Delatte Laurent
Fellous Ines
Senelier Mona

Rapport atelier Introduction Homme Machine

lien vers le github du projet: https://github.com/alphae-nix/Account_Manager

I. Le choix du projet

Nous avons eu l'idée d'une application de gestionnaire de finance personnelle, cela nous permettait de réaliser une première application de base relativement rapidement et par la suite de rajouter différents fonctionnalitées si nous en avions le temps.

II. Le but

Nous avons choisis de produire une interface se composant d'une page de connexion qui permet de créer un compte puis de s'y connecter. Il est donc possible de créer plusieurs comptes différents pour différencier, par exemple des revenus personnels et professionnels. Ainsi, cette page de connexion donne accès à une plateforme permettant de gérer les dépenses et les revenus.
Il est donc possible d'entrer une dépense, en indiquant une date, une catégorie de dépense, un montant ainsi que le nom de cette dépense et d'actualiser le solde du compte.
De plus, cette mise à jour rajoute une ligne de dépense à une liste qui regroupe toutes les dépenses une par une spécifiant les informations de cette dépenses ainsi qu'en les triant par leur domaines. Par exemple, si l'utilisateur fait une dépense de type loisir, une ligne de dépense s'ajoutera dans la partie loisir. Nous avons également ajouté une commande de retour en arrière afin de pouvoir annuler la dernière opération qui a été effectué si jamais l'utilisateur commet une erreur.

Nous voulions mettre en place un système de sauvegarde des données pour que si l'utilisateur se reconnecte il retrouve les dépenses et recettes qu'il a déjà entré mais cette étape était trop compliqué par rapport aux connaissances que nous avions ainsi que le temps disponible.

III. La réalisation

1) La base du projet

Nous avons commencé par créer une grande partie des frames afin d'avoir la base du projet et de pouvoir mieux se départager le travail. Ainsi, on a créé quatre frames principales, une à gauche de la page, une en haut afin de faire un menu, une à droite qui a été complété par la frame de l'ajout d'une dépense ou d'une recette et enfin, une dernière frame au milieu.

Ensuite, nous avons ajouté les éléments principaux de la page en mettant des labels, Entry, menu déroulant ou boutons qui auront une utilisation plus tard.

2) Les fonctions principales

Puis, nous nous sommes rendus compte que la partie la plus difficile à écrire de notre code était la fonction qui permet d'actualiser les 4 parties(Loisirs, Alimentation, Loyer, Voyages) affichant les dépenses et entrées.
Nous nous sommes d'abord concentré sur le fait que le bouton ajoute une nouvelle liste de dépense dans une frame. Nous avons donc créé une fonction qui ajoute une nouvelle frame à chaque actualisation, frame contenant le montant de la dépense, ainsi que la date et la description. C'est à dire que la fonction doit récupérer les valeurs entrées par l'utilisateur dans la frame de droite. Ensuite, nous avons dû mettre en place le fait que la ligne de dépense s'ajoute dans la bonne frame en fonction du type de dépense qui découle d'un menu déroulant.

Ensuite, nous nous sommes concentrés sur l'actualisation du solde, c'est à dire que la fonction fasse plusieurs actions simultanément à l'appuie du bouton "actualiser".

Pour le solde la grande difficulté était de conserver la valeur précédente et de lui appliquer une nouvelle opération. Après recherche nous avons repris l'exemple d'une calculatrice ou l'on peut conserver un résultat et lui appliquer une autre opération. Il a fallu créer plusieurs variable pour tous les cas possible celui d'une dépense ou recette. En effet l'opération n'est pas la même soit une soustraction soit une addition. La variable x est commune pour les deux cas car elle permet de calculer la valeur du solde, y et z sont respectivement pour depense et recette.

Puis, nous avons complété en ajoutant une actualisation des recettes au total ainsi que des dépenses. Nous avions d'abord décidé de faire deux frames différentes, une pour les recettes et l'autre pour les dépenses. Cependant nous nous sommes rendu compte qu'il était plus intuitif de faire une seule et même frame qui contenait au début un radiobutton

dont on récupère la valeur à l'aide d'un .get() et d'un result à l'image de ce que nous avions déjà fait avant pour récupérer les autres informations.

3) Les exceptions que l'utilisateur peut rencontrer

Ensuite nous avons dû mettre en place un système assez complexe pour les exceptions afin d'anticiper les erreurs pouvant être commise par l'utilisateur. Par exemple, le fait que l'utilisateur ne puisse pas ajouter une nouvelle ligne s'il n'a pas indiqué le montant de la dépense (ou de la recette).

