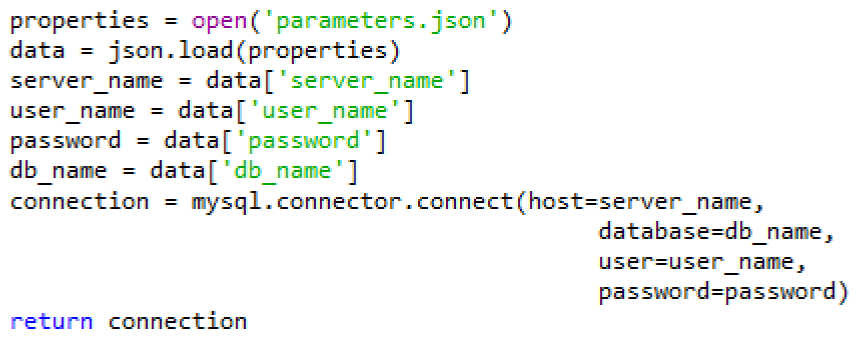
Scraper Service-Linked In

Scrapper.py Functionality :

- db_connect(properties):

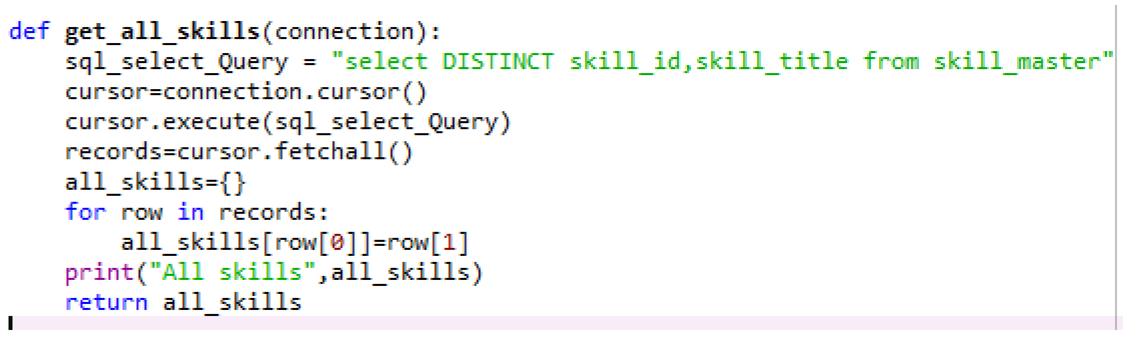
This function takes properties which basically contains database parameters loaded from parameters.json file.



```
properties = open('parameters.json')
data = json.load(properties)
server_name = data['server_name']
user_name = data['user_name']
password = data['password']
db_name = data['db_name']
connection = mysql.connector.connect(host=server_name,
                                     database=db_name,
                                     user=user_name,
                                     password=password)

return connection
```

