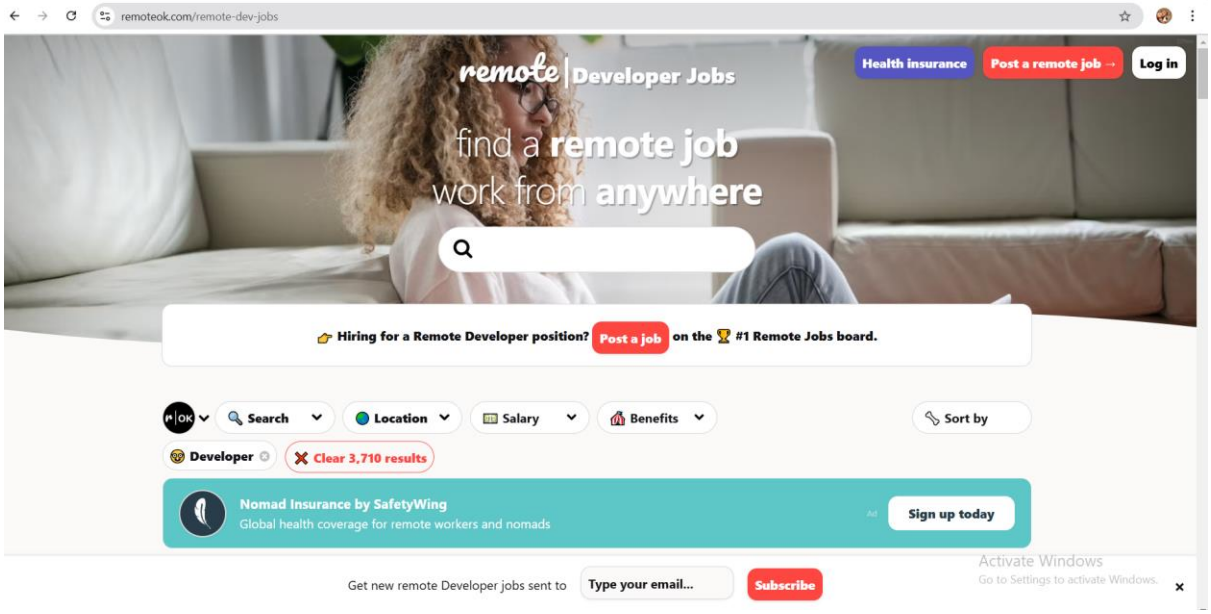
Final Project:- Python(Web Scrapping)

Project for Web Scrapping:

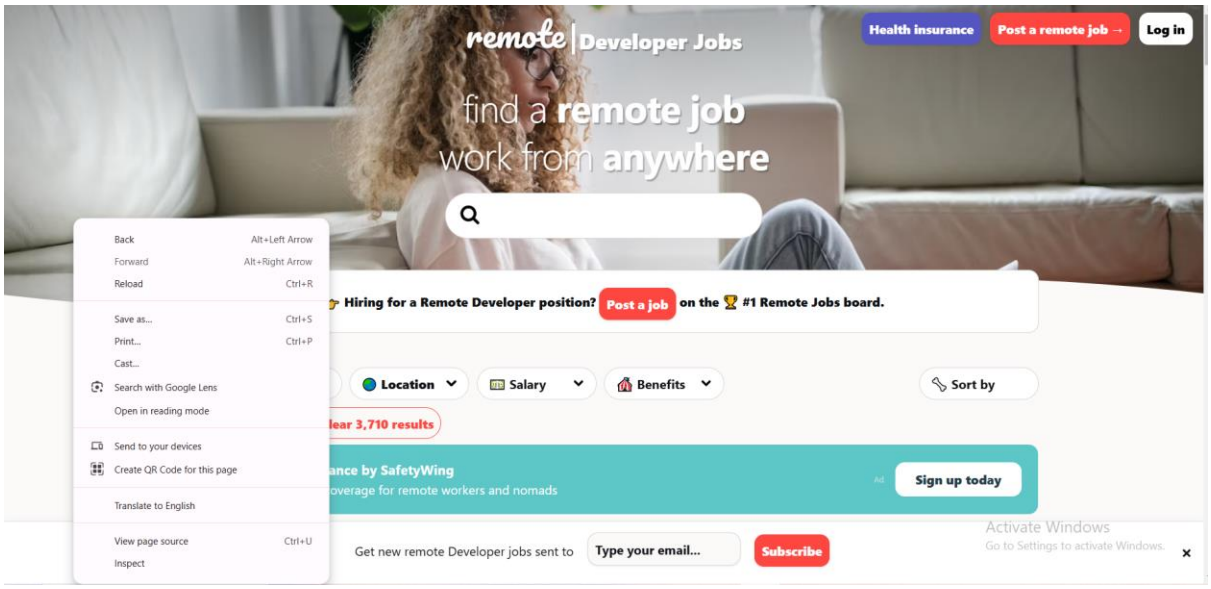
Python code fetches HTML content from a web page, extracts job details, and saves them to a text file.

