior-flutter-developer-proxify-382219'}, {'title': 'Web Frontend Engineer JS CSS React Flutter', 'company': 'Canonical - Jobs', 'location': 'NG Nigeria', 'pay': '$58k - $118k', 'link': 'https://remoteok.com/remote-jobs/remote-web-frontend-engineer-js-css-react-flutter-canonical-jobs-472776'}, {'title': 'Senior Mobile Engineer', 'company': 'Careers at Tide', 'location': 'IN India', 'pay': '$68k - $118k', 'link': 'https://remoteok.com/remote-jobs/remote-senior-mobile-engineer-careers-at-tide-472714'}, {'title': 'Senior Mobile Developer Latin America', 'company': 'FullStack Labs', 'location': 'BR Brazil', 'pay': '$58k - $103k', 'link': 'https://remoteok.com/remote-jobs/remote-senior-mobile-developer-latin-america-fullstack-labs-469370'}, {'title': 'Full Stack Web3 Co Founder', 'company': 'Senny', 'location': 'Worldwide', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/118288-remote-full-stack-web3-co-founder-senny'}, {'title': 'Golang Backend Engineer', 'company': 'NEKOM CC GmbH', 'location': 'Probably worldwide', 'pay': '$65k - $120k', 'link': 'https://remoteok.com/remote-jobs/107942-remote-golang-backend-engineer-nekomcc-gmbh'}, {'title': 'Senior Developer', 'company': 'Toptal', 'location': 'Worldwide', 'pay': None, 'link': 'https://remoteok.com/remote-jobs/99898-remote-senior-developer-toptal'}, {'title': 'Golang Flutter Developer', 'company': 'GetCourageNow', 'location': 'Probably worldwide', 'pay': '$68k - $130k', 'link': 'https://remoteok.com/remote-jobs/83868-remote-golang-flutter-developer-getcourage-now'}, {'title': 'Golang Developer', 'company': 'Veganbase', 'location': 'Probably worldwide', 'pay': '$68k - $130k', 'link': 'https://remoteok.com/remote-jobs/72496-remote-golang-developer-veganbase'}]

```

[3단계] : 위에서 만들어진 값을 csv 파일에 저장한다!

```

from playwright.sync_api import sync_playwright
import time
from bs4 import BeautifulSoup
import re
import csv

all_jobs = []

def keywordJob(keyWord):
    # 동적으로 읽어야 읽히는 상황
    p = sync_playwright().start() # playwright 시작!

    browser = p.chromium.launch(headless=False) # 브라우저 : 크롬

    page = browser.new_page() # 브라우저에 새로운 탭 생성

    page.goto(f"https://remoteok.com/remote-{keyWord}-jobs") # keywords에 있는 직업을 검색!

    time.sleep(3)

    content = page.content()

    p.stop()

    return content

```

```

def BeautifulSoup(jobsContent, keyword):
    soup = BeautifulSoup(jobsContent, "html.parser")

    jobsboard = soup.find('table', id='jobsboard'
                           ).find_all('td', class_='company position company_and_position')[1:]

    for job in jobsboard:
        link = f"https://remoteok.com{job.find('a')['href']}"
        title = job.find("h2", itemprop="title").text.strip()
        company = job.find("h3", itemprop="name").text.strip()

        location_check = job.find('div', class_='location') # 이렇게만 가져오면 location_tooltip,
        location_tooltip-set 까지 같이 가져오짐. 필터링 필요.
        if location_check is None: # 내용이 없어서 None 으로 불러와진 경우 편집
            location='정보 없음'
        elif 'tooltip' in location_check.get('class', []): # 'tooltip'이 class 속성에 포함되어 있는지 확
            인 후 편집
                location='정보 없음'
        elif 'tooltip-set' in location_check.get('class', []): # 'tooltip-set'이 class 속성에 포함되어 있
            는지 확인 후 편집
                location='정보 없음'
        else:
            location=location_check.text.strip()

        pay = job.find("div", class_="location tooltip", title=re.compile("No salary data published"))

        if pay:
            pay = pay.get_text(strip=True) # 텍스트만 가져오고 양쪽 공백을 제거
            print(pay) # 💰 $60k - $130k*
        else:
            pay = "None" # 값이 없을 경우!

        job = {
            "title" : title,
            "company" : company,
            "location" : location,
            "pay" : pay,
            "link" : link
        }
        all_jobs.append(job)

    file = open(f"{keyword}.csv", "w", encoding="utf-8")

```

```
writer = csv.writer(file)
```

```
writer.writerow([  
    "Title",  
    "Company",  
    "Location",  
    "Pay",  
    "Link"  
])
```

```
for job in all_jobs:  
    writer.writerow(job.values())
```

```
all_jobs.clear()
```

```
keywords = [  
    'flutter',  
    'python',  
    'golang'  
]
```

```
for key in keywords:  
    jobs_content = keywordJob(key) # keywords의 검색 결과를 content에 저장  
    BeautifulSoup(jobs_content, key)
```

이것도 성공했다!!



각각 나눠서 csv 파일로 저장하기 성공했다!

이제 완성된 파일을 class로 만들어서 다시 수정하기로 결정!

하면서 사용자가 마냥 기다리기는 지루하니까 print문을 사용해서 진행상황을 알려주는게 좋을 것 같다고 생각함 (그래서 했습니다~)

#### 수정된 코드

```
from playwright.sync_api import sync_playwright
import time
from bs4 import BeautifulSoup
import re
import csv

class webScrap:
    def __init__(self, keyWord):
        self.keyword = keyWord
        self.all_jobs = []

    def keywordJob(self): # 전달받은 keyword를 사용해서 페이지 내부의 내용을 저장하기

        print(f"{self.keyword} page scraping start!")

