Scrapping

```
import requests

import csv

import time

from bs4 import BeautifulSoup

import pandas as pd



# Define the locations and job search keywords

#locations = [

    # Asia

    #Mumbai', 'Delhi', 'Bangalore', 'Hyderabad', 'Ahmedabad', 'Chennai',
'Kolkata','Hong Kong',

    # Africa

     #'Johannesburg', 'Cape Town', 'Nairobi','Accra','Kampala','Addis Ababa',
'Durban', 'Pretoria', 'Kimberley'

#]

#job_search_keywords = [

    # Tech industry

    #'Data+Scientist', 'Software+Engineer', 'Full+Stack+Developer',
'Front+End+Developer', 'Back+End+Developer',

    #'DevOps+Engineer', 'Database+Administrator', 'Data+Analyst',

    # Finance industry

    #'Financial+Analyst', 'Accountant', 'Financial+Advisor', 'Auditor',
'Insurance+Agent',

    #'Investment+Banker', 'Tax+Consultant', 'Budget+Analyst', 'Cost+Estimator',
'Loan+Officer',

    # Healthcare industry
```

```
    #'Nurse', 'Physician', 'Pharmacist', 'Dietitian', 'Physical+Therapist',
    #'Radiologic+Technologist', 'Medical+Technologist', 'Home+Health+Aide',
'Medical+Transcriptionist', 'Phlebotomist',
    # Education industry
    #'Teacher', 'Teaching+Assistant',
    #'Childcare+Worker',  'Preschool+Teacher', 'College+Counselor',
'School+Psychologist',
    # Marketing industry
    #'Marketing+Manager', 'SEO+Specialist', 'Content+Marketing+Manager',
'Brand+Manager', 'Public+Relations+Specialist',
    #'Media+Planner', 'Product+Manager', 'Digital+Marketing+Manager',
'Social+Media+Manager', 'Marketing+Analyst'
#]
locations = ['London']
job_search_keywords = ['Business+Analyst']
# Loop through each location and job search keyword and extract the job data
job_data = []
for location in locations:
    for job_search_keyword in job_search_keywords:
        # Define the URL of the job search page
        url =
f'https://www.linkedin.com/jobs/search/?keywords={job_search_keyword}&location={l
ocation}'

        # Send a GET request to the URL and store the response
        response = requests.get(url)

        # Parse the HTML content of the response using Beautiful Soup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find all the job listings on the page
```

```
        #job_listings = soup.find_all('li', class_='job-search-card')


        # print(soup)
        # Find all of the job listings on the page
        job_listings = soup.find_all(class_='job-search-card')
        # For each job listing, extract the title, company, and location
        for i, listing in enumerate(job_listings):
            title = listing.find('h3').text.strip()
            company = listing.find('h4').text.strip()
            location = listing.find('span', class_='job-search-
card__location').text.strip()
            job_link = listing.find("a", href=True)["href"]
            resp = requests.get(job_link)
            soup=BeautifulSoup(resp.text,'html.parser')
            #time.sleep(5)  # Pause to avoid overwhelming the server
            job_criteria_list = soup.find("ul", {"class": "description__job-
criteria-list"})
            if job_criteria_list:
               job_criteria = job_criteria_list.find_all("li")
            else:
               break
            try:
                level=job_criteria[0].text.replace("Seniority level","").strip()
            except:
                level=None
            try:
                empType=job_criteria[1].text.replace("Employment
type","").strip()
            except:
                empType=None
```

```
            try:
                empFunction=job_criteria[2].text.replace("Job
function","").strip()
            except:
                empFunction=None
            try:
                industries=job_criteria[3].text.replace("Industries","").strip()
            except:
                industries=None
            try:
                salary=soup.find("div", {"class": "salary
compensation__salary"}).text.strip()
            except:
                salary=None

            job_data.append({'job_id':i, 'title': title, 'location': location ,
'department':None, 'salary_range': salary ,
                             'company_profile':company,'description':None,
'requirements':None, 'benefits':None,
                             'telecommuting':0 ,'has_company_logo':0 ,
'has_questions':0 , 'employment_type': empType,
                             'required_experience':
level,'required_education':None,'industry': industries,
                             'function': empFunction , 'fraudulent':0 })
            # Define the file name and header row


# Convert the job data to a pandas DataFrame
df_linkedin = pd.DataFrame(job_data)


# Save the DataFrame to a CSV file
df_linkedin.to_csv('jobs_linkedin.csv', index=False, encoding='utf-8')
```

```
print("Web Scrapping done!")
```

- Le script commence par importer les bibliothèques nécessaires : requests, csv, time, BeautifulSoup de bs4, et pandas.
- Il définit des variables pour les emplacements (dans ce cas, uniquement 'London') et les mots-clés de recherche d'emploi (dans ce cas, uniquement 'Business+Analyst').
- Ensuite, il initialise une liste vide pour stocker les données des offres d'emploi.
- Le script boucle à travers chaque emplacement et chaque mot-clé de recherche d'emploi.
- Pour chaque combinaison emplacement-mot-clé, il construit une URL de recherche d'emploi sur LinkedIn.
- Ensuite, il envoie une requête GET à l'URL et analyse la réponse HTML avec BeautifulSoup.
- Il trouve toutes les offres d'emploi sur la page.
- Pour chaque offre d'emploi, il extrait le titre, l'entreprise, l'emplacement et le lien de l'offre d'emploi.
- Ensuite, pour chaque offre d'emploi, il suit le lien pour obtenir plus d'informations sur l'offre.
- Il extrait des informations supplémentaires telles que le niveau de seniorité, le type d'emploi, la fonction d'emploi, les industries et le salaire.
- Il ajoute toutes ces informations extraites à la liste des données d'emploi.
- Une fois toutes les offres d'emploi traitées, il convertit la liste de données d'emploi en un DataFrame pandas.
- Enfin, il enregistre le DataFrame dans un fichier CSV nommé 'jobs_linkedin.csv' et imprime un message indiquant que le web scraping est terminé.