Website for scrapping:- " https://remoteok.io/remote-dev-jobs"

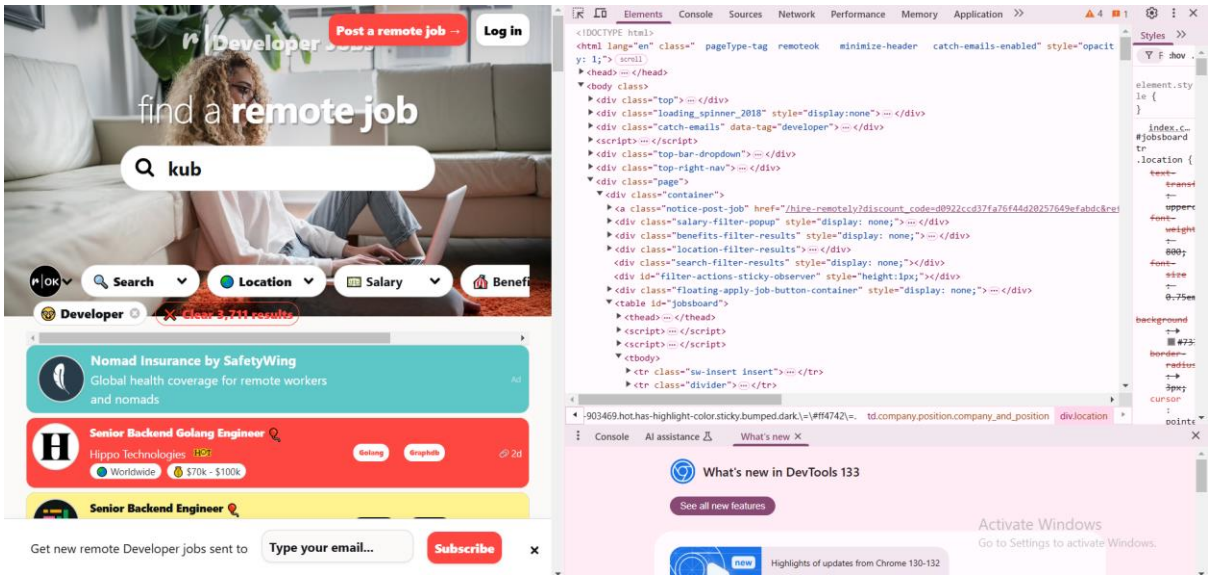
Open website



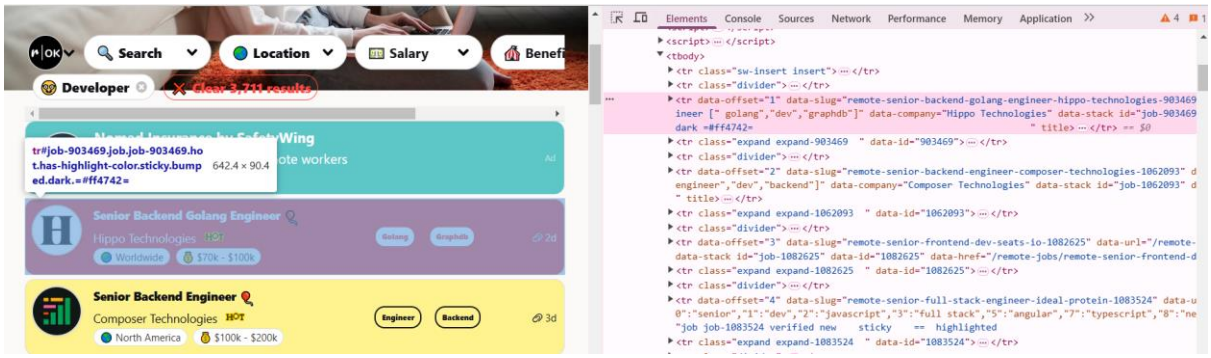
Right click on page and click on Inspect



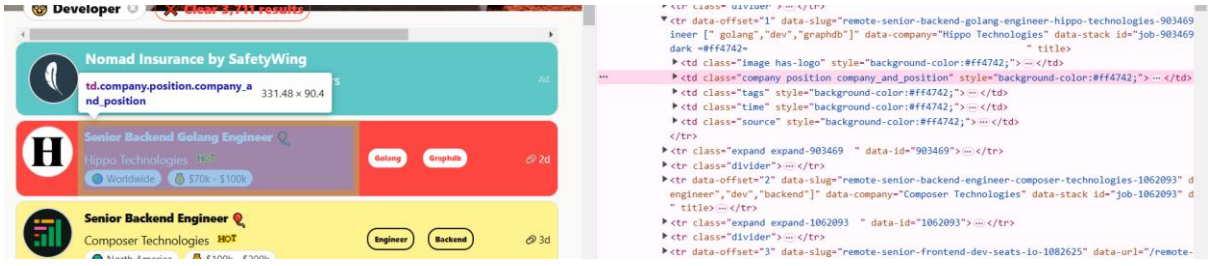
It will open html page



We can study html page, If you click on different tags it will highlight area for that tag



We can open each tag and check for which tag is for the job name, location and company name likewise



We can inspect that above h2 tag is for Job Title

We can inspect that above h3 tag is for Company Name

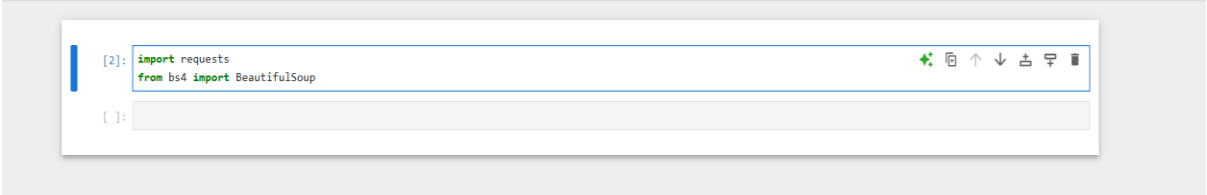
We can inspect that above div ‘location’ is for job location

We ca code it with hlp of above data

Step1:-

Code:-

```
import requests
from bs4 import BeautifulSoup
```



Explanation:-

requests: This library is used to send HTTP requests and handle responses.

BeautifulSoup: This library is used for parsing HTML and XML documents. It helps in extracting data from HTML content.

Step2:-

Code:-

```
def fetch_html(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.text
```

```
    else:
        raise ConnectionError("Failed to retrieve content from ", url)

def main():
    url = "https://remoteok.io/remote-dev-jobs"
    html_content = fetch_html(url)
    print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:
        file.write(html_content)

main()
```

Explanation:-

* This function retrieves the HTML content of the given URL.

```
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
    }
```

headers: This dictionary contains the User-Agent header, which mimics a browser request to avoid being blocked by the website.