        # 동적으로 읽어야 읽히는 상황
        p = sync_playwright().start() # playwright 시작!
```



```

browser = p.chromium.launch() # 브라우저 : 크롬

page = browser.new_page() # 브라우저에 새로운 탭 생성

page.goto(f"https://remoteok.com/remote-{self.keyword}-jobs") # keywords에 있는 작업을 검
색!

time.sleep(3) # 화면 로딩 대기

# 스크롤을 더이상 내릴 수 없을 때까지 내리는 코드도 짜봤는데, python이나 golang이 경우 1년
이상 지난 게시물까지 전부 로딩되는걸 확인하고 아래 코드로 최종 결정.
for i in range(4): # 스크롤 4번만 내리면 내 컴퓨터 기준 120개 정도의 게시물이 로딩됨.
    page.keyboard.down('End')
    time.sleep(5)

content = page.content() # 페이지 내용 가져오기
browser.close()
p.stop()

self.content = content
print(f"{self.keyword} page scraping finish!")

def BeautifulSoup(self): # BeautifulSoup를 사용해서 원하는 데이터를 뽑아내기

    print(f"{self.keyword} job list making...") # 현재 스크랩 중인 잡을 알려줌

    soup = BeautifulSoup(self.content, "html.parser")

    jobsboard = soup.find('table', id='jobsboard'
                           ).find_all('td', class_='company position company_and_position')[1:]

    for job in jobsboard:
        link = f"https://remoteok.com{job.find('a')['href']}" # 각각 직업에 걸려져있는 링크완성하
        기!

        title = job.find("h2", itemprop="title").text.strip() # 제목
        company = job.find("h3", itemprop="name").text.strip() # 회사

        location_check = job.find('div', class_='location') # 이렇게만 가져오면 location_tooltip,
        location_tooltip-set 까지 같이 가져오짐.
        if location_check is None: # 내용이 없어서 None 으로 불러와진 경우 편집
            location='정보 없음'
        elif 'tooltip' in location_check.get('class', []): # 'tooltip'이 class 속성에 포함되어 있는지

```

확인 후 편집

```
        location='정보 없음'
    elif 'tooltip-set' in location_check.get('class', []): # 'tooltip-set'이 class 속성에 포함되
어 있는지 확인 후 편집
        location='정보 없음'
    else:
        location=location_check.text.strip()

    pay = job.find("div", class_="location tooltip", title=re.compile("No salary data
published"))

    if pay:
        pay = pay.get_text(strip=True) # 텍스트만 가져오고 양쪽 공백을 제거
        # print(pay) # ₩ $60k - $130k*
    else:
        pay = "None" # 값이 없을 경우!

    job = {
        "title" : title,
        "company" : company,
        "location" : location,
        "pay" : pay,
        "link" : link
    }

    self.all_jobs.append(job) # 생성된 job 딕셔너리를 all_jobs 리스트에 추가!

file = open(f"{self.keyword}.csv", "w", encoding="utf-8")
writer = csv.writer(file)

writer.writerow([
    "Title",
    "Company",
    "Location",
    "Pay",
    "Link"
])

for job in self.all_jobs: # 읽은 정보를 이제 csv 파일에 쓰기!
    writer.writerow(job.values())
file.close() # 다 썼으면 file 닫기
```

```
print(f'{self.keyword} job list making finish.') # 진행상황 브리핑
print(f'정리된 {self.keyword} job 수는 {len(self.all_jobs)} 입니다. 자세한 내용은 csv 파일을 확인
해주세요.') # 결과 확인용 메시지
```

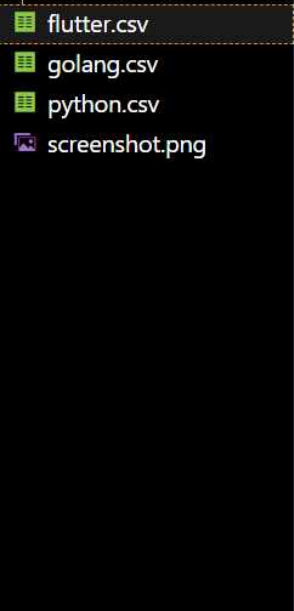
# 우리가 이제 검색하고자 하는 직업의 키워드 리스트

```
keywords = [
    'flutter',
    'python',
    'golang'
]
```

# keywords에 저장된 갯수만큼 반복

```
for i in range(len(keywords)):
    keyword=webScrap(keywords[i]) #
    keyword.keywordJob() # page의 content를 읽기
    keyword.beautiSoup() # 읽은 content에서 원하는 정보 읽기 실행!
```

이제 각 페이지 스크랩이 끝나면 글자가 보인다

|   |  |
|---|--|
|  | <pre>PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders&gt; &amp; C:/Use ython_Web/CodeChallenge_클래스까지.py" flutter page scraping start! flutter page scraping finish! flutter job list making... flutter job list making finish. 정리된 flutter job 수는 81 입니다. 자세한 내용은 csv 파일을 확인해주세요. python page scraping start! python page scraping finish! python job list making... python job list making finish. 정리된 python job 수는 119 입니다. 자세한 내용은 csv 파일을 확인해주세요. golang page scraping start! golang page scraping finish! golang job list making... golang job list making finish. 정리된 golang job 수는 60 입니다. 자세한 내용은 csv 파일을 확인해주세요. PS C:\Users\jgy09\OneDrive\바탕 화면\PythonWorkspace\Nomadcoders&gt; []</pre> |
|---|--|