Web Scraping Method

Website: SAM.gov

Tools: Python, VS CODE

TEAM AMPLYTICS

UC DAVIS 2025

JENNY, HENRI, REBECCA, SONIA, ALEX

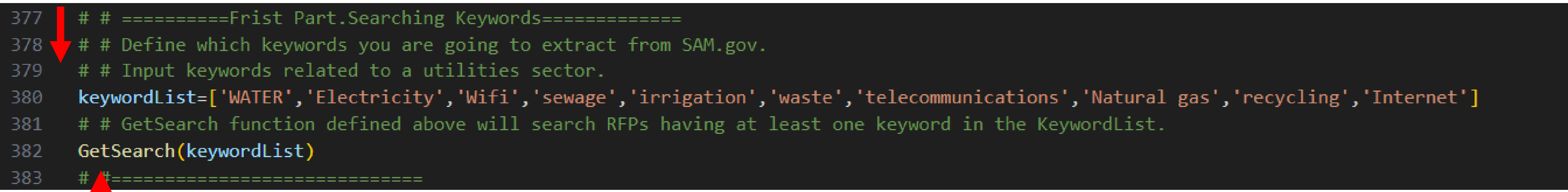
Table of contents

Brief introduction of code

The entire script is divided into three main parts. In the top part, I defined the functions necessary for installing the Chrome browser, sending an API call, retrieving specific details of each RFP and uploading the complete data to Supabase. On the bottom part, I specified the logic for executing the script, including keywords insertion and data iterations for consecutive data extractions from the web page and uploads to Supabase. Therefore, you can better understand the code by reviewing it starting from the bottom and moving back and forth to find how the functions interact with the main execution rather than reviewing it sequentially.

Part 1. Retrieving data by sending an API call.

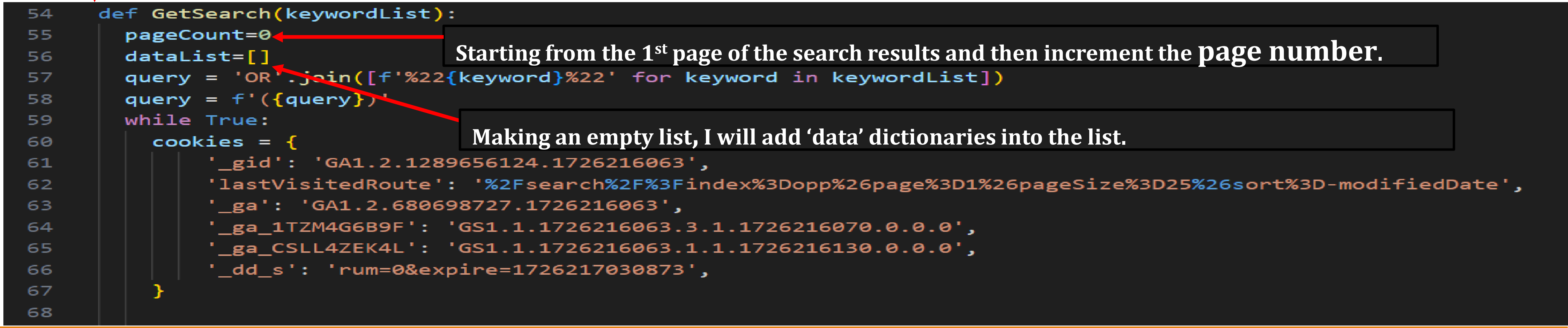
Define keywords you want to search for on SAM.gov in the ‘keywordList’



```
# # ==========Frist Part.Searching Keywords=============
# # Define which keywords you are going to extract from SAM.gov.
# # Input keywords related to a utilities sector.
keywordList=['WATER','Electricity','Wifi','sewage','irrigation','waste','telecommunications','Natural gas','recycling','Internet']
# # GetSearch function defined above will search RFPs having at least one keyword in the KeywordList.
GetSearch(keywordList)
# #============================
```

Scroll up to find a ‘GetSearch’ function defined above.

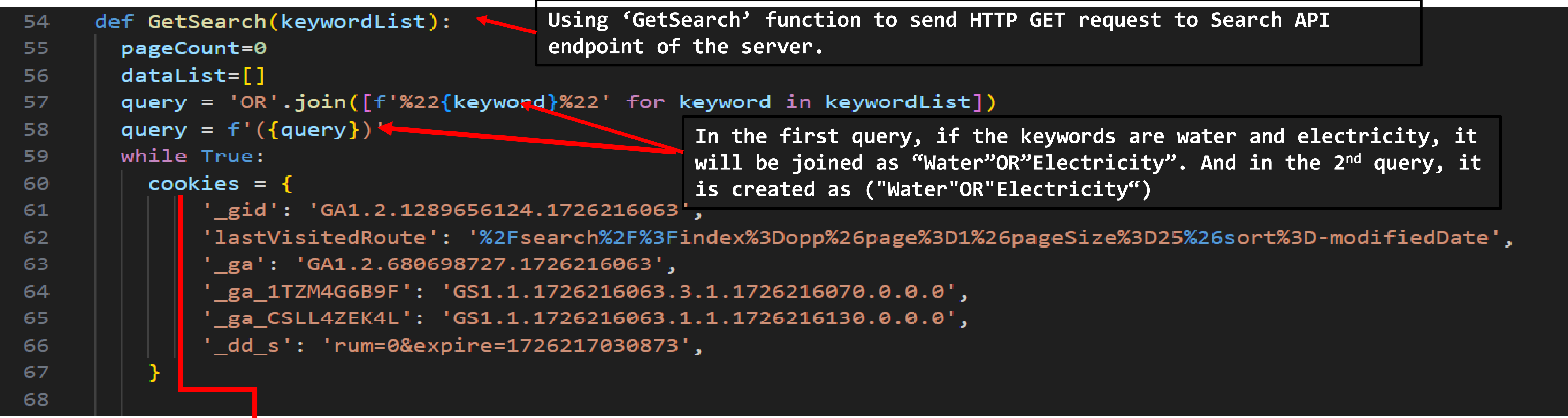
You can find a ‘GetSearch’ function where it sends HTTP GET request to the server.
Here, I sent a HTTP GET request to Search API endpoint to get search data using keywords listed in the KeywordList. This process starts when accessing ‘Advanced search’ and adding keywords in a Search Editor to extract RFPs details before clicking a specific document.



```
def GetSearch(keywordList):
    pageCount=0
    dataList=[]
    query = 'OR'.join([f'%22{keyword}%22' for keyword in keywordList])
    query = f'({query})'
    while True:
        cookies = {
            '_gid': 'GA1.2.1289656124.1726216063',
            'lastVisitedRoute': '%2Fsearch%2F%3Findex%3Dopp%26page%3D1%26pageSize%3D25%26sort%3D-modifiedDate',
            '_ga': 'GA1.2.680698727.1726216063',
            '_ga_1TZM4G6B9F': 'GS1.1.1726216063.3.1.1726216070.0.0.0',
            '_ga_CSLL4ZEK4L': 'GS1.1.1726216063.1.1.1726216130.0.0.0',
            '_dd_s': 'rum=0&expire=1726217030873',
        }
```

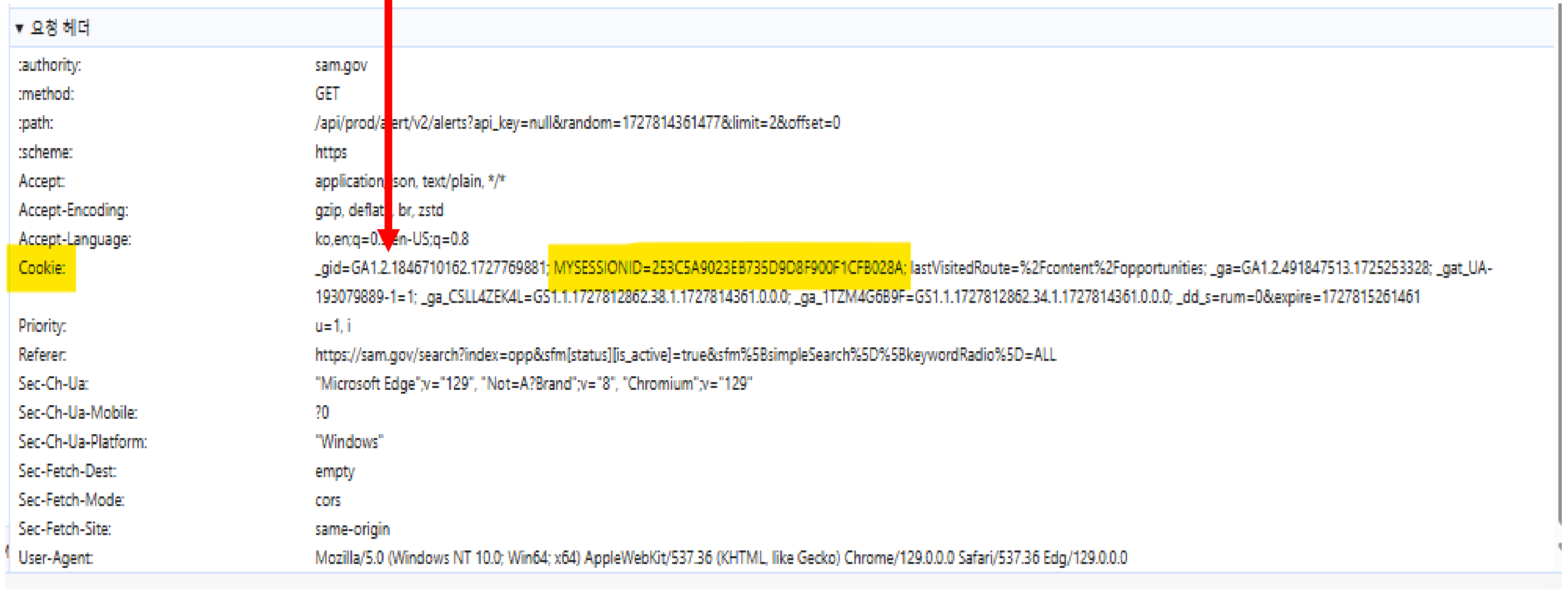
Starting from the 1st page of the search results and then increment the **page number**.

Making an empty list, I will add ‘data’ dictionaries into the list.

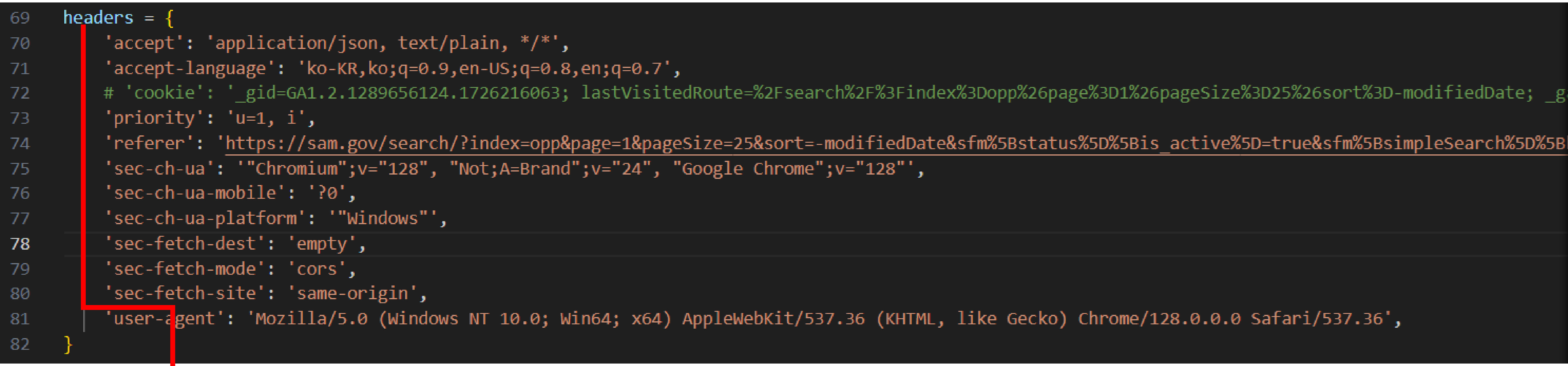

```python
def GetSearch(keywordList):
    pageCount=0
    dataList=[]
    query = 'OR'.join([f'%22{keyword}%22' for keyword in keywordList])
    query = f'({query})'
    while True:
        cookies = {
            '_gid': 'GA1.2.1289656124.1726216063',
            'lastVisitedRoute': '%2Fsearch%2F%3Findex%3Dopp%26page%3D1%26pageSize%3D25%26sort%3D-modifiedDate',
            '_ga': 'GA1.2.680698727.1726216063',
            '_ga_1TZM4G6B9F': 'GS1.1.1726216063.3.1.1726216070.0.0.0',
            '_ga_CSLL4ZEK4L': 'GS1.1.1726216063.1.1.1726216130.0.0.0',
            '_dd_s': 'rum=0&expire=1726217030873',
        }

```

Using 'GetSearch' function to send HTTP GET request to Search API endpoint of the server.

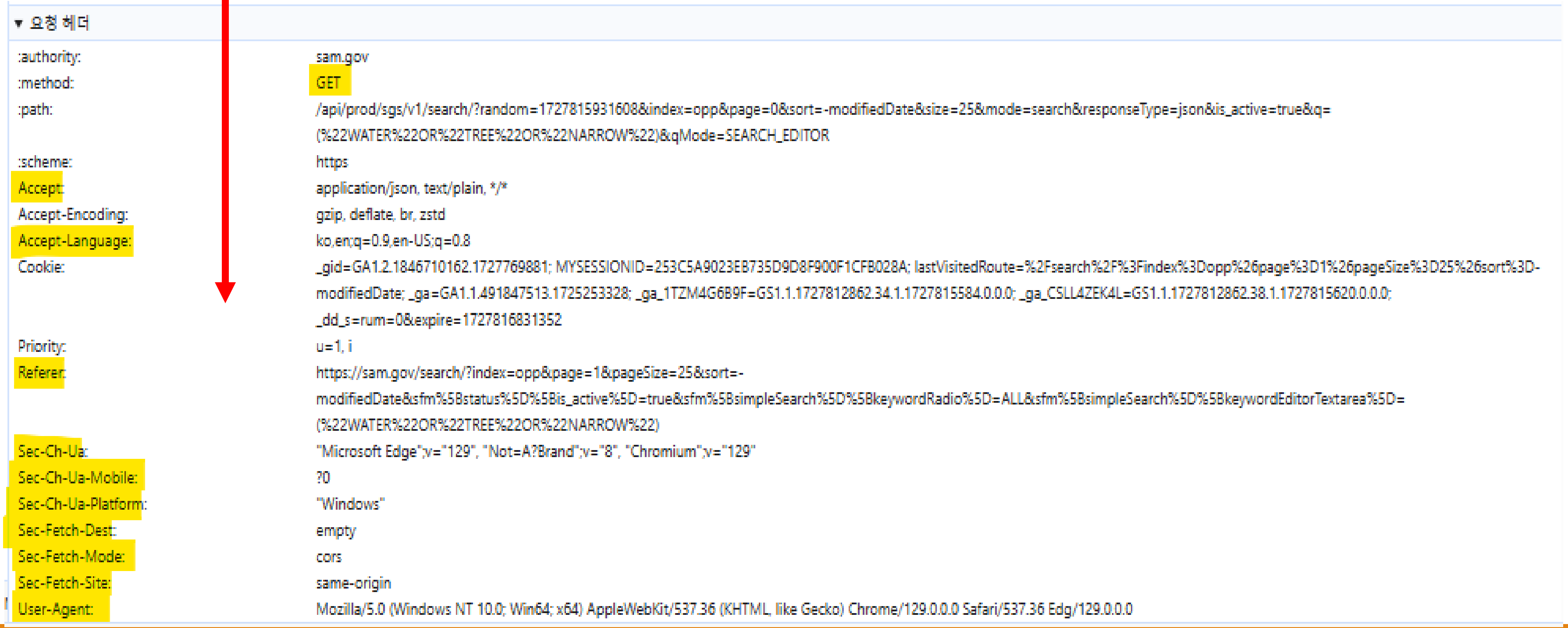
In the first query, if the keywords are water and electricity, it will be joined as "Water"OR"Electricity". And in the 2nd query, it is created as ("Water"OR"Electricity")


```
▾ 요청 헤더
:authority:              sam.gov
:method:                 GET
:path:                   /api/prod/a ert/v2/alerts?api_key=null&random=1727814361477&limit=2&offset=0
:scheme:                 https
Accept:                  application son, text/plain, */*
Accept-Encoding:         gzip, deflat , br, zstd
Accept-Language:         ko,en;q=0. en-US;q=0.8
Cookie:                  _gid=GA1.2.1846710162.1727769881; MYSESSIONID=253C5A9023EB735D9D8F900F1CFB028A; lastVisitedRoute=%2Fcontent%2Fopportunities; _ga=GA1.2.491847513.1725253328; _gat_UA-
                         193079889-1=1; _ga_CSLL4ZEK4L=GS1.1.1727812862.38.1.1727814361.0.0.0; _ga_1TZM4G6B9F=GS1.1.1727812862.34.1.1727814361.0.0.0; _dd_s=rum=0&expire=1727815261461
Priority:                u=1, i
Referer:                 https://sam.gov/search?index=opp&sfm[status][is_active]=true&sfm%5BsimpleSearch%5D%5BkeywordRadio%5D=ALL
Sec-Ch-Ua:               "Microsoft Edge";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
Sec-Ch-Ua-Mobile:        ?0
Sec-Ch-Ua-Platform:      "Windows"
Sec-Fetch-Dest:          empty
Sec-Fetch-Mode:          cors
Sec-Fetch-Site:          same-origin
User-Agent:              Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36 Edg/129.0.0.0
```

We can find 'cookie' data in 'DevTools'. The 'cookie' data usually involves session data for users. However, SAM.gov possesses public data so I did not include MYSESSIONID info in the 'cookie'. You can find 'cookie' data in a 'Request Header'.



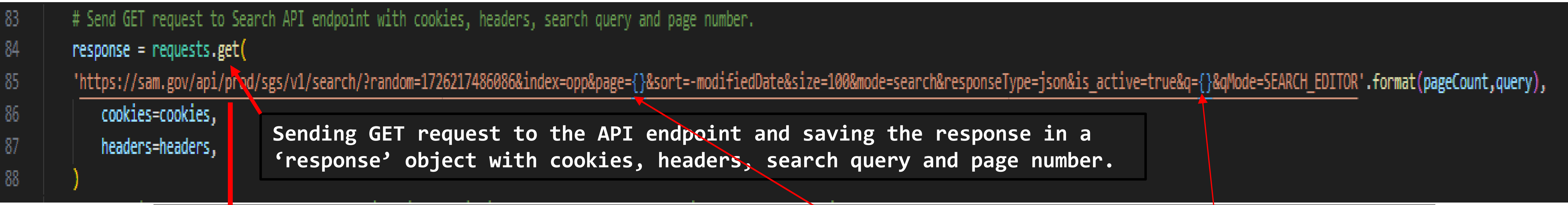
```python
headers = {
    'accept': 'application/json, text/plain, */*',
    'accept-language': 'ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7',
    # 'cookie': '_gid=GA1.2.1289656124.1726216063; lastVisitedRoute=%2Fsearch%2F%3Findex%3Dopp%26page%3D1%26pageSize%3D25%26sort%3D-modifiedDate; _g
    'priority': 'u=1, i',
    'referer': 'https://sam.gov/search/?index=opp&page=1&pageSize=25&sort=-modifiedDate&sfm%5Bstatus%5D%5Bis_active%5D=true&sfm%5BsimpleSearch%5D%5B
    'sec-ch-ua': '"Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"',
    'sec-ch-ua-mobile': '?0',
    'sec-ch-ua-platform': '"Windows"',
    'sec-fetch-dest': 'empty',
    'sec-fetch-mode': 'cors',
    'sec-fetch-site': 'same-origin',
    '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',
}
```



▼ 요청 헤더

:authority:	sam.gov
:method:	GET
:path:	/api/prod/sgs/v1/search/?random=1727815931608&index=opp&page=0&sort=-modifiedDate&size=25&mode=search&responseType=json&is_active=true&q=(%22WATER%22OR%22TREE%22OR%22NARROW%22)&qMode=SEARCH_EDITOR
:scheme:	https
Accept:	application/json, text/plain, */*
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	ko,en;q=0.9,en-US;q=0.8
Cookie:	_gid=GA1.2.1846710162.1727769881; MYSESSIONID=253C5A9023EB735D9D8F900F1CFB028A; lastVisitedRoute=%2Fsearch%2F%3Findex%3Dopp%26page%3D1%26pageSize%3D25%26sort%3D-modifiedDate; _ga=GA1.1.491847513.1725253328; _ga_1TZM4G6B9F=GS1.1.1727812862.34.1.1727815584.0.0.0; _ga_CSLL4ZEK4L=GS1.1.1727812862.38.1.1727815620.0.0.0; _dd_s=rum=0&expire=1727816831352
Priority:	u=1, i
Referer:	https://sam.gov/search/?index=opp&page=1&pageSize=25&sort=-modifiedDate&sfm%5Bstatus%5D%5Bis_active%5D=true&sfm%5BsimpleSearch%5D%5BkeywordRadio%5D=ALL&sfm%5BsimpleSearch%5D%5BkeywordEditorTextarea%5D=(%22WATER%22OR%22TREE%22OR%22NARROW%22)
Sec-Ch-Ua:	"Microsoft Edge";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"
Sec-Fetch-Dest:	empty
Sec-Fetch-Mode:	cors
Sec-Fetch-Site:	same-origin
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36 Edg/129.0.0.0

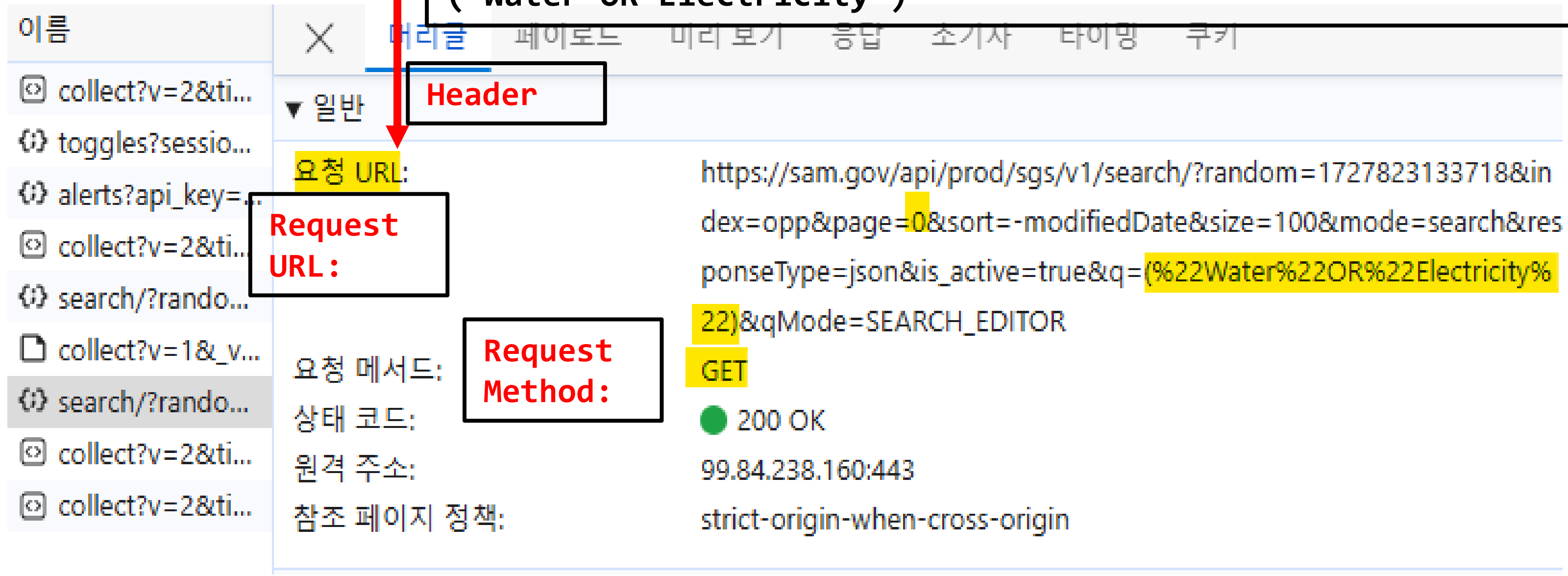
I need to send 'headers' information to the server when sending HTTP GET request to Search API.



```python
# Send GET request to Search API endpoint with cookies, headers, search query and page number.
response = requests.get(
'https://sam.gov/api/prod/sgs/v1/search/?random=1726217486086&index=opp&page={}&sort=-modifiedDate&size=100&mode=search&responseType=json&is_active=true&q={}&qMode=SEARCH_EDITOR'.format(pageCount,query),
    cookies=cookies,
    headers=headers,
)
```

Sending GET request to the API endpoint and saving the response in a 'response' object with cookies, headers, search query and page number.

