Compte rendu du SAÉ 15

RICHIER Erwann RT1G2

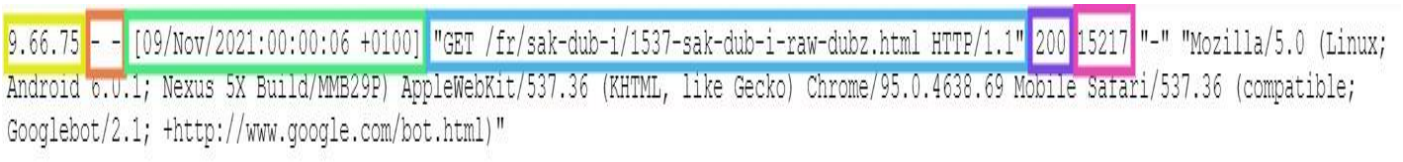
iut réseaux & télécoms Mont de Marsan

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

SOMMAIRE

I°/ Explication du fichier log Apache

Le fichier log Apache se constitue comme suit :



- **9.66.75**
  Il s'agit de l'adresse IP du client qui a envoyé la requête au serveur.

- **-**
  Le tiret dans la sortie indique que l'information demandée n'est pas disponible. Dans ce cas, l'information qui n'est pas disponible est l'identité de la machine client.

- **-**
  Il s'agit cette fois du nom d'utilisateur qui demande le document.

- **[09/Nov/2021:00:00:06 +0100]**
  L'heure à laquelle la demande a été reçue. Le format est le suivant : [jour/mois/année:heure:minute:seconde timezone]

- **"GET /fr/sak-dub-i/1537-sak-dub-i-raw-dubz.html HTTP/1.1"**
  La ligne de requête du client est indiquée entre guillemets. La ligne de requête contient un grand nombre d'informations utiles. Premièrement, la méthode utilisée par le client est GET. Deuxièmement, le client a demandé la ressource /fr/sak-dub-i/1537-sak-dub-i-raw-dubz.html, et troisièmement, le client a utilisé le protocole HTTP/1.1.

- **200**
  Il s'agit du code d'état que le serveur renvoie au client. Cette information est très précieuse, car elle révèle si la requête a donné lieu à une réponse positive (2xx), à une redirection (3xx), à une client (4xx) ou à une erreur serveur (5xx). La plus connue étant 404.

- **15127**
  Cette partie indique la taille du paquet renvoyé au client.Le reste n'étant que des informations relatives au client comme son navigateur ou son OS.

II°/ Explication du code Python

A°/ Structure de la méthode ***parsing(file)***

```python
# Pairing each regex code to its data type$
import re #regex library
regex = '([(\d\.)]+) - - \[(.*?)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"'
#format : IP, - -, Date, Request Type, Status Code, Packet Length, -,Browser
def parsing(file):
    ans = []
    f = open(file, 'r')
    for line in f.readlines():
        ans.append(re.match(regex, line).groups())
    return ans
```

Dans cette fonction j'ai utilisé le regex, un format de texte qui détecte les expressions régulières afin de faciliter les recherches d'informations dans des quantités importantes de données.
Ici, grâce à une boucle for, je fait correspondre le regex aux lignes du fichier log ce qui me permet donc de délimiter chaque requête et rendre plus lisible les données car elles sont maintenant des éléments d'une liste de tuples nommée "ans" pour answer.

L'ouput ressemble donc à cela :

[('9.66.75','-','-','09/Nov/2021:00:00:06 +0100', 'GET /fr/sak-dub-i/1537-sak-dub-i-raw-dubz.html HTTP/1.1', '200', '15217', '-', 'Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Mobile Safari/537.36 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)')]

B°/ Description des fonctions demandées

La première fonction demandée est "***getIP_infos(ip)***" :

```python
# Getting IPs data with json format
def getIP_infos(ip):
    response = requests.get('https://api.freegeoip.app/json/{}?apikey=338b0fe0-5144-11ec-9435-15403feef841'.format(ip))
    # api key freegeoip.app '338b0fe0-5144-11ec-9435-15403feef841' pour faire 15000 requets par heures
    result = response.content.decode()
    result = json.loads(result)
    return result
```

J'ai utilisé le format json délivré par l'API "freegeoip" de façon à extraire les informations telles que le nom du pays où se trouve l'adresse IP et d'autres informations qui ne m'intéressent pas. Tout cela se trouve donc dans une variable "result".

