RAPPORT SAE15 - Andréa HAVARD et Elouann LEGUEN - Groupe B2

Introduction

Voici notre rapport de SAE15, dans celui-ci vous retrouverez nos différents script bash et leurs csv réspectifs, ainsi que les fonctions python que nous avons du réaliser pour avoir différents statiques, et enfin 3 diagrammes utilisant nos données et quelques unes de nos statistiques. Tout les tests de fonctionnement que ce soit pour les script bash ou les fonctions python sont présents, nous vous fournissons également des captures d'écran des 3 diagrammes en cas de problème sur votre machine. Nous tenons également à préciser que nous avons réaliser l'ensemble du projet à deux.

Partie I - Les Scripts

Récupération de donnée

In [ ]:
```
#!/bin/bash
#description : recupére la durée de vie d'un paquet envoyés à google et OpenDNS
#ELOUANN & ANDREA
#Exécution : ./recup_donnee.sh avec chmod u+x recup_donnee.sh
serveur1='8.8.8.8'
serveur2='208.67.222.222'
for i in $serveur1 $serveur2
do
trace=$(traceroute $i -I | tail -1)
time=$(echo $trace | cut -d " " -f 4)
nb_saut=$(echo $trace | cut -d " " -f 1)
dns=$(echo $trace | cut -d " " -f 2)
echo $dns, $nb_saut, $time >> /root/Documents/SAE15/tmp/donnee.csv
done
```

```
  Cell In[29], line 7
    for i in $serveur1 $serveur2
             ^
SyntaxError: invalid syntax
```

Après execution du script, nous avons remarqué que notre fichier donnee.csv n'était pas exploitable. En effet, le dns et le nombre de saut ne change jamais, il est donc impossible de faire des graphiques avec ce genre de donnée. Nous avons donc réalisé un csv fictif, dans celui-ci le nombre de saut change a chaque fois, ce qui rend nos données beaucoup plus exploitables

Test dans un terminal :

```
root@debian11:~# serveur1='8.8.8.8'
serveur2='208.67.222.222'
for i in $serveur1 $serveur2
do
trace=$(traceroute $i -I | tail -1)
time=$(echo $trace | cut -d " " -f 4)
nb_saut=$(echo $trace | cut -d " " -f 1)
dns=$(echo $trace | cut -d " " -f 2)
echo $dns, $nb_saut, $time
> done
dns.google, 8, 16.507
dns.opendns.com, 6, 17.714
root@debian11:~#
```

CSV (Réel)

In [ ]:
```
dns.google, 8, 15.957
dns.opendns.com, 6, 17.462
dns.google, 8, 15.873
dns.opendns.com, 6, 17.280
dns.google, 8, 15.966
dns.opendns.com, 6, 17.221
dns.google, 8, 16.056
dns.opendns.com, 6, 17.638
dns.google, 8, 18.170
dns.opendns.com, 6, 17.174
dns.google, 8, 16.439
dns.opendns.com, 6, 17.737
dns.google, 8, 16.277
dns.opendns.com, 6, 17.540
dns.google, 8, 16.269
dns.opendns.com, 6, 17.319
dns.google, 8, 16.104
dns.opendns.com, 6, 17.628
dns.google, 8, 15.953
dns.opendns.com, 6, 17.711
dns.google, 8, 18.173
dns.opendns.com, 6, 17.475
```

CSV (Fictif)

In [ ]:
```
dns.google, 8, 15.957
dns.opendns.com, 6, 17.462
dns.google, 10, 15.873
dns.opendns.com, 4, 17.280
dns.google, 2, 15.966
dns.opendns.com, 2, 17.221
dns.google, 6, 16.056
dns.opendns.com, 6, 17.638
dns.google, 8, 18.170
dns.opendns.com, 8, 17.174
dns.google, 3, 16.439
dns.opendns.com, 10, 17.737
```

```
dns.google, 12, 16.277
dns.opendns.com, 3, 17.540
dns.google, 20, 16.269
dns.opendns.com, 9, 17.319
dns.google, 8, 16.104
dns.opendns.com, 7, 17.628
dns.google, 1, 15.953
dns.opendns.com, 1, 17.711
dns.google, 5, 18.173
dns.opendns.com, 8, 17.475
dns.google, 3, 17.637
dns.opendns.com, 6, 17.050
```

Mise a jour des données

In [ ]:
```
#!/bin/bash
#description: mise à jour des données dans la commande ip route
#ELOUANN & ANDREA
#Exécution : ./maj_donnee avec chmod u+x maj_donnee.sh
min=1
max=30
val=$(shuf -i $min-$max -n 1)
ip addr flush docker0
ip addr add 172.17.0.$val/16 dev docker0
interface=$(ip route | tail -1 | cut -d " " -f 3)
ip=$(ip route | tail -1 | cut -d " " -f 9)
echo $interface, $ip >> /root/Documents/SAE15/tmp/maj.cs
```

Nous avons réalisé ce script pour la commande ip route car après discussion avec les professeurs, nous avons remarqué que la mise a jour de donnée avec notre commande traceroute allait être compliquée. Nous avons donc réalisé le script comme montré ci-dessus et mis nos données dans un nouveau fichier csv. Celui-ci ne seras pas utile mais permet de montrer le bon changement de l'adresse ip toutes les minutes.

Test dans un terminal :

```
root@debian11:~# min=1
max=30
val=$(shuf -i $min-$max -n 1)
ip addr flush docker0
ip addr add 172.17.0.$val/16 dev docker0
interface=$(ip route | tail -1 | cut -d " " -f 3)
ip=$(ip route | tail -1 | cut -d " " -f 9)
echo $interface, $ip
docker0, 172.17.0.12
root@debian11:~#
```

CSV

In [ ]:
```
docker0, 172.17.0.26
docker0, 172.17.0.26
docker0, 172.17.0.7
```

```
docker0, 172.17.0.25
docker0, 172.17.0.23
docker0, 172.17.0.25
docker0, 172.17.0.27
docker0, 172.17.0.9
docker0, 172.17.0.10
docker0, 172.17.0.21
docker0, 172.17.0.15
docker0, 172.17.0.17
docker0, 172.17.0.25
docker0, 172.17.0.9
docker0, 172.17.0.4
docker0, 172.17.0.9
docker0, 172.17.0.7
docker0, 172.17.0.20
docker0, 172.17.0.10
docker0, 172.17.0.22
docker0, 172.17.0.7
```