This helps prevent your requests from being flagged or blocked by the website's security mechanisms.

"User-Agent": This specifies the user-agent string, which is a string that web browsers and other HTTP clients send to web servers to identify themselves.

The specific user-agent string used here is for a modern version of Google Chrome running on Windows 10.

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

Above code sends an HTTP GET request to the specified URL with the provided headers.

```
if response.status_code == 200:
        return response.text
```

else:

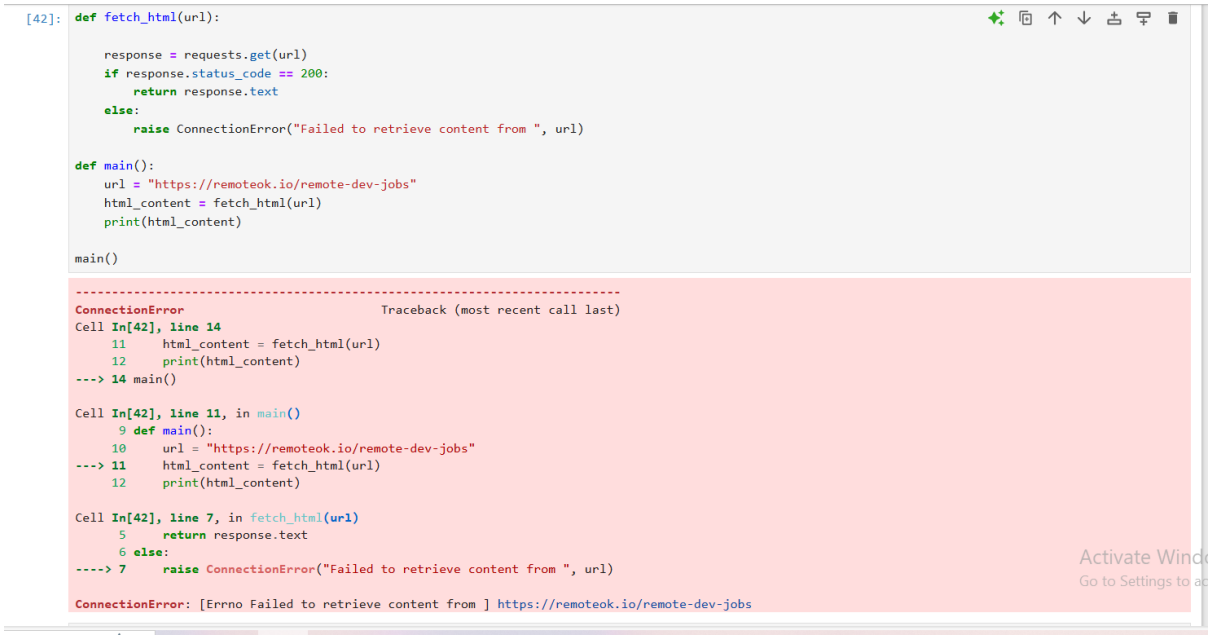
raise ConnectionError(“Failed to retrieve content from ",url)

response.status_code: This retrieves the status code from the HTTP response. The status code is a numeric code that indicates the outcome of the HTTP request.

200: This is the standard status code for a successful HTTP request. It means that the server has successfully processed the request and returned the requested content.

response.text: Returns the HTML content of the page if the request was successful.