This is the 'request URL' with page number and search query.
I found this 'request URL' in the header of a 'DevTools'.
In the image below, I am on the 1st page and I search keywords like ("Water"OR"Electricity")



Header

요청 URL: https://sam.gov/api/prod/sgs/v1/search/?random=1727823133718&index=opp&page=0&sort=-modifiedDate&size=100&mode=search&responseType=json&is_active=true&q=(%22Water%22OR%22Electricity%22)&qMode=SEARCH_EDITOR

Request URL:

요청 메서드: GET

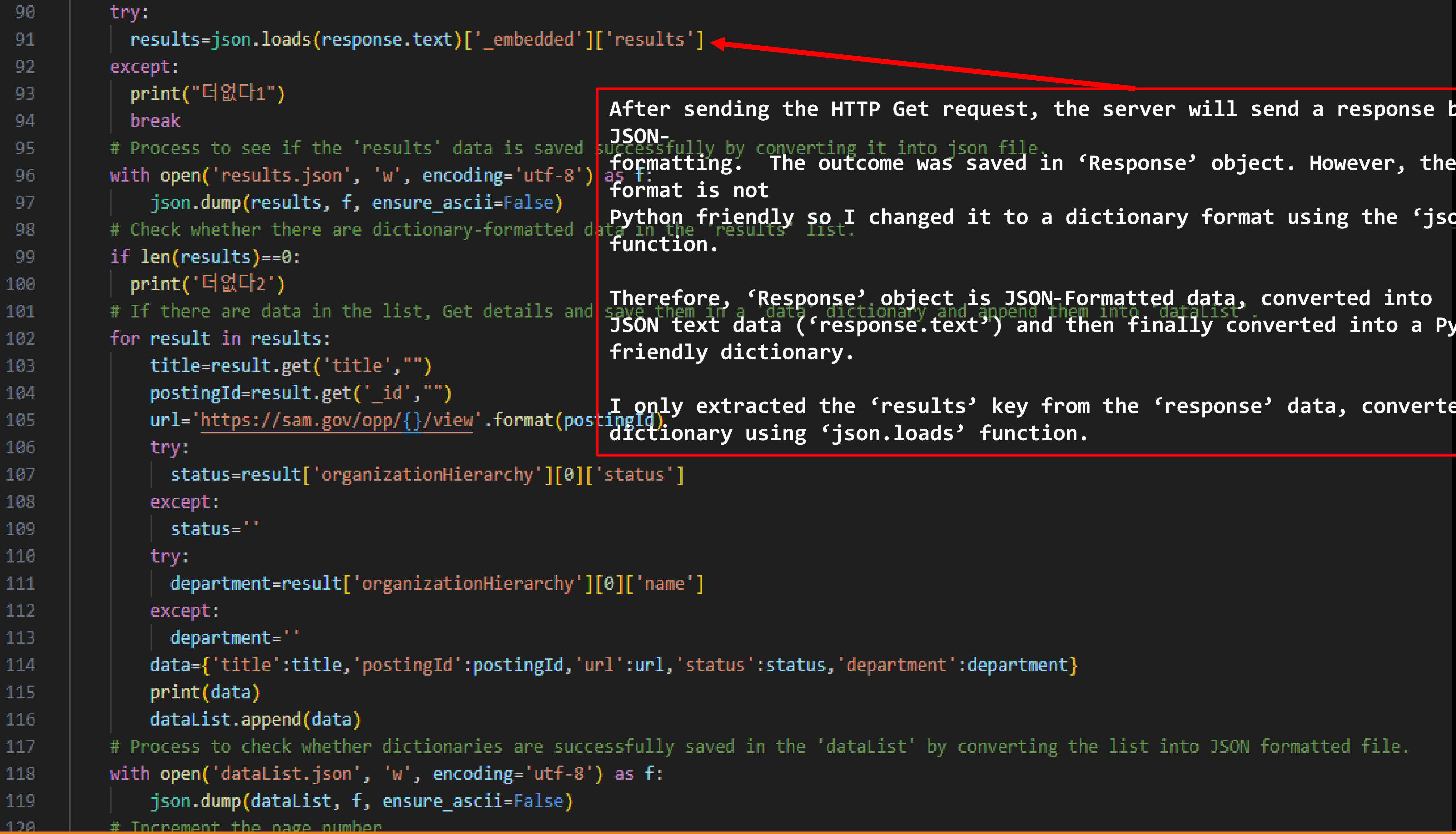
Request Method:

상태 코드: ● 200 OK

원격 주소: 99.84.238.160:443

참조 페이지 정책: strict-origin-when-cross-origin

When I sent the HTTP GET request with the Request, I had to dynamically change the page number and keyword query.
Therefore, I used {} to indicate a placeholder where dynamic values must be included.

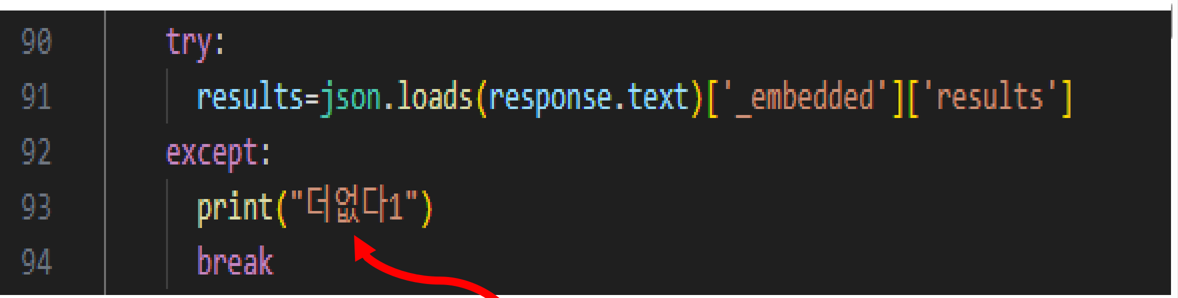


```python
    try:
        results=json.loads(response.text)['_embedded']['results']
    except:
        print("더없다1")
        break
    # Process to see if the 'results' data is saved successfully by converting it into json file.
    with open('results.json', 'w', encoding='utf-8') as f:
        json.dump(results, f, ensure_ascii=False)
    # Check whether there are dictionary-formatted data in the 'results' list.
    if len(results)==0:
        print('더없다2')
    # If there are data in the list, Get details and save them in a 'data' dictionary and append them into 'dataList'.
    for result in results:
        title=result.get('title',"")
        postingId=result.get('_id',"")
        url='https://sam.gov/opp/{}/view'.format(postingId)
        try:
            status=result['organizationHierarchy'][0]['status']
        except:
            status=''
        try:
            department=result['organizationHierarchy'][0]['name']
        except:
            department=''
        data={'title':title,'postingId':postingId,'url':url,'status':status,'department':department}
        print(data)
        dataList.append(data)
    # Process to check whether dictionaries are successfully saved in the 'dataList' by converting the list into JSON formatted file.
    with open('dataList.json', 'w', encoding='utf-8') as f:
        json.dump(dataList, f, ensure_ascii=False)
    # Increment the page number
```

After sending the HTTP Get request, the server will send a response back in JSON-formatting. The outcome was saved in 'Response' object. However, the JSON format is not Python friendly so I changed it to a dictionary format using the 'json.loads' function.

Therefore, 'Response' object is JSON-Formatted data, converted into JSON text data ('response.text') and then finally converted into a Python friendly dictionary.

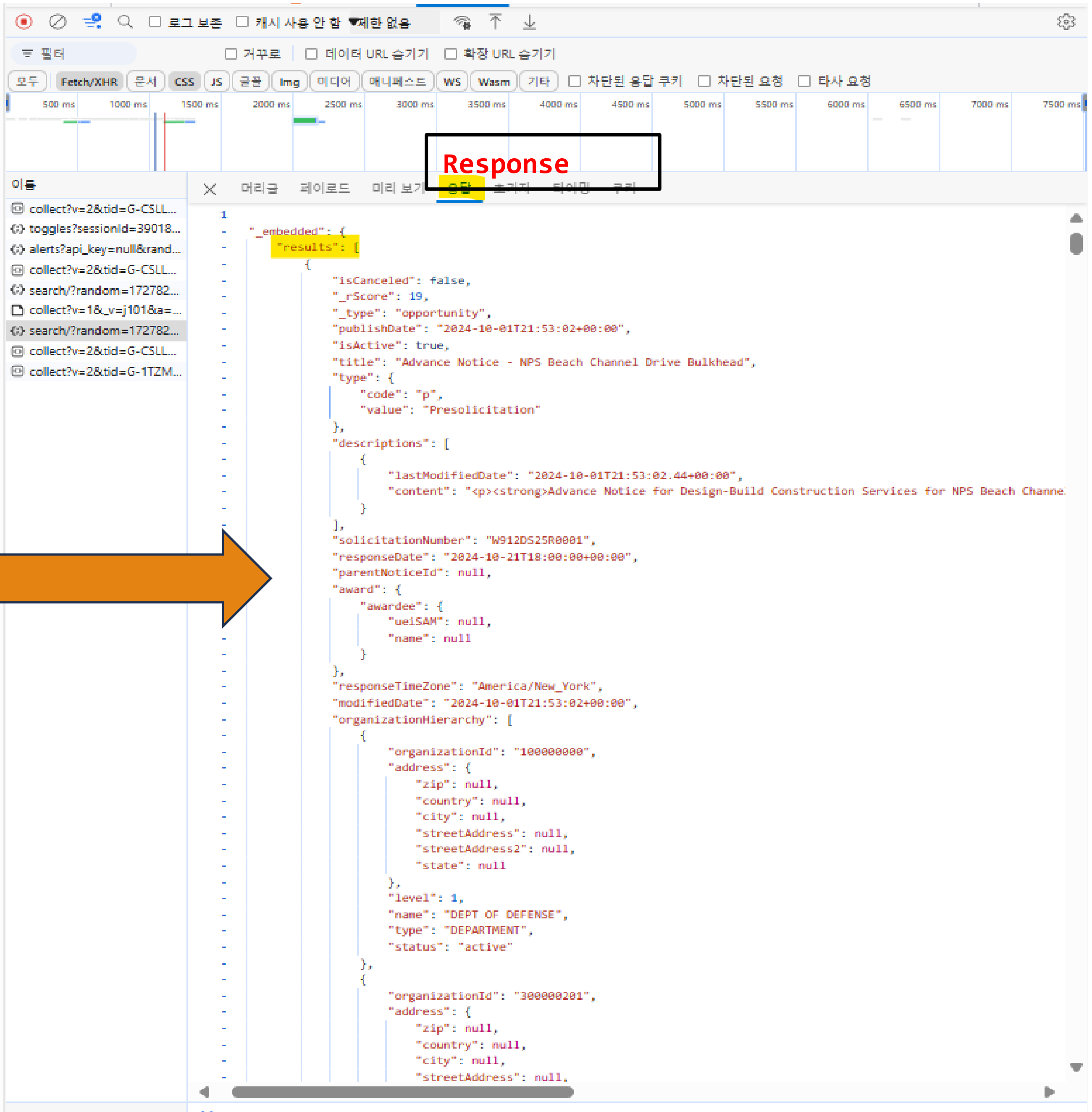
I only extracted the 'results' key from the 'response' data, converted into a dictionary using 'json.loads' function.



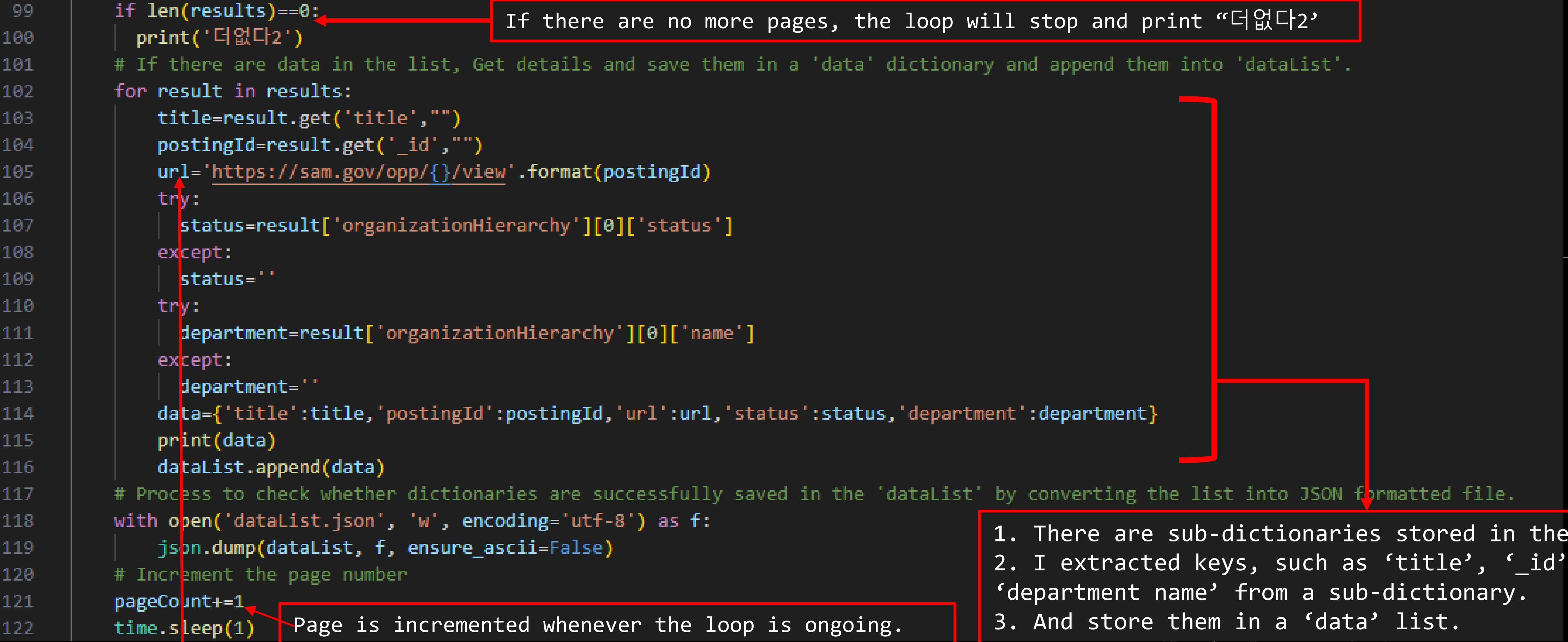
```python
    try:
        results=json.loads(response.text)['_embedded']['results']
    except:
        print("더없다1")
        break
```

If the ‘response’ data is not parsed correctly, the loop will stop.

JSON-Formatted data.
And ‘results’ is one of the keys in a dictionary, and its value is a list containing sub-dictionaries.



```
1
"_embedded": {
    "results": [
        {
            "isCanceled": false,
            "_rScore": 19,
            "_type": "opportunity",
            "publishDate": "2024-10-01T21:53:02+00:00",
            "isActive": true,
            "title": "Advance Notice - NPS Beach Channel Drive Bulkhead",
            "type": {
                "code": "p",
                "value": "Presolicitation"
            },
            "descriptions": [
                {
                    "lastModifiedDate": "2024-10-01T21:53:02.44+00:00",
                    "content": "<p><strong>Advance Notice for Design-Build Construction Services for NPS Beach Channe
                }
            ],
            "solicitationNumber": "W912DS25R0001",
            "responseDate": "2024-10-21T18:00:00+00:00",
            "parentNoticeId": null,
            "award": {
                "awardee": {
                    "ueiSAM": null,
                    "name": null
                }
            },
            "responseTimeZone": "America/New_York",
            "modifiedDate": "2024-10-01T21:53:02+00:00",
            "organizationHierarchy": [
                {
                    "organizationId": "100000000",
                    "address": {
                        "zip": null,
                        "country": null,
                        "city": null,
                        "streetAddress": null,
                        "streetAddress2": null,
                        "state": null
                    },
                    "level": 1,
                    "name": "DEPT OF DEFENSE",
                    "type": "DEPARTMENT",
                    "status": "active"
                },
                {
                    "organizationId": "300000201",
                    "address": {
                        "zip": null,
                        "country": null,
                        "city": null,
                        "streetAddress": null,
```

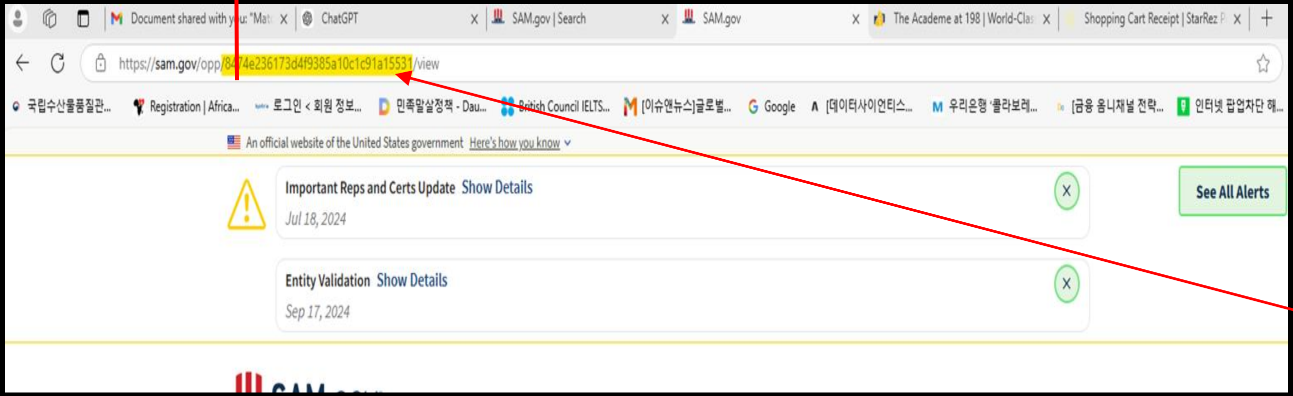


```python
    if len(results)==0:
      print('더없다2')
    # If there are data in the list, Get details and save them in a 'data' dictionary and append them into 'dataList'.
    for result in results:
        title=result.get('title',"")
        postingId=result.get('_id',"")
        url='https://sam.gov/opp/{}/view'.format(postingId)
        try:
          status=result['organizationHierarchy'][0]['status']
        except:
          status=''
        try:
          department=result['organizationHierarchy'][0]['name']
        except:
          department=''
        data={'title':title,'postingId':postingId,'url':url,'status':status,'department':department}
        print(data)
        dataList.append(data)
    # Process to check whether dictionaries are successfully saved in the 'dataList' by converting the list into JSON formatted file.
    with open('dataList.json', 'w', encoding='utf-8') as f:
        json.dump(dataList, f, ensure_ascii=False)
    # Increment the page number
    pageCount+=1
    time.sleep(1)
```

If there are no more pages, the loop will stop and print "더없다2'

1. There are sub-dictionaries stored in the 'results' list.
2. I extracted keys, such as 'title', '_id', 'status' and 'department name' from a sub-dictionary.
3. And store them in a 'data' list.
4. I repeatedly implemented the same process for the rest of the sub-dictionaries included in the 'results' object.
5. Finally, I appended all the 'data' dictionaries into the 'dataList' that I created above.

Page is incremented whenever the loop is ongoing.



https://sam.gov/opp/8474e236173d4f9385a10c1c91a15531/view

An official website of the United States government Here's how you know

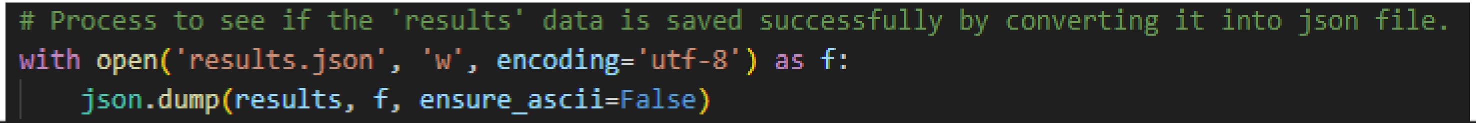
Important Reps and Certs Update Show Details
Jul 18, 2024

Entity Validation Show Details
Sep 17, 2024

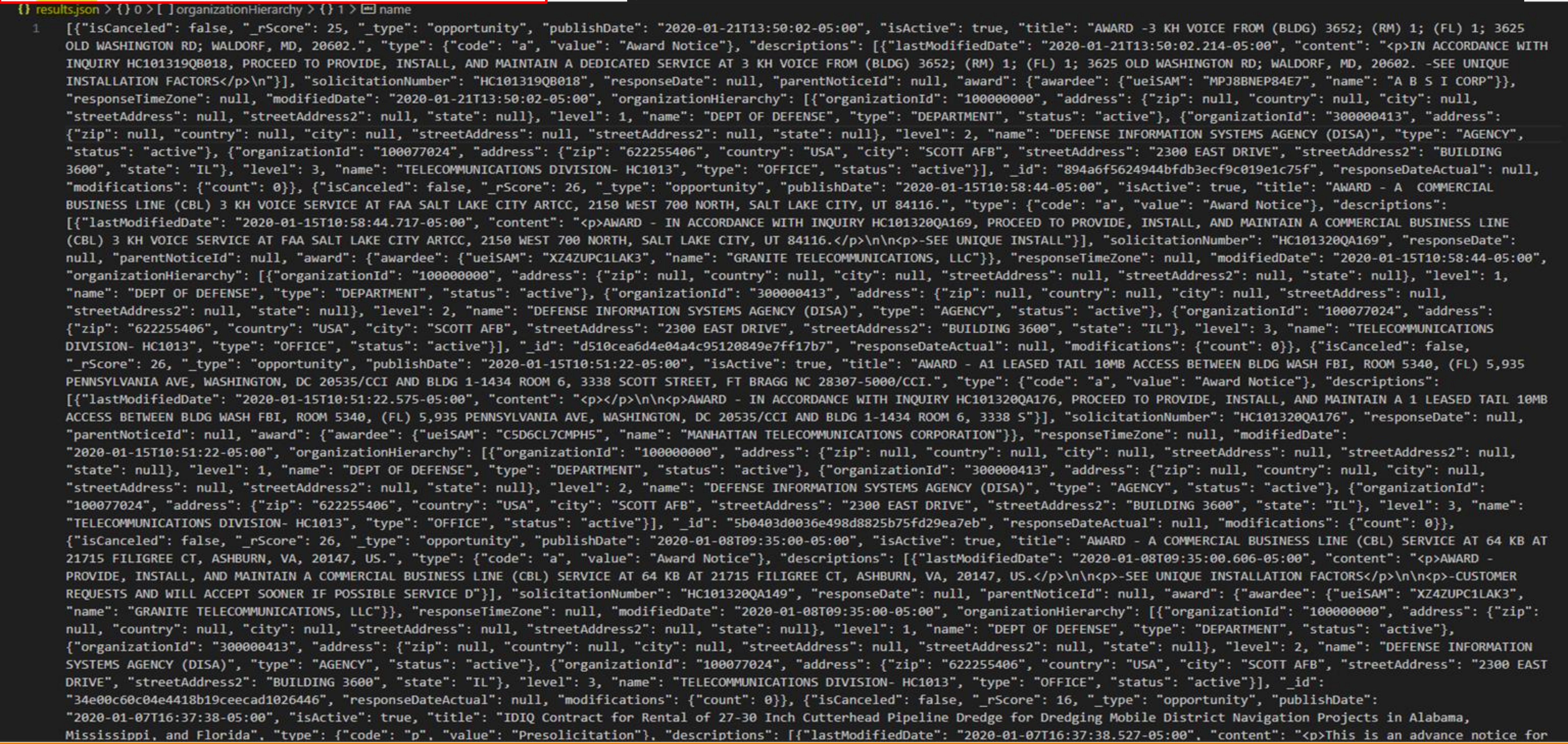
See All Alerts

When I click a document, the URL format is url='https://sam.gov/opp/{}/view'. In {}, the postingID, which is an identifier of each RFP, must be included.
I dynamically added the ID info using format function.

The result of ‘result.json’ file.