Partie II - Crontab

Pour modifier le crontab on tappe : crontab -e

```
  GNU nano 5.4                /tmp/crontab.SC8fup/crontab
* * * * * /root/Documents/SAE15/recup_donnee.sh >> /tmp/logfile 2>&1
* * * * * /root/Documents/SAE15/maj_donnee.sh >> /tmp/logfile 2>&1
```

Cette commande nous ouvres le fichier crontab avec nano, ce qui nous permet d'ajouter nos deux lignes

In [ ]:
```
* * * * * /root/Documents/SAE15/recup_donnee.sh >> /tmp/logfile 2>&1
* * * * * /root/Documents/SAE15/maj_donnee.sh >> /tmp/logfile 2>&1
```

Pour visualiser nos modifications on tappe : crontab - l

```
root@debian11:~# crontab -l
* * * * * /root/Documents/SAE15/recup_donnee.sh >> /tmp/logfile 2>&1
* * * * * /root/Documents/SAE15/maj_donnee.sh >> /tmp/logfile 2>&1
```

Partie III - Les fonctions Python

In [ ]:
```
import csv
from math import sqrt
from typing import List
from datetime import datetime
import matplotlib.pyplot as plt
```

Lecture du fichier donnee_fictif.csv

In [ ]:
```python
def lecture_fichier(fichier : str) -> list[list[str]]:
    """Lit un fichier csv et le renvoie sous forme de tableau

    Args:
        fichier (str): fichier csv en question

    Returns:
        list[list[int]]: La liste générée par la fonction du fichier cvs
    """
    listeR : list[list[int]] = []
    with open(fichier, newline='') as csvfile :
        datareader = csv.reader(csvfile, delimiter= ',', quotechar='|')
        for ligne in datareader :
            # listeR.append(ligne)
            listeR.append([
                ligne[1],
                ligne[2],
                ligne[0]
            ])
    return listeR

liste=lecture_fichier("donnee_fictif.csv")
print(liste)
```

```
[[' 8', ' 15.957', 'dns.google'], [' 6', ' 17.462', 'dns.opendns.com'], [' 10', '
15.873', 'dns.google'], [' 4', ' 17.280', 'dns.opendns.com'], [' 2', ' 15.966',
'dns.google'], [' 2', ' 17.221', 'dns.opendns.com'], [' 6', ' 16.056', 'dns.googl
e'], [' 6', ' 17.638', 'dns.opendns.com'], [' 8', ' 18.170', 'dns.google'], ['
8', ' 17.174', 'dns.opendns.com'], [' 3', ' 16.439', 'dns.google'], [' 10', ' 17.
737', 'dns.opendns.com'], [' 12', ' 16.277', 'dns.google'], [' 3', ' 17.540', 'dn
s.opendns.com'], [' 20', ' 16.269', 'dns.google'], [' 9', ' 17.319', 'dns.opendn
s.com'], [' 8', ' 16.104', 'dns.google'], [' 7', ' 17.628', 'dns.opendns.com'],
[' 1', ' 15.953', 'dns.google'], [' 1', ' 17.711', 'dns.opendns.com'], [' 5', ' 1
8.173', 'dns.google'], [' 8', ' 17.475', 'dns.opendns.com'], [' 3', ' 17.637', 'd
ns.google'], [' 6', ' 17.050', 'dns.opendns.com'], [' 8', ' 19.723', 'dns.googl
e'], [' 9', ' 17.564', 'dns.opendns.com'], [' 6', ' 16.611', 'dns.google'], ['
2', ' 17.137', 'dns.opendns.com'], [' 11', ' 16.753', 'dns.google'], [' 7', ' 17.
283', 'dns.opendns.com'], [' 7', ' 16.754', 'dns.google'], [' 11', ' 18.352', 'dn
s.opendns.com'], [' 2', ' 16.948', 'dns.google'], [' 9', ' 18.245', 'dns.opendns.
com'], [' 9', ' 17.558', 'dns.google'], [' 6', ' 17.147', 'dns.opendns.com'], ['
13', ' 17.661', 'dns.google'], [' 8', ' 17.270', 'dns.opendns.com'], [' 8', ' 16.
887', 'dns.google'], [' 3', ' 17.185', 'dns.opendns.com'], [' 6', ' 16.676', 'dn
s.google'], [' 12', ' 18.188', 'dns.opendns.com'], [' 4', ' 16.403', 'dns.googl
e'], [' 2', ' 17.214', 'dns.opendns.com'], [' 14', ' 17.138', 'dns.google'], ['
5', ' 18.853', 'dns.opendns.com'], [' 9', ' 16.894', 'dns.google'], [' 4', ' 17.8
78', 'dns.opendns.com'], [' 8', ' 16.852', 'dns.google'], [' 6', ' 17.253', 'dns.
opendns.com']]
```

Test :

- [[' 8', ' 15.957', 'dns.google'], [' 6', ' 17.462', 'dns.opendns.com'], [' 10', ' 15.873', 'dns.google'], [' 4', ' 17.280', 'dns.opendns.com'], [' 2', ' 15.966', 'dns.google'], [' 2', ' 17.221', 'dns.opendns.com'], [' 6', ' 16.056', 'dns.google'], [' 6', ' 17.638', 'dns.opendns.com'], [' 8', ' 18.170', 'dns.google'], [' 8', ' 17.174', 'dns.opendns.com'], [' 3', ' 16.439', 'dns.google'], [' 10', ' 17.737', 'dns.opendns.com'], [' 12', ' 16.277', 'dns.google'], [' 3', ' 17.540', 'dns.opendns.com'], [' 20', ' 16.269', 'dns.google'], [' 9', ' 17.319', 'dns.opendns.com'], [' 8', ' 16.104', 'dns.google'], [' 7', ' 17.628',

