Rapport de projet

I. Installation et utilisation

L'application nécessite l'installation d'une version 3.x de Python pour une utilisation optimale. Pour lancer l'application, il suffit d'ouvrir le répertoire où elle est stocké à partir de l'invite de commande et de l'exécuter à partir de la commande ***python Mobile.py*** ou bien ***python3.x Mobile.py*** (3.x étant la version installé sur l'ordinateur. Sur les ordinateurs de l'université, il s'agit de la version 3.1).
Pour définir la taille de la fenêtre de l'application, il faudra y apporter deux paramètres supplémentaires : la largeur et la hauteur. La commande pour l'exécution devient donc ***python Mobile.py 800 600*** ou bien ***python3.x Mobile.py 800 600***.

L'utilisation de l'application est très simple : toutes les commandes sont disponibles dans la barre de menu.

Dans le menu Fichier, vous pouvez :
- définir un nouveau mobile sous la forme d'une liste (par exemple [[5, 3], 8]) ;
- ouvrir un mobile à partir d'un fichier (deux exemples de fichiers sont disponibles) ;
- générer un mobile aléatoire après avoir défini deux paramètres : le nombre total d'objet et la taille maximal des objets ;
- modifier le mobile courant ;
- enregistrer le mobile courant dans un fichier sous la forme d'une liste ;
- quitter l'application.

Dans le menu Affichage, vous pouvez classer le mobile courant dans l'ordre croissant ou décroissant des poids des objets (normal permettant de ré-afficher le mobile d'origine).

II. Fonctionnalités

Le code de l'application est séparé en deux classes.

La classe Fenetre s'occupe de l'affichage de l'application dans une fenêtre grâce au module tkinter. Les différentes fonctions permettent d'afficher le menu et d'exécuter les différentes options disponibles à partir de ce même menu.

La classe Arbre, quant à elle, s'occupe de la structure du mobile qui ressemble à celle d'un arbre : les tiges sont des branches et les objets sont des feuilles. C'est cette classe qui permettra toutes les transformations des différentes formes de mobile :
- d'une liste vers un arbre et inversement ;
- d'un fichier vers une liste ;
- d'une liste de la forme [5, 3, 8] à une liste de la forme [[5, 3], 8] (qui respecte les conditions d'une liste pour mobile)...

D'autres fonctions parcourent l'arbre pour calculer certaines valeurs :
- le poids total d'un arbre ;
- l'objet de poids minimal et maximal appartenant à l'arbre
- le nombre d'objets dans l'arbre...

III. Comportements erronés ou inattendus

Le code a été pensé de façon a ce que l'affichage du mobile soit le moins ambiguë possible. Cependant, si la largeur de la fenêtre de l'application est trop petite par rapport au nombre d'objets dans le mobile, il se peut que le mobile soit très serré, ce qui empêche d'avoir une bonne vision du mobile. De même, si la hauteur de la fenêtre est plus petite que la hauteur du mobile, ce dernier sera rogné par le bas.

IV. Fonctionnalités non-implémentées

Selon les instructions du projet, toutes les fonctionnalités obligatoires sont présentes dans l'application. Quelques fonctionnalités supplémentaires ont été ajouté, mais une fonctionnalité définie comme supplémentaire n'est pas présente : celle qui permet l'affichage d'un mobile non équilibré.

V. Algorithme utilisé

L'algorithme utilisé pour la construction du mobile est un algorithme équilibré qui dispose les objets dans le mobile de façon récursive, de telle sorte que la répartition soit équitable de part et d'autre du mobile.

L'algorithme fonctionne sur deux fonctions de la façon suivante : dans la première fonction, il va tout d'abord réunir les objets dans une liste simple, puis appeler la deuxième fonction ayant pour paramètre la liste. Ensuite, il va créer une nouvelle liste de la forme [None, None] dont le premier élément sera un appel récursif avec cette fois-ci pour paramètre la première moitié de la liste, le deuxième élément étant un appel récursif avec comme paramètre la deuxième moitié de la liste. Lorsque la longueur de la liste en paramètre est égal à 1, la nouvelle liste prend pour valeur la valeur de la liste en paramètre.

Exemple :
Soit la liste [2, 4, 6, 8]. L'algorithme algo(liste) va fonctionner de la manière suivante :
- création d'une nouvelle liste [None, None] ;
- le premier élément de la nouvelle liste est algo([2, 4]), le deuxième est algo([6, 8]) ;
- dans algo([2, 4]), nouvelle liste est [algo(2), algo(4)],
- dans algo([6, 8]), nouvelle liste est [algo(6), algo(8)] ;
- dans algo(2), la taille de la liste est 1 donc nouvelle liste = 2, même chose pour algo(4), algo(6) et algo(8) ;
- notre nouvelle liste final a donc la forme [[2, 4], [6, 8]]

VI. Bilan du travail réalisé

Ce projet de LS4 était très épanouissant dans notre apprentissage en Python. Il nous a permis d'assimiler beaucoup de notions vues de manière générale lors des TPs, notamment en rapport avec les différentes possibilités du module tkinter. Notre application est loin d'être parfaite, beaucoup d'éléments visuels aurait pu être ajoutés si nous avions eu plus de temps, comme par exemple des effets de mouvement, des interactions avec la souris, ...