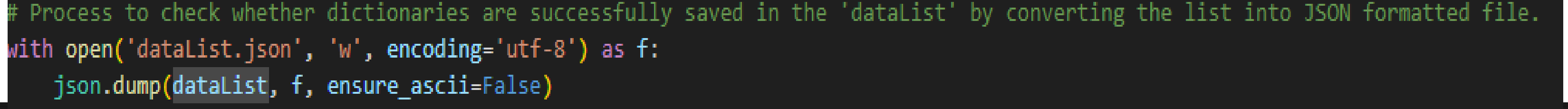


```python
# Process to see if the 'results' data is saved successfully by converting it into json file.
with open('results.json', 'w', encoding='utf-8') as f:
    json.dump(results, f, ensure_ascii=False)
```

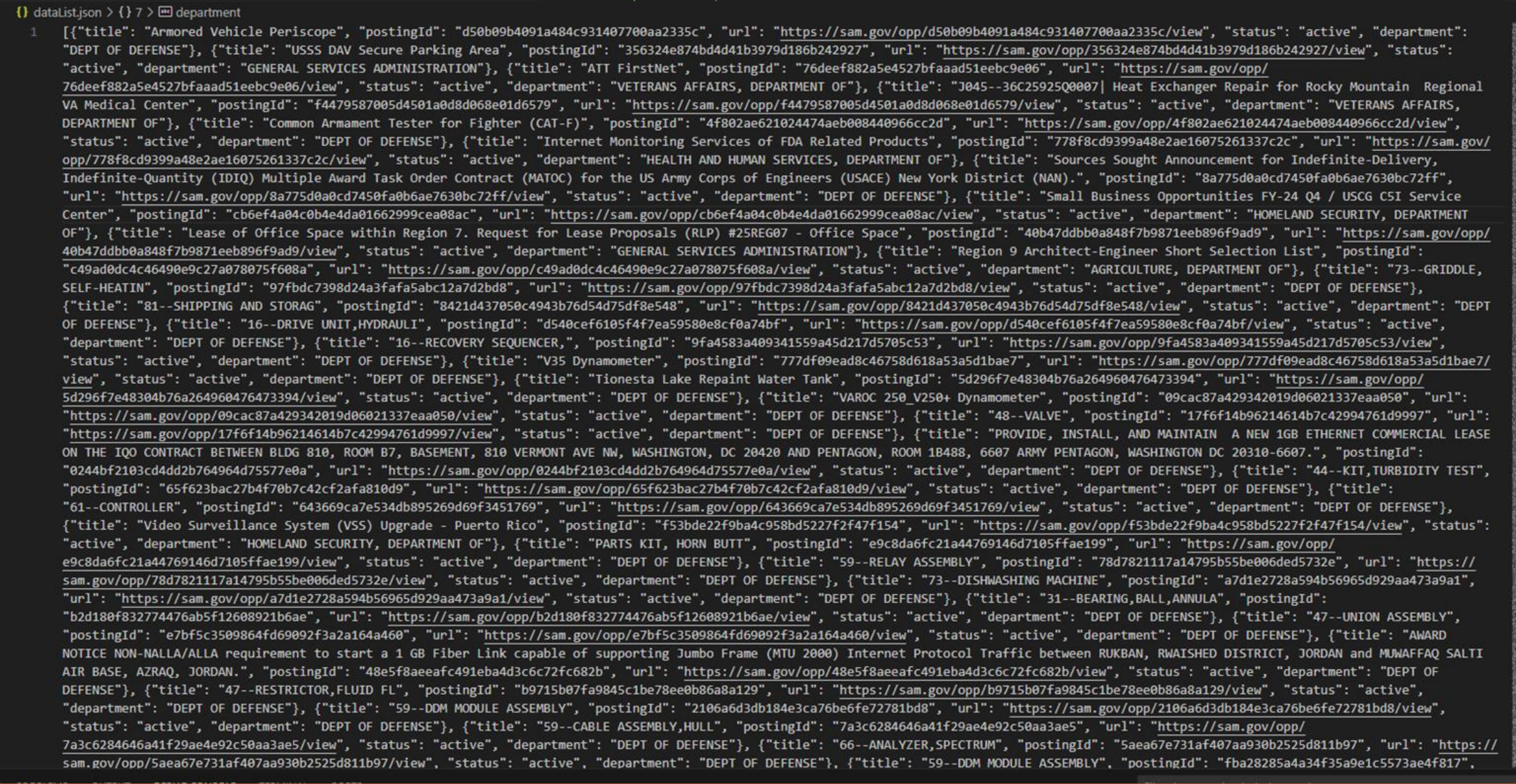

{} results.json > {} 0 > [ ] organizationHierarchy > {} 1 > name

```
[{"isCanceled": false, "_rScore": 25, "_type": "opportunity", "publishDate": "2020-01-21T13:50:02-05:00", "isActive": true, "title": "AWARD -3 KH VOICE FROM (BLDG) 3652; (RM) 1; (FL) 1; 3625
OLD WASHINGTON RD; WALDORF, MD, 20602.", "type": {"code": "a", "value": "Award Notice"}, "descriptions": [{"lastModifiedDate": "2020-01-21T13:50:02.214-05:00", "content": "<p>IN ACCORDANCE WITH
INQUIRY HC101319QB018, PROCEED TO PROVIDE, INSTALL, AND MAINTAIN A DEDICATED SERVICE AT 3 KH VOICE FROM (BLDG) 3652; (RM) 1; (FL) 1; 3625 OLD WASHINGTON RD; WALDORF, MD, 20602. -SEE UNIQUE
INSTALLATION FACTORS</p>\n"}], "solicitationNumber": "HC101319QB018", "responseDate": null, "parentNoticeId": null, "award": {"awardee": {"ueiSAM": "MPJ8BNEP84E7", "name": "A B S I CORP"}},
"responseTimeZone": null, "modifiedDate": "2020-01-21T13:50:02-05:00", "organizationHierarchy": [{"organizationId": "100000000", "address": {"zip": null, "country": null, "city": null,
"streetAddress": null, "streetAddress2": null, "state": null}, "level": 1, "name": "DEPT OF DEFENSE", "type": "DEPARTMENT", "status": "active"}, {"organizationId": "300000413", "address":
{"zip": null, "country": null, "city": null, "streetAddress": null, "streetAddress2": null, "state": null}, "level": 2, "name": "DEFENSE INFORMATION SYSTEMS AGENCY (DISA)", "type": "AGENCY",
"status": "active"}, {"organizationId": "100077024", "address": {"zip": "622255406", "country": "USA", "city": "SCOTT AFB", "streetAddress": "2300 EAST DRIVE", "streetAddress2": "BUILDING
3600", "state": "IL"}, "level": 3, "name": "TELECOMMUNICATIONS DIVISION- HC1013", "type": "OFFICE", "status": "active"}], "_id": "894a6f5624944bfdb3ecf9c019e1c75f", "responseDateActual": null,
"modifications": {"count": 0}}, {"isCanceled": false, "_rScore": 26, "_type": "opportunity", "publishDate": "2020-01-15T10:58:44-05:00", "isActive": true, "title": "AWARD - A  COMMERCIAL
BUSINESS LINE (CBL) 3 KH VOICE SERVICE AT FAA SALT LAKE CITY ARTCC, 2150 WEST 700 NORTH, SALT LAKE CITY, UT 84116.", "type": {"code": "a", "value": "Award Notice"}, "descriptions":
[{"lastModifiedDate": "2020-01-15T10:58:44.717-05:00", "content": "<p>AWARD - IN ACCORDANCE WITH INQUIRY HC101320QA169, PROCEED TO PROVIDE, INSTALL, AND MAINTAIN A COMMERCIAL BUSINESS LINE
(CBL) 3 KH VOICE SERVICE AT FAA SALT LAKE CITY ARTCC, 2150 WEST 700 NORTH, SALT LAKE CITY, UT 84116.</p>\n\n<p>-SEE UNIQUE INSTALL"}], "solicitationNumber": "HC101320QA169", "responseDate":
null, "parentNoticeId": null, "award": {"awardee": {"ueiSAM": "XZ4ZUPC1LAK3", "name": "GRANITE TELECOMMUNICATIONS, LLC"}}, "responseTimeZone": null, "modifiedDate": "2020-01-15T10:58:44-05:00",
"organizationHierarchy": [{"organizationId": "100000000", "address": {"zip": null, "country": null, "city": null, "streetAddress": null, "streetAddress2": null, "state": null}, "level": 1,
"name": "DEPT OF DEFENSE", "type": "DEPARTMENT", "status": "active"}, {"organizationId": "300000413", "address": {"zip": null, "country": null, "city": null, "streetAddress": null,
"streetAddress2": null, "state": null}, "level": 2, "name": "DEFENSE INFORMATION SYSTEMS AGENCY (DISA)", "type": "AGENCY", "status": "active"}, {"organizationId": "100077024", "address":
{"zip": "622255406", "country": "USA", "city": "SCOTT AFB", "streetAddress": "2300 EAST DRIVE", "streetAddress2": "BUILDING 3600", "state": "IL"}, "level": 3, "name": "TELECOMMUNICATIONS
DIVISION- HC1013", "type": "OFFICE", "status": "active"}], "_id": "d510cea6d4e04a4c95120849e7ff17b7", "responseDateActual": null, "modifications": {"count": 0}}, {"isCanceled": false,
"_rScore": 26, "_type": "opportunity", "publishDate": "2020-01-15T10:51:22-05:00", "isActive": true, "title": "AWARD - A1 LEASED TAIL 10MB ACCESS BETWEEN BLDG WASH FBI, ROOM 5340, (FL) 5,935
PENNSYLVANIA AVE, WASHINGTON, DC 20535/CCI AND BLDG 1-1434 ROOM 6, 3338 SCOTT STREET, FT BRAGG NC 28307-5000/CCI.", "type": {"code": "a", "value": "Award Notice"}, "descriptions":
[{"lastModifiedDate": "2020-01-15T10:51:22.575-05:00", "content": "<p></p>\n\n<p>AWARD - IN ACCORDANCE WITH INQUIRY HC101320QA176, PROCEED TO PROVIDE, INSTALL, AND MAINTAIN A 1 LEASED TAIL 10MB
ACCESS BETWEEN BLDG WASH FBI, ROOM 5340, (FL) 5,935 PENNSYLVANIA AVE, WASHINGTON, DC 20535/CCI AND BLDG 1-1434 ROOM 6, 3338 S"}], "solicitationNumber": "HC101320QA176", "responseDate": null,
"parentNoticeId": null, "award": {"awardee": {"ueiSAM": "C5D6CL7CMPH5", "name": "MANHATTAN TELECOMMUNICATIONS CORPORATION"}}, "responseTimeZone": null, "modifiedDate":
"2020-01-15T10:51:22-05:00", "organizationHierarchy": [{"organizationId": "100000000", "address": {"zip": null, "country": null, "city": null, "streetAddress": null, "streetAddress2": null,
"state": null}, "level": 1, "name": "DEPT OF DEFENSE", "type": "DEPARTMENT", "status": "active"}, {"organizationId": "300000413", "address": {"zip": null, "country": null, "city": null,
"streetAddress": null, "streetAddress2": null, "state": null}, "level": 2, "name": "DEFENSE INFORMATION SYSTEMS AGENCY (DISA)", "type": "AGENCY", "status": "active"}, {"organizationId":
"100077024", "address": {"zip": "622255406", "country": "USA", "city": "SCOTT AFB", "streetAddress": "2300 EAST DRIVE", "streetAddress2": "BUILDING 3600", "state": "IL"}, "level": 3, "name":
"TELECOMMUNICATIONS DIVISION- HC1013", "type": "OFFICE", "status": "active"}], "_id": "5b0403d0036e498d8825b75fd29ea7eb", "responseDateActual": null, "modifications": {"count": 0}},
{"isCanceled": false, "_rScore": 26, "_type": "opportunity", "publishDate": "2020-01-08T09:35:00-05:00", "isActive": true, "title": "AWARD - A COMMERCIAL BUSINESS LINE (CBL) SERVICE AT 64 KB AT
21715 FILIGREE CT, ASHBURN, VA, 20147, US.", "type": {"code": "a", "value": "Award Notice"}, "descriptions": [{"lastModifiedDate": "2020-01-08T09:35:00.606-05:00", "content": "<p>AWARD -
PROVIDE, INSTALL, AND MAINTAIN A COMMERCIAL BUSINESS LINE (CBL) SERVICE AT 64 KB AT 21715 FILIGREE CT, ASHBURN, VA, 20147, US.</p>\n\n<p>-SEE UNIQUE INSTALLATION FACTORS</p>\n\n<p>-CUSTOMER
REQUESTS AND WILL ACCEPT SOONER IF POSSIBLE SERVICE D"}], "solicitationNumber": "HC101320QA149", "responseDate": null, "parentNoticeId": null, "award": {"awardee": {"ueiSAM": "XZ4ZUPC1LAK3",
"name": "GRANITE TELECOMMUNICATIONS, LLC"}}, "responseTimeZone": null, "modifiedDate": "2020-01-08T09:35:00-05:00", "organizationHierarchy": [{"organizationId": "100000000", "address": {"zip":
null, "country": null, "city": null, "streetAddress": null, "streetAddress2": null, "state": null}, "level": 1, "name": "DEPT OF DEFENSE", "type": "DEPARTMENT", "status": "active"},
{"organizationId": "300000413", "address": {"zip": null, "country": null, "city": null, "streetAddress": null, "streetAddress2": null, "state": null}, "level": 2, "name": "DEFENSE INFORMATION
SYSTEMS AGENCY (DISA)", "type": "AGENCY", "status": "active"}, {"organizationId": "100077024", "address": {"zip": "622255406", "country": "USA", "city": "SCOTT AFB", "streetAddress": "2300 EAST
DRIVE", "streetAddress2": "BUILDING 3600", "state": "IL"}, "level": 3, "name": "TELECOMMUNICATIONS DIVISION- HC1013", "type": "OFFICE", "status": "active"}], "_id":
"34e00c60c04e4418b19ceecad1026446", "responseDateActual": null, "modifications": {"count": 0}}, {"isCanceled": false, "_rScore": 16, "_type": "opportunity", "publishDate":
"2020-01-07T16:37:38-05:00", "isActive": true, "title": "IDIQ Contract for Rental of 27-30 Inch Cutterhead Pipeline Dredge for Dredging Mobile District Navigation Projects in Alabama,
Mississippi, and Florida", "type": {"code": "p", "value": "Presolicitation"}, "descriptions": [{"lastModifiedDate": "2020-01-07T16:37:38.527-05:00", "content": "<p>This is an advance notice for
```

The result of ‘result.json’ file.



```
# Process to check whether dictionaries are successfully saved in the 'dataList' by converting the list into JSON formatted file.
with open('dataList.json', 'w', encoding='utf-8') as f:
    json.dump(dataList, f, ensure_ascii=False)
```

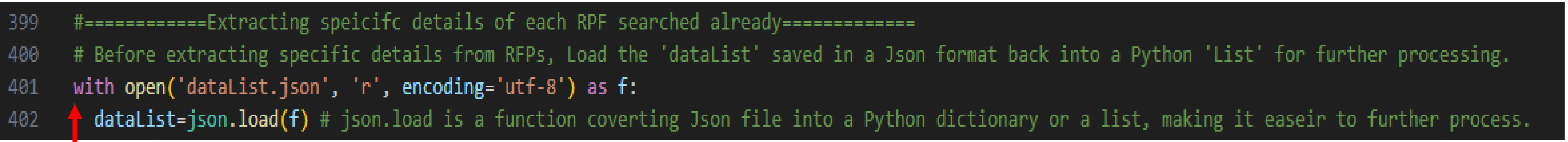


dataList.json > {} 7 > department

```
[{"title": "Armored Vehicle Periscope", "postingId": "d50b09b4091a484c931407700aa2335c", "url": "https://sam.gov/opp/d50b09b4091a484c931407700aa2335c/view", "status": "active", "department":
"DEPT OF DEFENSE"}, {"title": "USSS DAV Secure Parking Area", "postingId": "356324e874bd4d41b3979d186b242927", "url": "https://sam.gov/opp/356324e874bd4d41b3979d186b242927/view", "status":
"active", "department": "GENERAL SERVICES ADMINISTRATION"}, {"title": "ATT FirstNet", "postingId": "76deef882a5e4527bfaaad51eebc9e06", "url": "https://sam.gov/opp/
76deef882a5e4527bfaaad51eebc9e06/view", "status": "active", "department": "VETERANS AFFAIRS, DEPARTMENT OF"}, {"title": "J045--36C25925Q0007| Heat Exchanger Repair for Rocky Mountain  Regional
VA Medical Center", "postingId": "f4479587005d4501a0d8d068e01d6579", "url": "https://sam.gov/opp/f4479587005d4501a0d8d068e01d6579/view", "status": "active", "department": "VETERANS AFFAIRS,
DEPARTMENT OF"}, {"title": "Common Armament Tester for Fighter (CAT-F)", "postingId": "4f802ae621024474aeb008440966cc2d", "url": "https://sam.gov/opp/4f802ae621024474aeb008440966cc2d/view",
"status": "active", "department": "DEPT OF DEFENSE"}, {"title": "Internet Monitoring Services of FDA Related Products", "postingId": "778f8cd9399a48e2ae16075261337c2c", "url": "https://sam.gov/
opp/778f8cd9399a48e2ae16075261337c2c/view", "status": "active", "department": "HEALTH AND HUMAN SERVICES, DEPARTMENT OF"}, {"title": "Sources Sought Announcement for Indefinite-Delivery,
Indefinite-Quantity (IDIQ) Multiple Award Task Order Contract (MATOC) for the US Army Corps of Engineers (USACE) New York District (NAN).", "postingId": "8a775d0a0cd7450fa0b6ae7630bc72ff",
"url": "https://sam.gov/opp/8a775d0a0cd7450fa0b6ae7630bc72ff/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "Small Business Opportunities FY-24 Q4 / USCG C5I Service
Center", "postingId": "cb6ef4a04c0b4e4da01662999cea08ac", "url": "https://sam.gov/opp/cb6ef4a04c0b4e4da01662999cea08ac/view", "status": "active", "department": "HOMELAND SECURITY, DEPARTMENT
OF"}, {"title": "Lease of Office Space within Region 7. Request for Lease Proposals (RLP) #25REG07 - Office Space", "postingId": "40b47ddbb0a848f7b9871eeb896f9ad9", "url": "https://sam.gov/opp/
40b47ddbb0a848f7b9871eeb896f9ad9/view", "status": "active", "department": "GENERAL SERVICES ADMINISTRATION"}, {"title": "Region 9 Architect-Engineer Short Selection List", "postingId":
"c49ad0dc4c46490e9c27a078075f608a", "url": "https://sam.gov/opp/c49ad0dc4c46490e9c27a078075f608a/view", "status": "active", "department": "AGRICULTURE, DEPARTMENT OF"}, {"title": "73--GRIDDLE,
SELF-HEATIN", "postingId": "97fbdc7398d24a3fafa5abc12a7d2bd8", "url": "https://sam.gov/opp/97fbdc7398d24a3fafa5abc12a7d2bd8/view", "status": "active", "department": "DEPT OF DEFENSE"},
{"title": "81--SHIPPING AND STORAG", "postingId": "8421d437050c4943b76d54d75df8e548", "url": "https://sam.gov/opp/8421d437050c4943b76d54d75df8e548/view", "status": "active", "department": "DEPT
OF DEFENSE"}, {"title": "16--DRIVE UNIT,HYDRAULI", "postingId": "d540cef6105f4f7ea59580e8cf0a74bf", "url": "https://sam.gov/opp/d540cef6105f4f7ea59580e8cf0a74bf/view", "status": "active",
"department": "DEPT OF DEFENSE"}, {"title": "16--RECOVERY SEQUENCER,", "postingId": "9fa4583a409341559a45d217d5705c53", "url": "https://sam.gov/opp/9fa4583a409341559a45d217d5705c53/view",
"status": "active", "department": "DEPT OF DEFENSE"}, {"title": "V35 Dynamometer", "postingId": "777df09ead8c46758d618a53a5d1bae7", "url": "https://sam.gov/opp/777df09ead8c46758d618a53a5d1bae7/
view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "Tionesta Lake Repaint Water Tank", "postingId": "5d296f7e48304b76a264960476473394", "url": "https://sam.gov/opp/
5d296f7e48304b76a264960476473394/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "VAROC 250_V250+ Dynamometer", "postingId": "09cac87a429342019d06021337eaa050", "url":
"https://sam.gov/opp/09cac87a429342019d06021337eaa050/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "48--VALVE", "postingId": "17f6f14b96214614b7c42994761d9997", "url":
"https://sam.gov/opp/17f6f14b96214614b7c42994761d9997/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "PROVIDE, INSTALL, AND MAINTAIN  A NEW 1GB ETHERNET COMMERCIAL LEASE
ON THE IQO CONTRACT BETWEEN BLDG 810, ROOM B7, BASEMENT, 810 VERMONT AVE NW, WASHINGTON, DC 20420 AND PENTAGON, ROOM 1B488, 6607 ARMY PENTAGON, WASHINGTON DC 20310-6607.", "postingId":
"0244bf2103cd4dd2b764964d75577e0a", "url": "https://sam.gov/opp/0244bf2103cd4dd2b764964d75577e0a/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "44--KIT,TURBIDITY TEST",
"postingId": "65f623bac27b4f70b7c42cf2afa810d9", "url": "https://sam.gov/opp/65f623bac27b4f70b7c42cf2afa810d9/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title":
"61--CONTROLLER", "postingId": "643669ca7e534db895269d69f3451769", "url": "https://sam.gov/opp/643669ca7e534db895269d69f3451769/view", "status": "active", "department": "DEPT OF DEFENSE"},
{"title": "Video Surveillance System (VSS) Upgrade - Puerto Rico", "postingId": "f53bde22f9ba4c958bd5227f2f47f154", "url": "https://sam.gov/opp/f53bde22f9ba4c958bd5227f2f47f154/view", "status":
"active", "department": "HOMELAND SECURITY, DEPARTMENT OF"}, {"title": "PARTS KIT, HORN BUTT", "postingId": "e9c8da6fc21a44769146d7105ffae199", "url": "https://sam.gov/opp/
e9c8da6fc21a44769146d7105ffae199/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "59--RELAY ASSEMBLY", "postingId": "78d7821117a14795b55be006ded5732e", "url": "https://
sam.gov/opp/78d7821117a14795b55be006ded5732e/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "73--DISHWASHING MACHINE", "postingId": "a7d1e2728a594b56965d929aa473a9a1",
"url": "https://sam.gov/opp/a7d1e2728a594b56965d929aa473a9a1/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "31--BEARING,BALL,ANNULA", "postingId":
"b2d180f832774476ab5f12608921b6ae", "url": "https://sam.gov/opp/b2d180f832774476ab5f12608921b6ae/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "47--UNION ASSEMBLY",
"postingId": "e7bf5c3509864fd69092f3a2a164a460", "url": "https://sam.gov/opp/e7bf5c3509864fd69092f3a2a164a460/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "AWARD
NOTICE NON-NALLA/ALLA requirement to start a 1 GB Fiber Link capable of supporting Jumbo Frame (MTU 2000) Internet Protocol Traffic between RUKBAN, RWAISHED DISTRICT, JORDAN and MUWAFFAQ SALTI
AIR BASE, AZRAQ, JORDAN.", "postingId": "48e5f8aeeafc491eba4d3c6c72fc682b", "url": "https://sam.gov/opp/48e5f8aeeafc491eba4d3c6c72fc682b/view", "status": "active", "department": "DEPT OF
DEFENSE"}, {"title": "47--RESTRICTOR,FLUID FL", "postingId": "b9715b07fa9845c1be78ee0b86a8a129", "url": "https://sam.gov/opp/b9715b07fa9845c1be78ee0b86a8a129/view", "status": "active",
"department": "DEPT OF DEFENSE"}, {"title": "59--DDM MODULE ASSEMBLY", "postingId": "2106a6d3db184e3ca76be6fe72781bd8", "url": "https://sam.gov/opp/2106a6d3db184e3ca76be6fe72781bd8/view",
"status": "active", "department": "DEPT OF DEFENSE"}, {"title": "59--CABLE ASSEMBLY,HULL", "postingId": "7a3c6284646a41f29ae4e92c50aa3ae5", "url": "https://sam.gov/opp/
7a3c6284646a41f29ae4e92c50aa3ae5/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "66--ANALYZER,SPECTRUM", "postingId": "5aea67e731af407aa930b2525d811b97", "url": "https://
sam.gov/opp/5aea67e731af407aa930b2525d811b97/view", "status": "active", "department": "DEPT OF DEFENSE"}, {"title": "59--DDM MODULE ASSEMBLY", "postingId": "fba28285a4a34f35a9e1c5573ae4f817",
```

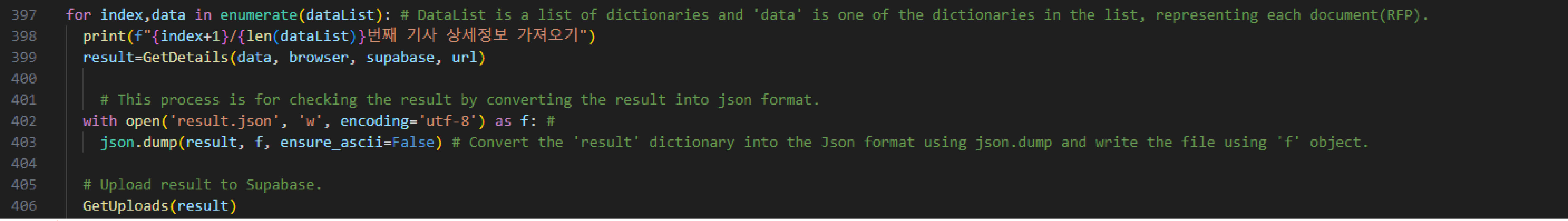
Part 2. Web scraping using Selenium and Beautiful Soup.

Now we are going to extract specific details from each RFP!



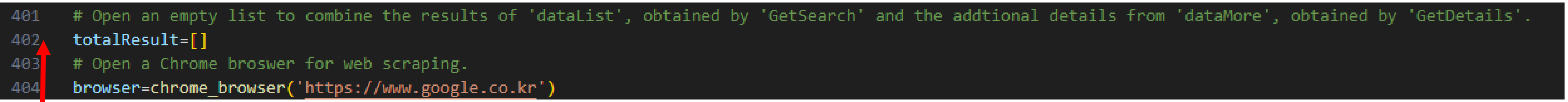
```python
#============Extracting speicifc details of each RPF searched already=============
# Before extracting specific details from RFPs, Load the 'dataList' saved in a Json format back into a Python 'List' for further processing.
with open('dataList.json', 'r', encoding='utf-8') as f:
    dataList=json.load(f) # json.load is a function coverting Json file into a Python dictionary or a list, making it easeir to further process.
```

1. We have already searched RFPs above and saved them into a ‘dataList’ and then converted them into a JSON file.
2. From this line, I am going to extract specific details of RFPs.
3. In order to further process it in Python, I need to convert the dataList in the JSON file format back into a Python ‘List’ using the ‘json.load’ function.
4. For your reference, ‘r’ means I am going to open the ‘dataList’ file in read mode using ‘f’ file object.
5. The function of the ‘f’ object is helping Python connect with the opened file. When the file opens, Python uses the ‘f’ file object to read from it.
6. The ‘encoding=utf-8’ helps Python read from and write to text files, managing various languages including English, Korean, Chinese and so on.



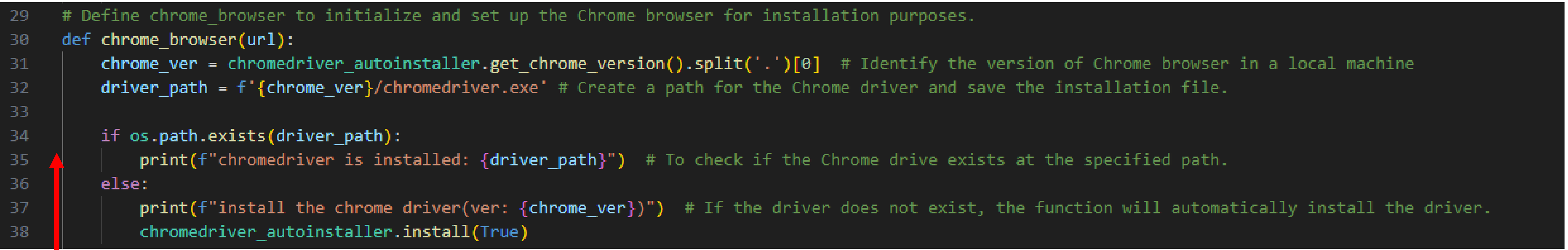
```python
for index,data in enumerate(dataList): # DataList is a list of dictionaries and 'data' is one of the dictionaries in the list, representing each document(RFP).
    print(f"{index+1}/{len(dataList)}번째 기사 상세정보 가져오기")
    result=GetDetails(data, browser, supabase, url)

     # This process is for checking the result by converting the result into json format.
    with open('result.json', 'w', encoding='utf-8') as f: #
        json.dump(result, f, ensure_ascii=False) # Convert the 'result' dictionary into the Json format using json.dump and write the file using 'f' object.

    # Upload result to Supabase.
    GetUploads(result)
```

1. We have already opened the ‘dataList’ file, and partial data extracted via an API call are stored.
2. The for loop iterates over data and retrieves the complete dataset using the ‘GetDetails’ function that is used when web scraping data using Selenium and bs4.
3. After extracting the full dataset combining partial data with the web scraping data, we store them in the ‘result’ object and convert it into a JSON file.
4. The result will be uploaded to Supabase by using the ‘GetUpload’ function in Python.