'dns.opendns.com'], [' 1', ' 15.953', 'dns.google'], [' 1', ' 17.711', 'dns.opendns.com'], [' 5', ' 18.173', 'dns.google'], [' 8', ' 17.475', 'dns.opendns.com'], [' 3', ' 17.637', 'dns.google'], [' 6', ' 17.050', 'dns.opendns.com'], [' 8', ' 19.723', 'dns.google'], [' 9', ' 17.564', 'dns.opendns.com'], [' 6', ' 16.611', 'dns.google'], [' 2', ' 17.137', 'dns.opendns.com'], [' 11', ' 16.753', 'dns.google'], [' 7', ' 17.283', 'dns.opendns.com'], [' 7', ' 16.754', 'dns.google'], [' 11', ' 18.352', 'dns.opendns.com'], [' 2', ' 16.948', 'dns.google'], [' 9', ' 18.245', 'dns.opendns.com'], [' 9', ' 17.558', 'dns.google'], [' 6', ' 17.147', 'dns.opendns.com'], [' 13', ' 17.661', 'dns.google'], [' 8', ' 17.270', 'dns.opendns.com'], [' 8', ' 16.887', 'dns.google'], [' 3', ' 17.185', 'dns.opendns.com'], [' 6', ' 16.676', 'dns.google'], [' 12', ' 18.188', 'dns.opendns.com'], [' 4', ' 16.403', 'dns.google'], [' 2', ' 17.214', 'dns.opendns.com'], [' 14', ' 17.138', 'dns.google'], [' 5', ' 18.853', 'dns.opendns.com'], [' 9', ' 16.894', 'dns.google'], [' 4', ' 17.878', 'dns.opendns.com'], [' 8', ' 16.852', 'dns.google'], [' 6', ' 17.253', 'dns.opendns.com']]

Transposement du fichier sous forme d'une liste

In [ ]:

```python
def transpose(liste) -> tuple[list[int], list[float]]:
    """Mise en forme de la liste pour faire une liste des sauts et liste des tem

    Args:
        liste (list): la liste en question

    Returns:
        tuple[list[int], list[float]] : liste des sauts et liste des temps
    """
    liste_google_sauts,liste_google_temps = [],[]
    liste_opendns_sauts,liste_opendns_temps = [],[]
    liste_global_sauts,liste_global_temps=[],[]
    retour=[]
    for el in liste:
        if len(el) == 2 :
            liste_global_sauts.append(int(el[0]))
            liste_global_temps.append(float(el[1]))
        else :
            if el[2] == "dns.google":
                liste_google_sauts.append(int(el[0]))
                liste_google_temps.append(float(el[1]))
            elif el[2] == "dns.opendns.com":
                liste_opendns_sauts.append(int(el[0]))
                liste_opendns_temps.append(float(el[1]))

    if len(liste_global_sauts) > 0:
        retour = [liste_global_sauts, liste_global_temps]
    else :
        retour = [liste_google_sauts,liste_google_temps,liste_opendns_sauts,list

    return retour

liste2=transpose(liste)
print(liste2)
```

```
[[8, 10, 2, 6, 8, 3, 12, 20, 8, 1, 5, 3, 8, 6, 11, 7, 2, 9, 13, 8, 6, 4, 14, 9,
8], [15.957, 15.873, 15.966, 16.056, 18.17, 16.439, 16.277, 16.269, 16.104, 15.95
3, 18.173, 17.637, 19.723, 16.611, 16.753, 16.754, 16.948, 17.558, 17.661, 16.88
7, 16.676, 16.403, 17.138, 16.894, 16.852], [6, 4, 2, 6, 8, 10, 3, 9, 7, 1, 8, 6,
9, 2, 7, 11, 9, 6, 8, 3, 12, 2, 5, 4, 6], [17.462, 17.28, 17.221, 17.638, 17.174,
17.737, 17.54, 17.319, 17.628, 17.711, 17.475, 17.05, 17.564, 17.137, 17.283, 18.
352, 18.245, 17.147, 17.27, 17.185, 18.188, 17.214, 18.853, 17.878, 17.253]]
```

Test :

- [[8, 10, 2, 6, 8, 3, 12, 20, 8, 1, 5, 3, 8, 6, 11, 7, 2, 9, 13, 8, 6, 4, 14, 9, 8], [15.957, 15.873, 15.966, 16.056, 18.17, 16.439, 16.277, 16.269, 16.104, 15.953, 18.173, 17.637, 19.723, 16.611, 16.753, 16.754, 16.948, 17.558, 17.661, 16.887, 16.676, 16.403, 17.138, 16.894, 16.852], [6, 4, 2, 6, 8, 10, 3, 9, 7, 1, 8, 6, 9, 2, 7, 11, 9, 6, 8, 3, 12, 2, 5, 4, 6], [17.462, 17.28, 17.221, 17.638, 17.174, 17.737, 17.54, 17.319, 17.628, 17.711, 17.475, 17.05, 17.564, 17.137, 17.283, 18.352, 18.245, 17.147, 17.27, 17.185, 18.188, 17.214, 18.853, 17.878, 17.253]]

Calcul de la valeur minimum

In [ ]:
```python
def calculValMini(listeParam : list[int]) -> int:
    """fonction qui renvoie le minimum d'une liste

    Args:
        listeParam (list[int]): liste pour laquelle on veut le minimum

    Returns:
        int: minimum de la liste
    """
    mini : int = listeParam[0]

    for elt in listeParam:
        if elt < mini:
            mini = elt

    return mini

#Variable avec un appel de la fonction pour avoir la valeur minimum des sauts et
google_sauts_min=calculValMini(liste2[0])
google_temps_min=calculValMini(liste2[1])
opendns_sauts_min=calculValMini(liste2[2])
opendns_temps_min=calculValMini(liste2[3])

#On range ces variables dans une liste
listMini:list=[[google_sauts_min, google_temps_min], [opendns_sauts_min, opendns

#print de cette liste
print(listMini)
```

```
[[1, 15.873], [1, 17.05]]
```

Test :

- [[1, 15.873], [1, 17.05]]

Calcul de la valeur maximum

In [ ]:
```python
def calculValMaxi(listeParam : list[int]) -> int:
    """fonction qui renvoie le maximum d'une liste

    Args:
        listeParam (list[int]): liste pour laquelle on veut le maximum

    Returns:
        int: maximum de la liste
    """
    maxi : int = listeParam[0]

    for elt in listeParam:
        if elt > maxi:
            maxi = elt

    return maxi

#Variable avec un appel de la fonction pour avoir la valeur maximum des sauts et
google_sauts_max=calculValMaxi(liste2[0])
google_temps_max=calculValMaxi(liste2[1])
opendns_sauts_max=calculValMaxi(liste2[2])
opendns_temps_max=calculValMaxi(liste2[3])

#On range ces variables dans une liste
listMaxi:list=[[google_sauts_max, google_temps_max], [opendns_sauts_max, opendns

#print de cette liste
print(listMaxi)
```

```
[[20, 19.723], [12, 18.853]]
```

