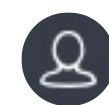




Piotr Kojalowicz

Wrocław, 06.02.2018

Traffic report



Personal Info

Adres

Zachodnia 20/67
53-644 Wrocław

Telefon

889 29 59 75

E-mail

piotr.kojalowicz@gmail.com

1. For the first task I decided to solve the problem of finding the http address in a single line.

```
def cutter(line=str, line_no=int):          ### > the function takes: line and its number <
    http_index_to=find(line,'HTTP/')->1     ### > searched for the end of address <
    http_index_from=find(line,'http:')+7     ### > searched for the beginning of address <
    find_question_mark=find(line[0:http_index_to], '?') ### > checks if they are question
    if find_question_mark > -1:               ### send by http address if they are
        http_index_to=find_question_mark     ### deletes them <
    if line[http_index_to-1] == '/':
        http_index_to-=1
    if http_index_to>-2 and http_index_from>6:
        return line[http_index_from:http_index_to] ### > if http is correct add them to list
    else: ### if not send information "Invalid..." <
        return 'Invalid log lines: %s' %(line_no)
```

2. Next step was to create a list of all addresses.

```
def only_http(log_file):                    ### > the function takes log file
    list_http = []
    line_no = 1
    file=open(log_file,'r')
    for line in file:
        list_http.append(cutter(line, line_no)) ### > adds more records (address)
        line_no+=1 ### and possibly errors along
    file.close() ### with the line number <
    return list_http ### > returns a list of all addresses <
```

3. The last defined function in the program checks how many unique addresses are and gives the number of their repetitions.

```
def log_out(list_http):                    ### > the function takes list of http <
    http_dict = dict()
    for i in list_http:
        key=i
        if "Invalid" in key: ### > checks if the line has an error if it gives 0 <
            http_dict[key]= 0
        elif key in http_dict: ### > checks if the address is already on the list
            http_dict[key] += 1 ### if so increases the value by 1
        else: ### if it does not give the value 1 <
            http_dict[key] = 1
    return http_dict
```

4. In main function the result will be displayed with compliance conditions specified in the task.

```
if __name__ == "__main__":
    log = str(sys.argv[1])
    http_dict=log_out(only_http(log))
    for i in [k for k, v in sorted(http_dict.iteritems(), key=lambda(k, v): (-v, k))]:
        if "Invalid" in i:
            print ("%s" %i)
        else:
            print("%s",%s' %(i,http_dict[i]))
    sys.exit()
```

CONCLUSIONS:

- "+":
- the level of difficulty required from me to look for some solutions,
 - simple and functional code,
 - fast and efficient script work (checked for logos with more than 1,000,000 records)
- "-"
- the new challenge took more time than I expected
 - does not have error handling - my knowledge is little in this subject