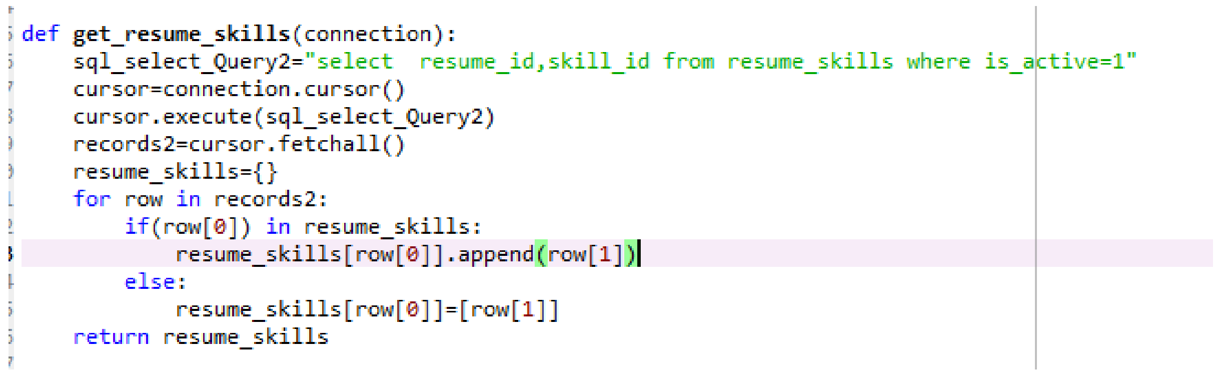
Returns: database connection object

- get_all_skills(connection)
  Argument: connection object
  Returns : List of all skill id’s and skill names from our skill_master table.



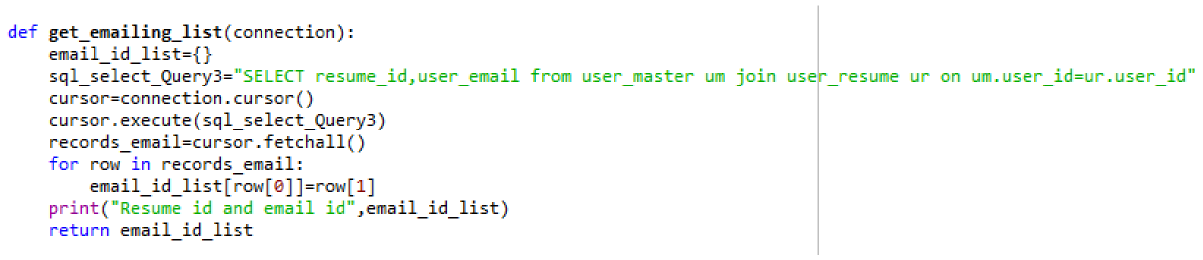
```
def get_all_skills(connection):
    sql_select_Query = "select DISTINCT skill_id,skill_title from skill_master"
    cursor=connection.cursor()
    cursor.execute(sql_select_Query)
    records=cursor.fetchall()
    all_skills={}
    for row in records:
        all_skills[row[0]]=row[1]
    print("All skills",all_skills)
    return all_skills
```

- get_resume_skills(connection)
- Argument: connection object
- Returns : List of all resume id and resume skills from our resume_skills table.



```
def get_resume_skills(connection):
    sql_select_Query2="select  resume_id,skill_id from resume_skills where is_active=1"
    cursor=connection.cursor()
    cursor.execute(sql_select_Query2)
    records2=cursor.fetchall()
    resume_skills={}
    for row in records2:
        if(row[0]) in resume_skills:
            resume_skills[row[0]].append(row[1])
        else:
            resume_skills[row[0]]=[row[1]]
    return resume_skills
```

- get_emailing_list(connection):



```
def get_emailing_list(connection):
    email_id_list={}
    sql_select_Query3="SELECT resume_id,user_email from user_master um join user_resume ur on um.user_id=ur.user_id"
    cursor=connection.cursor()
    cursor.execute(sql_select_Query3)
    records_email=cursor.fetchall()
    for row in records_email:
        email_id_list[row[0]]=row[1]
    print("Resume id and email id",email_id_list)
    return email_id_list
```

Argument: connection object returned by db_connect function()

Returns: Returns a dictionary containing a particular resume_id and the corresponding email id of the user to whom the resume belongs to.

- get_job_description(keyword,no_of_jobs_to_retrieve,data):

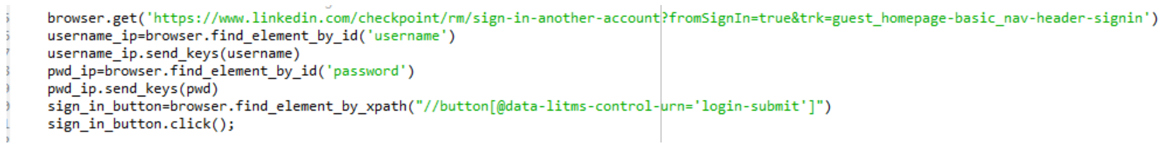
Arguments:

1. Keyword: takes the search query i.e which job profile is the user looking for to be used as filter later.
2. No_of_jobs_to_retrieve: Takes the number of jobs user wants to search and retrieve from each site .
3. data basically has information from parameters.json which has the password for the dummy linked in account.