Test :

- [[20, 19.723], [12, 18.853]]

Calcul de la valeur moyenne

In [ ]:
```python
def calculValMoy(listeParam : list[int]) -> int:
    """fonction qui renvoie la moyenne d'une liste

    Args:
        listeParam (list[int]): liste pour laquelle on veut la moyenne

    Returns:
        int: moyenne de la liste
    """
    diviseur : int = len(listeParam)
    dividende: int = 0
    for elt in listeParam:
        dividende += elt
    moy : float = (dividende/diviseur)
    return moy

#Variable avec un appel de la fonction pour avoir la valeur moyenne des sauts et
google_sauts_moy=calculValMoy(liste2[0])
google_temps_moy=calculValMoy(liste2[1])
opendns_sauts_moy=calculValMoy(liste2[2])
opendns_temps_moy=calculValMoy(liste2[3])
```

```
#On range ces variables dans une liste
listMoy:list=[google_sauts_moy, google_temps_moy, opendns_sauts_moy, opendns_tem

#print de cette liste
print(listMoy)
```

```
[7.64, 16.869279999999996, 6.16, 17.552159999999997]
```

Test :

- [7.64, 16.869279999999996, 6.16, 17.552159999999997]

Calcul de la valeur médiane

In [ ]:
```
def calculValMed(listeParam : list[int]) -> float:
    """fonction qui renvoie la médiane d'une liste

    Args:
        listeParam (list[int]): liste pour laquelle on veut la médiane

    Returns:
        float: médiane de la liste
    """
    listeT : list[int] = sorted(listeParam)
    taille : int = len(listeT)
    milieu : int = taille // 2
    retour : float = 0

    if taille%2 == 0:
        retour = (listeT[milieu] + listeT[milieu-1])/2
    else:
        retour = listeT[milieu]
    return retour

#Variable avec un appel de la fonction pour avoir la valeur médiane des sauts et
google_sauts_med=calculValMed(liste2[0])
google_temps_med=calculValMed(liste2[1])
opendns_sauts_med=calculValMed(liste2[2])
opendns_temps_med=calculValMed(liste2[3])

#On range ces variables dans une liste
listMed:list=[[google_sauts_med, google_temps_med], [opendns_sauts_med, opendns_

#print de cette liste
print(listMed)
```

```
[[8, 16.753], [6, 17.462]]
```

Test :

- [[8, 16.753], [6, 17.462]]

Calcul de la variance

In [ ]:
```
def variance(sauts:list[int], temps:list[float]) -> float:
    """fonction qui renvoie la variance d'une liste

    Args:
        sauts (list[int])
```

```
        temps (list[float])

    Returns:
        float: variance de la liste
    """
    esperance=0
    variance=0

    for i in range(len(sauts)) :
        esperance=sauts[i]*(temps[i]*10**-3)+esperance

    for j in range(len(sauts)) :
        variance=(((sauts[j]-esperance)**2)*(temps[i]*10**-3))+variance

    return variance

#Variable des variances de google et opendns
variance_google=variance(liste2[0], liste2[1])
variance_opendns=variance(liste2[2], liste2[3])

#On range ces variables dans une liste
listeVar:list=[variance_google, variance_opendns]

#print de cette liste
print(listeVar)
```

```
[15.650380243517594, 8.870258930806669]
```

Test :

- [15.650380243517594, 8.870258930806669]

Calcul de l'écart type

In [ ]:

```
def ecart_type(variance:float) -> float :
    """fonction qui renvoie l'écart type d'une liste

    Args:
        variance (float): la variance calculée précedemment

    Returns:
        float: ecart type de la liste
    """
    return sqrt(variance)

#Variable des écarts type de google et opendns
ecart_google=ecart_type(variance_google)
ecart_opendns=ecart_type(variance_opendns)

#On range ces variables dans une liste
listeEcartT:list=[ecart_google, ecart_opendns]

#print de cette liste
print(listeEcartT)
```

```
[3.956056147669999, 2.9782979922779167]
```

Test :

- [3.956056147669999, 2.9782979922779167]

Lecture du fichier donnee_fictif.csv pour le mettre sous forme d'un dictionnaire

In [ ]:
```
def csvToDict(f : str) -> dict[str,list[list[int]]]:
    """! Fonction chargeant les données stockées dans un fichier csv avec un sép
    `,` et retournant un dictionnaire où les clés sont les différents noms de fi
    décrits et les valeurs [taille en ko, timestamp] sont stockées dans une list
    listes

    Args:
        @param fichier (str): Le nom du fichier à charger
        @return dict[str,list]: les clés sont les noms des fichiers observés et

    Returns:
        valeurs une liste de dimension 2 [[taille,timestamp]]


    liste : list = [[ ??? , ??? ]] # on crée une liste à deux dimensions
    dictionnaire[cle] = liste
    #Attention à bien stocker des valeurs numériques dans votre liste

    dictionnaire[cle].append([???, ???]) # on crée une liste simple
    """

    dictR : dict[str,list[list[int]]] = {}

    with open(f, newline="") as csvfile :
        datareader = csv.reader(csvfile, delimiter= ',', quotechar='|')
        for ligne in datareader :
            if ligne[0] not in dictR.keys() : # clé non présente
                dictR[ligne[0]] = []

            dictR[ligne[0]].append([int(ligne[1]), float(ligne[2])])


    return dictR

#stockage du dictionnaire dans une variable
liste3=csvToDict('donnee_fictif.csv')
print(liste3)
```

```
{'dns.google': [[8, 15.957], [10, 15.873], [2, 15.966], [6, 16.056], [8, 18.17],
[3, 16.439], [12, 16.277], [20, 16.269], [8, 16.104], [1, 15.953], [5, 18.173],
[3, 17.637], [8, 19.723], [6, 16.611], [11, 16.753], [7, 16.754], [2, 16.948],
[9, 17.558], [13, 17.661], [8, 16.887], [6, 16.676], [4, 16.403], [14, 17.138],
[9, 16.894], [8, 16.852]], 'dns.opendns.com': [[6, 17.462], [4, 17.28], [2, 17.22
1], [6, 17.638], [8, 17.174], [10, 17.737], [3, 17.54], [9, 17.319], [7, 17.628],
[1, 17.711], [8, 17.475], [6, 17.05], [9, 17.564], [2, 17.137], [7, 17.283], [11,
18.352], [9, 18.245], [6, 17.147], [8, 17.27], [3, 17.185], [12, 18.188], [2, 17.
214], [5, 18.853], [4, 17.878], [6, 17.253]]}
```