Exception Handling: If the request fails, a ConnectionError is raised with an error message.



```
[42]: def fetch_html(url):

          response = requests.get(url)
          if response.status_code == 200:
              return response.text
          else:
              raise ConnectionError("Failed to retrieve content from ", url)

      def main():
          url = "https://remoteok.io/remote-dev-jobs"
          html_content = fetch_html(url)
          print(html_content)

      main()
```

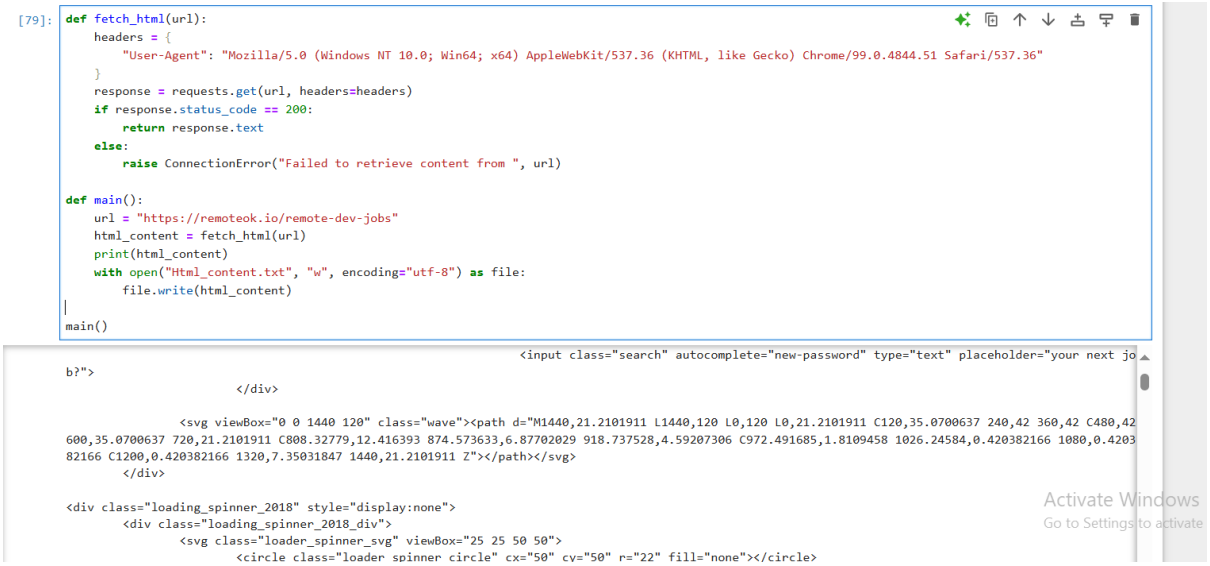
```
---------------------------------------------------------------------------
ConnectionError                           Traceback (most recent call last)
Cell In[42], line 14
     11     html_content = fetch_html(url)
     12     print(html_content)
---> 14 main()

Cell In[42], line 11, in main()
      9 def main():
     10     url = "https://remoteok.io/remote-dev-jobs"
---> 11     html_content = fetch_html(url)
     12     print(html_content)

Cell In[42], line 7, in fetch_html(url)
      5     return response.text
      6 else:
----> 7     raise ConnectionError("Failed to retrieve content from ", url)

ConnectionError: [Errno Failed to retrieve content from ] https://remoteok.io/remote-dev-jobs
```


When there is no error HTML content saved in txt file:-



```
[79]: def fetch_html(url):
          headers = {
              "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
          }
          response = requests.get(url, headers=headers)
          if response.status_code == 200:
              return response.text
          else:
              raise ConnectionError("Failed to retrieve content from ", url)

      def main():
          url = "https://remoteok.io/remote-dev-jobs"
          html_content = fetch_html(url)
          print(html_content)
          with open("Html_content.txt", "w", encoding="utf-8") as file:
              file.write(html_content)

      main()
```

```
                                                                    <input class="search" autocomplete="new-password" type="text" placeholder="your next jo
b?">
                        </div>

                <svg viewBox="0 0 1440 120" class="wave"><path d="M1440,21.2101911 L1440,120 L0,120 L0,21.2101911 C120,35.0700637 240,42 360,42 C480,42
600,35.0700637 720,21.2101911 C808.32779,12.416393 874.573633,6.87702029 918.737528,4.59207306 C972.491685,1.8109458 1026.24584,0.420382166 1080,0.4203
82166 C1200,0.420382166 1320,7.35031847 1440,21.2101911 Z"></path></svg>
        </div>

<div class="loading_spinner_2018" style="display:none">
        <div class="loading_spinner_2018_div">
                <svg class="loader_spinner_svg" viewBox="25 25 50 50">
                        <circle class="loader_spinner_circle" cx="50" cy="50" r="22" fill="none"></circle>
```


