**Bayt website Saudi Arabia job data extraction script documentation**

**Overview:**

This Python script is designed to automate the process of collecting **Saudi Arabia** job listings and related information from the Bayt.com website. The script accomplishes this task by utilizing a range of programming libraries and techniques. It is meant to serve as a flexible and efficient tool for extracting job data for analysis and reporting.

**Code Structure:**

The code is organized into distinct sections and functions, each serving a specific role in the data extraction process. Below, we provide an in-depth explanation of each major section and function

**Package and versions:**

Here's a list of the packages used in the code along with their versions:

1. **Python** – version: **3.7**
2. **requests** – version: **2.2.0**
3. **bs4 (BeautifulSoup4)** – version: **4.12.2**
4. **random –** version: **1.0.1**
5. **mtranslate(translate) –** version: **1.8**
6. **pip –** version: **10.0.1**
7. **logging**
8. **os**
9. **csv**
10. **time**

**Import statements:**

The script begins by importing the necessary Python libraries and modules. These imports are essential for enabling various functionalities within the script:

* **os.path:** Used for file and directory path manipulation.
* **requests:** Allows making HTTP requests to web servers.
* **csv:** Provides functionality for working with CSV (Comma-Separated Values) files.
* **time:** Offers functions for handling time-related operations.
* **BeautifulSoup (from bs4):** A library for parsing HTML and XML documents, facilitating web scraping.
* **logging:** A module for creating log messages and managing application logs.
* **mtranslate.translate (from mtranslate):** Provides translation capabilities.
* **random.uniform (from random):** Generates random floating-point numbers within specified ranges, often used for introducing delays when interacting with web resources

**Constants and setup:**

This section defines important constants and performs initial setup tasks:

* **dir\_path**: A constant representing the absolute path of the directory where the script is located. This ensures that the script knows its own location.
* **folder\_name**: The name of the folder where output data will be stored.
* **file\_name**: The name of the CSV file in which extracted data will be saved.
* **headers**: A dictionary containing HTTP headers with a user-agent string. This is included in HTTP requests to mimic a web browser request and avoid being blocked by the website's security mechanisms.

**Directory creation:**

Before proceeding with data extraction, the script attempts to create a directory specified in `folder\_name`. This directory stores the output data, such as the extracted job listings and details. If the directory already exists, the script continues without raising an error. This ensures a clean and organized structure for storing data.

**Main function:**

The `main` function serves as the central entry point for the script's execution:

* It initializes the URL for job listings on Bayt.com and the location is set to Saudi Arabia in the Bayt website.
* Calls the `fetch\_job\_ids` function to retrieve job IDs from the website.
* Initiates the data retrieval and processing using synchronous calls, enhancing performance by allowing multiple requests to be made simultaneously.
* Handles scenarios where data retrieval for certain job IDs may fail or result in missing fields.
* Finally, it saves the extracted data to a CSV file and removes any empty rows to ensure data consistency.

**Fetch\_job\_ids function:**

The `fetch\_job\_ids` function handles the retrieval of job IDs from Bayt.com. It does so by iterating through paginated job listings:

* The function initializes an empty set to store all job IDs.
* It starts from the first page of job listings and continues until there are no more job IDs to fetch.
* The script uses the beautifulsoup library to parse HTML content and extract job IDs.
* Once all job IDs are collected, they are returned as a list.

**Translate to English function**:

This function translates non-English text in a CSV file to English. It takes the following parameters:

**input\_csv\_file**: The input CSV file path.

**output\_csv\_file**: The output CSV file path.

**column1\_name**: The name of the first column to translate.

**column2\_name**: The name of the second column to translate.

* Opens the input CSV file for reading and the output CSV file for writing.
* Reads the CSV data using csv.DictReader and writes the translated data using csv.DictWriter.
* For each row in the input CSV, it checks if the text in column1\_name and column2\_name is non-English.
* If non-English text is detected, it attempts to translate the text to English using the mtranslate.translate function.
* Writes the updated row with translated text to the output CSV file.

**Remove empty rows from csv function:**

This function removes empty rows from a CSV file. It takes the following parameters:

**input\_csv\_file:** The input CSV file path**.**

**output\_csv\_file:** The output CSV file path**.**

* Opens the input CSV file for reading and the output CSV file for writing.
* Reads the CSV data using csv.DictReader and writes non-empty rows to the output CSV file using csv.DictWriter.
* Removes rows where all values are empty.