Test :

- {'dns.google': [[8, 15.957], [10, 15.873], [2, 15.966], [6, 16.056], [8, 18.17], [3, 16.439], [12, 16.277], [20, 16.269], [8, 16.104], [1, 15.953], [5, 18.173], [3, 17.637], [8, 19.723], [6, 16.611], [11, 16.753], [7, 16.754], [2, 16.948], [9, 17.558], [13, 17.661], [8, 16.887], [6, 16.676], [4, 16.403], [14, 17.138], [9, 16.894], [8, 16.852]], 'dns.opendns.com': [[6,

17.462], [4, 17.28], [2, 17.221], [6, 17.638], [8, 17.174], [10, 17.737], [3, 17.54], [9, 17.319], [7, 17.628], [1, 17.711], [8, 17.475], [6, 17.05], [9, 17.564], [2, 17.137], [7, 17.283], [11, 18.352], [9, 18.245], [6, 17.147], [8, 17.27], [3, 17.185], [12, 18.188], [2, 17.214], [5, 18.853], [4, 17.878], [6, 17.253]]}

Appel des fonctions avec le dictionnaire

In [ ]:
```
for element in liste3.keys():
        print(element)
        liste4 = transpose(liste3[element])
        print(liste4)
        print(f"Sauts Min : {calculValMini(liste4[0])}")
        print(f"Temps Min : {calculValMini(liste4[1])}")
        print(f"Sauts Max : {calculValMaxi(liste4[0])}")
        print(f"Temps Max : {calculValMaxi(liste4[1])}")
        print(f"Sauts Moy : {calculValMoy(liste4[0])}")
        print(f"Temps Moy : {calculValMoy(liste4[1])}")
        print(f"Sauts Med : {calculValMed(liste4[0])}")
        print(f"Temps Med : {calculValMed(liste4[1])}")
        print(f"Variance : {variance(liste4[0], liste4[1])}")
        print(f"Ecart Type : {ecart_type(variance(liste4[0], liste4[1]))}")
        print("----------------")
```

```
dns.google
[[8, 10, 2, 6, 8, 3, 12, 20, 8, 1, 5, 3, 8, 6, 11, 7, 2, 9, 13, 8, 6, 4, 14, 9,
8], [15.957, 15.873, 15.966, 16.056, 18.17, 16.439, 16.277, 16.269, 16.104, 15.95
3, 18.173, 17.637, 19.723, 16.611, 16.753, 16.754, 16.948, 17.558, 17.661, 16.88
7, 16.676, 16.403, 17.138, 16.894, 16.852]]
Sauts Min : 1
Temps Min : 15.873
Sauts Max : 20
Temps Max : 19.723
Sauts Moy : 7.64
Temps Moy : 16.869279999999996
Sauts Med : 8
Temps Med : 16.753
Variance : 15.650380243517594
Ecart Type : 3.956056147669999
----------------
dns.opendns.com
[[6, 4, 2, 6, 8, 10, 3, 9, 7, 1, 8, 6, 9, 2, 7, 11, 9, 6, 8, 3, 12, 2, 5, 4, 6],
[17.462, 17.28, 17.221, 17.638, 17.174, 17.737, 17.54, 17.319, 17.628, 17.711, 1
7.475, 17.05, 17.564, 17.137, 17.283, 18.352, 18.245, 17.147, 17.27, 17.185, 18.1
88, 17.214, 18.853, 17.878, 17.253]]
Sauts Min : 1
Temps Min : 17.05
Sauts Max : 12
Temps Max : 18.853
Sauts Moy : 6.16
Temps Moy : 17.552159999999997
Sauts Med : 6
Temps Med : 17.462
Variance : 8.870258930806669
Ecart Type : 2.9782979922779167
----------------
```

Test :

- dns.google
- [[8, 10, 2, 6, 8, 3, 12, 20, 8, 1, 5, 3, 8, 6, 11, 7, 2, 9, 13, 8, 6, 4, 14, 9, 8], [15.957, 15.873, 15.966, 16.056, 18.17, 16.439, 16.277, 16.- 269, 16.104, 15.953, 18.173, 17.637, 19.723, 16.611, 16.753, 16.754, 16.948, 17.558, 17.661, 16.887, 16.676, 16.403, 17.138, 16.894, 16.852]]
- Sauts Min : 1
- Temps Min : 15.873
- Sauts Max : 20
- Temps Max : 19.723
- Sauts Moy : 7.64
- Temps Moy : 16.869279999999996
- Sauts Med : 8
- Temps Med : 16.753
- Variance : 15.650380243517594
- Ecart Type : 3.956056147669999

- dns.opendns.com
- [[6, 4, 2, 6, 8, 10, 3, 9, 7, 1, 8, 6, 9, 2, 7, 11, 9, 6, 8, 3, 12, 2, 5, 4, 6], [17.462, 17.28, 17.221, 17.638, 17.174, 17.737, 17.54, 17.319, 17.628, 17.711, 17.475, 17.05, 17.564, 17.137, 17.283, 18.352, 18.245, 17.147, 17.27, 17.185, 18.188, 17.214, 18.853, 17.878, 17.253]]
- Sauts Min : 1
- Temps Min : 17.05
- Sauts Max : 12
- Temps Max : 18.853
- Sauts Moy : 6.16
- Temps Moy : 17.552159999999997
- Sauts Med : 6
- Temps Med : 17.462
- Variance : 8.870258930806669
- Ecart Type : 2.9782979922779167

Partie IV - Les Diagrammes

Fonction pour compter le nombre de saut

In [ ]:
```
# Fonction permettant de compter les sauts identiques

def count_liste(liste : list) :
    listeCount = [[],[]]
    for i in range(len(liste)):
        if liste[i] not in listeCount[1]:
            listeCount[0].append(liste.count(liste[i]) )
            listeCount[1].append(liste[i])
    return listeCount
```

```
print(count_liste(liste2[0])) # liste2[0] correspond aux sauts de Google
```

```
[[6, 1, 2, 3, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1], [8, 10, 2, 6, 3, 12, 20, 1, 5, 1
1, 7, 9, 13, 4, 14]]
```

Test :

- [[6, 1, 2, 3, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1], [8, 10, 2, 6, 3, 12, 20, 1, 5, 11, 7, 9, 13, 4, 14]]