Execution:

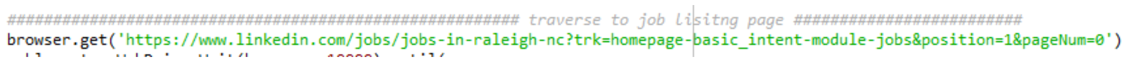
1. Login in to a dummy linked in profile.

Code snippet



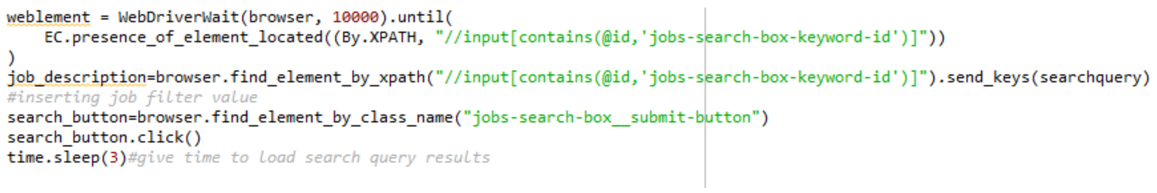
```
browser.get('https://www.linkedin.com/checkpoint/rm/sign-in-another-account?fromSignIn=true&trk=guest_homepage-basic_nav-header-signin')
username_ip=browser.find_element_by_id('username')
username_ip.send_keys(username)
pwd_ip=browser.find_element_by_id('password')
pwd_ip.send_keys(pwd)
sign_in_button=browser.find_element_by_xpath("//button[@data-litms-control-urn='login-submit']")
sign_in_button.click();
```

2. After logging in the web browser takes us to the home page from which we are redirecting the browser via selenium to go the job listing page.



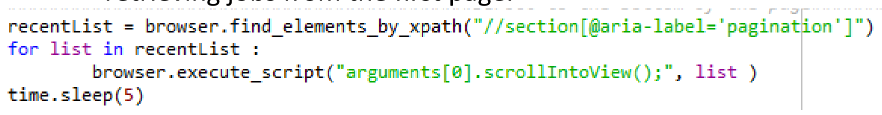
```
############################################################ traverse to job lisitng page ##########################
browser.get('https://www.linkedin.com/jobs/jobs-in-raleigh-nc?trk=homepage-basic_intent-module-jobs&position=1&pageNum=0')
```

3. At the job listing page we enter the search query which is stored in our keyword variable.



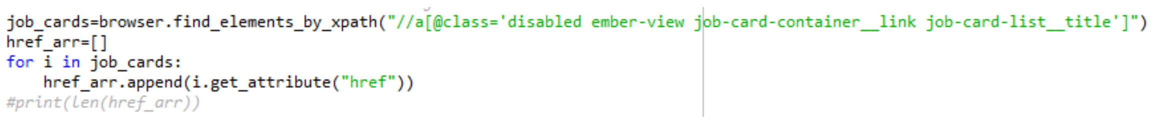
```
weblement = WebDriverWait(browser, 10000).until(
    EC.presence_of_element_located((By.XPATH, "//input[contains(@id,'jobs-search-box-keyword-id')]"))
)
job_description=browser.find_element_by_xpath("//input[contains(@id,'jobs-search-box-keyword-id')]").send_keys(searchquery)
#inserting job filter value
search_button=browser.find_element_by_class_name("jobs-search-box__submit-button")
search_button.click()
time.sleep(3)#give time to load search query results
```

4. After the search query is executed we scroll done to the end of the job listings page so that all possible job listings on the current page can be seen. Currently we are only retrieving jobs from the first page.



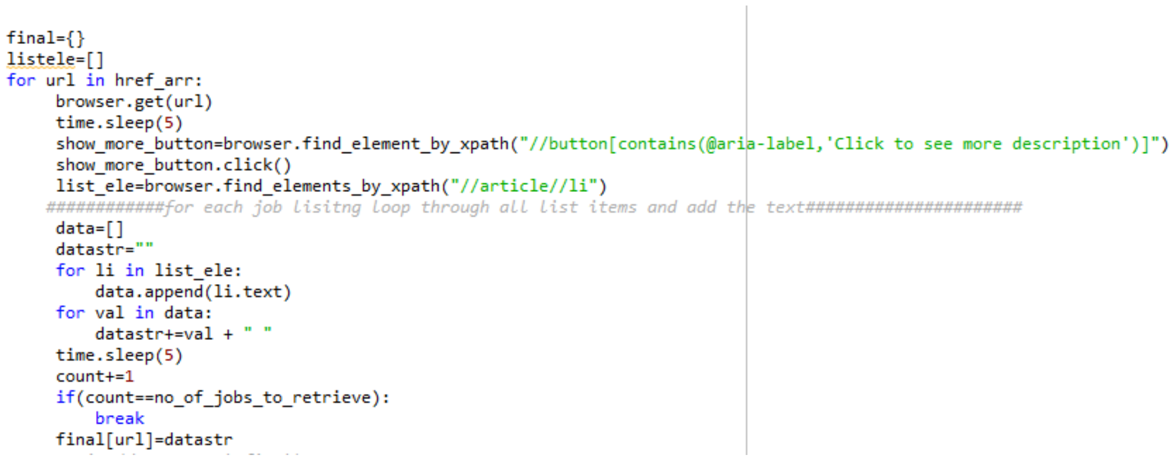
```
recentList = browser.find_elements_by_xpath("//section[@aria-label='pagination']")
for list in recentList :
        browser.execute_script("arguments[0].scrollIntoView();", list )
time.sleep(5)
```

5. Now we retrieve all the job links from the job listing panel and store them in a list



```
job_cards=browser.find_elements_by_xpath("//a[@class='disabled ember-view job-card-container__link job-card-list__title']")
href_arr=[]
for i in job_cards:
    href_arr.append(i.get_attribute("href"))
#print(len(href_arr))
```

6. Now for each of the job listing from that list we scrape all the relevant data such as responsibilities,skills and so on(usually written in list tags on each job posting) and storing it in a list.After that, we are converting all the entries in that list to a single string variable



```
final={}
listele=[]
for url in href_arr:
     browser.get(url)
     time.sleep(5)
     show_more_button=browser.find_element_by_xpath("//button[contains(@aria-label,'Click to see more description')]")
     show_more_button.click()
     list_ele=browser.find_elements_by_xpath("//article//li")
    ############for each job lisitng loop through all list items and add the text#####################
     data=[]
     datastr=""
     for li in list_ele:
         data.append(li.text)
     for val in data:
         datastr+=val + " "
     time.sleep(5)
     count+=1
     if(count==no_of_jobs_to_retrieve):
         break
     final[url]=datastr
```

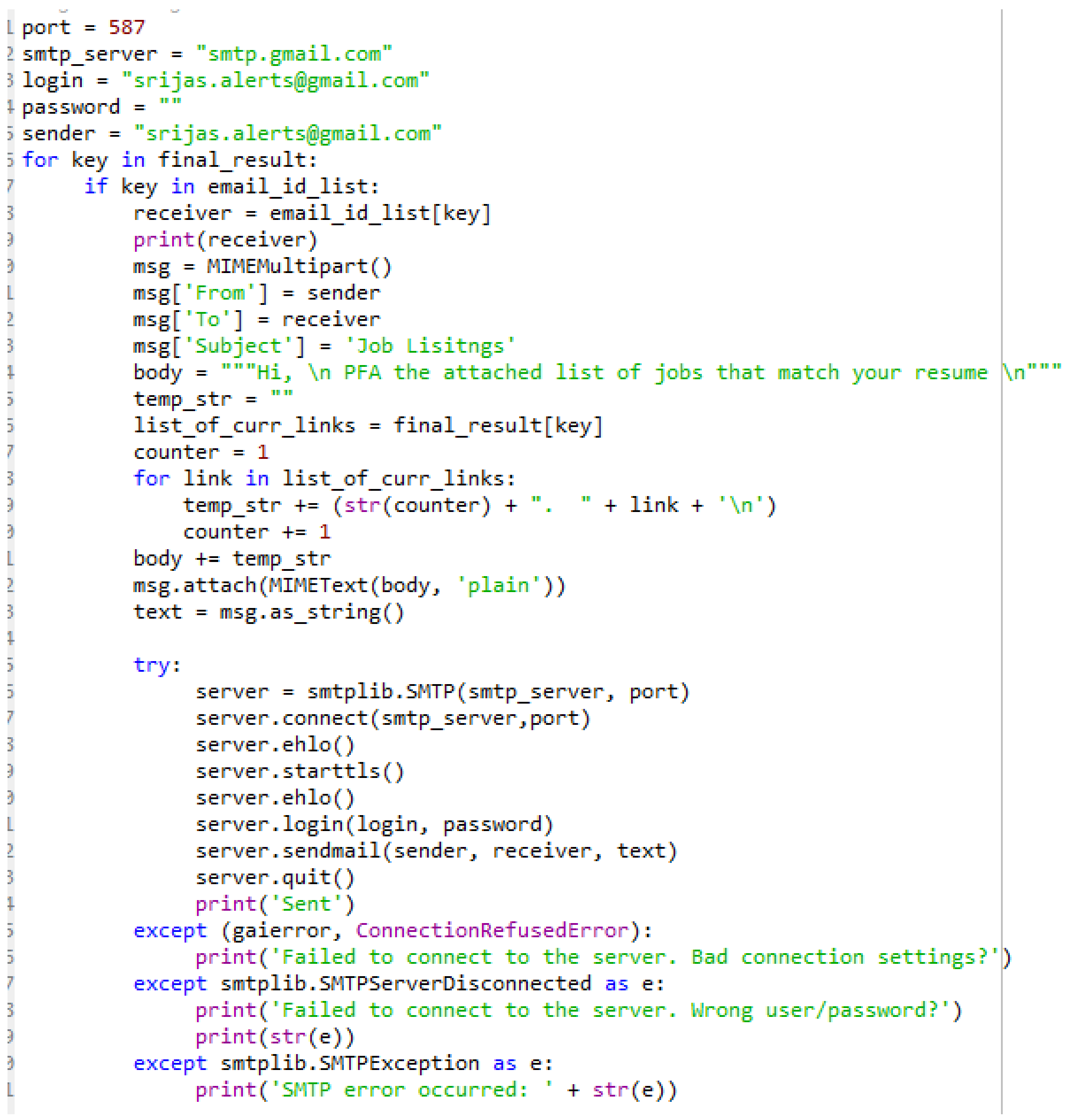
7. This is then passed to the get_user_id_to_list_of_job_ids() function.

Returns:

Final list containing email_id and their corresponding job links is returned.

Email service

We are using the smtp protocol to send email containing the job links to each of the email id



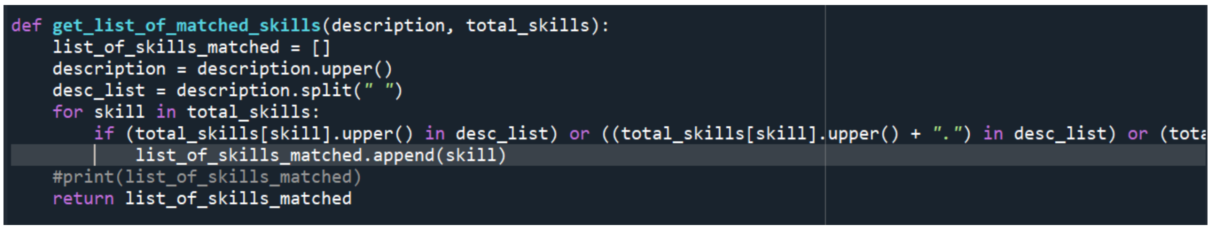
```python
port = 587
smtp_server = "smtp.gmail.com"
login = "srijas.alerts@gmail.com"
password = ""
sender = "srijas.alerts@gmail.com"
for key in final_result:
    if key in email_id_list:
        receiver = email_id_list[key]
        print(receiver)
        msg = MIMEMultipart()
        msg['From'] = sender
        msg['To'] = receiver
        msg['Subject'] = 'Job Lisitngs'
        body = """Hi, \n PFA the attached list of jobs that match your resume \n"""
        temp_str = ""
        list_of_curr_links = final_result[key]
        counter = 1
        for link in list_of_curr_links:
            temp_str += (str(counter) + ".  " + link + '\n')
            counter += 1
        body += temp_str
        msg.attach(MIMEText(body, 'plain'))
        text = msg.as_string()

        try:
             server = smtplib.SMTP(smtp_server, port)
             server.connect(smtp_server,port)
             server.ehlo()
             server.starttls()
             server.ehlo()
             server.login(login, password)
             server.sendmail(sender, receiver, text)
             server.quit()
             print('Sent')
        except (gaierror, ConnectionRefusedError):
             print('Failed to connect to the server. Bad connection settings?')
        except smtplib.SMTPServerDisconnected as e:
             print('Failed to connect to the server. Wrong user/password?')
             print(str(e))
        except smtplib.SMTPException as e:
             print('SMTP error occurred: ' + str(e))
```

keyword_extraction_modules.py

This file uses four different functions:

get_list_of_matched_skills(description, total_skills)

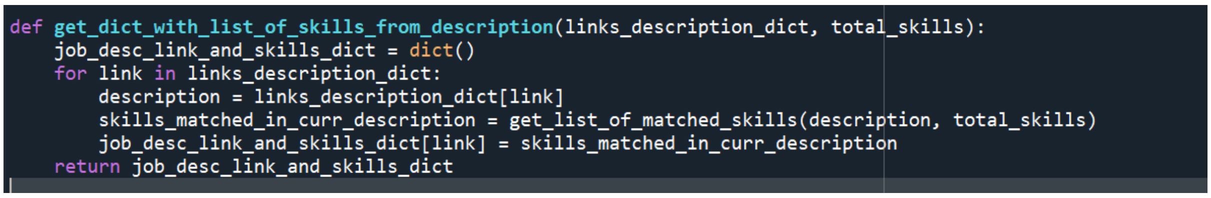
This function takes the description and matches with skills from the skills table. It returns the list of matches skills



```python
def get_list_of_matched_skills(description, total_skills):
    list_of_skills_matched = []
    description = description.upper()
    desc_list = description.split(" ")
    for skill in total_skills:
        if (total_skills[skill].upper() in desc_list) or ((total_skills[skill].upper() + ".") in desc_list) or (tota
            list_of_skills_matched.append(skill)
    #print(list_of_skills_matched)
    return list_of_skills_matched
```

get_dict_with_list_of_skills_from_description(links_description_dict, total_skills)

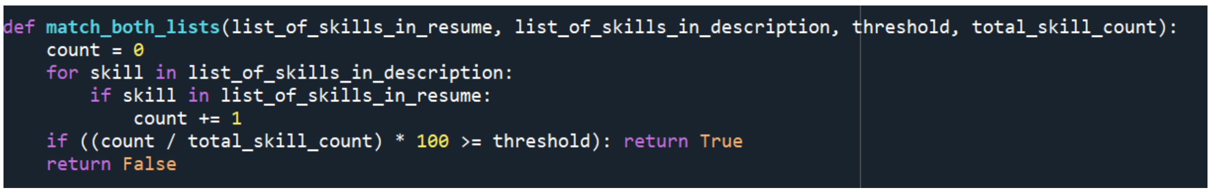
This function creates a dictionary of job description and skills using the get_list_of_matched_skills function.



```python
def get_dict_with_list_of_skills_from_description(links_description_dict, total_skills):
    job_desc_link_and_skills_dict = dict()
    for link in links_description_dict:
        description = links_description_dict[link]
        skills_matched_in_curr_description = get_list_of_matched_skills(description, total_skills)
        job_desc_link_and_skills_dict[link] = skills_matched_in_curr_description
    return job_desc_link_and_skills_dict
```

match_both_lists(list_of_skills_in_resume, list_of_skills_in_description, threshold, total_skill_count)

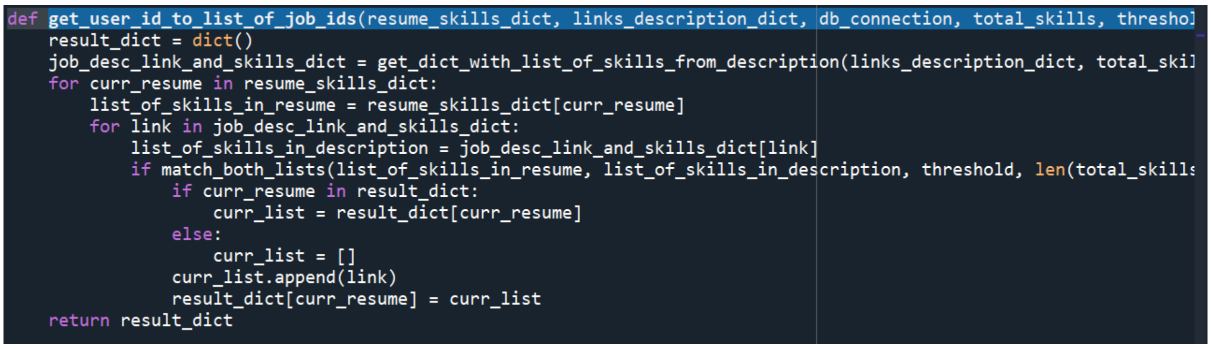
This functions returns true if the % of skills in the resume matches skills in the job description is greater than the threshold.



```python
def match_both_lists(list_of_skills_in_resume, list_of_skills_in_description, threshold, total_skill_count):
    count = 0
    for skill in list_of_skills_in_description:
        if skill in list_of_skills_in_resume:
            count += 1
    if ((count / total_skill_count) * 100 >= threshold): return True
    return False
```

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.



```python
def get_user_id_to_list_of_job_ids(resume_skills_dict, links_description_dict, db_connection, total_skills, threshol
    result_dict = dict()
    job_desc_link_and_skills_dict = get_dict_with_list_of_skills_from_description(links_description_dict, total_skil
    for curr_resume in resume_skills_dict:
        list_of_skills_in_resume = resume_skills_dict[curr_resume]
        for link in job_desc_link_and_skills_dict:
            list_of_skills_in_description = job_desc_link_and_skills_dict[link]
            if match_both_lists(list_of_skills_in_resume, list_of_skills_in_description, threshold, len(total_skills
                if curr_resume in result_dict:
                    curr_list = result_dict[curr_resume]
                else:
                    curr_list = []
                curr_list.append(link)
                result_dict[curr_resume] = curr_list
    return result_dict
```