Step3:-

Code:-

```
def fetch_html(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.text
    else:
        raise ConnectionError("Failed to retrieve content from ", url)
def extract_job_details(html_content):
    soup = BeautifulSoup(html_content, 'lxml')
    print(soup)
def main():
    url = "https://remoteok.io/remote-dev-jobs"
```

```
    html_content = fetch_html(url)
    # print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:
        file.write(html_content)
    extract_job_details(html_content)

main()


def extract_job_details(html_content):
    soup = BeautifulSoup(html_content, 'lxml')
    print(soup)
```

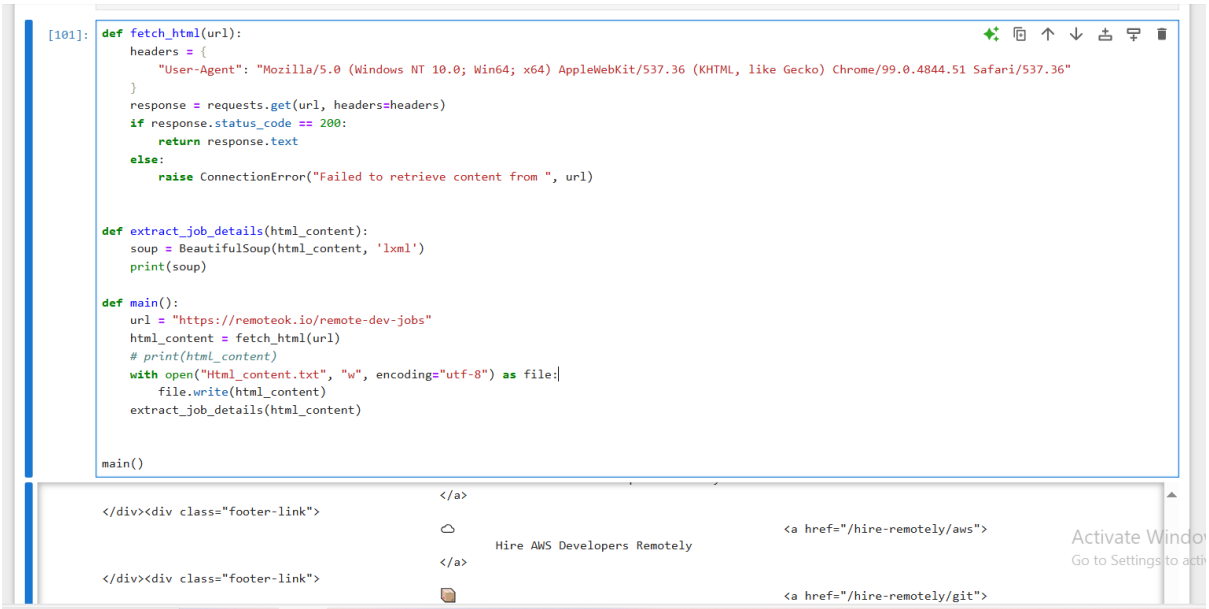
Explanation:-

soup = BeautifulSoup(html_content, 'lxml')

Above line creates soup object by passing the html_content to the BeautifulSoup constructor. 'lxml' parser is used.

BeautifulSoup: It's a Python library used for parsing HTML and XML documents, transforming them into a tree structure that can be easily navigated and searched.

Output after running:



Step4:-

```
def extract_job_details(html_content):
    soup = BeautifulSoup(html_content, 'lxml')
    jobs = []
    job_listings = soup.find_all('tr', class_='job')
    print(job_listings)
```

Explanation:-

job_listings = soup.find_all('tr', class_='job')

soup.find_all():-

used to search through the parsed HTML and find all instances of a specified tag. It returns a list of all matching elements.