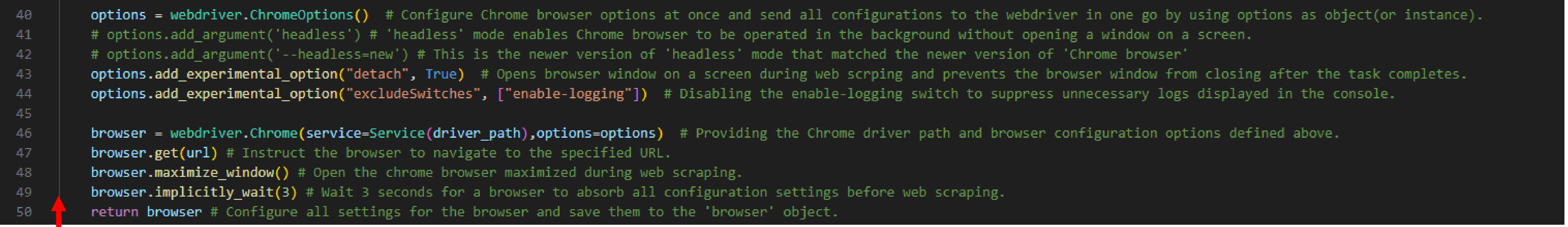
```python
# Open an empty list to combine the results of 'dataList', obtained by 'GetSearch' and the addtional details from 'dataMore', obtained by 'GetDetails'.
totalResult=[]
# Open a Chrome broswer for web scraping.
browser=chrome_browser('https://www.google.co.kr')
```

1. I made an empty list to combine the results of 'dataList', obtained by API call sending HTTP GET request to the Search API endpoint of the server with additional information obtained by Web scraping using Selenium's 'GetDetails' function.
2. I opened the Chrome browser using the 'browser' object, already defined above. Now, I am going to explain how to install ChromeDriver, which helps Selenium interact with Chrome browser in your local computer.



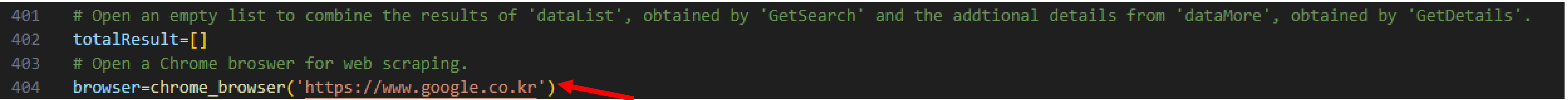
```python
# Define chrome_browser to initialize and set up the Chrome browser for installation purposes.
def chrome_browser(url):
    chrome_ver = chromedriver_autoinstaller.get_chrome_version().split('.')[0]  # Identify the version of Chrome browser in a local machine
    driver_path = f'{chrome_ver}/chromedriver.exe' # Create a path for the Chrome driver and save the installation file.

    if os.path.exists(driver_path):
        print(f"chromedriver is installed: {driver_path}")  # To check if the Chrome drive exists at the specified path.
    else:
        print(f"install the chrome driver(ver: {chrome_ver})")  # If the driver does not exist, the function will automatically install the driver.
        chromedriver_autoinstaller.install(True)
```

1. Define the 'chrome_browser' function and set up all configurations for the Chrome browser.
2. 'chromedriver_autoinstaller' function will get the version of Chrome installed in your local machine automatically and will index the major version.
3. We need a Chrome Driver to control the Chrome browser and helps Selenium interact with the browser.
4. After designating the path for 'chromedriver.exe' installation file, the 'if' code checks whether the Chrome Driver already exists at the specified path.
5. If not, 'chromedriver_autoinstaller' installs the Chrome Driver automatically that matches the version of Chrome installed in my local computer.
6. For instance, if the major version of Chrome installed is 95, the Chrome driver that matches the version of 95 is automatically installed.



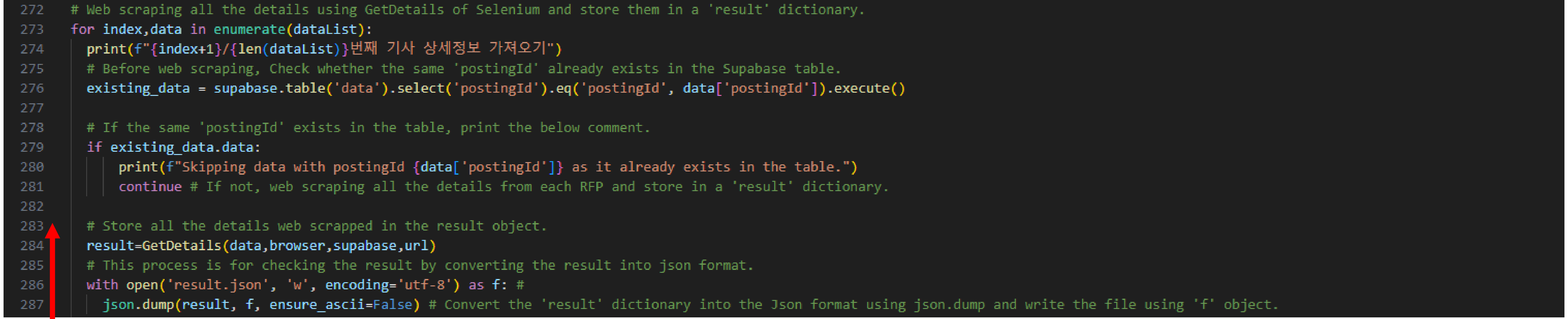
```python
    options = webdriver.ChromeOptions()  # Configure Chrome browser options at once and send all configurations to the webdriver in one go by using options as object(or instance).
    # options.add_argument('headless') # 'headless' mode enables Chrome browser to be operated in the background without opening a window on a screen.
    # options.add_argument('--headless=new') # This is the newer version of 'headless' mode that matched the newer version of 'Chrome browser'
    options.add_experimental_option("detach", True)  # Opens browser window on a screen during web scrping and prevents the browser window from closing after the task completes.
    options.add_experimental_option("excludeSwitches", ["enable-logging"])  # Disabling the enable-logging switch to suppress unnecessary logs displayed in the console.

    browser = webdriver.Chrome(service=Service(driver_path),options=options)  # Providing the Chrome driver path and browser configuration options defined above.
    browser.get(url) # Instruct the browser to navigate to the specified URL.
    browser.maximize_window() # Open the chrome browser maximized during web scraping.
    browser.implicitly_wait(3) # Wait 3 seconds for a browser to absorb all configuration settings before web scraping.
    return browser # Configure all settings for the browser and save them to the 'browser' object.
```

1. Configure all necessary settings for Chrome browser that we are going to control using Chrome Driver using a 'ChromeOptions' function in a 'webdriver' library in Selenium.
2. All configurations will be saved in 'options' object.
3. 'add_experimental_option' function sets up the Chrome browser opened on a screen during web scraping.("detach", True)
4. This function also disables 'enable-logging' switch to suppress unnecessary logs displayed in the VS CODE console when opening the Chrome browser.
5. After setting up the configurations, we provide the path for Chrome driver and all setting options to 'browser' object using 'Chrome' function in 'webdriver' library.
6. 'browser' also configures settings for itself, such as navigating to the specified URL, maximizing the window screen size during web scraping and waiting for 3 seconds for the web page being loaded.
7. All these settings will be saved into 'browser' object and then returned back.



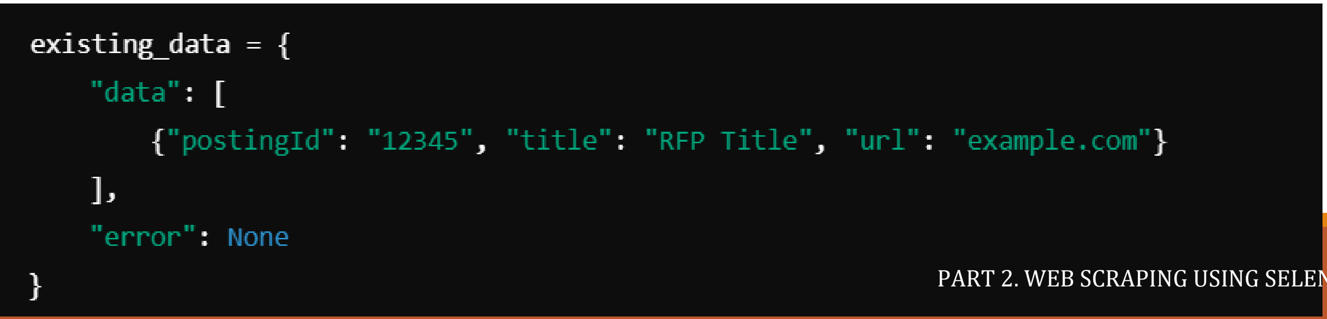
```python
# Open an empty list to combine the results of 'dataList', obtained by 'GetSearch' and the addtional details from 'dataMore', obtained by 'GetDetails'.
totalResult=[]
# Open a Chrome broswer for web scraping.
browser=chrome_browser('https://www.google.co.kr')
```

The 'browser' object is used when assigning a specific URL for web scraping.

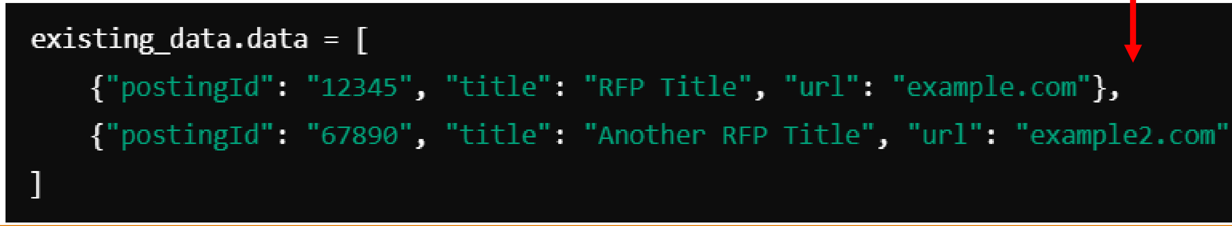


```python
# Web scraping all the details using GetDetails of Selenium and store them in a 'result' dictionary.
for index,data in enumerate(dataList):
    print(f"{index+1}/{len(dataList)}번째 기사 상세정보 가져오기")
    # Before web scraping, Check whether the same 'postingId' already exists in the Supabase table.
    existing_data = supabase.table('data').select('postingId').eq('postingId', data['postingId']).execute()

    # If the same 'postingId' exists in the table, print the below comment.
    if existing_data.data:
        print(f"Skipping data with postingId {data['postingId']} as it already exists in the table.")
        continue # If not, web scraping all the details from each RFP and store in a 'result' dictionary.

    # Store all the details web scrapped in the result object.
    result=GetDetails(data,browser,supabase,url)
    # This process is for checking the result by converting the result into json format.
    with open('result.json', 'w', encoding='utf-8') as f: #
        json.dump(result, f, ensure_ascii=False) # Convert the 'result' dictionary into the Json format using json.dump and write the file using 'f' object.
```

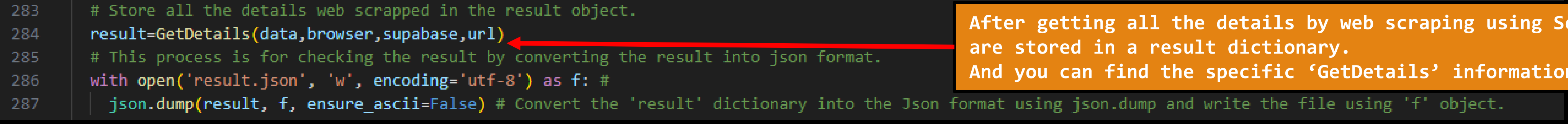
1. 'enumerate' function retrieves 'index' and 'data' dictionary in 'dataList' whose data we got from the server by API call.
2. And then print 'index+1' / the total number of data in dataList to indicate which RFP is currently being retrieved by web scraping.
3. 'existing_data' object will check whether the 'postingId' obtained by API call matches the one uploaded in a Supabase table.
4. (Below images for references) 'existing_data' retrieves data from the Supabase as a list, and there is 'data' key with 'postingID' of each RFP.
5. If the 'postingId' uploaded in Supabase matches the 'postingId' extracted from the server, it will skip the RFP.
6. If not, the code will continue and get details from the server by web scraping and store all the details in 'result' as a dictionary.



```
existing_data = {
    "data": [
        {"postingId": "12345", "title": "RFP Title", "url": "example.com"}
    ],
    "error": None
}
```

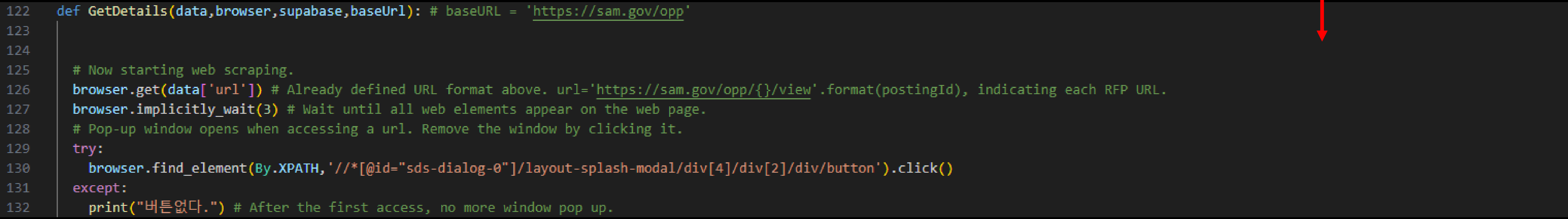


```
existing_data.data = [
    {"postingId": "12345", "title": "RFP Title", "url": "example.com"},
    {"postingId": "67890", "title": "Another RFP Title", "url": "example2.com"
]
```

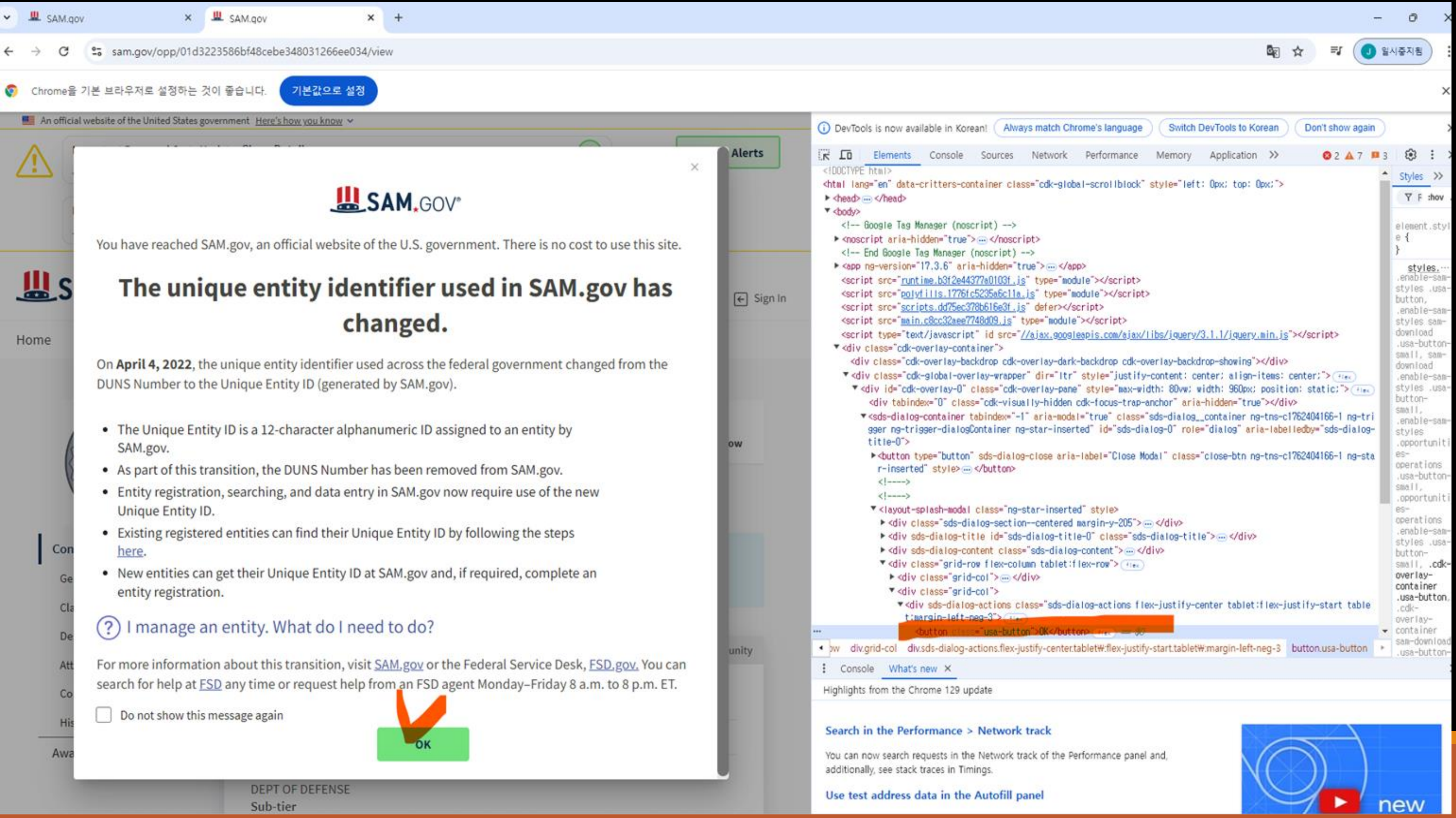


```python
    # Store all the details web scrapped in the result object.
    result=GetDetails(data,browser,supabase,url)
    # This process is for checking the result by converting the result into json format.
    with open('result.json', 'w', encoding='utf-8') as f: #
      json.dump(result, f, ensure_ascii=False) # Convert the 'result' dictionary into the Json format using json.dump and write the file using 'f' object.
```

After getting all the details by web scraping using Selenium, all the details are stored in a result dictionary.
And you can find the specific 'GetDetails' information from here.



```python
def GetDetails(data,browser,supabase,baseUrl): # baseURL = 'https://sam.gov/opp'


  # Now starting web scraping.
  browser.get(data['url']) # Already defined URL format above. url='https://sam.gov/opp/{}/view'.format(postingId), indicating each RFP URL.
  browser.implicitly_wait(3) # Wait until all web elements appear on the web page.
  # Pop-up window opens when accessing a url. Remove the window by clicking it.
  try:
    browser.find_element(By.XPATH,'//*[@id="sds-dialog-0"]/layout-splash-modal/div[4]/div[2]/div/button').click()
  except:
    print("버튼없다.") # After the first access, no more window pop up.
```



SAM.gov

You have reached SAM.gov, an official website of the U.S. government. There is no cost to use this site.

The unique entity identifier used in SAM.gov has changed.

On **April 4, 2022**, the unique entity identifier used across the federal government changed from the DUNS Number to the Unique Entity ID (generated by SAM.gov).

- The Unique Entity ID is a 12-character alphanumeric ID assigned to an entity by SAM.gov.
- As part of this transition, the DUNS Number has been removed from SAM.gov.
- Entity registration, searching, and data entry in SAM.gov now require use of the new Unique Entity ID.
- Existing registered entities can find their Unique Entity ID by following the steps here.
- New entities can get their Unique Entity ID at SAM.gov and, if required, complete an entity registration.

I manage an entity. What do I need to do?

For more information about this transition, visit SAM.gov or the Federal Service Desk, FSD.gov. You can search for help at FSD any time or request help from an FSD agent Monday–Friday 8 a.m. to 8 p.m. ET.

☐ Do not show this message again

OK

```
<!DOCTYPE html>
<html lang="en" data-critters-container class="cdk-global-scrollblock" style="left: 0px; top: 0px;">
 ▶<head> ⋯ </head>
 ▼<body>
   <!-- Google Tag Manager (noscript) -->
  ▶<noscript aria-hidden="true"> ⋯ </noscript>
   <!-- End Google Tag Manager (noscript) -->
  ▶<app ng-version="17.3.6" aria-hidden="true"> ⋯ </app>
   <script src="runtime.b3f2e44377a0103f.js" type="module"></script>
   <script src="polyfills.1776fc5235a6c11a.js" type="module"></script>
   <script src="scripts.dd75ec378b616e3f.js" defer></script>
   <script src="main.c8cc32aee7748d09.js" type="module"></script>
   <script type="text/javascript" id src="//ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  ▼<div class="cdk-overlay-container">
     <div class="cdk-overlay-backdrop cdk-overlay-dark-backdrop cdk-overlay-backdrop-showing"></div>
    ▼<div class="cdk-global-overlay-wrapper" dir="ltr" style="justify-content: center; align-items: center;"> flex
     ▼<div id="cdk-overlay-0" class="cdk-overlay-pane" style="max-width: 80vw; width: 960px; position: static;"> flex
        <div tabindex="0" class="cdk-visually-hidden cdk-focus-trap-anchor" aria-hidden="true"></div>
       ▼<sds-dialog-container tabindex="-1" aria-modal="true" class="sds-dialog__container ng-tns-c1762404166-1 ng-trigger ng-trigger-dialogContainer ng-star-inserted" id="sds-dialog-0" role="dialog" aria-labelledby="sds-dialog-title-0">
         ▶<button type="button" sds-dialog-close aria-label="Close Modal" class="close-btn ng-tns-c1762404166-1 ng-star-inserted" style> ⋯ </button>
          <!---->
          <!---->
         ▼<layout-splash-modal class="ng-star-inserted" style>
           ▶<div class="sds-dialog-section--centered margin-y-205"> ⋯ </div>
           ▶<div sds-dialog-title id="sds-dialog-title-0" class="sds-dialog-title"> ⋯ </div>
           ▶<div sds-dialog-content class="sds-dialog-content"> ⋯ </div>
           ▼<div class="grid-row flex-column tablet:flex-row"> flex
             ▶<div class="grid-col"> ⋯ </div>
             ▼<div class="grid-col">
               ▼<div sds-dialog-actions class="sds-dialog-actions flex-justify-center tablet:flex-justify-start tablet:margin-left-neg-3"> flex
                  <button class="usa-button">OK</button> flex == $0
```

Highlights from the Chrome 129 update

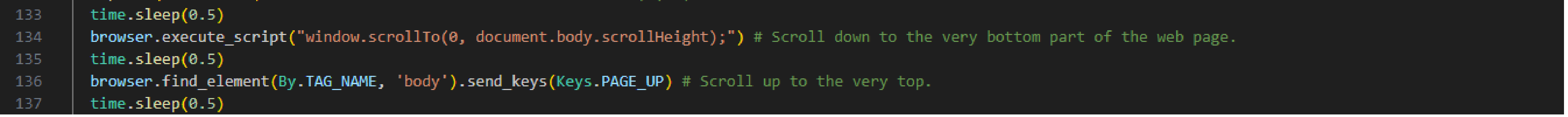
Search in the Performance > Network track

You can now search requests in the Network track of the Performance panel and, additionally, see stack traces in Timings.