Nous avons donc mis en place liste de conditions qui implique la création des frames d'ajout de ligne. Ainsi, uniquement si les conditions sont respectés, la ligne est créée.

Pour anticiper les erreurs de l'utilisateur nous avons utiliser les try et catch permettant de tester un bout de code qui est susceptible de produire une erreur, puis dans le catch on a utilisé soit ValueError pour préciser à l'utilisateur que la valeur attendue était un nombre et AssertionError pour limité la valeur des nombres pour certaines cases (par exemple pour les mois le nombre est limité à 12)

4) Le design

Enfin, nous avons entrepris d'améliorer le design de notre interface. En effet lorsque nous avons créé chaque frame et placé les éléments dans la page nous avons mis des couleurs à chaque frame afin de bien réussir à les différencier pour placer nos éléments aux bons endroits. Ainsi, nous avons dû retirer toutes les couleurs et de faires quelque chose d'un peu plus esthétique dans la limite de tkinter. Nous avons donc ajouté des reliefs dans les frames ainsi que des couleurs un peu plus harmonieuses. Nous avons également changé des polices afin de finaliser les détails.

IV. Les problèmes rencontrés

L'un des premiers problème que nous avons rencontrés fut lorsque nous avons voulu gérer les exceptions pour que l'utilisateur ne puisse pas ne pas remplir un des éléments.
Nous avions en premier lieu, créé un grand nombre de boucle conditionnelles pour prendre en considération toutes les exceptions possible et également le fait que le programme ne rentre pas dans les autres boucles si une condition n'était pas respecté. L'inconvénient majeur de cette méthode est le fait que dès qu'une erreur était détectée il y avait 5 messages d'erreur qui s'affichait, cela rendait l'utilisation un peu complexe et beaucoup moins intuitive qu'elle devait l'être. On a donc mis en place une deuxième solution (décrite dans le paragraphe précédent) à l'aide de la méthode try/exept.

Ensuite le problème majeur de ce projet fut de tenter de créer une fonction de sauvegarde des données entrées par l'utilisateur que nous n'avons pas réussi à mettre en place à cause d'un manque de temps et de connaissance. Nous avons pensé à associer les listes de dépenses/recettes à un fichier texte qui serait enregistrer automatiquement puis réutilisé dès la réouverture du programme mais nous n'avons pas réussi à créer quelque chose de viable. De plus, nous avons créé un espace d'identification qui complique encore plus le projet puisque cela signifie qu'il faut associer des fichiers textes spécifiquement à chaque sessions créée. Cela nous a semblé un peu trop compliqué à réaliser en 3 semaines sans avoir particulièrement de base en Python. Cependant ce serait une très bonne amélioration pour ce projet.

De plus, nous voulions ajouter des scrolls dans les quatres frames du milieu afin de pouvoir s'adapter si jamais le nombre de ligne dépassait la taille des frames de base. Cependant, même en reprenant le code que nous avions vu en cours nous n'avons pas réussi à mettre en place un scroll qualitatif car dans le cas où il apparaissait il était sur la mauvaise frame et créait un nouveau scroll à chaque ligne ajoutée ce qui ne nous convenait pas et en changeant de frame, le scroll ne s'affichait pas et même en demandant de l'aide nous n'avons pas réussi à arriver à un résultat satisfaisant.
Nous avons donc cherché une autre solution en essayant de mettre en place une liste dans chaque frame (les 4 du milieux). Cependant, même en essayant de multiple chose, en cherchant sur internet et en demandant de l'aide nous n'avons pas réussi à obtenir un résultat qui fonctionnait. Nous n'avons pas réussi à faire en sorte qu'à chaque actualisation une ligne s'ajoute et se remplisse avec les informations que l'utilisateur avait rentré. Nous n'avons d'ailleurs pas compris pourquoi cela ne fonctionnait pas. Nous y avons passé des jours entiers, à plusieurs, sans réussir.

Lors de la phase de design de notre interface, nous avons essayé d'ajouter les thèmes comme vu en cours mais nous avons eu beaucoup de mal à les importer sur nos pc personnels et le résultat n'était pas forcément au rendez-vous. Nous nous sommes donc dit qu'il fallait mieux faire le design nous même afin de ne pas risquer que le design soit différent sur un autre ordinateur.