Unit4_Challenge

December 19, 2023

1 Unit 4: Challenge

1.1 Problem 1

1. Open the `exam.log` file.
2. Write a function `ip_result` that:

- Searches for lines with IP
- Counts the number of each IP
- Puts the results in a dictionary
- Sorts the dictionary
- Puts the results into a file

3. Write a function `invalid_user_count` that:

- Searches for invalid user logins
- Counts the invalid logins for each user
- Puts the results in a dictionary
- Sorts the dictionary
- Puts the result into a file

4. Write a function `failed_logins` that:

- Searches for wrong passwords
- Counts the failed logins
- Puts the results in a dictionary
- Sorts the dictionary
- Puts the result in a file

5. Call the functions

[21]:
```
import re
import os
#2------------------------------------------------------
#counts IPs and how many times they show up

def ip_result(file_path):
    with open (file_path) as f:
        dict_ip = {}
        for ip in f.readlines():
            match_ip = re.search("\d{1,3}\.\d{1,3}\.\d{1,3}.\d{1,3}", ip)
```

```
            if match_ip:
                ip_count = match_ip.group(0)
                if ip_count in dict_ip:
                    dict_ip[ip_count] += 1

                else:
                    dict_ip[ip_count] = 1
            else:
                pass

        #sort
        sorted_ip_dict = dict(sorted(dict_ip.items()))

    #save results to file
    with open("ip_results.txt", "w") as ip_file:
        for ip, count in sorted_ip_dict.items():
            ip_file.write(f"{ip}: {count} times\n")

    return sorted_ip_dict


#3--------------------------------------------------------------------------
#count invalid user logins - "Invalid user *user* from IP". lists users

def invalid_user_count(file_path):
    with open(file_path) as f:
        invalid_count= {}
        for line in f.readlines():
            match_ip = re.search("\d{1,3}\.\d{1,3}\.\d{1,3}.\d{1,3}", line)
            match_invalid_user = re.search(r"Invalid user (\S+)", line)

            if match_ip and match_invalid_user:
                inv_count = match_ip.group(0)
                user_name = match_invalid_user.group(1)

                if inv_count not in invalid_count:
                    invalid_count[inv_count] = {}

                if user_name in invalid_count[inv_count]:
                    invalid_count[inv_count][user_name] += 1
                else:
                    invalid_count[inv_count][user_name] = 1
            else:
                pass

        #sort dictionary
        sorted_inv_count = dict(sorted(invalid_count.items()))
```

```
    #save results to file
    with open("invalid_user_count.txt", "w") as invalid_file:
        for ip, users_count in sorted_inv_count.items():
            user_str = ', '.join([f"{user} ({count} times)" for user, count in␣
 ↪users_count.items()])
            invalid_file.write(f"{ip}: {len(users_count)} unique user(s) with␣
 ↪invalid login(s): {user_str}\n")

    return sorted_inv_count
#4--------------------------------------------------------------------------

    #find any users and count their failed login attempts. List users

def failed_logins(file_path):
    with open(file_path) as file:
        dict_flogs = {}
        for attempts in file.readlines():
            match_ip = re.search("\d{1,3}\.\d{1,3}\.\d{1,3}.\d{1,3}", attempts)
            match_user = re.search(r"Failed password for (?!Invalid)\S+\s+user␣
 ↪(\S+)",attempts)

            if match_ip and match_user:
                failed_count = match_ip.group(0)
                user_name = match_user.group(1)

                if failed_count not in dict_flogs:
                    dict_flogs[failed_count] = {}

                if user_name in dict_flogs[failed_count]:
                    dict_flogs[failed_count][user_name] +=1
                else:
                    dict_flogs[failed_count][user_name] = 1
            else:
                pass

        #sort dictionary
        sorted_failed_count = dict(sorted(dict_flogs.items()))

    #save results to a file
    with open("failed_logins.txt", "w") as failed_file:
        for ip, users_count in sorted_failed_count.items():
