**Consignes.** Avant toute chose, prenez conscience que les corrections sont effectuées à l'aide d'un automate, par conséquent toute violation des règles conduira inévitablement à la délivrance d'une note nulle.

- Comment livrer mon code ?
  Si mon nom de famille est **yunès** et que je souhaite rendre le travail pour l'exercice 12 du tp 3 alors je dois construire une archive au format ZIP de nom `yunes-tp3-ex12.zip`, le nom ayant été converti en minuscules, sans accent et sans espace (*ie* : **de Gaulle** doit être traduit en `degaulle`). Si vous avez un homonyme rajoutez des initiales, par exemple `yunesjb-tp1-ex1`. Respectez votre nommage pour tout le semestre.
- Que mettre dans l'archive ?
  Le code source (et uniquement le code source) correspondant à l'exercice.

De plus, suivez bien les instructions complémentaires des livrables.

Pour ce TP, il faut récupérer les fichiers de trace du TP n°2 (`access.log` et `error.log`).

# 1 Exercice 1

Écrire un programme python qui analyse le fichier des erreurs pris en argument :

- et affiche pour chaque journée le nombre d'erreurs
- puis la moyenne des erreurs/jours

Exemple :

```
01/01/2019 157
02/01/2019 10
05/01/2019 17
...
Moyenne : 45.8
```

Livrable : l'archive contenant le programme `dayerror.py`.

# 2 Exercice 2

Écrire un programme python qui analyse le fichier des erreurs pris en argument :

- et affiche pour chaque url en défaut, le nombre de fois qu'elle a été en défaut
- puis le nombre d'urls en défaut

Exemple de sortie :

```
/truc/bidule.txt 18
/truc/machin.jpg 29
...
207 urls en défaut
```

Livrable : l'archive contenant le programme `urlerror.py`.

# 3 Exercice 3

Écrire un programme python qui analyse le fichier des erreurs pris en argument :

- et affiche pour chaque url en défaut le nombre de fois qu'elle l'a été, la première fois et la dernière fois (date/heure)

Exemple de sortie :

```
/truc/bidule.txt 18 01/02/2018 05/06/2019
/truc/machin.jpg 29 04/09/2015 05/04/2018
...
```

Livrable : l'archive contenant le programme `filetypeerror.py`.

# 4 Exercice 4

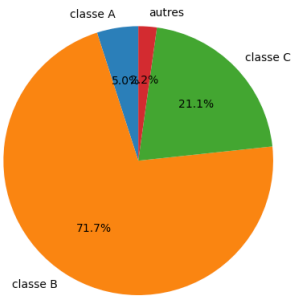
Écrire un programme python qui analyse le fichier des traces standards pris en argument :

- collecte les adresses réseau pour afficher en sortie le nombre de connexions par classe IP
- et affiche ce résultat sous la forme d'un camembert à l'aide du package `matplotlib` (classe IP vs nombre de connexions)

Exemple de sortie :

```
classe A 3289
class B 340
...
```

Exemple d'image générée :



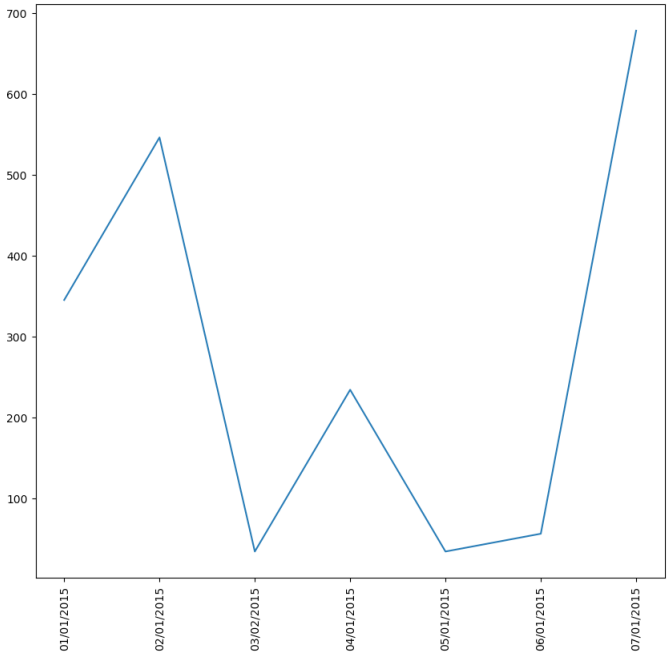

Livrable : l'archive contenant le programme `ipclass.py`.

# 5 Exercice 5

Écrire un programme python qui analyse le fichier des traces standards pris en argument :

- collecte le nombre de connexions pour chaque jour
- affiche ce résultat sous la forme de plusieurs graphes (pas de camembert, une fonction plutôt) `matplotlib` (jour-année/connexions) (semaine-année) (mois-année/connexions).

Exemple d'image générée :




Livrable : l'archive contenant le programme `connexions.py`.

# 6 Exercice 6 (difficile ?)

Écrire un programme python qui analyse le fichier des traces standards pris en argument :

- collecte le nombre de connexions par type de navigateur ainsi que par type de système. Lorsqu'il n'est pas possible de reconstruire ces données, on peut ne pas tenir compte de l'entrée correspondante,
- affiche ce résultat sous la forme de camemberts `matplotlib` (navigateur/nombre) (système/nombre)

Le décodage du type de navigateur est assez complexe (les données ne sont pas normalisées), mais on peut trouver sur internet des expressions régulières ou des descriptifs de techniques pour résoudre ce problème.
Livrable : l'archive contenant le programme `navtype.py`.