J'ai utilisé cet API (qui fonctionne exactement de la même manière que "ip-api") car le nombre de requêtes était bien trop insuffisant pour pouvoir analyser un fichier aussi gros que celui fourni. Cependant en y réfléchissant il aurait simplement suffit de mettre une action sur le temps telle que ***time.sleep(2)*** mais je doute que ça aurait été assez rapide.

Pour la deuxième fonction demandée ("***exportToCSVFile(liste)***") :

```
# Creating a CSV file with the log
def exportToCSVFile(log_parsing):
    CSVFile = open("CSVFile.csv", "w")
    writer = csv.writer(CSVFile)
    writer.writerows(log_parsing)
    CSVFile.close()
```

J'ai la liste "log-parsing", provenant d'une fonction annexe, en paramètre de la fonction. Cette dernière me permet de créer un fichier au format CSV avec les données de la liste.

L'output ressemble à ceci :

A	B	C	D	E
9.66.75	09/Nov/2021:00:00:06 +0100	GET /fr/sak-dub-i/1537-sak-dub-i-raw-dubz.html HTTP/1.1	200	15217

Pour la troisième fonction demandée ("***exportToJSONFile(liste)***") :

```
# Exporting into JSON file
def exportToJSONFile(list_ip_log_info):
    writer = open("out.json", "w")
    writer.write(json.dumps(list_ip_log_info, indent=4, separators=(";", ":")))
    writer.close()
```

J'ai la même liste "log-parsing" que dans la fonction précédente en paramètre de la fonction. Cette dernière me permet de créer un fichier au format JSON avec les données de la liste. J'ai utilisé la méthode json.dumps(liste,indentation,séparateurs) afin de faciliter la création du fichier JSON.

Voici l'output :

Pour la quatrième et dernière fonction demandée ("***exportToXMLFile(liste)***"):

```python
# Exporting into XML file
def export_xml(list_ip_log_info):
    # create root and sub element
    usrconfig = ET.Element("usrconfig")
    usrconfig = ET.SubElement(usrconfig, "usrconfig")
    # insert list element into sub elements
    for user in range(len(list_ip_log_info)):
        usr = ET.SubElement(usrconfig, "usr")
        usr.text = str(list_ip_log_info[user])
    tree = ET.ElementTree(usrconfig)
    # write the file
    tree.write("out.xml", encoding='utf-8', xml_declaration=True)
```

J'ai la même liste "log-parsing" que dans la fonction précédente en paramètre de la fonction. Cette dernière me permet de créer un fichier au format XML avec les données de la liste.

Voici l'output :

```xml
<?xml version="1.0" encoding="utf-8"?>
<usrconfig>
    <usr>
        ('66.249.66.75', '09/Nov/2021:00:00:06 +0100', 'GET
    </usr>
</usrconfig>
```

NB : Le format JSON est le meilleur format pour faire des statistiques selon moi car les champs qu'on recherchent ont des noms bien précis, il est donc simple de récupérer ces champs dans un programme afin d'obtenir l'info recherchée.

III°/ Utilisation de l'API

A°/ Comment il fonctionne

freegeoip.app ou *ip-api.com* fonctionne de manière parfaitement similaire. Il suffit de rentrer une ip et l'API nous donne en retour sa localisation et d'autres informations au format JSON.

Voici un exemple avec le DNS de l'IUT :

B°/ Comment je l’ai utilisé

Afin d’obtenir un nombre de requêtes par heure pouvant supporter la pression qu’impose le programme avec une recherche de 4700 ip différentes, je me suis inscrit pour obtenir une clef API.
Je fais donc une requête à l’API via le module ***requests*** et j’insère via la méthode ***.format()*** chaque ip correspondant à l’index 0 de la liste “*list_ip_log_info*” contenant toutes les informations sur chaque ip.

```
response = requests.get(
    'https://api.freegeoip.app/json/{}?apikey=045324a0-5747-11ec-819a-43a1b701a3ac'.format(i[0]))
```

IV°/ Statistiques du fichier log

A°/ Comment le module se manipule

Le module qui permet de faire des statistiques s'appelle ***matplotlib***, il faut donc l'importer. Il faut également importer ***numpy*** afin de manipuler les chiffres.

```
import numpy as np
import matplotlib.pyplot as plt
```

J'ai utilisé ce module dans le but de créer un histogramme du nombre de connexions par pays via la fonction "***graphic_creation(dictionnaire)***" :

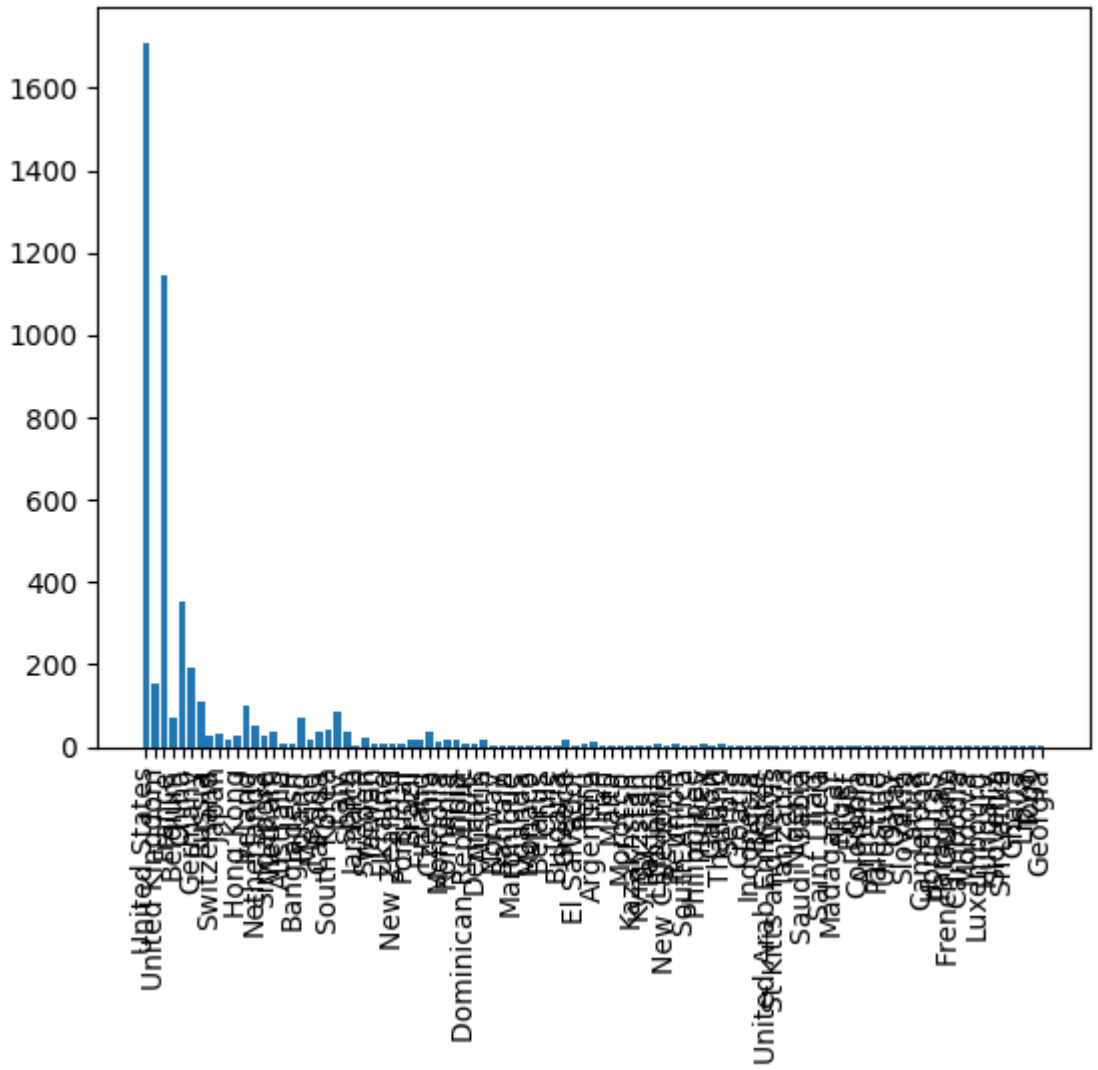
```
# Creating graph of country connexions
def graphic_creation(country_info):
    print(country_info)
    height = []
    bars = []
    for key, value in country_info.items():
        if key != 'total':
            height.append(value)
            bars.append(key)

    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars, rotation = 90)
    plt.savefig('graph.png', bbox_inches='tight')
    plt.title(label="Country graph",fontsize=30,color="red")
    plt.show()
```

Pour créer cet histogramme j'ai d'abord défini les valeurs des axes Ox et Oy, ici respectivement "*bars*" et "height", tel que chaque barre correspond à un pays (donc les clefs du dictionnaire) et sa valeur soit le nombre d'ip dans ce pays (donc les valeurs des clefs du dictionnaire).

Maintenant que chaque “item” du graphique est défini, on peut alors le tracer. La variable “*y_pos*” correspond à la hauteur de la barre.
La méthode plt.bar(x,y) permet de tracer chaque barre.
Pour ce qui est de plt.xtricks(y,x) ça permet d’écrire le nom du pays à la verticale sous la barre.
La méthode plt.title() permet, comme son nom l’indique, de donner un titre au graphique.
Et pour finir la méthode plt.show() affiche le graphique une fois finis.

B°/ Finalité et présentation des statistiques

La méthode plt.save() permet de sauvegarder le fichier du graphique.



Voilà ce que donne le graphique avec près de 4700 ip différentes et un total de plus de 150 pays.

V°/ Cartographie

A°/ Explication de son fonctionnement

Afin de rendre la lecture de la provenance des ip, il nous a été demandé de mettre ses ip sur une carte de type OpenStreetMap.

Pour cela j'ai utilisé le module ***folium*** afin de créer une carte avec ces ip assez facilement. J'ai donc créer la map grâce à la fonction "***position_on_map***" :

```
# Creating marks on map
map = folium.Map(location=[48.85, 2.34], zoom_start=3)  # creating map with start point at Paris
def position_on_map(list_ip_log_info):
    print("-- Debut positionnement sur la carte --")
    for i in list_ip_log_info:
        folium.Marker([i["latitude"], i["longitude"]], popup="<i>{}</i>".format(i), tooltip=i['ip']).add_to(map)
    map.save("index.html")
    print("-- Toute les ip on était placé sur la carte disponible dans le 'index.html' --")
```

Cette fonction a pour entrée la même liste qui me sert pour tout et en sort une map avec toutes les ip. Grâce à la méthode *Marker()* de folium, qui crée des balises sur la map avec la latitude et la longitude du json de chaque ip, ainsi que la méthode *add_to(map)*, qui permet de mettre ces balises sur la map, j'ai pu obtenir une carte du monde avec toutes les ip.

B°/ Rendu final

Le fichier index.html que me sort la fonction ressemble à cela :

```
marker_bb4d117bc03a45d2ab479965f43284ce.bindTooltip(
    `<div>
         66.249.66.75
     </div>`,
    {"sticky": true}
);


var marker_cf9ed866ff134bfb866f83def948ec24 = L.marker(
    [51.5095, -0.0955],
    {}
).addTo(map_024296faa43744e18657edca7ceefc2f);
```

Ce qui donne cela sur la map :