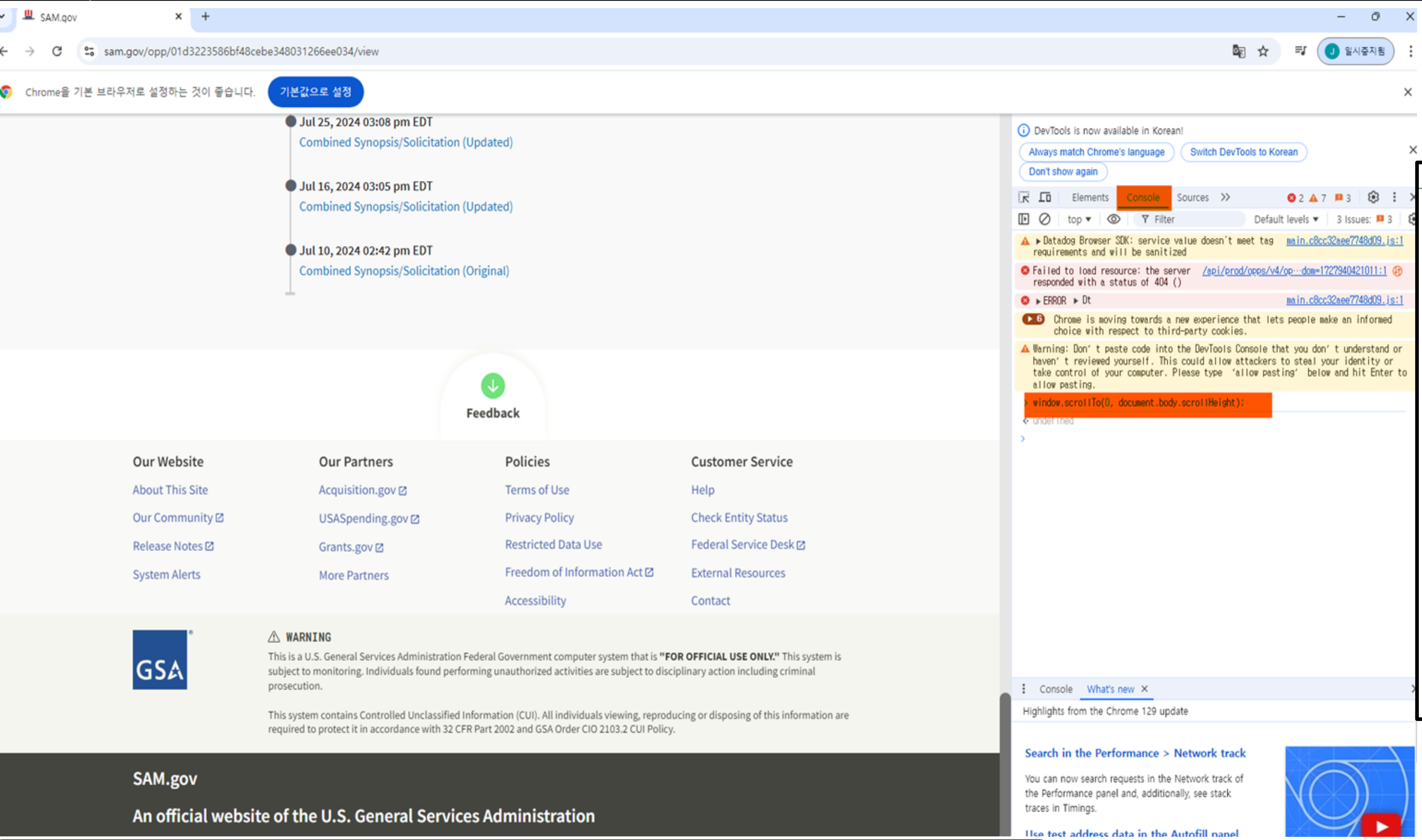
Use test address data in the Autofill panel

SAM.gov
sam.gov/opp/01d3223586bf48cebe348031266ee034/view
Chrome을 기본 브라우저로 설정하는 것이 좋습니다.
기본값으로 설정
An official website of the United States government Here's how you know
Alerts
Sign In
Home
DEPT OF DEFENSE
Sub-tier
DevTools is now available in Korean!
Always match Chrome's language
Switch DevTools to Korean
Don't show again
Elements
Console
Sources
Network
Performance
Memory
Application
Styles
div.grid-col
div.sds-dialog-actions.flex-justify-center.tablet₩:flex-justify-start.tablet₩:margin-left-neg-3
button.usa-button
What's new
new

1. **Define 'GetDetails' function to web scrape data by using Selenium.**
2. Provide 'url' info, retrieved by API call, to access each RPF.
3. Wait until all web elements on the web page have loaded for 3 seconds.
4. When accessing the first RFP, **a small pop up window appears**.
5. **Find the path of the 'OK' button in a DevTools**, copy the 'XPATH' of it and execute 'click' action to close the window.
6. However, when accessing the 2nd RFP on the same window, I found that no more pop-ups appeared. I added **'except'** command to avoid any error when a button is not found.
7. **'버튼없다' means 'no button'.**



```python
time.sleep(0.5)
browser.execute_script("window.scrollTo(0, document.body.scrollHeight);") # Scroll down to the very bottom part of the web page.
time.sleep(0.5)
browser.find_element(By.TAG_NAME, 'body').send_keys(Keys.PAGE_UP) # Scroll up to the very top.
time.sleep(0.5)
```



sam.gov/opp/01d3223586bf48cebe348031266ee034/view

Chrome을 기본 브라우저로 설정하는 것이 좋습니다. 기본값으로 설정

Jul 25, 2024 03:08 pm EDT
Combined Synopsis/Solicitation (Updated)

Jul 16, 2024 03:05 pm EDT
Combined Synopsis/Solicitation (Updated)

Jul 10, 2024 02:42 pm EDT
Combined Synopsis/Solicitation (Original)

Feedback

Our Website	Our Partners	Policies	Customer Service
About This Site	Acquisition.gov	Terms of Use	Help
Our Community	USASpending.gov	Privacy Policy	Check Entity Status
Release Notes	Grants.gov	Restricted Data Use	Federal Service Desk
System Alerts	More Partners	Freedom of Information Act	External Resources
		Accessibility	Contact

WARNING
This is a U.S. General Services Administration Federal Government computer system that is "FOR OFFICIAL USE ONLY." This system is subject to monitoring. Individuals found performing unauthorized activities are subject to disciplinary action including criminal prosecution.

This system contains Controlled Unclassified Information (CUI). All individuals viewing, reproducing or disposing of this information are required to protect it in accordance with 32 CFR Part 2002 and GSA Order CIO 2103.2 CUI Policy.

SAM.gov
An official website of the U.S. General Services Administration

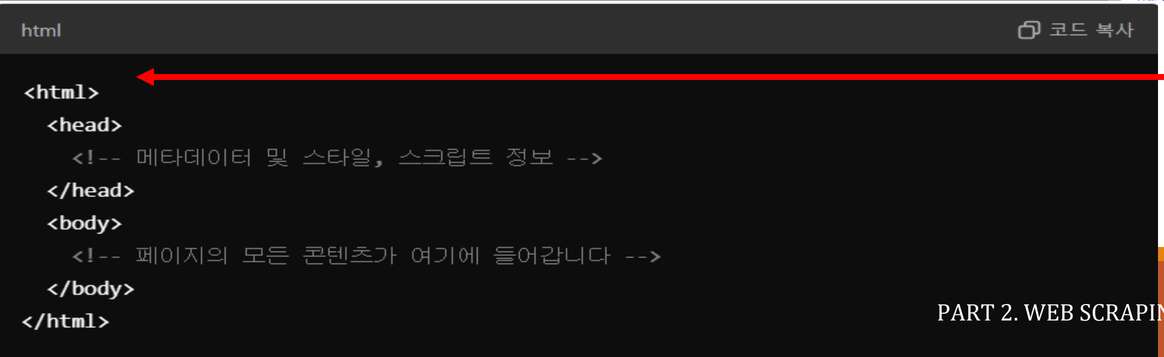
DevTools is now available in Korean!
Always match Chrome's language | Switch DevTools to Korean | Don't show again

Elements | Console | Sources

```
▸ Datadog Browser SDK: service value doesn't meet tag requirements and will be sanitized    main.c8cc32aee7748d09.js:1
Failed to load resource: the server responded with a status of 404 ()    /api/prod/opps/v4/op…dom=1727940421011:1
▸ ERROR ▸ Dt    main.c8cc32aee7748d09.js:1
Chrome is moving towards a new experience that lets people make an informed choice with respect to third-party cookies.
Warning: Don't paste code into the DevTools Console that you don't understand or haven't reviewed yourself. This could allow attackers to steal your identity or take control of your computer. Please type 'allow pasting' below and hit Enter to allow pasting.
> window.scrollTo(0, document.body.scrollHeight);
< undefined
```

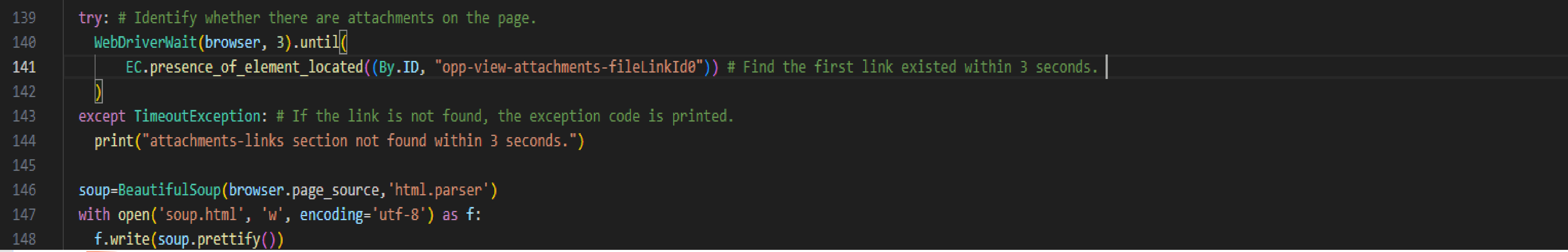
Console | What's new
Highlights from the Chrome 129 update
Search in the Performance > Network track
You can now search requests in the Network track of the Performance panel and, additionally, see stack traces in Timings.

1. The 'execute.script' function is used in Selenium to execute JavaScript commands for controlling dynamic actions like scrolling.
2. The function controls dynamic actions like scrolling down to the very bottom part.
3. You can input "window.scrollTo(0, document.body.scrollHeight);" command in the Console of the DevTools to check whether or not it works.
4. 'find_element' is used to find elements like text and images on the webpage. Here, you can find a 'body' tag using 'By' and then send a key 'Page up' to scroll up to the very top.

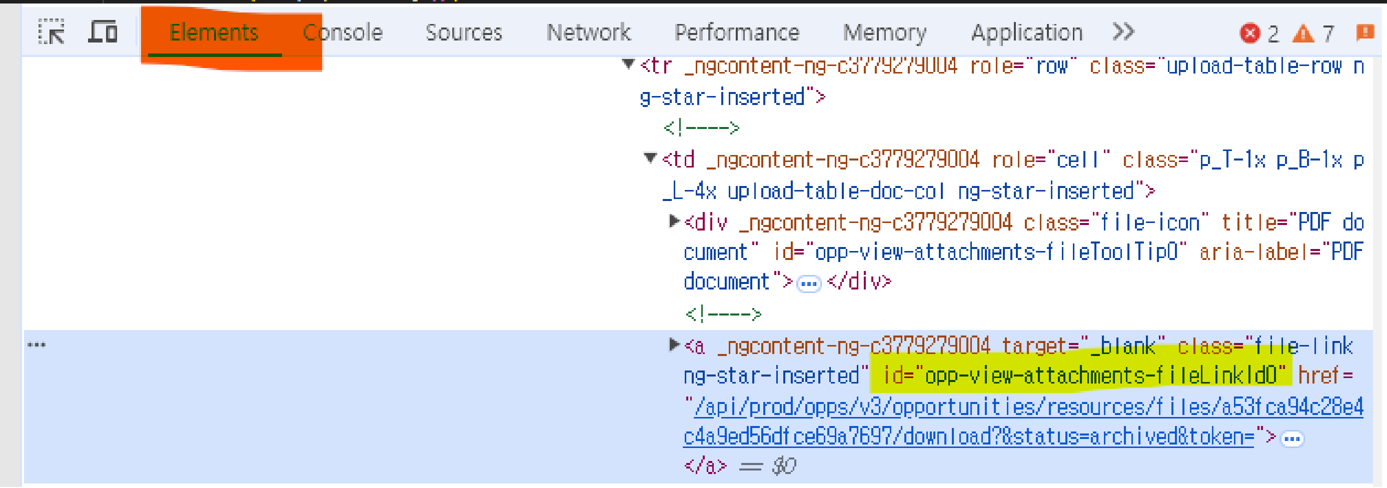


```html
<html>
  <head>
    <!-- 메타데이터 및 스타일, 스크립트 정보 -->
  </head>
  <body>
    <!-- 페이지의 모든 콘텐츠가 여기에 들어갑니다 -->
  </body>
</html>
```

<head> is one of the tags in the HTML of the web page, indicating the whole web page where texts, images, links and buttons are placed.



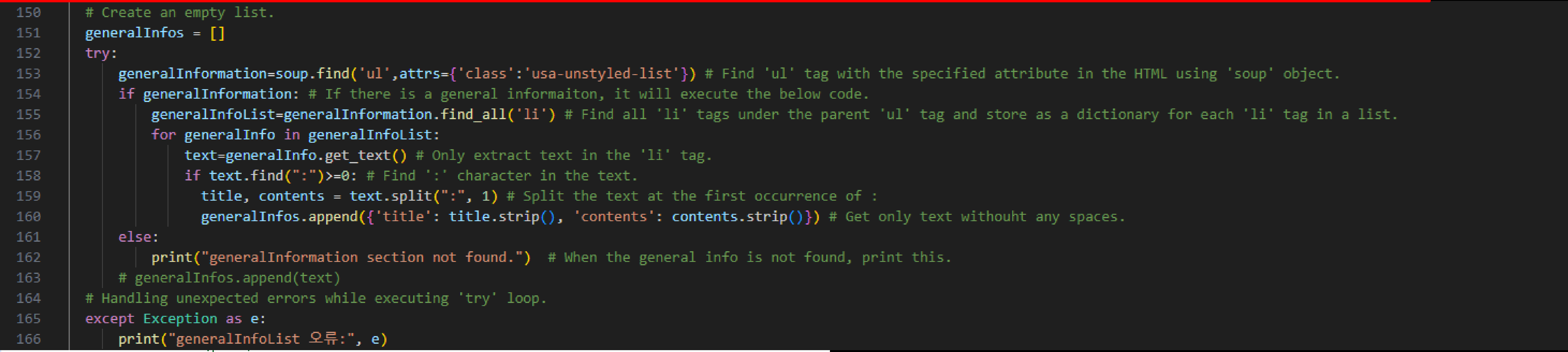
```python
try: # Identify whether there are attachments on the page.
  WebDriverWait(browser, 3).until(
      EC.presence_of_element_located((By.ID, "opp-view-attachments-fileLinkId0")) # Find the first link existed within 3 seconds.
  )
except TimeoutException: # If the link is not found, the exception code is printed.
  print("attachments-links section not found within 3 seconds.")

soup=BeautifulSoup(browser.page_source,'html.parser')
with open('soup.html', 'w', encoding='utf-8') as f:
  f.write(soup.prettify())
```



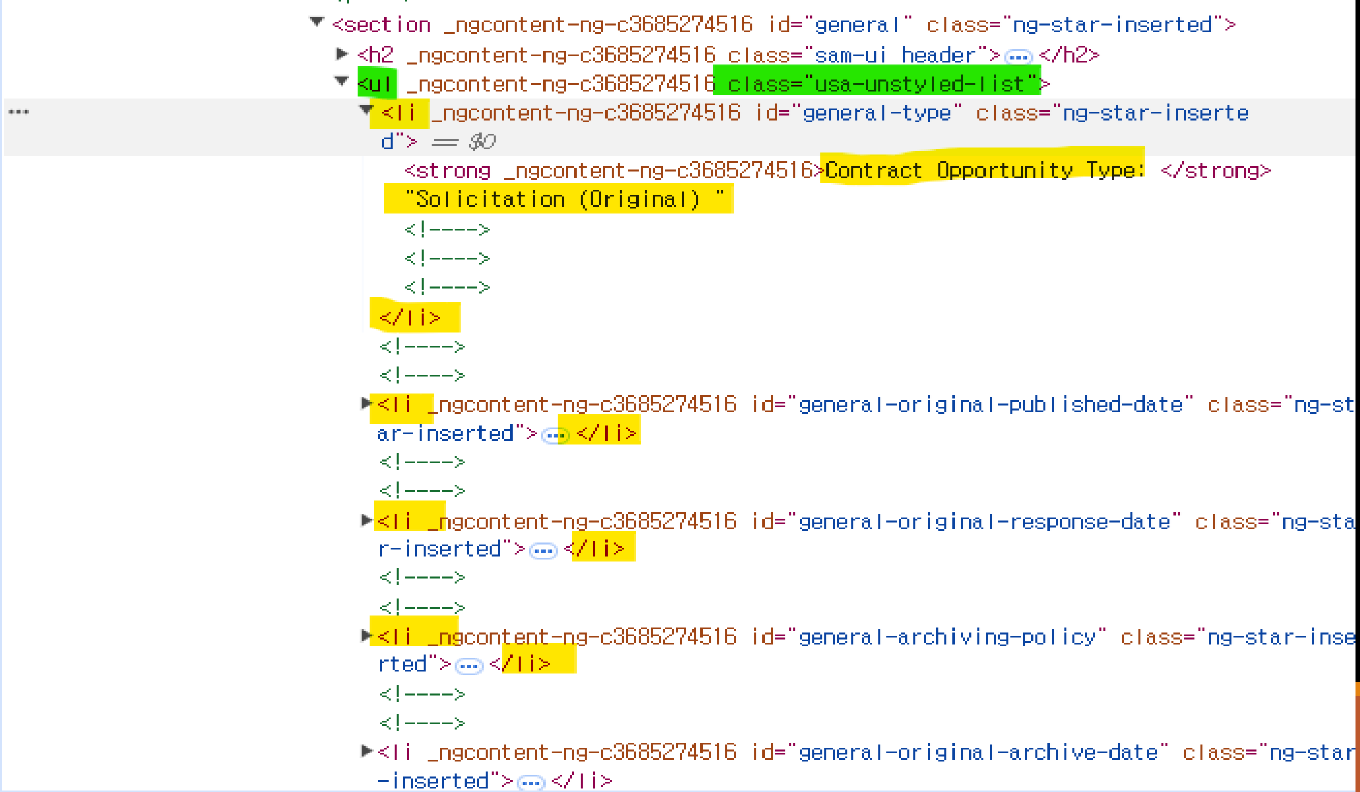
```html
▼<tr _ngcontent-ng-c3779279004 role="row" class="upload-table-row ng-star-inserted">
   <!---->
  ▼<td _ngcontent-ng-c3779279004 role="cell" class="p_T-1x p_B-1x p_L-4x upload-table-doc-col ng-star-inserted">
    ▶<div _ngcontent-ng-c3779279004 class="file-icon" title="PDF document" id="opp-view-attachments-fileToolTip0" aria-label="PDF document">…</div>
     <!---->
    ▶<a _ngcontent-ng-c3779279004 target="_blank" class="file-link ng-star-inserted" id="opp-view-attachments-fileLinkId0" href="/api/prod/opps/v3/opportunities/resources/files/a53fca94c28e4c4a9ed56dfce69a7697/download?&status=archived&token=">…
     </a> == $0
```

1. The try-except command is used to let the browser pause for at least 3 seconds to find the first link of attachments located on the web page within an <a> (anchor) tag that includes the ‘id’ element for the first link.
2. I used ‘Beautifulsoup’ for web scraping while controlling the Chrome browser using Selenium.
3. In the ‘Beautifulsoup’ library, ‘html.parser’ function can be used to scrape the entire HTML source code of the loaded web page.
4. Browser.page_source indicates the entire HTML source code of the loaded web page.
5. The HTML source code is stored in a ‘soup’ object as soup.html, and made well-organized using the ‘prettify’ function.

From now on, I am going to web scrape every detail for each RFP using ‘beautifulsoup’.

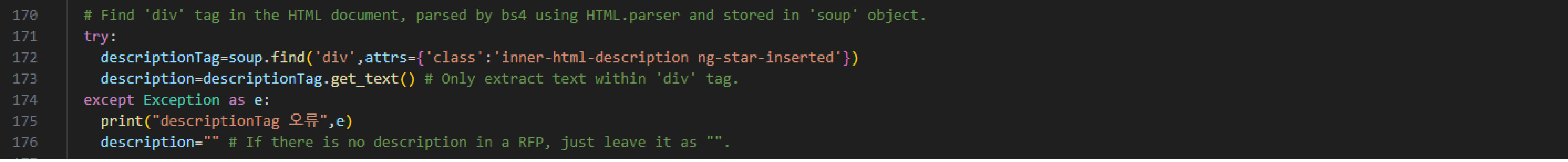


```python
# Create an empty list.
generalInfos = []
try:
    generalInformation=soup.find('ul',attrs={'class':'usa-unstyled-list'}) # Find 'ul' tag with the specified attribute in the HTML using 'soup' object.
    if generalInformation: # If there is a general informaiton, it will execute the below code.
        generalInfoList=generalInformation.find_all('li') # Find all 'li' tags under the parent 'ul' tag and store as a dictionary for each 'li' tag in a list.
        for generalInfo in generalInfoList:
            text=generalInfo.get_text() # Only extract text in the 'li' tag.
            if text.find(":")>=0: # Find ':' character in the text.
              title, contents = text.split(":", 1) # Split the text at the first occurrence of :
              generalInfos.append({'title': title.strip(), 'contents': contents.strip()}) # Get only text withouht any spaces.
    else:
        print("generalInformation section not found.")  # When the general info is not found, print this.
    # generalInfos.append(text)
# Handling unexpected errors while executing 'try' loop.
except Exception as e:
    print("generalInfoList 오류:", e)
```

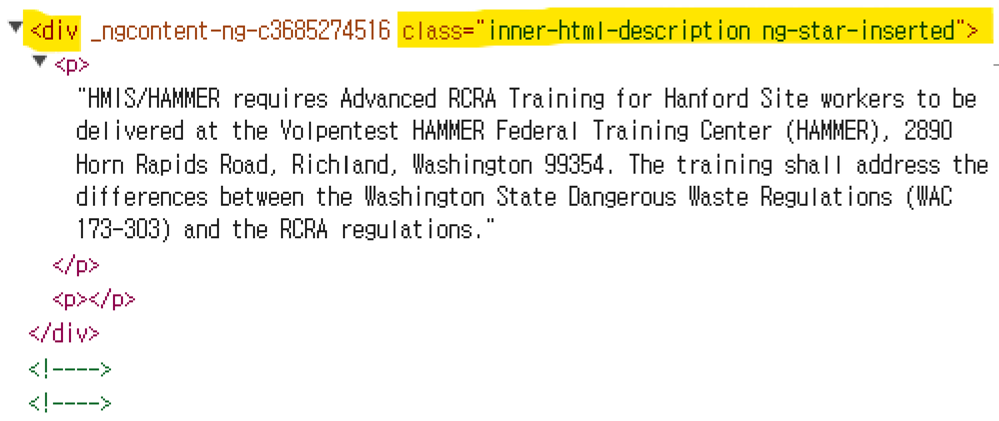


```html
▼<section _ngcontent-ng-c3685274516 id="general" class="ng-star-inserted">
  ▶<h2 _ngcontent-ng-c3685274516 class="sam-ui header">…</h2>
  ▼<ul _ngcontent-ng-c3685274516 class="usa-unstyled-list">
    ▼<li _ngcontent-ng-c3685274516 id="general-type" class="ng-star-inserte
     d"> == $0
        <strong _ngcontent-ng-c3685274516>Contract Opportunity Type: </strong>
       "Solicitation (Original) "
        <!---->
        <!---->
        <!---->
     </li>
     <!---->
     <!---->
    ▶<li _ngcontent-ng-c3685274516 id="general-original-published-date" class="ng-st
     ar-inserted">…</li>
     <!---->
     <!---->
    ▶<li _ngcontent-ng-c3685274516 id="general-original-response-date" class="ng-sta
     r-inserted">…</li>
     <!---->
     <!---->
    ▶<li _ngcontent-ng-c3685274516 id="general-archiving-policy" class="ng-star-inse
     rted">…</li>
     <!---->
     <!---->
    ▶<li _ngcontent-ng-c3685274516 id="general-original-archive-date" class="ng-star
     -inserted">…</li>
```

1. Find ‘ul’ (unordered list) tags in the HTML structure which includes ‘General Information’ details on the web page, and store them in ‘generalInformation’ object.
2. Under the ‘ul’ tags, find all ‘li’ tags which are the child tags of the ‘ul’ tag and store them in ‘generalInfoList’.
3. Under the ‘li’ tag, there is text data which consists of the exact details for web scraping.
4. Find ‘:’ in the text and split the text based on the character and store them in a dictionary.
5. Add all dictionaries to the ‘generalInfos’ List.

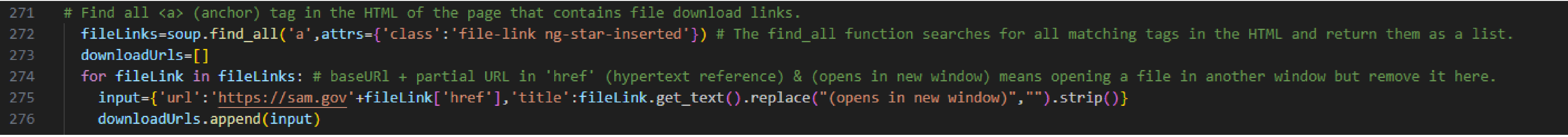


```python
# Find 'div' tag in the HTML document, parsed by bs4 using HTML.parser and stored in 'soup' object.
try:
  descriptionTag=soup.find('div',attrs={'class':'inner-html-description ng-star-inserted'})
  description=descriptionTag.get_text() # Only extract text within 'div' tag.
except Exception as e:
  print("descriptionTag 오류",e)
  description="" # If there is no description in a RFP, just leave it as "".
```



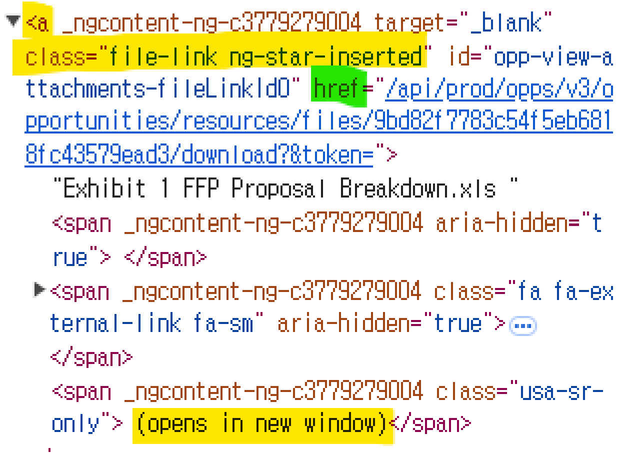
```html
▼<div _ngcontent-ng-c3685274516 class="inner-html-description ng-star-inserted">
  ▼<p>
     "HMIS/HAMMER requires Advanced RCRA Training for Hanford Site workers to be
     delivered at the Volpentest HAMMER Federal Training Center (HAMMER), 2890
     Horn Rapids Road, Richland, Washington 99354. The training shall address the
     differences between the Washington State Dangerous Waste Regulations (WAC
     173-303) and the RCRA regulations."
   </p>
   <p></p>
 </div>
 <!---->
 <!---->
```

1. Find ‘div’ tag with ‘class’ element equal to ‘inner-html-description ng-star-inserted.’ that contains all details in ‘description’ for each RFP, and store them in ‘descriptiontag’ object.
2. Under the ‘div’ tag, there is the exact description details that we need to web scrape.
3. Use ‘get_text’ function.
4. If there is no description details on a specific RFP, leave it as a blank with an empty string (“”).



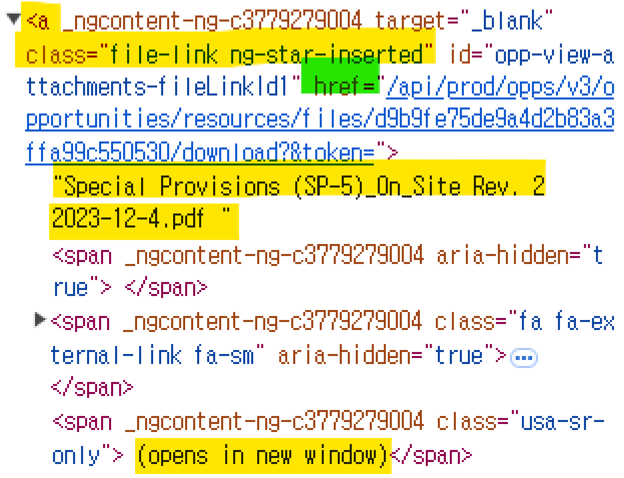
```python
# Find all <a> (anchor) tag in the HTML of the page that contains file download links.
  fileLinks=soup.find_all('a',attrs={'class':'file-link ng-star-inserted'}) # The find_all function searches for all matching tags in the HTML and return them as a list.
  downloadUrls=[]
  for fileLink in fileLinks: # baseURl + partial URL in 'href' (hypertext reference) & (opens in new window) means opening a file in another window but remove it here.
    input={'url':'https://sam.gov'+fileLink['href'],'title':fileLink.get_text().replace("(opens in new window)","").strip()}
    downloadUrls.append(input)
```

First File



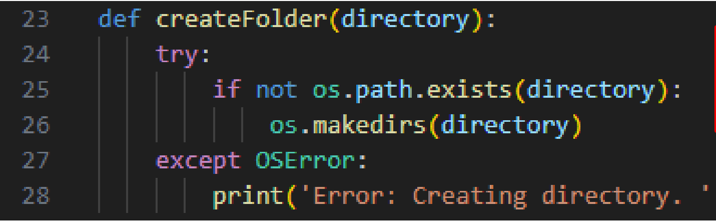
```html
▼<a _ngcontent-ng-c3779279004 target="_blank" class="file-link ng-star-inserted" id="opp-view-attachments-fileLinkId0" href="/api/prod/opps/v3/opportunities/resources/files/9bd82f7783c54f5eb6818fc43579ead3/download?&token=">
   "Exhibit 1 FFP Proposal Breakdown.xls "
   <span _ngcontent-ng-c3779279004 aria-hidden="true"> </span>
 ▶<span _ngcontent-ng-c3779279004 class="fa fa-external-link fa-sm" aria-hidden="true"> ⋯
   </span>
   <span _ngcontent-ng-c3779279004 class="usa-sr-only"> (opens in new window)</span>
```

Second File



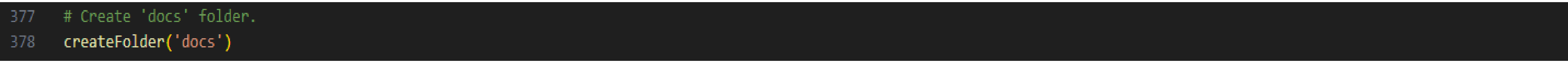
```html
▼<a _ngcontent-ng-c3779279004 target="_blank" class="file-link ng-star-inserted" id="opp-view-attachments-fileLinkId1" href="/api/prod/opps/v3/opportunities/resources/files/d9b9fe75de9a4d2b83a3ffa99c550530/download?&token=">
   "Special Provisions (SP-5)_On_Site Rev. 2
   2023-12-4.pdf "
   <span _ngcontent-ng-c3779279004 aria-hidden="true"> </span>
 ▶<span _ngcontent-ng-c3779279004 class="fa fa-external-link fa-sm" aria-hidden="true"> ⋯
   </span>
   <span _ngcontent-ng-c3779279004 class="usa-sr-only"> (opens in new window)</span>
```

1. Find the ‘a’ (anchor) tag with class ‘file-link-ng-star-inserted’ and store the HTML structure into the ‘fileLinks’ object.
2. There will be multiple <a> tags with the same class if there are several attachments on a specific RFP.
3. In each file link of a specific RFP, extract the ‘href’ key to combine it with the base URL.
4. Extract the title using ‘get.text’ function and remove unnecessary text such as “(opens in new window)” by replacing it with an empty string “”.

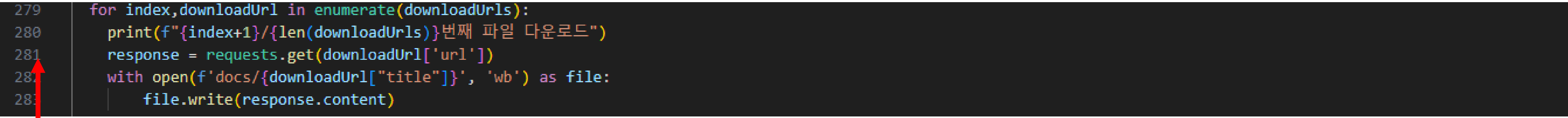


```python
def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except OSError:
        print('Error: Creating directory. ' + directory)
```

1. Create a new ‘docs’ folder in your current directory where the script is running.
2. If the ‘docs’ folder does not exist in the current directory, it creates a new folder.

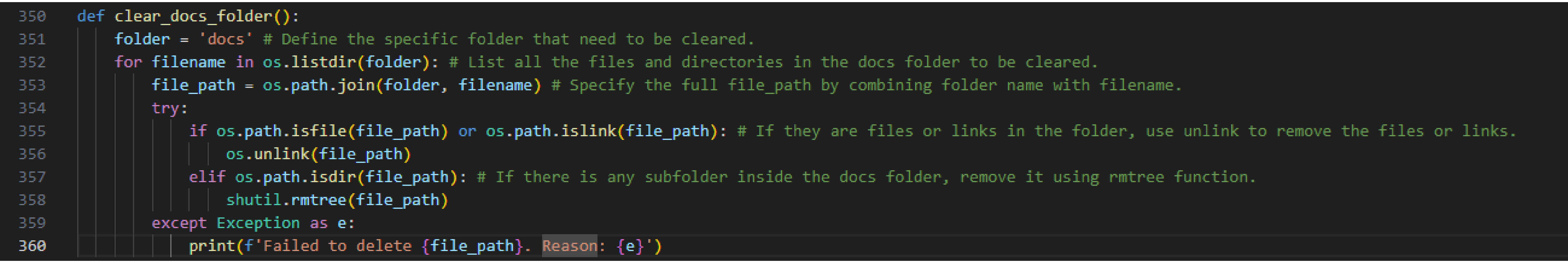


```python
# Create 'docs' folder.
createFolder('docs')
```

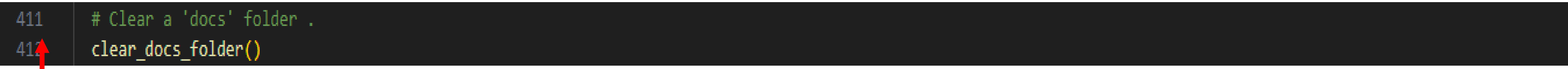


```python
    for index,downloadUrl in enumerate(downloadUrls):
      print(f"{index+1}/{len(downloadUrls)}번째 파일 다운로드")
      response = requests.get(downloadUrl['url'])
      with open(f'docs/{downloadUrl["title"]}', 'wb') as file:
          file.write(response.content)
```

1. It sends an API call to get each file downloaded into the newly created ‘docs’ folder. Then the files are stored in the ‘response’ object.
2. In the ‘docs’ folder, there will be downloaded files, and in the back end, the ‘file’ handler opens each file and writes the content in a binary format.

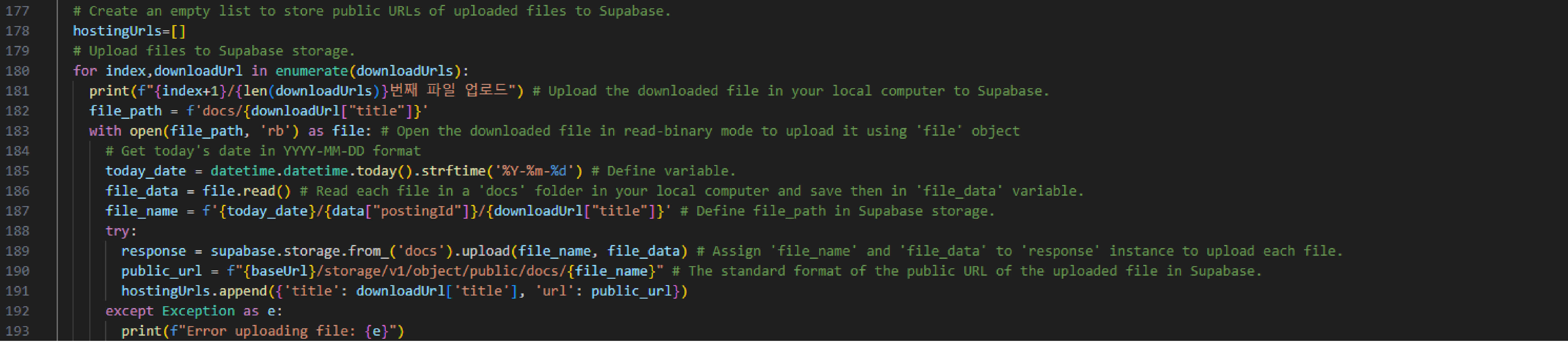


```python
def clear_docs_folder():
    folder = 'docs' # Define the specific folder that need to be cleared.
    for filename in os.listdir(folder): # List all the files and directories in the docs folder to be cleared.
        file_path = os.path.join(folder, filename) # Specify the full file_path by combining folder name with filename.
        try:
            if os.path.isfile(file_path) or os.path.islink(file_path): # If they are files or links in the folder, use unlink to remove the files or links.
                os.unlink(file_path)
            elif os.path.isdir(file_path): # If there is any subfolder inside the docs folder, remove it using rmtree function.
                shutil.rmtree(file_path)
        except Exception as e:
            print(f'Failed to delete {file_path}. Reason: {e}')
```

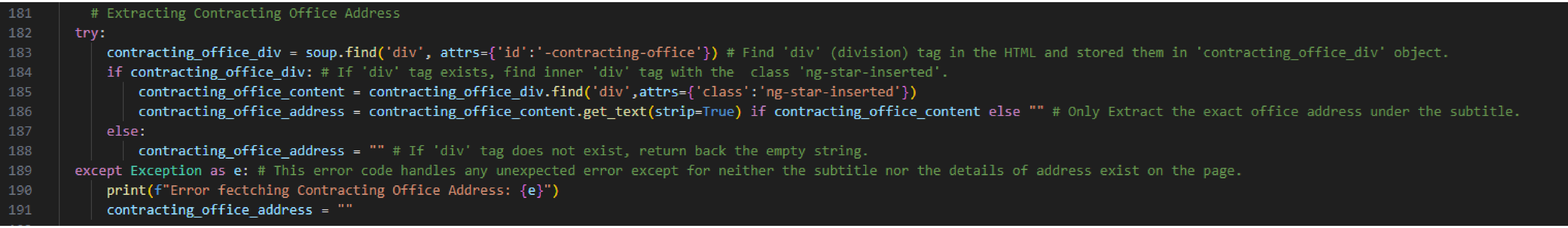


```python
# Clear a 'docs' folder .
clear_docs_folder()
```

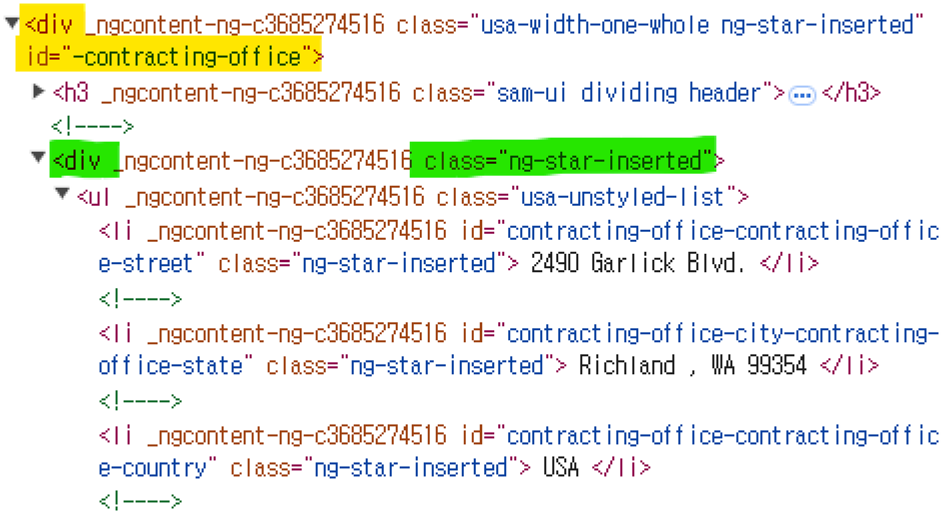
1. After getting details from the web page and uploading all the details to Supabase, delete the files downloaded into the ‘docs’ folder to avoid any capacity issue.


```
    # Create an empty list to store public URLs of uploaded files to Supabase.
    hostingUrls=[]
    # Upload files to Supabase storage.
    for index,downloadUrl in enumerate(downloadUrls):
      print(f"{index+1}/{len(downloadUrls)}번째 파일 업로드") # Upload the downloaded file in your local computer to Supabase.
      file_path = f'docs/{downloadUrl["title"]}'
      with open(file_path, 'rb') as file: # Open the downloaded file in read-binary mode to upload it using 'file' object
        # Get today's date in YYYY-MM-DD format
        today_date = datetime.datetime.today().strftime('%Y-%m-%d') # Define variable.
        file_data = file.read() # Read each file in a 'docs' folder in your local computer and save then in 'file_data' variable.
        file_name = f'{today_date}/{data["postingId"]}/{downloadUrl["title"]}' # Define file_path in Supabase storage.
        try:
          response = supabase.storage.from_('docs').upload(file_name, file_data) # Assign 'file_name' and 'file_data' to 'response' instance to upload each file.
          public_url = f"{baseUrl}/storage/v1/object/public/docs/{file_name}" # The standard format of the public URL of the uploaded file in Supabase.
          hostingUrls.append({'title': downloadUrl['title'], 'url': public_url})
        except Exception as e:
          print(f"Error uploading file: {e}")
```

1. ‘hostingURLs’ are used to upload the downloaded files to Supabase and retrieve these files from Supabase.
2. ‘번째 파일 업로드’ translates to ‘uploading the downloaded files.’
3. Open the downloaded files in read-binary mode using the ‘file’ handler.
4. Specify the file path in the ‘storage’ of Supabase.
5. I have already created a new ‘docs’ bucket in Supabase and read all the file data stored in the ‘docs’ folder on my local computer.
6. The public URL is the same as the hostingURL that is used in Supabase when downloading the files.



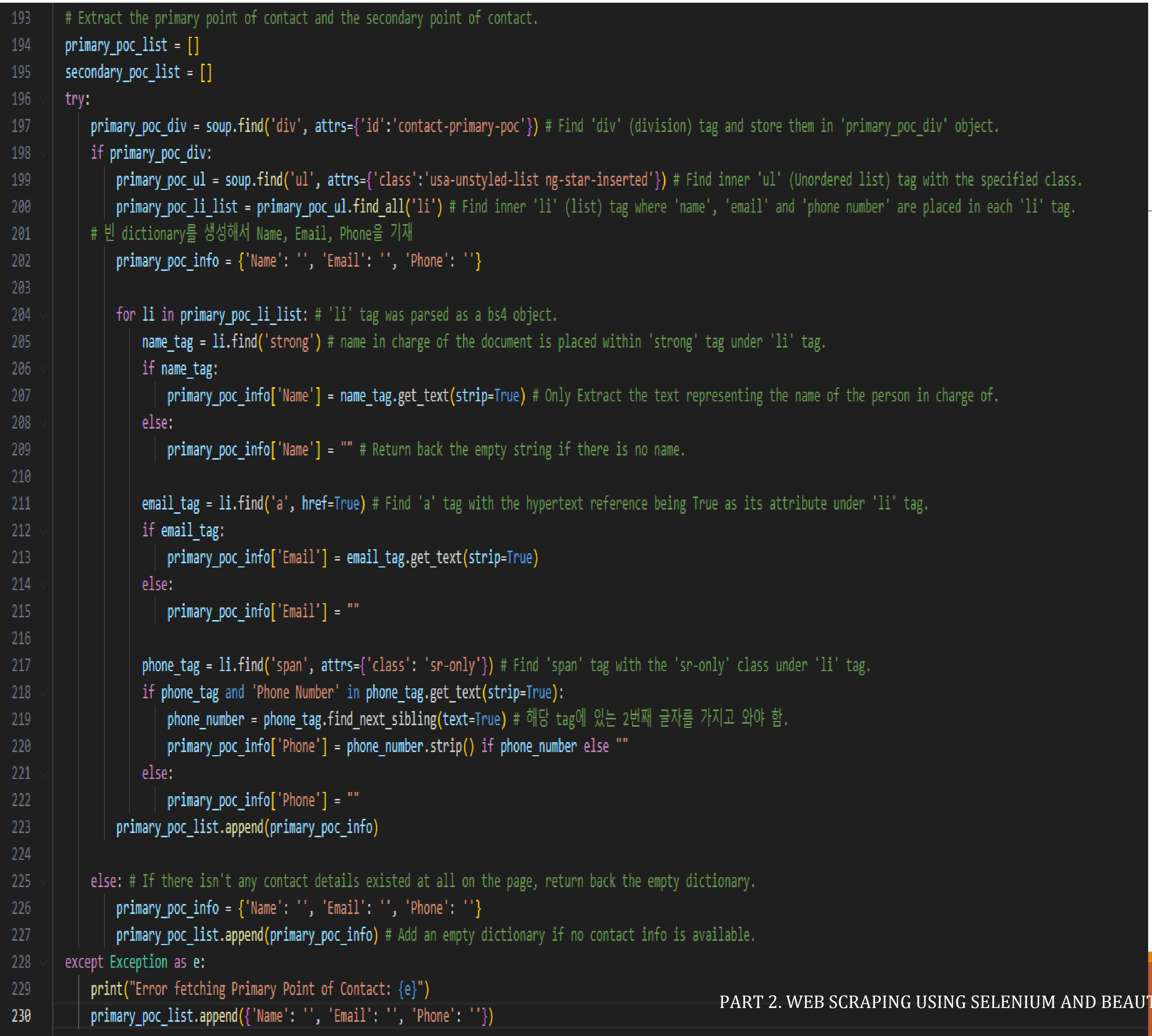
```python
 # Extracting Contracting Office Address
try:
    contracting_office_div = soup.find('div', attrs={'id':'-contracting-office'}) # Find 'div' (division) tag in the HTML and stored them in 'contracting_office_div' object.
    if contracting_office_div: # If 'div' tag exists, find inner 'div' tag with the  class 'ng-star-inserted'.
        contracting_office_content = contracting_office_div.find('div',attrs={'class':'ng-star-inserted'})
        contracting_office_address = contracting_office_content.get_text(strip=True) if contracting_office_content else "" # Only Extract the exact office address under the subtitle.
    else:
        contracting_office_address = "" # If 'div' tag does not exist, return back the empty string.
except Exception as e: # This error code handles any unexpected error except for neither the subtitle nor the details of address exist on the page.
    print(f"Error fectching Contracting Office Address: {e}")
    contracting_office_address = ""
```



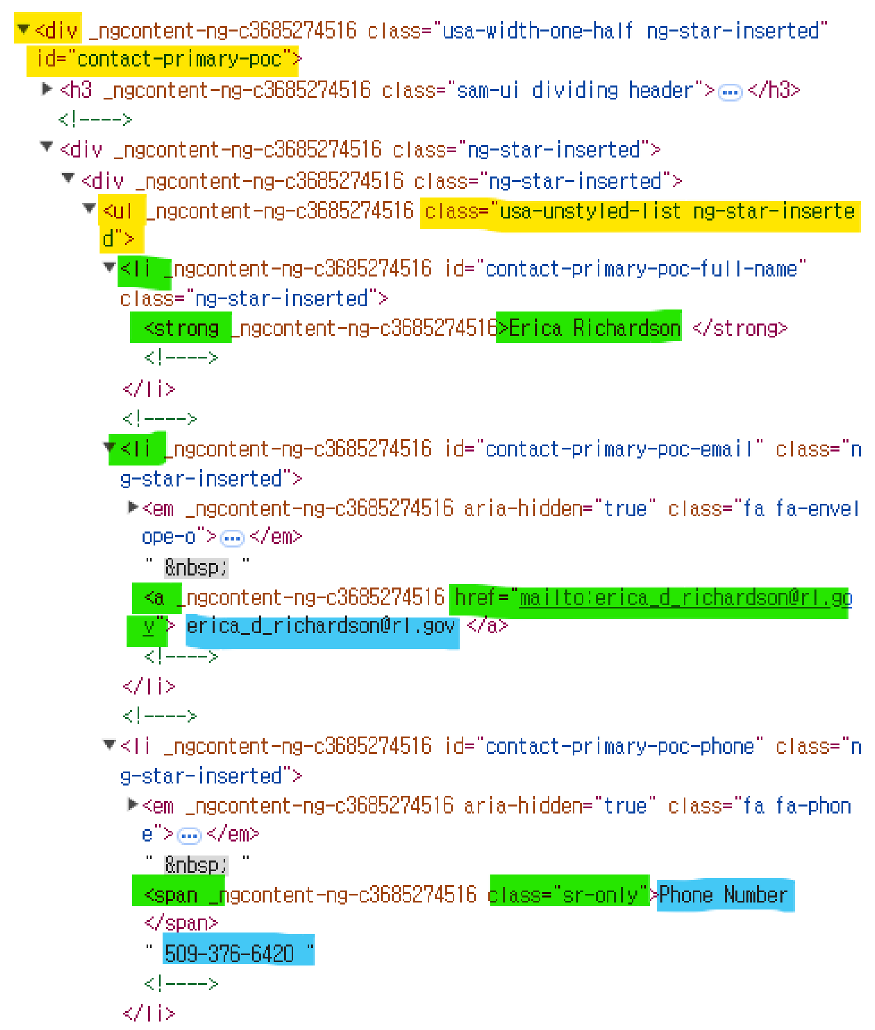
```html
▼<div _ngcontent-ng-c3685274516 class="usa-width-one-whole ng-star-inserted"
 id="-contracting-office">
  ▶<h3 _ngcontent-ng-c3685274516 class="sam-ui dividing header">…</h3>
  <!---->
  ▼<div _ngcontent-ng-c3685274516 class="ng-star-inserted">
    ▼<ul _ngcontent-ng-c3685274516 class="usa-unstyled-list">
       <li _ngcontent-ng-c3685274516 id="contracting-office-contracting-offic
       e-street" class="ng-star-inserted"> 2490 Garlick Blvd. </li>
       <!---->
       <li _ngcontent-ng-c3685274516 id="contracting-office-city-contracting-
       office-state" class="ng-star-inserted"> Richland , WA 99354 </li>
       <!---->
       <li _ngcontent-ng-c3685274516 id="contracting-office-contracting-offic
       e-country" class="ng-star-inserted"> USA </li>
       <!---->
```

1. Find the ‘div’ tag in the HTML of the web page with the ‘id’ attribute set to ‘-contracting-office’ and store it in the ‘contracting_office_div’ object.
2. After retrieving the HTML structure, find another ‘div’ tag with the class ‘ng-star-inserted’ that contains the contact address details.
3. If a sub ‘div’ tag under the main tag exists, use the ‘get_text’ function to retrieve the address details.