In HTML, table rows are represented using the <tr> element.

In HTML, classes are used to apply styles and identify specific elements. class_='job' is searching for <tr> elements that have the class attribute job. The class_ argument uses an underscore because class is a reserved keyword in Python.

After above step we have filtered data as below.

```
    if response.status_code == 200:
        return response.text
    else:
        raise ConnectionError("Failed to retrieve content from ", url)


def extract_job_details(html_content):
    soup = BeautifulSoup(html_content, 'lxml')
    # print(soup)
    jobs = []

    job_listings = soup.find_all('tr', class_='job')
    print(job_listings)


def main():
    url = "https://remoteok.io/remote-dev-jobs"
    html_content = fetch_html(url)
    # print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:
        file.write(html_content)
    extract_job_details(html_content)


main()
```

```
[<tr class="job job-1062093 hot sticky bumped == highlighted" data-company="Composer Technologies" data-epoch="1739664005" data-href="/remote-jobs/remo
te-senior-backend-engineer-composer-technologies-1062093" data-id="1062093" data-offset="1" data-search="Composer Technologies Senior Backend Engineer
[" data-slug="remote-senior-backend-engineer-composer-technologies-1062093" data-stack="" data-url="/remote-jobs/remote-senior-backend-engineer-compose
r-technologies-1062093" engineer="" id="job-1062093" title="">
<td class="image has-logo" style="">
<script type="application/ld+json">
                                                {"@context":"http://schema.org","@type":"JobPosting","datePosted":"2025-02-16T0
0:00:05+00:00",
```


In job_listing we have all list of job element

Step 5:-

```
for job in job_listings:
    job_title = job.find('h2').text.strip()
```

```
        company_name = job.find('h3').text.strip()
        location = job.find('div', class_='location').text.strip()
        jobs.append((job_title, company_name, location))
```

for job in job_listings:

This line starts a loop that iterates over each item in the job_listings list. Each item in the list is job during each iteration.

job_title = job.find('h2').text.strip()

job.find('h2'): Searches for the first <h2> element within the job object.

.text: Extracts the text content inside the <h2> tags, example "Backend Engineer".

.strip(): Removes any leading or trailing space.

company_name = job.find('h3').text.strip()

Similarly, this line finds the first <h3> element within the job object, extracts its text content, remove any leading or trailing space, as company_name.

location = job.find('div', class_='location').text.strip()

This line finds the first element within the job object that has a class attribute with the value 'location'. It extracts the text content of this , strips any leading or trailing space,

jobs.append((job_title, company_name, location))

This line appends a tuple containing job_title, company_name, and location to the jobs list. Each tuple represents a job listing with its title, company name, and location.

Step 6:-

```
def save_to_file(job_details, filename):
    with open(filename, 'w', encoding='utf-8') as file:
        for job in job_details:
            file.write(f"Job Title: {job[0]}\n")
            file.write(f"Company: {job[1]}\n")
            file.write(f"Location: {job[2]}\n")
            file.write("\n")
```

Function save_to_file(job_details, filename): Takes the job details and a filename as input.

Write to File: Opens the specified file in write mode ('w') with UTF-8 encoding(If don’t use encoding it gives error).

Loop Through Jobs: Iterates through the job_details list and writes each job's details to the file.

Job Title: Writes the job title.

Company: Writes the company name.

Location: Writes the location.

New Line: Adds a blank line after each job.

Main Function:-

```
def main():
    url = "https://remoteok.io/remote-dev-jobs"
    html_content = fetch_html(url)                  #Calling fetch_html func
    # print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:    #for getting html file in text
        file.write(html_content)
    job_details = extract_job_details(html_content)        #Calling extract_job_details func
    save_to_file(job_details, "job_listings.txt")          #Calling save_to_file func
    print("Job listings saved to job_listings.txt")


main()
```

Function main(): Coordinates the entire process.