            user_str = ', '.join([f"{user} ({count} times)" for user, count in␣
 ↪users_count.items()])
            failed_file.write(f"{ip}: {len(users_count)} unique user(s) with␣
 ↪failed login(s): {user_str}\n")
```

```
    return sorted_failed_count

# my main squeeze
def main():

    file_path = "/voc/public/exam.log"


    print(f"IP COUNT RESULTS: {ip_result(file_path)}\n")
    print(f"INVALID USER COUNTS: {invalid_user_count(file_path)}\n")
    print(f"FAILED LOGIN COUNTS: {failed_logins(file_path)}\n")




main()
```

```
IP COUNT RESULTS: {'0.0.0.0': 2, '112.112.102.38': 10, '123.150.200.121': 5,
'137.74.1.62': 284, '138.19.133.51': 5, '146.0.75.199': 1, '151.40.221.171': 6,
'163.172.69.249': 1, '168.70.37.89': 5, '178.15.105.238': 1, '178.68.209.126':
10, '185.156.177.53': 1, '185.156.177.8': 1, '185.182.56.85': 450,
'185.222.211.18': 1, '192.129.227.186': 26, '196.52.43.94': 1, '197.245.39.234':
5, '212.237.47.236': 358, '213.202.230.144': 1, '221.148.234.252': 10,
'41.59.225.121': 5, '42.159.246.3': 5, '5.255.68.179': 20, '5.39.216.134': 2,
'5.8.10.202': 3, '71.6.165.200': 2, '80.211.140.131': 4, '80.237.75.2': 481,
'87.27.92.12': 70, '91.121.158.124': 1}

INVALID USER COUNTS: {'112.112.102.38': {'admin': 1}, '137.74.1.62': {'RPM': 3,
'openvpn': 3, 'm202': 3, 'redmine': 3, 'leo': 3, 'odoo': 3, 'apc': 3, 'user': 3,
'admin': 3}, '178.68.209.126': {'admin': 1}, '185.182.56.85': {'ubnt': 4,
'r00t': 4, 'test': 4, 'administrator': 4, 'admin': 4, 'support': 4, 'nagios': 4,
'Admin': 4, 'ftpuser': 4, 'operator': 4, '1': 4}, '192.129.227.186': {'user': 2,
'miner': 1, 'ethos': 1, 'oracle': 1}, '212.237.47.236': {'antivirus': 1,
'apache': 1, 'backuppc': 1, 'bank': 1, 'board': 1, 'build': 1, 'byte': 1,
'bytes': 1, 'centos': 1, 'chat': 1, 'cinema': 1, 'control': 1, 'counterstrike':
1, 'cpanel': 1, 'cs': 1, 'csgo': 1, 'cvs': 1, 'cyber': 1, 'cyrus': 1, 'data': 1,
'db1': 1, 'db2inst1': 1, 'debian': 2, 'developer': 1, 'downloads': 1, 'dvd': 1,
'dvs': 1, 'edu': 1, 'education': 1, 'fedora': 1, 'forum': 1, 'ftp': 1,
'ftpadmin': 1, 'gaming': 1, 'gb': 1, 'ghani': 1, 'ghost': 1, 'git': 7, 'gnu': 1,
'gnuworld': 1, 'hdsf': 1, 'home': 1, 'html': 1, 'informix': 1, 'ircd': 1,
'kernel': 1, 'linux': 1, 'linuxmint': 1, 'mdb': 1, 'mysql': 1, 'mythtv': 1,
'ns2': 1, 'ns': 1, 'office': 1, 'operator': 1, 'oracle': 2, 'otrs': 1, 'pc': 1,
'php': 1, 'portal': 1, 'postgres': 2}, '221.148.234.252': {'admin': 1},
'5.255.68.179': {'miner': 1, 'user': 1, 'm1': 1, 'ethos': 2}, '80.237.75.2':
{'ubnt': 4, 'test': 4, 'administrator': 4, 'admin': 4, 'support': 4, 'ftpuser':
4, 'operator': 4, 'pi': 4, 'service': 4, 'r': 4, '1': 4}}
```

```
FAILED LOGIN COUNTS: {'112.112.102.38': {'admin': 6}, '137.74.1.62': {'RPM': 15,
'openvpn': 15, 'm202': 15, 'redmine': 15, 'leo': 15, 'odoo': 15, 'apc': 15,
'user': 15, 'admin': 15}, '178.68.209.126': {'admin': 6}, '185.182.56.85':
{'ubnt': 22, 'r00t': 22, 'test': 22, 'administrator': 22, 'admin': 22,
'support': 22, 'nagios': 22, 'Admin': 22, 'ftpuser': 22, 'operator': 22, '1':
22}, '192.129.227.186': {'user': 2, 'miner': 1, 'ethos': 1, 'oracle': 1},
'212.237.47.236': {'antivirus': 1, 'apache': 1, 'backuppc': 1, 'bank': 1,
'board': 1, 'build': 1, 'byte': 1, 'bytes': 1, 'centos': 1, 'chat': 1, 'cinema':
1, 'control': 1, 'counterstrike': 1, 'cpanel': 1, 'cs': 1, 'csgo': 1, 'cvs': 1,
'cyber': 1, 'cyrus': 1, 'data': 1, 'db1': 1, 'db2inst1': 1, 'debian': 2,
'developer': 1, 'downloads': 1, 'dvd': 1, 'dvs': 1, 'edu': 1, 'education': 1,
'fedora': 1, 'forum': 1, 'ftp': 1, 'ftpadmin': 1, 'gaming': 1, 'gb': 1, 'ghani':
1, 'ghost': 1, 'git': 7, 'gnu': 1, 'gnuworld': 1, 'hdsf': 1, 'home': 1, 'html':
1, 'informix': 1, 'ircd': 1, 'kernel': 1, 'linux': 1, 'linuxmint': 1, 'mdb': 1,
'mysql': 1, 'mythtv': 1, 'ns2': 1, 'ns': 1, 'office': 1, 'operator': 1,
'oracle': 2, 'otrs': 1, 'pc': 1, 'php': 1, 'portal': 1, 'postgres': 2},
'221.148.234.252': {'admin': 6}, '5.255.68.179': {'miner': 1, 'user': 1, 'm1':
1, 'ethos': 2}, '80.237.75.2': {'ubnt': 24, 'test': 24, 'administrator': 24,
'admin': 24, 'support': 24, 'ftpuser': 24, 'operator': 24, 'pi': 24, 'service':
24, 'r': 24, '1': 24}}
```