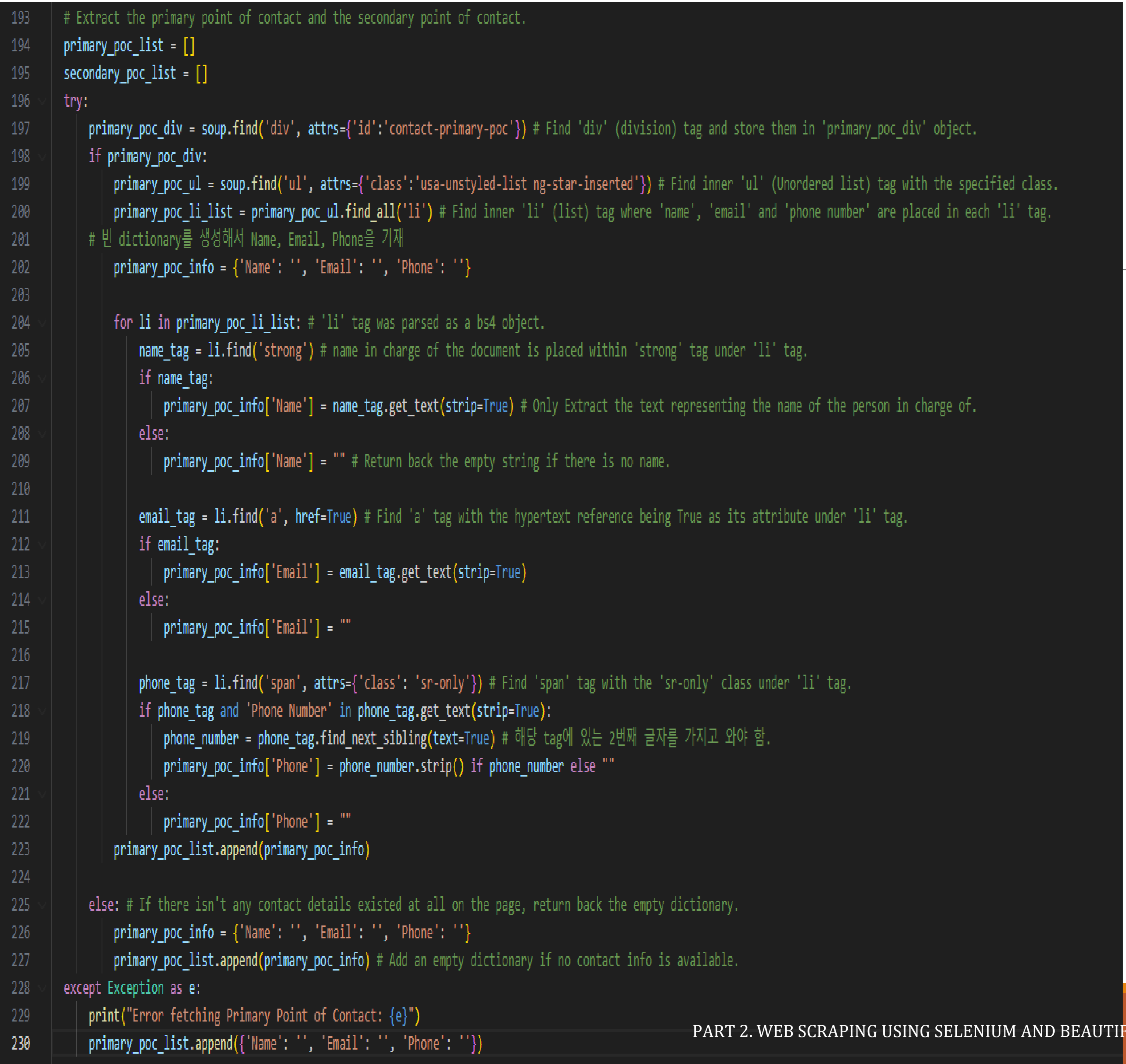
Find all tag details necessary for web scraping the point of contact details.



```python
# Extract the primary point of contact and the secondary point of contact.
primary_poc_list = []
secondary_poc_list = []
try:
    primary_poc_div = soup.find('div', attrs={'id':'contact-primary-poc'}) # Find 'div' (division) tag and store them in 'primary_poc_div' object.
    if primary_poc_div:
        primary_poc_ul = soup.find('ul', attrs={'class':'usa-unstyled-list ng-star-inserted'}) # Find inner 'ul' (Unordered list) tag with the specified class.
        primary_poc_li_list = primary_poc_ul.find_all('li') # Find inner 'li' (list) tag where 'name', 'email' and 'phone number' are placed in each 'li' tag.
    # 빈 dictionary를 생성해서 Name, Email, Phone을 기재
        primary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}

        for li in primary_poc_li_list: # 'li' tag was parsed as a bs4 object.
            name_tag = li.find('strong') # name in charge of the document is placed within 'strong' tag under 'li' tag.
            if name_tag:
                primary_poc_info['Name'] = name_tag.get_text(strip=True) # Only Extract the text representing the name of the person in charge of.
            else:
                primary_poc_info['Name'] = "" # Return back the empty string if there is no name.

            email_tag = li.find('a', href=True) # Find 'a' tag with the hypertext reference being True as its attribute under 'li' tag.
            if email_tag:
                primary_poc_info['Email'] = email_tag.get_text(strip=True)
            else:
                primary_poc_info['Email'] = ""

            phone_tag = li.find('span', attrs={'class': 'sr-only'}) # Find 'span' tag with the 'sr-only' class under 'li' tag.
            if phone_tag and 'Phone Number' in phone_tag.get_text(strip=True):
                phone_number = phone_tag.find_next_sibling(text=True) # 해당 tag에 있는 2번째 글자를 가지고 와야 함.
                primary_poc_info['Phone'] = phone_number.strip() if phone_number else ""
            else:
                primary_poc_info['Phone'] = ""
        primary_poc_list.append(primary_poc_info)

    else: # If there isn't any contact details existed at all on the page, return back the empty dictionary.
        primary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}
        primary_poc_list.append(primary_poc_info) # Add an empty dictionary if no contact info is available.
except Exception as e:
    print("Error fetching Primary Point of Contact: {e}")
    primary_poc_list.append({'Name': '', 'Email': '', 'Phone': ''})
```



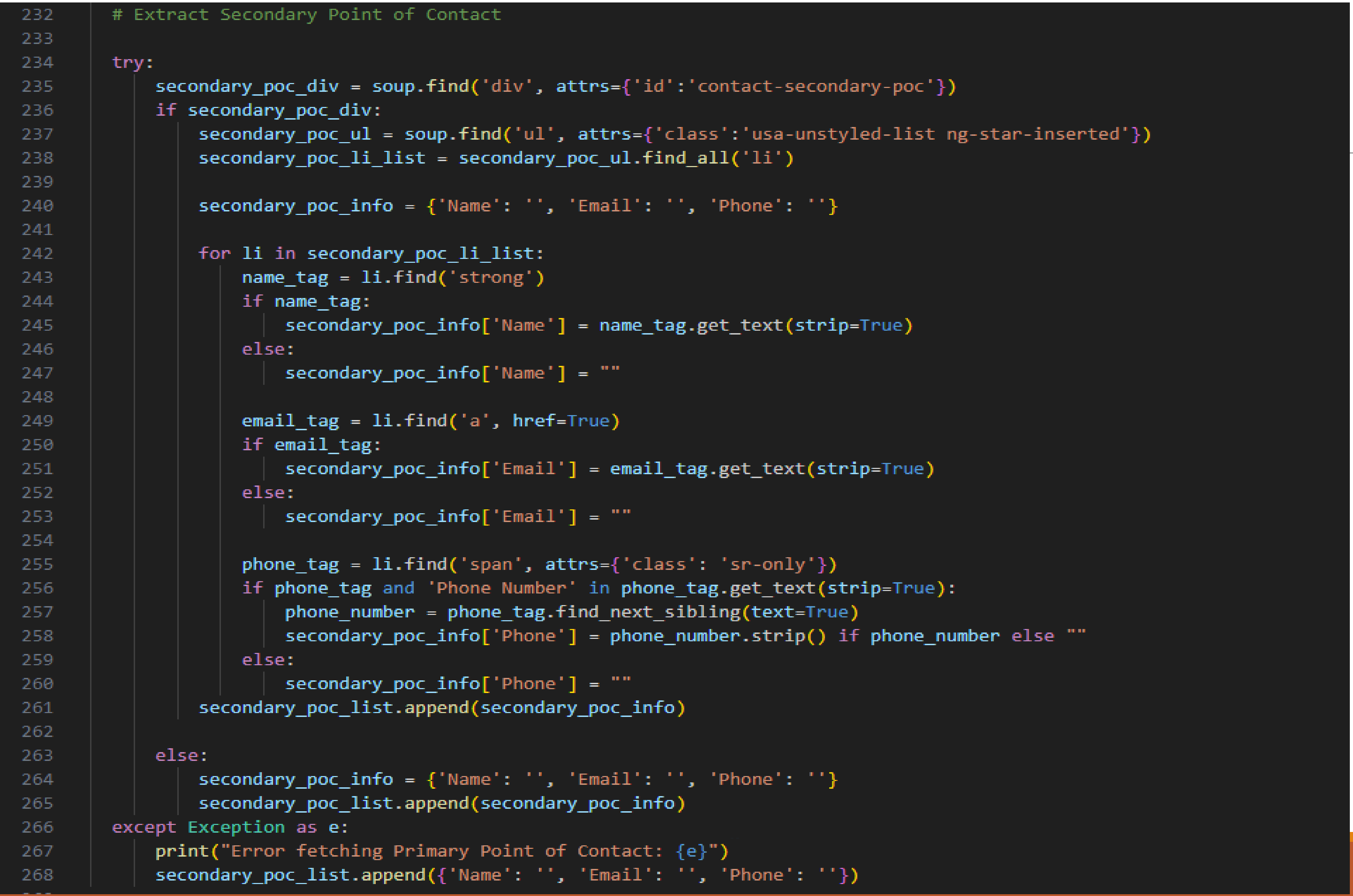
```html
▼<div _ngcontent-ng-c3685274516 class="usa-width-one-half ng-star-inserted"
 id="contact-primary-poc">
  ▶<h3 _ngcontent-ng-c3685274516 class="sam-ui dividing header">⋯</h3>
   <!---->
  ▼<div _ngcontent-ng-c3685274516 class="ng-star-inserted">
    ▼<div _ngcontent-ng-c3685274516 class="ng-star-inserted">
      ▼<ul _ngcontent-ng-c3685274516 class="usa-unstyled-list ng-star-inserte
       d">
        ▼<li _ngcontent-ng-c3685274516 id="contact-primary-poc-full-name"
         class="ng-star-inserted">
           <strong _ngcontent-ng-c3685274516>Erica Richardson </strong>
           <!---->
         </li>
         <!---->
        ▼<li _ngcontent-ng-c3685274516 id="contact-primary-poc-email" class="n
         g-star-inserted">
          ▶<em _ngcontent-ng-c3685274516 aria-hidden="true" class="fa fa-envel
           ope-o">⋯</em>
           " &nbsp; "
           <a _ngcontent-ng-c3685274516 href="mailto:erica_d_richardson@rl.go
           v"> erica_d_richardson@rl.gov </a>
           <!---->
         </li>
         <!---->
        ▼<li _ngcontent-ng-c3685274516 id="contact-primary-poc-phone" class="n
         g-star-inserted">
          ▶<em _ngcontent-ng-c3685274516 aria-hidden="true" class="fa fa-phon
           e">⋯</em>
           " &nbsp; "
           <span _ngcontent-ng-c3685274516 class="sr-only">Phone Number
           </span>
           " 509-376-6420 "
           <!---->
         </li>
```



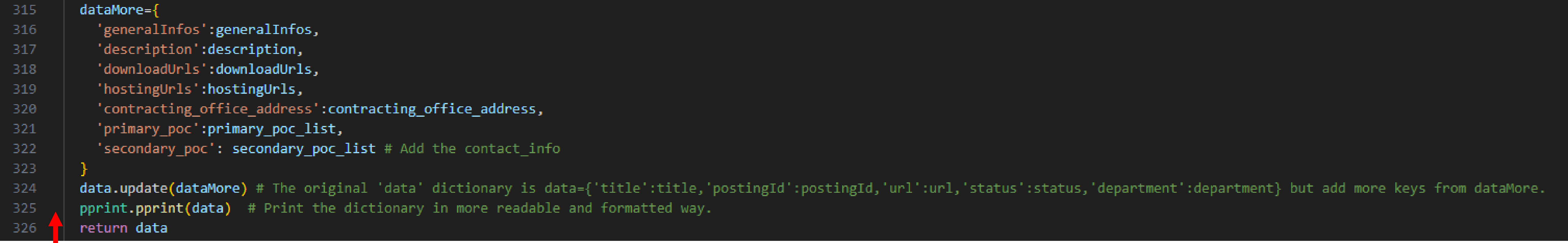
```python
# Extract the primary point of contact and the secondary point of contact.
primary_poc_list = []
secondary_poc_list = []
try:
    primary_poc_div = soup.find('div', attrs={'id':'contact-primary-poc'}) # Find 'div' (division) tag and store them in 'primary_poc_div' object.
    if primary_poc_div:
        primary_poc_ul = soup.find('ul', attrs={'class':'usa-unstyled-list ng-star-inserted'}) # Find inner 'ul' (Unordered list) tag with the specified class.
        primary_poc_li_list = primary_poc_ul.find_all('li') # Find inner 'li' (list) tag where 'name', 'email' and 'phone number' are placed in each 'li' tag.
    # 빈 dictionary를 생성해서 Name, Email, Phone을 기재
        primary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}

        for li in primary_poc_li_list: # 'li' tag was parsed as a bs4 object.
            name_tag = li.find('strong') # name in charge of the document is placed within 'strong' tag under 'li' tag.
            if name_tag:
                primary_poc_info['Name'] = name_tag.get_text(strip=True) # Only Extract the text representing the name of the person in charge of.
            else:
                primary_poc_info['Name'] = "" # Return back the empty string if there is no name.

            email_tag = li.find('a', href=True) # Find 'a' tag with the hypertext reference being True as its attribute under 'li' tag.
            if email_tag:
                primary_poc_info['Email'] = email_tag.get_text(strip=True)
            else:
                primary_poc_info['Email'] = ""

            phone_tag = li.find('span', attrs={'class': 'sr-only'}) # Find 'span' tag with the 'sr-only' class under 'li' tag.
            if phone_tag and 'Phone Number' in phone_tag.get_text(strip=True):
                phone_number = phone_tag.find_next_sibling(text=True) # 해당 tag에 있는 2번째 글자를 가지고 와야 함.
                primary_poc_info['Phone'] = phone_number.strip() if phone_number else ""
            else:
                primary_poc_info['Phone'] = ""
        primary_poc_list.append(primary_poc_info)

    else: # If there isn't any contact details existed at all on the page, return back the empty dictionary.
        primary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}
        primary_poc_list.append(primary_poc_info) # Add an empty dictionary if no contact info is available.
except Exception as e:
    print(f"Error fetching Primary Point of Contact: {e}")
    primary_poc_list.append({'Name': '', 'Email': '', 'Phone': ''})
```

1. Find the ‘div’ tag with the ‘id’ attribute set to ‘contact-primary-poc’ which contains the text details of the primary point of contact.
2. If this tag exists, find the ‘ul’ tag with the ‘class’ attribute set to ‘usa-unstyled-list ng-star-inserted’.
3. Under the ‘ul’ tag, find all ‘li’ sub tags that contain ‘Name’, ‘Email’ and ‘Phone number’.
4. For the ‘strong’ tag under the ‘li’ tag, store the HTML structure of the ‘strong’ tag and retrieve the ‘Name’ text using the get_text function.
5. For the ‘a’ tag under another ‘li’ tag with a hypertext reference, store the HTML structure of the ‘a’ tag in the ‘email-tag’ object and retrieve only the email text.
6. For the ‘span’ tag under another ‘li’ tag, there are two text elements: ‘Phone Number’ and the actual number. Extract the second text, which is the real phone number by using ‘find_next_sibling’ function.
7. Store all the details in a ‘primary_poc_info’ dictionary and append them into a ‘primary_poc_list’ list (already defined before the try loop).
8. If no primary contact details are found, leave the fields as empty strings.

The same method used for scraping the primary point of contact was used for the secondary point of contact.



```
# Extract Secondary Point of Contact

try:
    secondary_poc_div = soup.find('div', attrs={'id':'contact-secondary-poc'})
    if secondary_poc_div:
        secondary_poc_ul = soup.find('ul', attrs={'class':'usa-unstyled-list ng-star-inserted'})
        secondary_poc_li_list = secondary_poc_ul.find_all('li')

        secondary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}

        for li in secondary_poc_li_list:
            name_tag = li.find('strong')
            if name_tag:
                secondary_poc_info['Name'] = name_tag.get_text(strip=True)
            else:
                secondary_poc_info['Name'] = ""

            email_tag = li.find('a', href=True)
            if email_tag:
                secondary_poc_info['Email'] = email_tag.get_text(strip=True)
            else:
                secondary_poc_info['Email'] = ""

            phone_tag = li.find('span', attrs={'class': 'sr-only'})
            if phone_tag and 'Phone Number' in phone_tag.get_text(strip=True):
                phone_number = phone_tag.find_next_sibling(text=True)
                secondary_poc_info['Phone'] = phone_number.strip() if phone_number else ""
            else:
                secondary_poc_info['Phone'] = ""
        secondary_poc_list.append(secondary_poc_info)

    else:
        secondary_poc_info = {'Name': '', 'Email': '', 'Phone': ''}
        secondary_poc_list.append(secondary_poc_info)
except Exception as e:
    print("Error fetching Primary Point of Contact: {e}")
    secondary_poc_list.append({'Name': '', 'Email': '', 'Phone': ''})
```

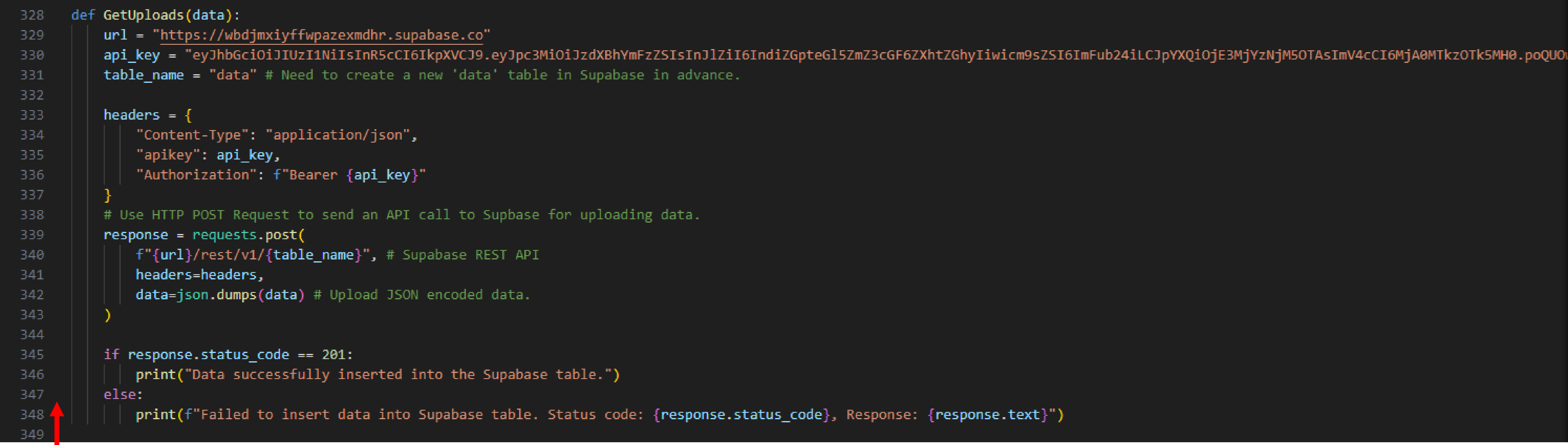


```
    dataMore={
      'generalInfos':generalInfos,
      'description':description,
      'downloadUrls':downloadUrls,
      'hostingUrls':hostingUrls,
      'contracting_office_address':contracting_office_address,
      'primary_poc':primary_poc_list,
      'secondary_poc': secondary_poc_list # Add the contact_info
    }
    data.update(dataMore) # The original 'data' dictionary is data={'title':title,'postingId':postingId,'url':url,'status':status,'department':department} but add more keys from dataMore.
    pprint.pprint(data)  # Print the dictionary in more readable and formatted way.
    return data
```

1. Create a ‘dataMore’ dictionary to store all retrieved data and update the existing ‘data’ extracted by sending an API call, for example:
   `data={'title':title,'postingId':postingId,'url':url,'status':status,'department':department}.`

2. Print the updated data in a structured way using a ‘pprint’ function.

Part 3. Upload data to Supabase by sending an API call.

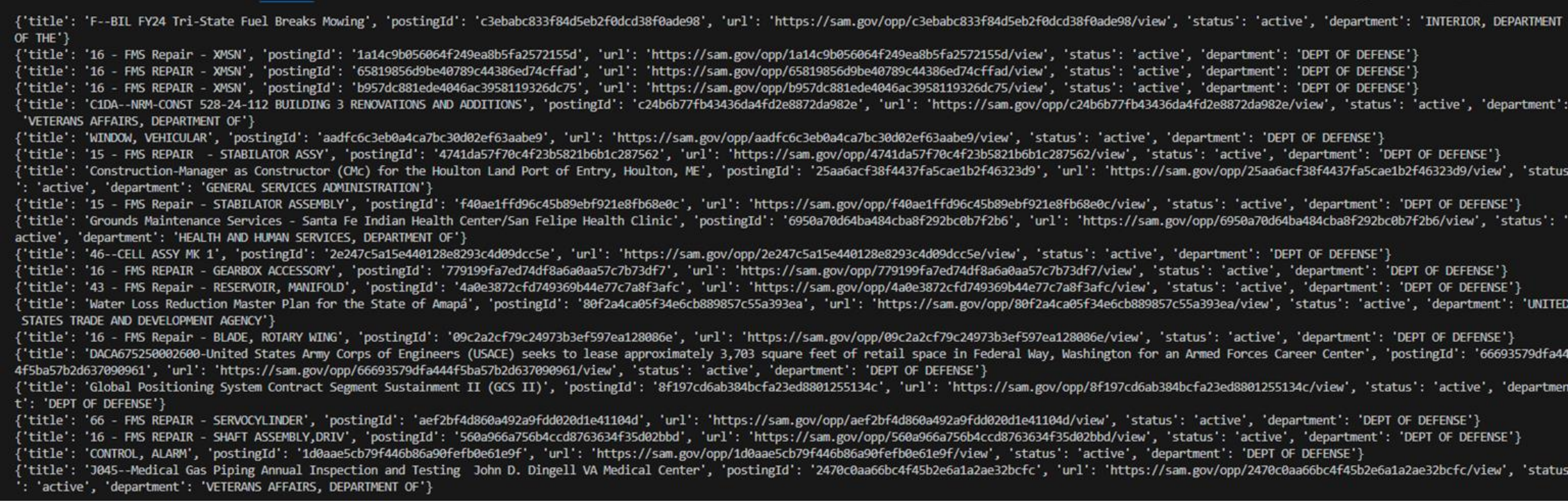


```python
def GetUploads(data):
    url = "https://wbdjmxiyffwpazexmdhr.supabase.co"
    api_key = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndiZGpteGl5ZmZ3cGF6ZXhtZGhyIiwicm9sZSI6ImFub24iLCJpYXQiOjE3MjYzNjM5OTAsImV4cCI6MjA0MTkzOTk5MH0.poQUO
    table_name = "data" # Need to create a new 'data' table in Supabase in advance.

    headers = {
        "Content-Type": "application/json",
        "apikey": api_key,
        "Authorization": f"Bearer {api_key}"
    }
    # Use HTTP POST Request to send an API call to Supbase for uploading data.
    response = requests.post(
        f"{url}/rest/v1/{table_name}", # Supabase REST API
        headers=headers,
        data=json.dumps(data) # Upload JSON encoded data.
    )

    if response.status_code == 201:
        print("Data successfully inserted into the Supabase table.")
    else:
        print(f"Failed to insert data into Supabase table. Status code: {response.status_code}, Response: {response.text}")
```

1. Define ‘**GetUploads**’ function in Python to **send an API call to the Supabase REST API**.
2. The ‘GetUploads’ function can send the **API call by using an HTTP POST request**.
3. If the ‘data’ is successfully uploaded, then ‘Data Successfully inserted into the Supabase table.’ is printed.
4. After the successful insertion, the program will proceed to the next iteration to extract and upload additional data.

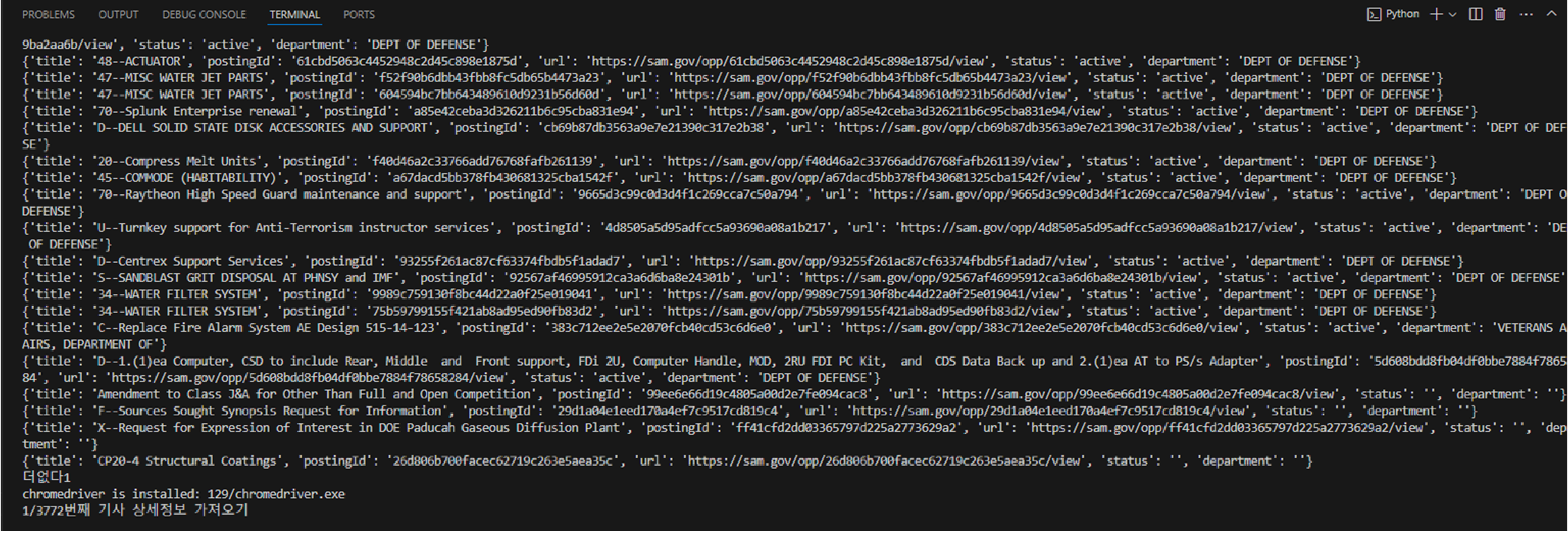
Appendix.

The outcomes shown on the terminal of VS CODE when running the Python script.



