Repository name: email-checker

Purpose: To write a server side function which receives a list of email addresses through a post request and responds with the number of unique emails in that list.

Requirements:

1. install python3.6
2. install flask

How to run ?

1. Start the email_checker.py function using the following command

   ```
   python3.6 email_check.py
   ```

   This starts the server side of the program and checks for any incoming post request with the route /unique_email

2. Can use the post_request.py to generate a post request

   ```
   python3.6 post_request.py
   ```

   We can also use postman to generate the post request in form, args or json formats.

Assumptions:

1. Using Flask for creating the web server side function, as this function be easily integrated with any other application through port forwarding.
2. A post request invokes the function to return unique emails.
3. The post request contains a field named “email_list” with the list of emails.
4. Post request is processed in args, form or json formats.
5. Usage of gmail username check in order to validate usernames [https://support.google.com/mail/answer/9211434?hl=en]

Code Logic Explanation

Email_checker.py:

1. /unique_email route invokes the method to check and return unique emails which are sent in the post request.
    a. Initial check for request format type
    b. Checking the domain name of the email address
    c. Check the validity of the username part
    d. Using a dictionary to check for uniqueness of the email
    e. Return number of unique emails found

Validity_check.py:

1. check_domain()
    a. Make sure the string contains a single @ character
    b. Obtain the string after @ using split()
    c. Check if we have already seen this domain using cached data structure
    d. If not seen before, check the validity of the domain name using socket.gethostbyname()
2. Check_username:
    a. Check username using rules provided by gmail username check [https://support.google.com/mail/answer/9211434?hl=en]

Time complexity:

N = number of emails
M = maximum number of characters in the username
S = Number of domains in the Domain name server list
O(N * M * S)

Space complexity:

S = Number of domains in the Domain name server list
P = Number of unique emails
O(P * S)