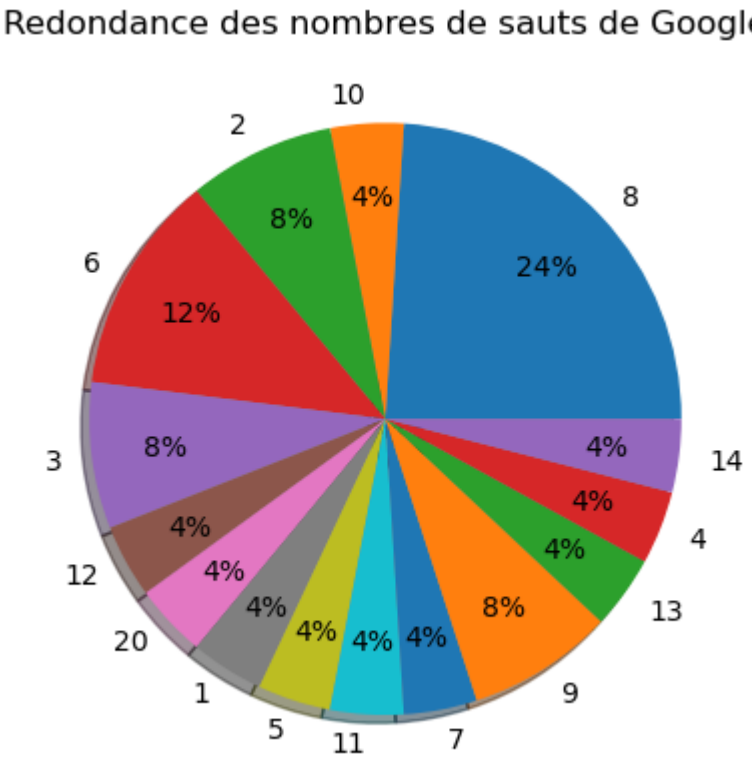
Nous avons réalisés cette fonction afin de pouvoir compter combien de fois un saut revenait dans notre liste. Cela nous serviras ensuite pour créer un camembert avec le pourcentage de redondance de chaque saut. Ici, nous le réaliserons avec les sauts de Google mais notre fonction fonctionne parfaitement avec les sauts d'Opendns.

Camembert

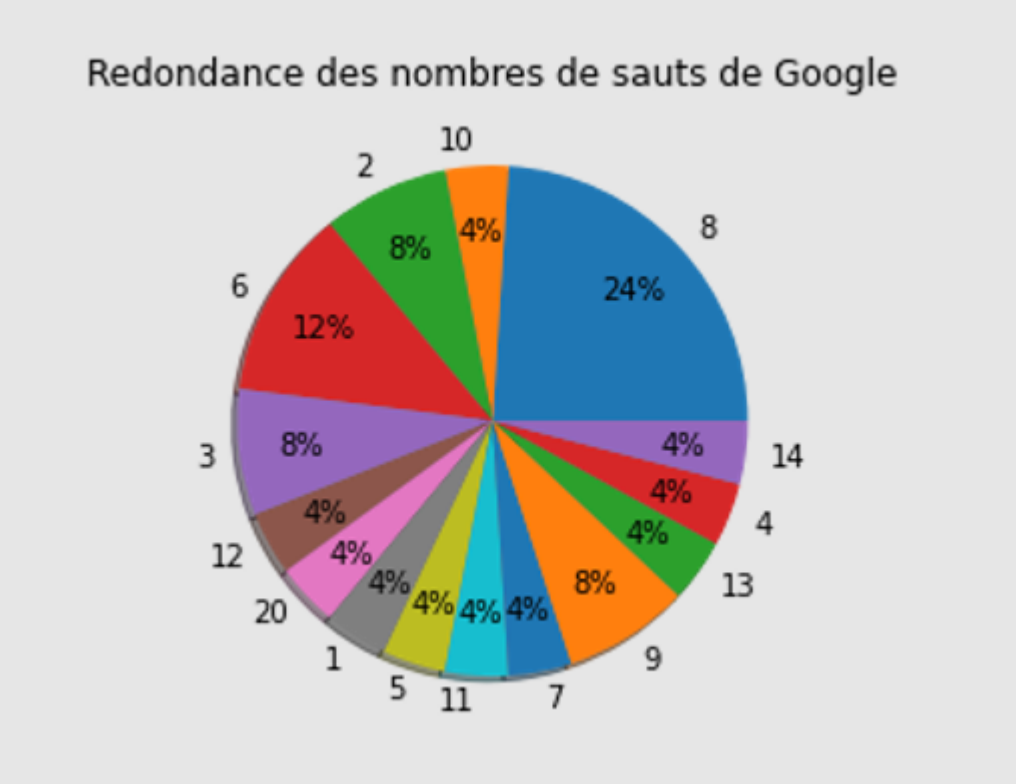
In [ ]:
```
# Fonction faisant apparaitre un diagramme camembert

def affCamembertSimple(valeurs:List[float] , libelles : List[str] ) :
    """!
    Procédure d'affichage d'un camenbert (Pie)
    @param valeurs :List[float] Liste des valeurs
    @param libelles :List[str] Liste des libellés de chaque valeur
    """
    plt.pie(valeurs, labels=libelles, normalize=True,autopct='%1.0f%%',pctdistan
    #plt.legend() # pour afficher la légende, inutile dans notre cas
    plt.title("Redondance des nombres de sauts de Google")
    plt.show()

liste_count = count_liste(liste2[0])
affCamembertSimple(liste_count[0],liste_count[1])
```



Capture d'écran :