1.2 Problem 2

Analyze the following code that reads the `apache_logs.txt` file. Determine what it does. Write your response as code comments.

[ ]:
```python
import sys #imports sys and os modules to provide access to some variables used␣
 ↪in these functions
import os
#function that reads given file and IP address, and then write results into␣
 ↪another file
def readfile(f):
    openfile = open(f,"r") #opens file in read mode
#opens files for unique IPs, all IPs, and IPs with urls associated
    unique_outfile = open("uniqueIP.txt","w")
    all_outfile = open("allIP.txt","w")
    ipAndUrl_outfile = open("ipAndUrl.txt","w")
#sets up list, dictionary, and set to store data
    lines = []
    ipAndUrl = {}
    ip_list = set()
#reads each line and appends it to the "lines" list that was established
    for line in openfile:
        lines.append(line.strip('\n'))#appends and adds new lines in between␣
 ↪data
# splits lines by spaces on the first item in the string. Assuming it's the IP␣
 ↪address as the first item
```

```
    for line in lines:
        ip = line.split(" ")[0]
# append lines to dictionary using ip as key and the associated value,
        if ip in ipAndUrl:
            ipAndUrl[ip].append(line.split(" ")[6])
        else:
            ipAndUrl[ip] = [line.split(" ")[6]]
# writes ip info to this file with spaces for readability
        all_outfile.write(ip)
        all_outfile.write("\n")
#adds ip to the set
        ip_list.add(ip)
# write unique IPs with spaces to this unique outfile
    for ip in ip_list:
        unique_outfile.write(ip)
        unique_outfile.write("\n")
#writes ip and value(urls) to the ipandurl file
    for key, value in ipAndUrl.items():
        ipAndUrl_outfile.write('%s  %s\n' % (key, value))
#closes files
    unique_outfile.close()
    ipAndUrl_outfile.close()
    all_outfile.close()
    openfile.close()
#main function to initiate user input for the file name and calls the readfile␣
 ↪function established above
def Main():
    file=input("please enter a file name: ")

    result = readfile(file)
# checks script if is being run as the main program
if __name__ == '__main__':

    Main()
Main()# run the main function
```

[ ]: