**Glassdoor Scrapper Documentation**

**Files:**

**Scrapper\_glassdoor.py**

This is the file which scrapes the data from the glassdoor job postings and sends email to the user. **properties.json**

This is the json file which contains the server name, username, password, database name. These are required to connect to the database.

**Variables:**

Data - to store the loaded json file for database connection.

final\_skills – to store the skills ids and the corresponding skills.

Mapping\_dict – to store the mapping between the resume id and the corresponding skill id.

Email\_dict – to store the resume id and the corresponding email id of the users.

Final\_dict – to store the job description links scrapped from glassdoor.

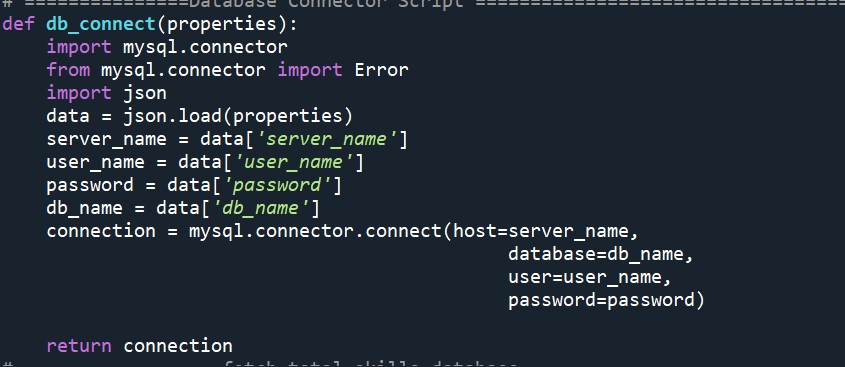
Total – to store the user id of the user and the job links which matches with skills in the resume of the user.

Job\_details – to store job url , role name and company name scrapped from glassdoor.

**Code segments:**

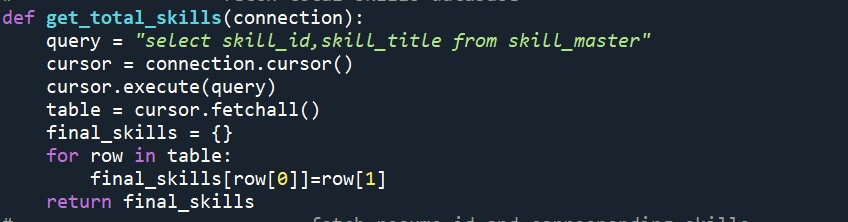
# 1. Db\_connect( properties )

This function loads the data from the properties.json file and connects to the database. It returns the connection object.



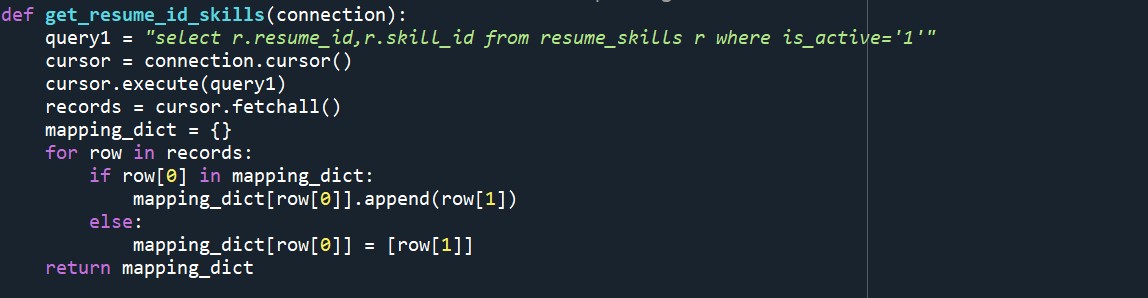
# 2. get\_total\_skills(connection)

This function executes the query to fetch the skill id and the corresponding skills from the skill\_master table. The result is stored in the dictionary name final\_skills.



# 3. get\_resume\_id\_skills(connection)

This function executes the query to fetch the resume\_id of the user and the corresponding skill id from the resume\_skills table. The result is stored in the dictionary name mapping\_dict.



# 4. get\_email\_id\_users(connection)

This function executes the query to fetch the resume id , first name and the email id of the user from user\_resume and user\_master tables based on a join. The result is stored in the dictionary name email\_dict.

Text

Description automatically generated

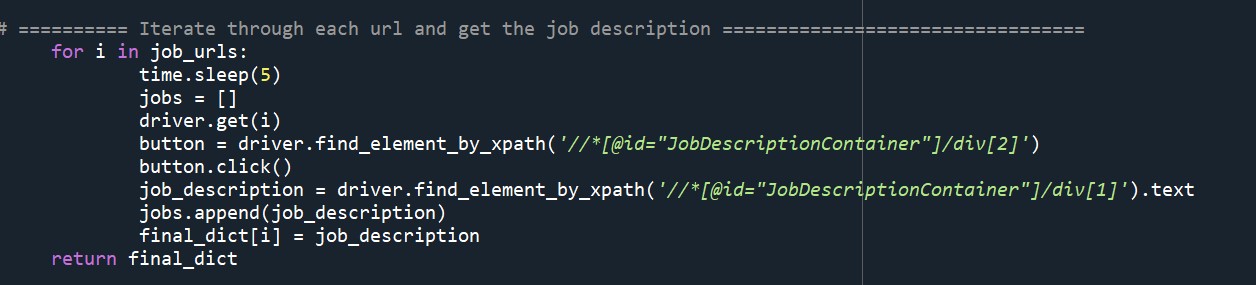
# 5. get\_job\_description(keyword,num\_jobs,verbose)

This function is used to scrape the glassdoor website, the function takes the keyword, i.e the title of the job which the user wants to apply and the number of jobs openings which the user wants to see.

First all the jobs postings urls are collected and stored in a list named job\_urls.



Then the job\_urls list is iterated to get the job description and each job description is stored in list named final\_dict



# 6. ke.get\_user\_id\_to\_list\_of\_job\_ids(mapping\_dict,final\_dict,connection,final\_skills,threshol d)

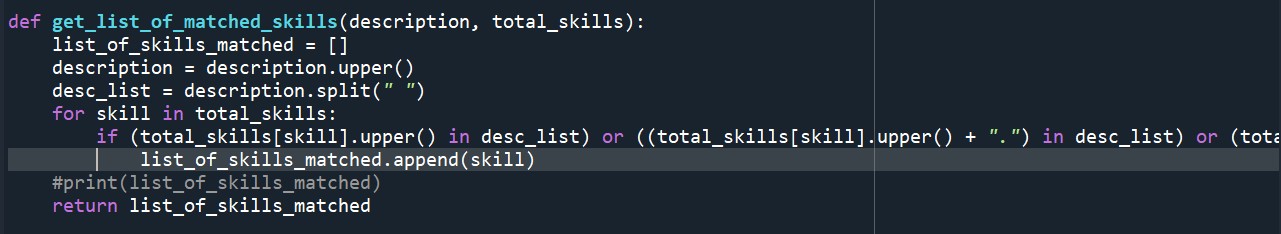
This function is taken from a different python file named keyword\_extraction\_modules.

This function returns user id with their corresponding job openings links that matches with the skills in the resume of the user.

This file uses four different functions:

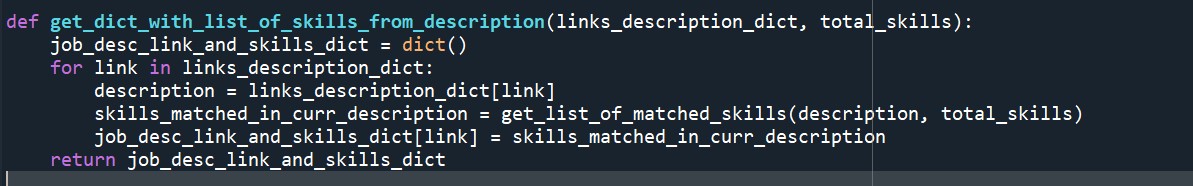
# a. get\_list\_of\_matched\_skills(description, total\_skills)

This function takes the description and matches with skills from the skills table. It returns the list of matches skills



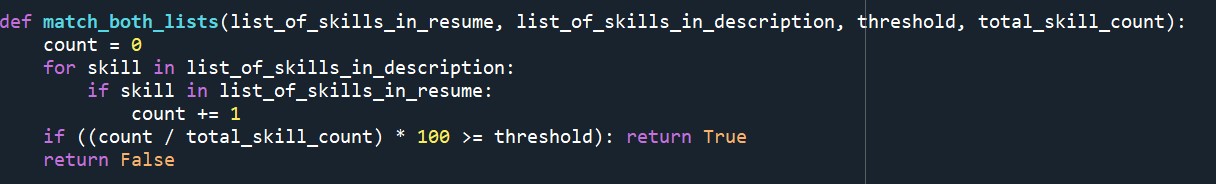
# b. get\_dict\_with\_list\_of\_skills\_from\_description(links\_description\_dict, total\_skills)

This function creates a dictionary of job description and skills using the get\_list\_of\_matched\_skills function.



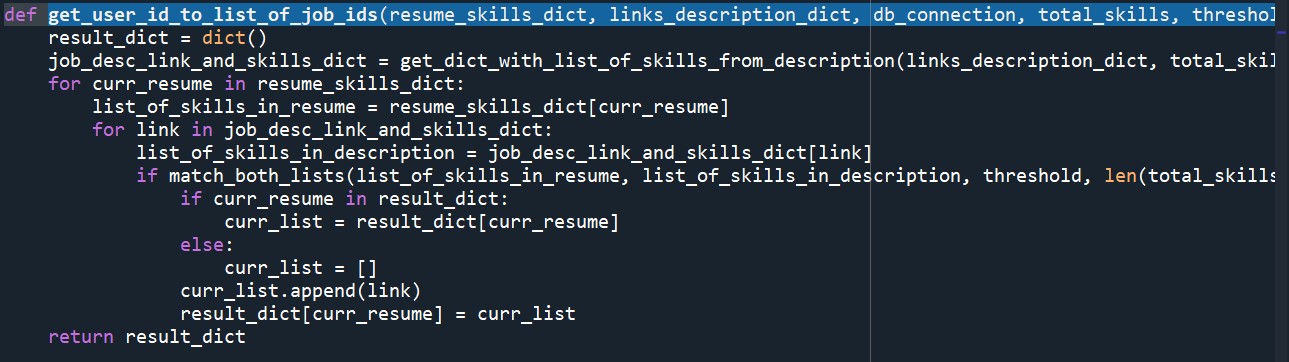
# c. match\_both\_lists(list\_of\_skills\_in\_resume, list\_of\_skills\_in\_description, threshold, total\_skill\_count)

This function returns true if the % of skills in the resume matches skills in the job description is greater than the threshold.



# d. get\_user\_id\_to\_list\_of\_job\_ids(resume\_skills\_dict, links\_description\_dict, db\_connection, total\_skills, threshold)

This function returns the user ids and the corresponding job links the matches with the skills in the resume.



**Sending email to the user**

7. The result from ke.get\_user\_id\_to\_list\_of\_job\_ids(mapping\_dict,final\_dict,connection,final\_skills,threshold) and get\_email\_id\_users are used to send email to the user using smtplib library

Text

Description automatically generated