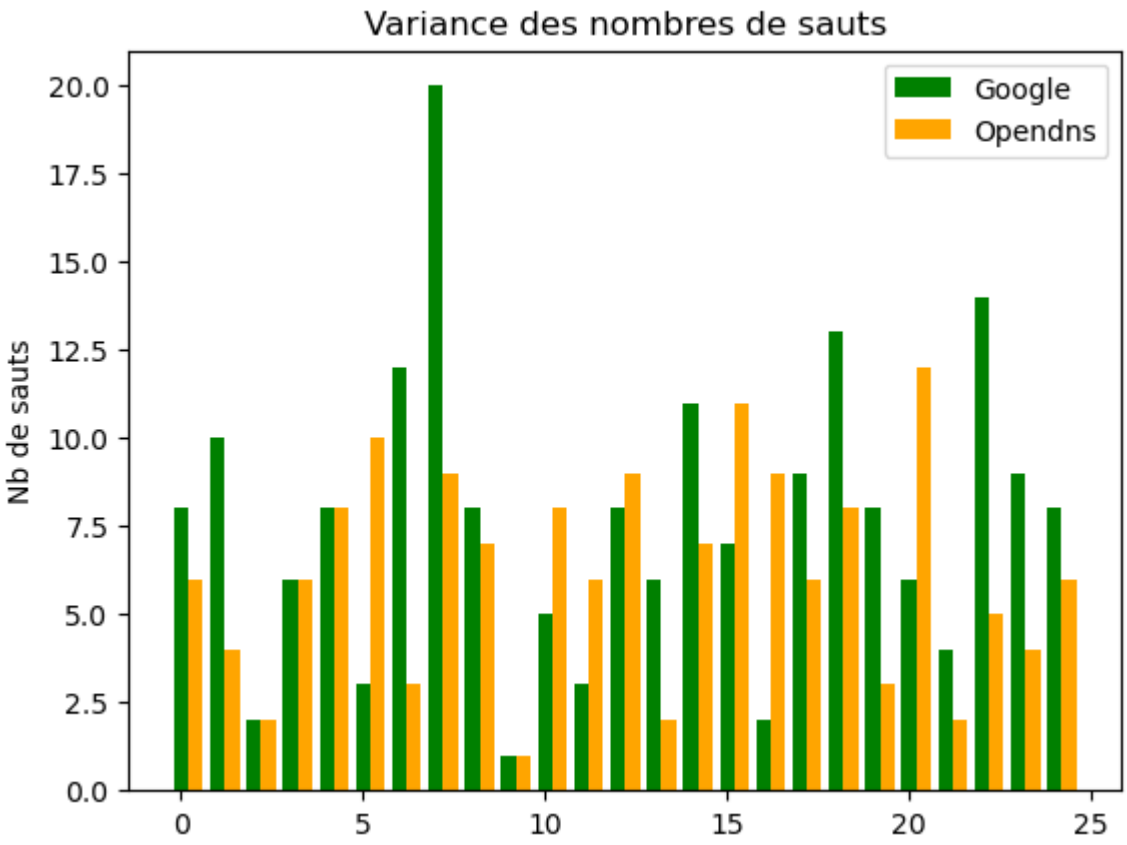


Histogramme

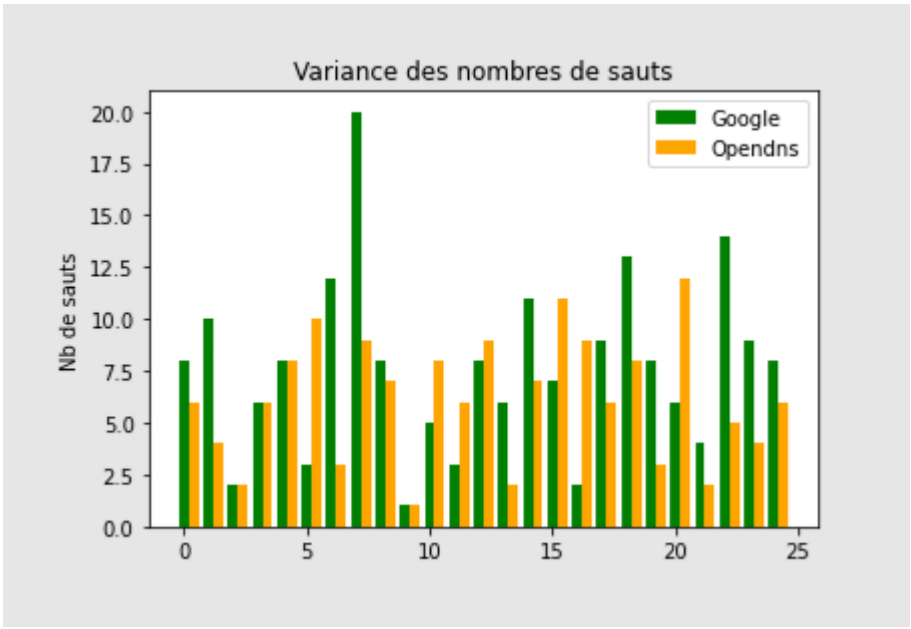
In [ ]:
```
# Fonction faisant apparaitre un digramme baton

def affBars (data :list[float], data2:list[float]) : # pour une série de données
    """!
    Procédure d'affichage sous la forme de barres.
    @param data :List[float] Liste de la hauteur des barres
    """
    barWidth=0.4
    r1 = range(len(data))
    r2 = [x + barWidth for x in r1]
    bar = plt.bar(r1,data,width=barWidth,color=["green" for i in data], linewidt
    plt.bar(r2,data2,width=barWidth,color=["orange" for i in data], linewidth=2,
    plt.ylabel("Nb de sauts")
    plt.title("Variance des nombres de sauts")
    plt.legend()
    plt.show()

affBars(liste2[0], liste2[2])
```



Capture d'écran :