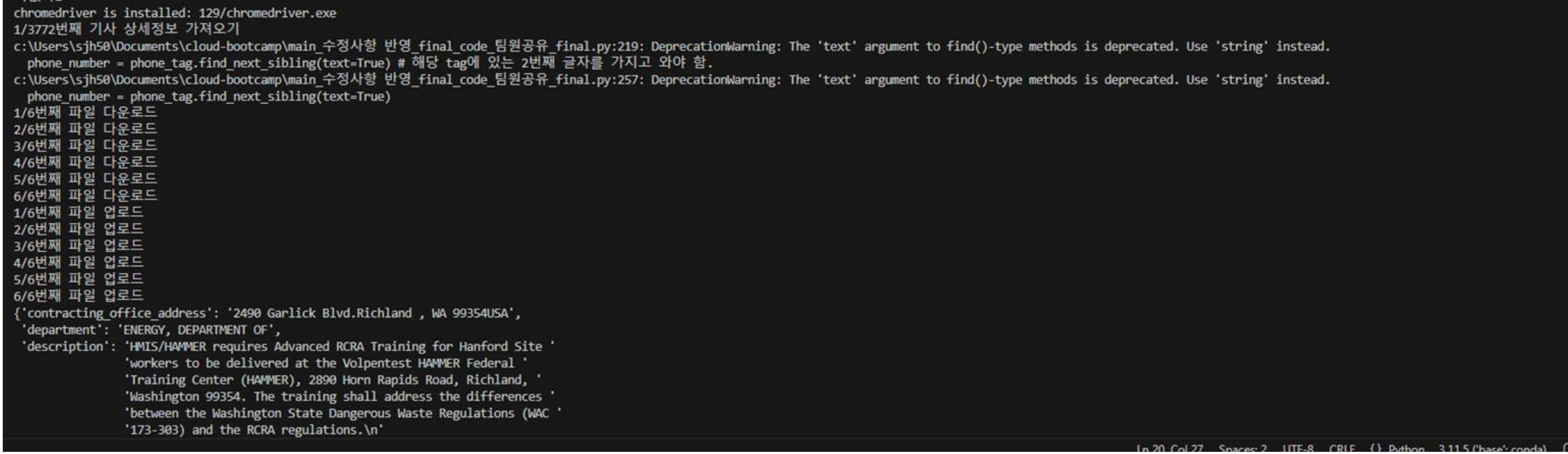
```
{'title': 'F--BIL FY24 Tri-State Fuel Breaks Mowing', 'postingId': 'c3ebabc833f84d5eb2f0dcd38f0ade98', 'url': 'https://sam.gov/opp/c3ebabc833f84d5eb2f0dcd38f0ade98/view', 'status': 'active', 'department': 'INTERIOR, DEPARTMENT
OF THE'}
{'title': '16 - FMS Repair - XMSN', 'postingId': '1a14c9b056064f249ea8b5fa2572155d', 'url': 'https://sam.gov/opp/1a14c9b056064f249ea8b5fa2572155d/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '16 - FMS REPAIR - XMSN', 'postingId': '65819856d9be40789c44386ed74cffad', 'url': 'https://sam.gov/opp/65819856d9be40789c44386ed74cffad/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '16 - FMS REPAIR - XMSN', 'postingId': 'b957dc881ede4046ac3958119326dc75', 'url': 'https://sam.gov/opp/b957dc881ede4046ac3958119326dc75/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'C1DA--NRM-CONST 528-24-112 BUILDING 3 RENOVATIONS AND ADDITIONS', 'postingId': 'c24b6b77fb43436da4fd2e8872da982e', 'url': 'https://sam.gov/opp/c24b6b77fb43436da4fd2e8872da982e/view', 'status': 'active', 'department':
 'VETERANS AFFAIRS, DEPARTMENT OF'}
{'title': 'WINDOW, VEHICULAR', 'postingId': 'aadfc6c3eb0a4ca7bc30d02ef63aabe9', 'url': 'https://sam.gov/opp/aadfc6c3eb0a4ca7bc30d02ef63aabe9/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '15 - FMS REPAIR  - STABILATOR ASSY', 'postingId': '4741da57f70c4f23b5821b6b1c287562', 'url': 'https://sam.gov/opp/4741da57f70c4f23b5821b6b1c287562/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'Construction-Manager as Constructor (CMc) for the Houlton Land Port of Entry, Houlton, ME', 'postingId': '25aa6acf38f4437fa5cae1b2f46323d9', 'url': 'https://sam.gov/opp/25aa6acf38f4437fa5cae1b2f46323d9/view', 'status
': 'active', 'department': 'GENERAL SERVICES ADMINISTRATION'}
{'title': '15 - FMS Repair - STABILATOR ASSEMBLY', 'postingId': 'f40ae1ffd96c45b89ebf921e8fb68e0c', 'url': 'https://sam.gov/opp/f40ae1ffd96c45b89ebf921e8fb68e0c/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'Grounds Maintenance Services - Santa Fe Indian Health Center/San Felipe Health Clinic', 'postingId': '6950a70d64ba484cba8f292bc0b7f2b6', 'url': 'https://sam.gov/opp/6950a70d64ba484cba8f292bc0b7f2b6/view', 'status': '
active', 'department': 'HEALTH AND HUMAN SERVICES, DEPARTMENT OF'}
{'title': '46--CELL ASSY MK 1', 'postingId': '2e247c5a15e440128e8293c4d09dcc5e', 'url': 'https://sam.gov/opp/2e247c5a15e440128e8293c4d09dcc5e/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '16 - FMS REPAIR - GEARBOX ACCESSORY', 'postingId': '779199fa7ed74df8a6a0aa57c7b73df7', 'url': 'https://sam.gov/opp/779199fa7ed74df8a6a0aa57c7b73df7/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '43 - FMS Repair - RESERVOIR, MANIFOLD', 'postingId': '4a0e3872cfd749369b44e77c7a8f3afc', 'url': 'https://sam.gov/opp/4a0e3872cfd749369b44e77c7a8f3afc/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'Water Loss Reduction Master Plan for the State of Amapá', 'postingId': '80f2a4ca05f34e6cb889857c55a393ea', 'url': 'https://sam.gov/opp/80f2a4ca05f34e6cb889857c55a393ea/view', 'status': 'active', 'department': 'UNITED
 STATES TRADE AND DEVELOPMENT AGENCY'}
{'title': '16 - FMS Repair - BLADE, ROTARY WING', 'postingId': '09c2a2cf79c24973b3ef597ea128086e', 'url': 'https://sam.gov/opp/09c2a2cf79c24973b3ef597ea128086e/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'DACA675250002600-United States Army Corps of Engineers (USACE) seeks to lease approximately 3,703 square feet of retail space in Federal Way, Washington for an Armed Forces Career Center', 'postingId': '66693579dfa44
4f5ba57b2d637090961', 'url': 'https://sam.gov/opp/66693579dfa444f5ba57b2d637090961/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'Global Positioning System Contract Segment Sustainment II (GCS II)', 'postingId': '8f197cd6ab384bcfa23ed8801255134c', 'url': 'https://sam.gov/opp/8f197cd6ab384bcfa23ed8801255134c/view', 'status': 'active', 'departmen
t': 'DEPT OF DEFENSE'}
{'title': '66 - FMS REPAIR - SERVOCYLINDER', 'postingId': 'aef2bf4d860a492a9fdd020d1e41104d', 'url': 'https://sam.gov/opp/aef2bf4d860a492a9fdd020d1e41104d/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '16 - FMS REPAIR - SHAFT ASSEMBLY,DRIV', 'postingId': '560a966a756b4ccd8763634f35d02bbd', 'url': 'https://sam.gov/opp/560a966a756b4ccd8763634f35d02bbd/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'CONTROL, ALARM', 'postingId': '1d0aae5cb79f446b86a90fefb0e61e9f', 'url': 'https://sam.gov/opp/1d0aae5cb79f446b86a90fefb0e61e9f/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'J045--Medical Gas Piping Annual Inspection and Testing  John D. Dingell VA Medical Center', 'postingId': '2470c0aa66bc4f45b2e6a1a2ae32bcfc', 'url': 'https://sam.gov/opp/2470c0aa66bc4f45b2e6a1a2ae32bcfc/view', 'status
': 'active', 'department': 'VETERANS AFFAIRS, DEPARTMENT OF'}
```

APPENDIX. THE OUTCOMES SHOWN ON THE TERMINAL OF VS CODE WHEN RUNNING THE PYTHON SCRIPT.



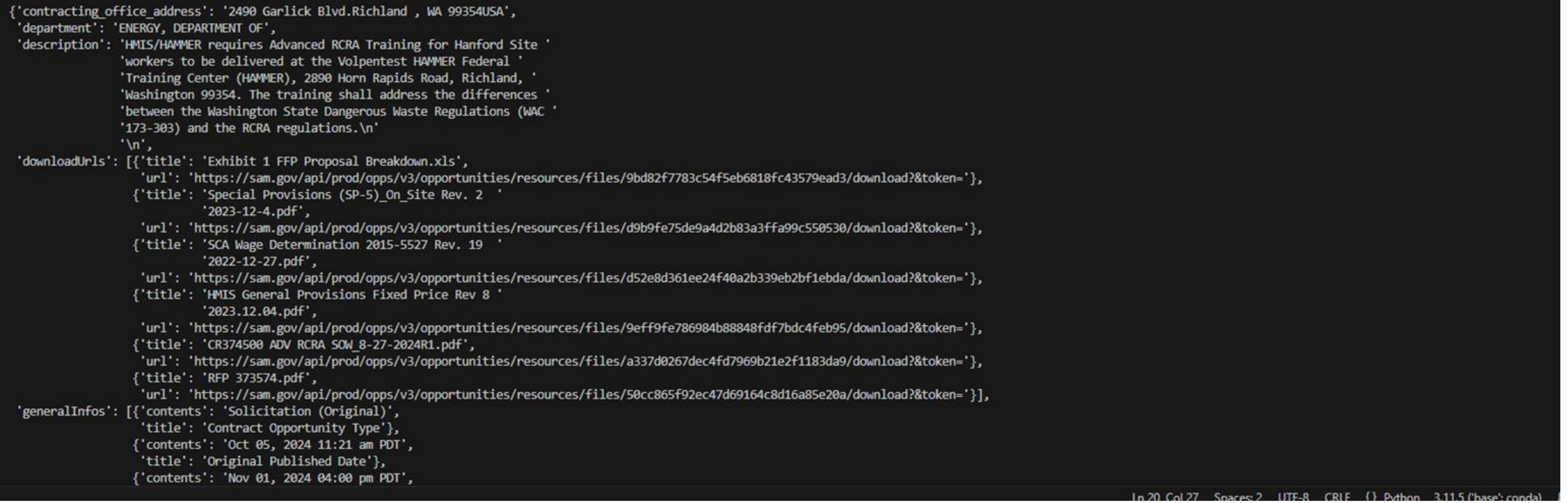
```
9ba2aa6b/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '48--ACTUATOR', 'postingId': '61cbd5063c4452948c2d45c898e1875d', 'url': 'https://sam.gov/opp/61cbd5063c4452948c2d45c898e1875d/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '47--MISC WATER JET PARTS', 'postingId': 'f52f90b6dbb43fbb8fc5db65b4473a23', 'url': 'https://sam.gov/opp/f52f90b6dbb43fbb8fc5db65b4473a23/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '47--MISC WATER JET PARTS', 'postingId': '604594bc7bb643489610d9231b56d60d', 'url': 'https://sam.gov/opp/604594bc7bb643489610d9231b56d60d/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '70--Splunk Enterprise renewal', 'postingId': 'a85e42ceba3d326211b6c95cba831e94', 'url': 'https://sam.gov/opp/a85e42ceba3d326211b6c95cba831e94/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'D--DELL SOLID STATE DISK ACCESSORIES AND SUPPORT', 'postingId': 'cb69b87db3563a9e7e21390c317e2b38', 'url': 'https://sam.gov/opp/cb69b87db3563a9e7e21390c317e2b38/view', 'status': 'active', 'department': 'DEPT OF DEF
SE'}
{'title': '20--Compress Melt Units', 'postingId': 'f40d46a2c33766add76768fafb261139', 'url': 'https://sam.gov/opp/f40d46a2c33766add76768fafb261139/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '45--COMMODE (HABITABILITY)', 'postingId': 'a67dacd5bb378fb430681325cba1542f', 'url': 'https://sam.gov/opp/a67dacd5bb378fb430681325cba1542f/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '70--Raytheon High Speed Guard maintenance and support', 'postingId': '9665d3c99c0d3d4f1c269cca7c50a794', 'url': 'https://sam.gov/opp/9665d3c99c0d3d4f1c269cca7c50a794/view', 'status': 'active', 'department': 'DEPT O
DEFENSE'}
{'title': 'U--Turnkey support for Anti-Terrorism instructor services', 'postingId': '4d8505a5d95adfcc5a93690a08a1b217', 'url': 'https://sam.gov/opp/4d8505a5d95adfcc5a93690a08a1b217/view', 'status': 'active', 'department': 'DE
 OF DEFENSE'}
{'title': 'D--Centrex Support Services', 'postingId': '93255f261ac87cf63374fbdb5f1adad7', 'url': 'https://sam.gov/opp/93255f261ac87cf63374fbdb5f1adad7/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'S--SANDBLAST GRIT DISPOSAL AT PHNSY and IMF', 'postingId': '92567af46995912ca3a6d6ba8e24301b', 'url': 'https://sam.gov/opp/92567af46995912ca3a6d6ba8e24301b/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'
{'title': '34--WATER FILTER SYSTEM', 'postingId': '9989c759130f8bc44d22a0f25e019041', 'url': 'https://sam.gov/opp/9989c759130f8bc44d22a0f25e019041/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': '34--WATER FILTER SYSTEM', 'postingId': '75b59799155f421ab8ad95ed90fb83d2', 'url': 'https://sam.gov/opp/75b59799155f421ab8ad95ed90fb83d2/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'C--Replace Fire Alarm System AE Design 515-14-123', 'postingId': '383c712ee2e5e2070fcb40cd53c6d6e0', 'url': 'https://sam.gov/opp/383c712ee2e5e2070fcb40cd53c6d6e0/view', 'status': 'active', 'department': 'VETERANS A
AIRS, DEPARTMENT OF'}
{'title': 'D--1.(1)ea Computer, CSD to include Rear, Middle  and  Front support, FDi 2U, Computer Handle, MOD, 2RU FDI PC Kit,  and  CDS Data Back up and 2.(1)ea AT to PS/s Adapter', 'postingId': '5d608bdd8fb04df0bbe7884f7865
84', 'url': 'https://sam.gov/opp/5d608bdd8fb04df0bbe7884f78658284/view', 'status': 'active', 'department': 'DEPT OF DEFENSE'}
{'title': 'Amendment to Class J&A for Other Than Full and Open Competition', 'postingId': '99ee6e66d19c4805a00d2e7fe094cac8', 'url': 'https://sam.gov/opp/99ee6e66d19c4805a00d2e7fe094cac8/view', 'status': '', 'department': ''}
{'title': 'F--Sources Sought Synopsis Request for Information', 'postingId': '29d1a04e1eed170a4ef7c9517cd819c4', 'url': 'https://sam.gov/opp/29d1a04e1eed170a4ef7c9517cd819c4/view', 'status': '', 'department': ''}
{'title': 'X--Request for Expression of Interest in DOE Paducah Gaseous Diffusion Plant', 'postingId': 'ff41cfd2dd03365797d225a2773629a2', 'url': 'https://sam.gov/opp/ff41cfd2dd03365797d225a2773629a2/view', 'status': '', 'dep
tment': ''}
{'title': 'CP20-4 Structural Coatings', 'postingId': '26d806b700facec62719c263e5aea35c', 'url': 'https://sam.gov/opp/26d806b700facec62719c263e5aea35c/view', 'status': '', 'department': ''}
더없다1
chromedriver is installed: 129/chromedriver.exe
1/3772번째 기사 상세정보 가져오기
```

APPENDIX. THE OUTCOMES SHOWN ON THE TERMINAL OF VS CODE WHEN RUNNING THE PYTHON SCRIPT.



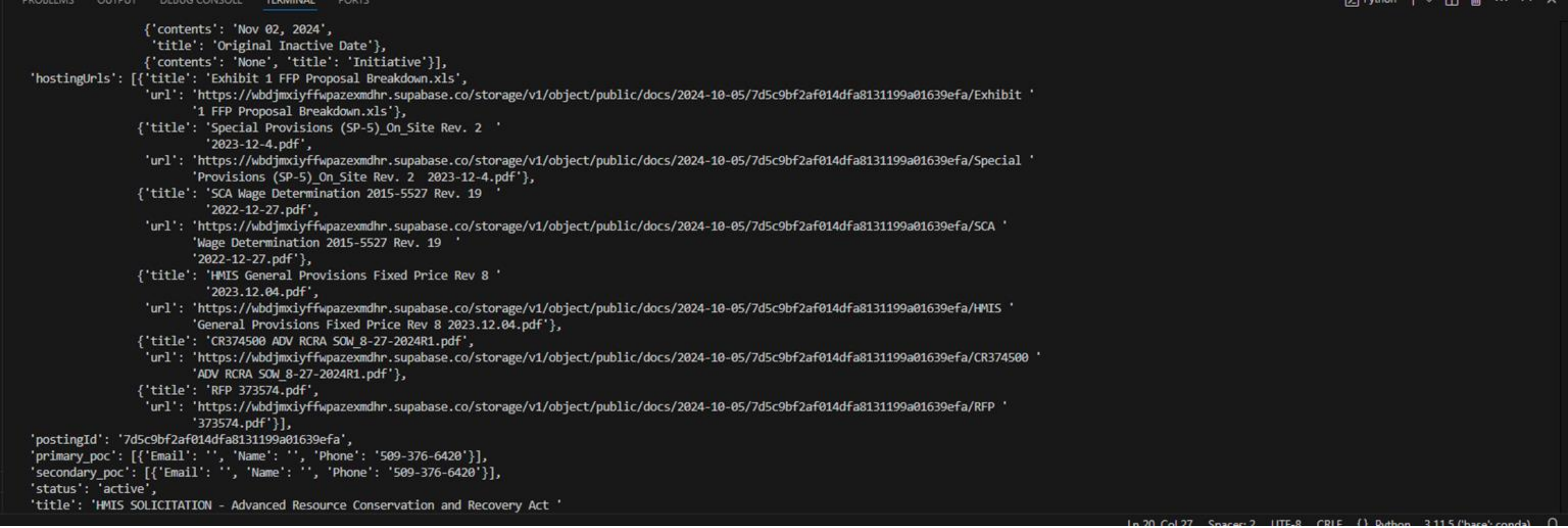
```
chromedriver is installed: 129/chromedriver.exe
1/3772번째 기사 상세정보 가져오기
c:\Users\sjh50\Documents\cloud-bootcamp\main_수정사항 반영_final_code_팀원공유_final.py:219: DeprecationWarning: The 'text' argument to find()-type methods is deprecated. Use 'string' instead.
  phone_number = phone_tag.find_next_sibling(text=True) # 해당 tag에 있는 2번째 글자를 가지고 와야 함.
c:\Users\sjh50\Documents\cloud-bootcamp\main_수정사항 반영_final_code_팀원공유_final.py:257: DeprecationWarning: The 'text' argument to find()-type methods is deprecated. Use 'string' instead.
  phone_number = phone_tag.find_next_sibling(text=True)
1/6번째 파일 다운로드
2/6번째 파일 다운로드
3/6번째 파일 다운로드
4/6번째 파일 다운로드
5/6번째 파일 다운로드
6/6번째 파일 다운로드
1/6번째 파일 업로드
2/6번째 파일 업로드
3/6번째 파일 업로드
4/6번째 파일 업로드
5/6번째 파일 업로드
6/6번째 파일 업로드
{'contracting_office_address': '2490 Garlick Blvd.Richland , WA 99354USA',
 'department': 'ENERGY, DEPARTMENT OF',
 'description': 'HMIS/HAMMER requires Advanced RCRA Training for Hanford Site '
                'workers to be delivered at the Volpentest HAMMER Federal '
                'Training Center (HAMMER), 2890 Horn Rapids Road, Richland, '
                'Washington 99354. The training shall address the differences '
                'between the Washington State Dangerous Waste Regulations (WAC '
                '173-303) and the RCRA regulations.\n'
```

Ln 20, Col 27   Spaces: 2   UTF-8   CRLF   {} Python   3.11.5 ('base': conda)

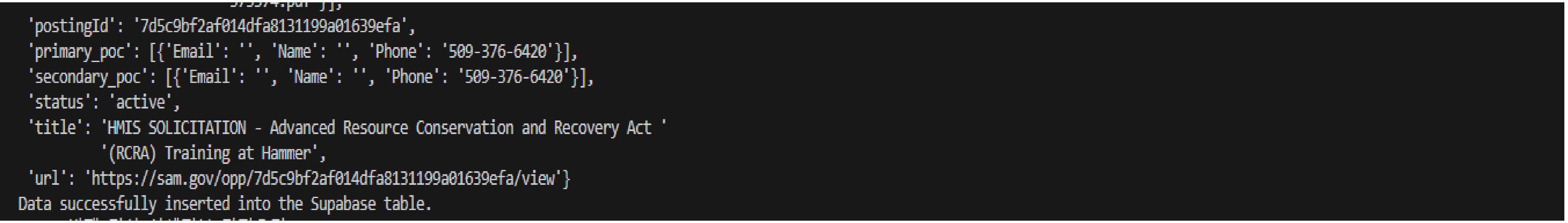


```
{'contracting_office_address': '2490 Garlick Blvd.Richland , WA 99354USA',
 'department': 'ENERGY, DEPARTMENT OF',
 'description': 'HMIS/HAMMER requires Advanced RCRA Training for Hanford Site '
                'workers to be delivered at the Volpentest HAMMER Federal '
                'Training Center (HAMMER), 2890 Horn Rapids Road, Richland, '
                'Washington 99354. The training shall address the differences '
                'between the Washington State Dangerous Waste Regulations (WAC '
                '173-303) and the RCRA regulations.\n'
                '\n',
 'downloadUrls': [{'title': 'Exhibit 1 FFP Proposal Breakdown.xls',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/9bd82f7783c54f5eb6818fc43579ead3/download?&token='},
                  {'title': 'Special Provisions (SP-5)_On_Site Rev. 2  '
                            '2023-12-4.pdf',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/d9b9fe75de9a4d2b83a3ffa99c550530/download?&token='},
                  {'title': 'SCA Wage Determination 2015-5527 Rev. 19  '
                            '2022-12-27.pdf',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/d52e8d361ee24f40a2b339eb2bf1ebda/download?&token='},
                  {'title': 'HMIS General Provisions Fixed Price Rev 8 '
                            '2023.12.04.pdf',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/9eff9fe786984b88848fdf7bdc4feb95/download?&token='},
                  {'title': 'CR374500 ADV RCRA SOW_8-27-2024R1.pdf',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/a337d0267dec4fd7969b21e2f1183da9/download?&token='},
                  {'title': 'RFP 373574.pdf',
                   'url': 'https://sam.gov/api/prod/opps/v3/opportunities/resources/files/50cc865f92ec47d69164c8d16a85e20a/download?&token='}],
 'generalInfos': [{'contents': 'Solicitation (Original)',
                   'title': 'Contract Opportunity Type'},
                  {'contents': 'Oct 05, 2024 11:21 am PDT',
                   'title': 'Original Published Date'},
                  {'contents': 'Nov 01, 2024 04:00 pm PDT',
```

APPENDIX. THE OUTCOMES SHOWN ON THE TERMINAL OF VS CODE WHEN RUNNING THE PYTHON SCRIPT.



```
                    {'contents': 'Nov 02, 2024',
                     'title': 'Original Inactive Date'},
                    {'contents': 'None', 'title': 'Initiative'}],
 'hostingUrls': [{'title': 'Exhibit 1 FFP Proposal Breakdown.xls',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/Exhibit '
                         '1 FFP Proposal Breakdown.xls'},
                 {'title': 'Special Provisions (SP-5)_On_Site Rev. 2  '
                           '2023-12-4.pdf',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/Special '
                         'Provisions (SP-5)_On_Site Rev. 2  2023-12-4.pdf'},
                 {'title': 'SCA Wage Determination 2015-5527 Rev. 19  '
                           '2022-12-27.pdf',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/SCA '
                         'Wage Determination 2015-5527 Rev. 19  '
                         '2022-12-27.pdf'},
                 {'title': 'HMIS General Provisions Fixed Price Rev 8 '
                           '2023.12.04.pdf',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/HMIS '
                         'General Provisions Fixed Price Rev 8 2023.12.04.pdf'},
                 {'title': 'CR374500 ADV RCRA SOW_8-27-2024R1.pdf',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/CR374500 '
                         'ADV RCRA SOW_8-27-2024R1.pdf'},
                 {'title': 'RFP 373574.pdf',
                  'url': 'https://wbdjmxiyffwpazexmdhr.supabase.co/storage/v1/object/public/docs/2024-10-05/7d5c9bf2af014dfa8131199a01639efa/RFP '
                         '373574.pdf'}],
 'postingId': '7d5c9bf2af014dfa8131199a01639efa',
 'primary_poc': [{'Email': '', 'Name': '', 'Phone': '509-376-6420'}],
 'secondary_poc': [{'Email': '', 'Name': '', 'Phone': '509-376-6420'}],
 'status': 'active',
 'title': 'HMIS SOLICITATION - Advanced Resource Conservation and Recovery Act '
```



```
 'postingId': '7d5c9bf2af014dfa8131199a01639efa',
 'primary_poc': [{'Email': '', 'Name': '', 'Phone': '509-376-6420'}],
 'secondary_poc': [{'Email': '', 'Name': '', 'Phone': '509-376-6420'}],
 'status': 'active',
 'title': 'HMIS SOLICITATION - Advanced Resource Conservation and Recovery Act '
          '(RCRA) Training at Hammer',
 'url': 'https://sam.gov/opp/7d5c9bf2af014dfa8131199a01639efa/view'}
Data successfully inserted into the Supabase table.
```

APPENDIX. THE OUTCOMES SHOWN ON THE TERMINAL OF VS CODE WHEN RUNNING THE PYTHON SCRIPT.