 # gestion des arguments

 # version basique
ouvrir un script et faire

```
import sys


print(sys.argv) # que voyez-vous ? type de l'objet renvoyé ?

# stocke l'argument 1 dans la variable filename
filename = sys.argv[1]
print(filename)
```

--> executer le script

```
$ script.py 1brs.pdb
```

Et maintenant ?

```
$ script.py
```

--> erreur ! car `sys.argv[1]` est vide puisque je n'ai pas renseigné de fichier.

On peut alors ajouter test pour tester les args de l'utilisateur

```
if len(sys.argv < 2) :
      print("erreur, rebtrez le nom de fichier")
      sys.exit()
else:
      filename = sys.argv[1]
      print(filename)
```

cela dit, même si sys.argv c'est pas mal car me retourne une liste contenant tous les éléments (séparés selon les espaces) de ma ligne de commande, cela me contraint à respecter l'ordre des arguments attendu dans le programme. Très contraignant et pas très lisible. J'aimerai une gestion plus fine des arguments avec un flag explicite (plus lisible) et des arguments par défaut par ex. Et pour ça, argparse c'est très bien !

 # avec argparse

```
$ pip install argparse
```

```
import argparse

# create the parser object
parser = argparse.ArgumentParser()
```

```
# add an argument
parser.add_argument("-f", "--file", help="nom du fichier a lire", type=str,
required = True)

args = parser.parse_args()
infile = args.file
print(infile)
```

j'ai crée un argument obligatoire qui récupère le nom du fichier d'entrée via le flag file et que je stocke ensuite dans la variable infile.

```
$ script.py file.txt

$ script.py

test.py: error: the following arguments are required: -f/--
file
```

L'appel du programme sans l'argument « file » plante et renvoie un message d'erreur.

dans le doute, je peux aussi accéder à « l'aide » de mon prgramme qui va afficher la liste des arguments, leur type et les lignes que j'aurai renseignées via l'argument « help » de la méthode parrser.add_argument()

```
$ script.py -h
$ script.py --help # équivalent
```

```
import argparse

# create the parser object (just an init, the command line is not parsed yet)
parser = argparse.ArgumentParser()

# add an argument
parser.add_argument("-f", "--file", help="nom du fichier a lire", type=str,
required = False)

# parse the command line according to the flags given with the method
parser.add_argument()
args = parser.parse_args()
print(args.file)
```

argument optionnel

```
parser.add_argument("-m", "--mode", help="mode de calcul de la distance inter-
residus", type=str, required = False)

mode_calcul = args.mode
```

#Ici l'argument « mode » est optionnel. L'appel du programme sans l'argument « mode » est fonctionnel.

#Maintenant, je veux renseigner une valeur par défaut à mode qui sera « atom »:

```
parser.add_argument("-m", "--mode", help="mode de calcul de la distance inter-
residus", type=str, required = False, default = "atom")

args = parser.parse_args()
mode_calcul = args.mode
print("le calcul se fera sur :", mode_calcul)
```

argument booléen, action= "store_true" permet de mettre à True la variable args.weigth si « --weight » ou « -w » est appelé

```
parser.add_argument("-w", "--weigth", action="store_true", help="weigth param X in
fun Y")

weigth_fun = args.weigth

if weigth_fun :
    print("ok je pondere !")
else:
    print("pas de ponderationpr l'operation")
```

advanced usage
nargs permet de spécifier qu'on ne sait pas à l'avance le nombre d'arguments, et const va me permettre de donner une valeur de pondération par défaut si jamais l'utilisateur voulait pondérer ; très pratique dans certaines situations. Par exemple, je ne sais ps si l'utilisateur va vouloir pondérer le terme X par ex dans la fct Y. Si il ne veut pas, il n'appellera pas l'arg -w ou --weigth et donc la longueur de cet arg sera de 0. Par contre, si l'utilisateur veut pondérer, il peut juste appeler l'argument --weight sans renseigner de valeur de poids, dans ce cas le poids par défaut est de 0.5. Si jamais il veut pondérer avec un autre poids, il suffit alors qu'il appelle le programme en précisant la valeur requise :

```
$ test.py -f 1brss.pdb --weigth 0.8 --mode centroid
```

```
parser.add_argument("-w", "--weigth", help="ponderation ou non, si pondere, val par
defaut = 0.5", type=float, default = False, nargs = "?", const=0.5)
```