Courbe

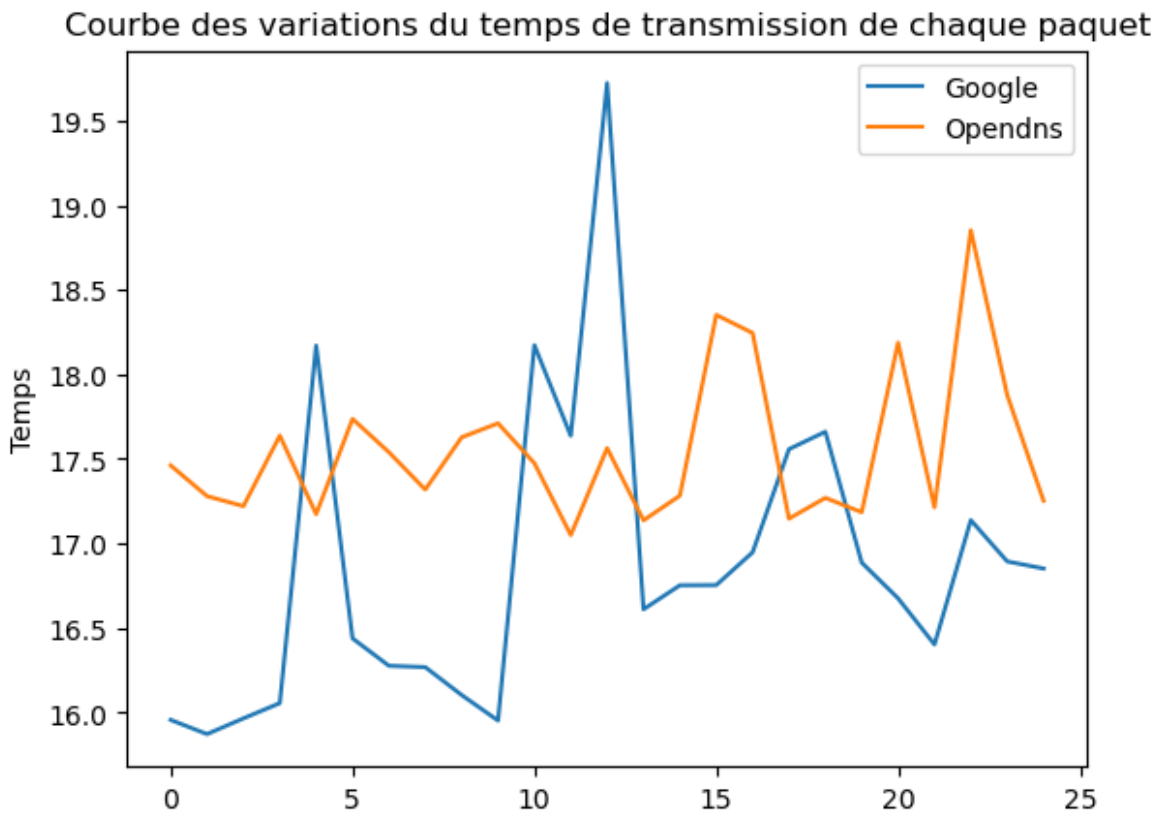
In [ ]:
```python
def affCourbe (data : list[list[int],list[float],list[int],list[float]]) :
    """Procédure d'affichage sous la forme de courbe

    Args:
        data (list[int]): Liste2 qui est la liste qui regroupe les sauts et temp
        Google et de OpenDNS
    """
    print(data)
    plt.plot(data[1], label="Google")
```

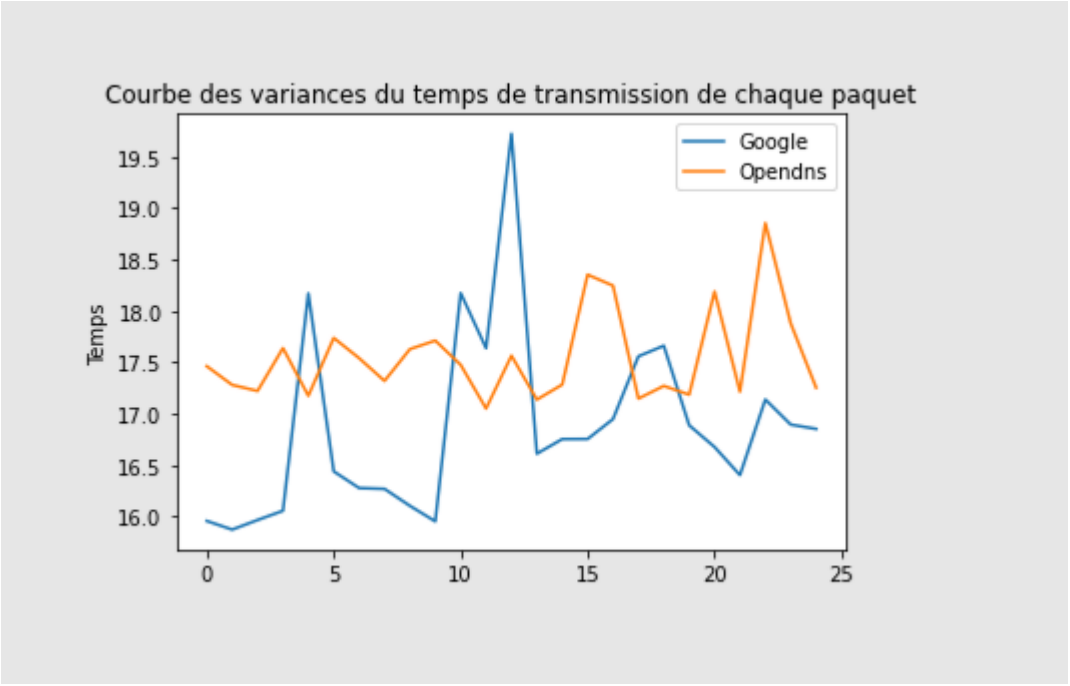
```
    plt.plot(data[3], label="Opendns")
    plt.legend()
    plt.title("Courbe des variations du temps de transmission de chaque paquet")
    plt.ylabel("Temps")
    plt.show()

affCourbe(liste2)
```

```
[[8, 10, 2, 6, 8, 3, 12, 20, 8, 1, 5, 3, 8, 6, 11, 7, 2, 9, 13, 8, 6, 4, 14, 9,
8], [15.957, 15.873, 15.966, 16.056, 18.17, 16.439, 16.277, 16.269, 16.104, 15.95
3, 18.173, 17.637, 19.723, 16.611, 16.753, 16.754, 16.948, 17.558, 17.661, 16.88
7, 16.676, 16.403, 17.138, 16.894, 16.852], [6, 4, 2, 6, 8, 10, 3, 9, 7, 1, 8, 6,
9, 2, 7, 11, 9, 6, 8, 3, 12, 2, 5, 4, 6], [17.462, 17.28, 17.221, 17.638, 17.174,
17.737, 17.54, 17.319, 17.628, 17.711, 17.475, 17.05, 17.564, 17.137, 17.283, 18.
352, 18.245, 17.147, 17.27, 17.185, 18.188, 17.214, 18.853, 17.878, 17.253]]
```



Capture d'écran :



Conclusion

Malgrès quelques complications concernant les données que l'on devait récupérer, nous avons tout de même réussi à réaliser des diagrammes cohérents. Cependant, nous voyons tout de même quelque point d'amélioration, en effet il était possible de créer des animations pour notre histogramme ou notre courbe mais nous n'avons pas compris comment utiliser FuncAnimation fournis avec mathplotlib. Si ce projet était à refaire nous essayerons d'appronfondir plus cette partie. Pour finir, ce projet était très intérréssant et nous à permis de monter en compétences que ce soit en systèmes ou en python.