Fetch HTML Content: Fetches the HTML content from the specified URL.

Save HTML Content: Saves the fetched HTML content to a file named Html_content.txt for reference.

Extract Job Details: Extracts job details from the HTML content.

Save Job Details: Saves the extracted job details to a file named job_listings.txt.

Print Confirmation: Prints a confirmation message.

Final Output:

```
def save_to_file(job_details, filename):
    # with open(filename, 'w') as file:                      #without encoding it gives error
   with open(filename, 'w', encoding='utf-8') as file:         #Opens the specified file in write mode ('w') with UTF-8 encoding
        for job in job_details:                                #Iterates through the job_details list and writes each job's details to the file
            file.write(f"Job Title: {job[0]}\n")               #Writes the job title.
            file.write(f"Company: {job[1]}\n")                 #Writes the Company name.
            file.write(f"Location: {job[2]}\n")                #Writes the Location name.
            file.write("\n")                                     #for new line
def main():
    url = "https://remoteok.io/remote-dev-jobs"
    html_content = fetch_html(url)                        #Calling fetch_html func
    # print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:    #for getting html file in text
        file.write(html_content)
    job_details = extract_job_details(html_content)          #Calling extract_job_details func
    save_to_file(job_details, "job_listings.txt")             #Calling save_to_file func
    print("Job listings saved to job_listings.txt")


main()
```

```
Job listings saved to job_listings.txt
```

In text file:

```
Job Title: Senior Solutions Engineer
Company: pganalyze
Location: CA Canada

Job Title: Senior Backend Golang Engineer
Company: Hippo Technologies
Location: 🌎 Worldwide

Job Title: Senior Backend Engineer
Company: Composer Technologies
Location: 🌎 North America

Job Title: Senior Frontend Dev
Company: Seats.io
Location: EU Europe

Job Title: Senior full stack engineer
Company: Ideal Protein
Location: 🌎 Worldwide

Job Title: Engineering Product Lead
Company: Seed Oil Scout
Location: 🌎 Worldwide

Job Title: Full Stack Developer with AI Expertise
Company: Stealth Start-up
Location: 🌎 Worldwide

Job Title: Internet of Things Technical Lead
Company: Sanctuary Computer
Location: 🌎 Worldwide
```


Code:-

```
def fetch_html(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36"
```

```
    }
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.text
    else:
        raise ConnectionError("Failed to retrieve content from ", url)


def extract_job_details(html_content):
    soup = BeautifulSoup(html_content, 'lxml')
    # print(soup)
    jobs = []

    job_listings = soup.find_all('tr', class_='job')
    # print(job_listings)
    for job in job_listings:
        job_title = job.find('h2').text.strip()
        company_name = job.find('h3').text.strip()
        location = job.find('div', class_='location').text.strip()
        jobs.append((job_title, company_name, location))

    return jobs


def save_to_file(job_details, filename):
    # with open(filename, 'w') as file:                    #without encoding it gives error
   with open(filename, 'w', encoding='utf-8') as file:         #Opens the specified file in write mode ('w') with UTF-8 encoding
        for job in job_details:                          #Iterates through the job_details list and writes each job's details to the file
            file.write(f"Job Title: {job[0]}\n")            #Writes the job title.
            file.write(f"Company: {job[1]}\n")              #Writes the Company name.
```

```
            file.write(f"Location: {job[2]}\n")              #Writes the Location name.
            file.write("\n")                                 #for new line
def main():
    url = "https://remoteok.io/remote-dev-jobs"
    html_content = fetch_html(url)                    #Calling fetch_html func
    # print(html_content)
    with open("Html_content.txt", "w", encoding="utf-8") as file:    #for getting html file in text
        file.write(html_content)
    job_details = extract_job_details(html_content)         #Calling extract_job_details func
    save_to_file(job_details, "job_listings.txt")           #Calling save_to_file func
    print("Job listings saved to job_listings.txt")



main()
```