**fetch\_data\_for\_job\_id function:**

The `fetch\_data\_for\_job\_id` function is responsible for fetching detailed job data for a given job ID.

It takes the following parameters:

**job\_id**: The ID of the job to fetch data for.

**retries**: The number of retries in case of errors (default is 3).

**backoff\_factor**: The backoff factor for exponential backoff in case of rate limiting (default is 2).

This function is called for each job ID:

* It constructs the URL for the specific job ID and sends an HTTP request to Bayt.com.
* Upon receiving a response, the function handles various response status codes, including 200 (success), 429 (Rate Limited), and 404 (Not Found).
* The HTML content of the page is parsed using BeautifulSoup to extract job details.
* The extracted information is stored in a dictionary.
* The function accounts for different types of job details, such as job names, company names, and additional attributes.
* Finally, the function returns a dictionary containing the job details.

**Fetch\_failed\_job\_data function:**

The `fetch\_failed\_job\_data` function is tasked with retrieving data for job IDs that previously failed to fetch. It utilizes synchronous calls to fetch data for job IDs:

* The function accepts a list of failed job IDs as input.
* It creates a list to store all the retrieved data for previously failed job ids.
* The script then uses synchronous calls to the `fetch\_data\_for\_job\_id` function for each job ID.
* The retrieved data is appended to the list.
* Once all data is fetched, the list of job details is returned.

**goto\_next\_page function:**

The `goto\_next\_page` function manages the navigation to the next page of job listings on Bayt.com. It handles various HTTP response codes and situations:

* The function constructs the URL for the next page based on the current page number.
* It makes an HTTP request to the website.
* Upon receiving a response, the function interprets the HTTP status code.
* The script handles various response status codes, including 200 (success), 502 (Bad Gateway), 429 (Rate Limited), and 404 (Not Found).
* To avoid overloading the website, the function introduces random delays before retrying.
* The function returns the page content and a boolean indicating whether it's the last page.

**Output file location:**

In the provided Python code, the output CSV file is stored in a specific directory with the filename as “**Bayt\_Saudi\_Arabia\_Job\_Data.csv**”. Here is a breakdown of where the CSV file will be located and what it will be named:

1. **Directory Path**:
   * The script determines the directory path where the CSV file will be stored using the **os.path.abspath(os.path.dirname(\_\_file\_\_))** method. This path represents the location of the Python script itself.
2. **Folder Name**:
   * The variable **folder\_name** specifies the name of the folder where the output CSV file will be saved. In this code, it is set to "**\_Output**"
3. **File Name**:
   * The variable **file\_name** specifies the name of the output CSV file. In this code, it is set to " **Bayt\_Saudi\_Arabia\_Job\_Data.csv**"

So the output csv will be stored in **\_output** folder with the file name **Bayt\_Saudi\_Arabia\_Job\_Data.csv.**

**Execution:**

The script's main execution logic is contained within an `if \_\_name\_\_ == '\_\_main\_\_':` block. This block is executed when the script is run as the main program:

* It calls the `main` function to initiate the data scraping process.

**Dependencies:**

The script depends on several external libraries and modules:

* BeautifulSoup is used for parsing HTML content.
* The `requests` library handles HTTP requests.
* `csv` assists in reading and writing CSV files.
* `random.uniform` generates random delays for more human-like behavior.

Additionally, a stable internet connection is required to fetch data from Bayt.com.

**Error Handling:**

The script effectively handles various types of errors:

1. **502 - Bad Gateway**: When the server reports this error, indicating a temporary issue, the script retries the request after a delay.

2. **429 - Too Many Requests (Rate Limiting)**: When rate limiting is in effect, the script waits for the recommended duration before retrying to avoid overloading the server.

3. **404 - Not Found**: If the script encounters this error, it considers it as reaching the last page of listings and stops data fetching.

4. **Request Timeouts**: In case of timeouts, the script retries the request after introducing a delay.

5. **Connection Errors**: The script handles connection errors and retries, ensuring resilience.

6. **Informative Messages**: It provides user-friendly messages for clarity during the scraping process.