**Projet d’ISN – Dossier**

**Automatisation d’une expérience sur la mémoire**

*Zoé GUILLEMAIN – TS4
(travail sans binôme)
2016-2017*

**PRESENTATION DU PROJET**

**Problématique :** comment optimiser le déroulement d’une expérience scientifique grâce à une application ?

**But du programme :**
J’ai étudié l’année dernière une expérience scientifique pour tester l’efficacité de la technique du Palais Mental (ou lieux de mémoire). L’expérience consiste à lire deux listes de mots à un groupe de sujets ; la première doit être retenue sans utiliser de moyen mnémotechnique, la deuxième en utilisant le Palais Mental : imaginer un lieu ou trajet bien connu, s’y promener mentalement et s’imaginer y déposer les objets de la liste. On a remarqué que le nombre de mots retenus avec la méthode était beaucoup plus élevé que sans la méthode.

J’ai mené le test sur ma classe, ce qui a posé certains problèmes d’organisation. Entre autre, la gestion du groupe, le risque de falsification des résultats si les sujets communiquent, l’incertitude liée à la vitesse de lecture des mots, la récupération et l’analyse des résultats… C’est pourquoi j’ai décidé d’automatiser le test pour régler ces problèmes :
-chaque sujet peut faire le test indépendamment, au moment qui lui convient, sans qu’il y ai besoin de gérer un groupe turbulent
-les mots sont lus à la même vitesse pour les deux listes et pour tous les sujets
-les résultats sont enregistrés dans une base de donnée et traités sous forme d’histogramme par le programme, ils sont donc plus facilement appréhendables

**Compétences informatiques :**

*Dimension algorithmique :* création des listes, lecture mot par mot, enregistrement des mots retenus
*Eléments de programmation :* langage utilisé : Python ; IDE : Idle (en classe) et PyCharm (à la maison)
*Représentation de l’information :* affichage des textes explicatifs, listes, zones de textes et boutons ; affichage des résultats sous forme de listes et d’histogrammes 🡺 utilisation de la bibliothèque Tkinter

**Déroulement de l’expérience :**

Le premier écran propose deux options : passer le test, ou voir les résultats des tests précédents.

*Le test :* il y a d’abord un texte explicatif du déroulement. Puis une liste de dix mots est affichée à l’écran, chaque mot affiché puis effacé un par un.
Ensuite, il y a une étape perturbatrice : on pose au sujet plusieurs questions de calcul mental (cela sert à empêcher qu’il utilise la mémoire à court terme pour retenir les mots). S’il répond mal à une question, elle est reposée.
Une fois toutes les questions répondues correctement, on passe à la restitution des mots. Le sujet doit entrer un par un tous les mots dont il se souvient.
Puis il y a un nouveau texte explicatif, décrivant le moyen mnémotechnique du Palais Mental. Ensuite, les mêmes étapes recommencent (avec une liste de mots différents évidemment).
Après les deux tests successifs, on annonce au sujet ses résultats aux deux tests, et on lui demande des informations supplémentaires : son âge et son sexe (ces questions sont facultatives ; si le sujet n’y répond pas, la valeur attribuée à ces deux données sera ‘’non renseigné ‘’).

*Les résultats :* on propose deux présentations des résultats.
D’abord, sujet par sujet : pour chaque sujet on donne ses résultats aux deux tests, son sexe et son âge.
 Les résultats sont également présentés sous forme d’histogramme. On peut modifier les critères : tous les résultats indifféremment (comparaisons tests 1 et 2 : nombre de mots retenus en abscisse, nombre de personnes concernées en ordonnées), en fonction de l’âge (les résultats pour toutes les personnes de 15 ans, ou pour toutes les personnes de moins de 20 ans…), en fonction du sexe (résultats des femmes par rapport aux résultats des hommes).

Organigramme de l’application :

****

**Fonctionnement du programme :**

Il y a une fenêtre principale, sur laquelle se superpose plusieurs cadres, un pour chaque étape de l’expérience. On créé tous les cadres dès le début du programme, mais ils sont affichés seulement lorsqu’on appuie sur le bouton qui les appelle.
Dans le programme principal on définit les caractéristiques de chaque cadre : son nom, et les widgets qui sont dessus. Toutes les actions faites par le programme (affichage des mots, changement de cadre, enregistrement des données…) sont effectuées par des fonctions. Certaines fonctions (comme la création des listes) sont appelées dès l’ouverture du programme. Les fonctions gérant l’affichage sont toutes appelées par des boutons : comme la partie graphique (écran, cadres) n’est effectuée qu’à la toute fin du programme (avec le .mainloop() ), on ne peut pas faire d’action dans le programme principal.

Certaines fonctions sont appelées par plusieurs boutons différents : par exemple changerecran, qui permet de passer d’un cadre au suivant. Les autres sont appelées par un ou deux boutons au plus, car il s’agit d’une action qui n’est effectuée qu’une seule fois dans tout le programme (ou deux pour l’affichage des mots, l’étape perturbatrice et la restitution des mots retenus, puisqu’il y a deux tests successifs). La plupart contiennent des boucles (par exemple l’affichage des mots, ou l’enregistrement des résultats dans la base de données).

**Contraintes :**

Avant de commencer le projet je maitrisais peu Tkinter, et je ne savais pas utiliser de gestion du temps (faire attendre le programme un certain nombre de secondes avant de passer à l’étape suivante), ni les bases de données. J’ai utilisé des tutoriels sur des sites et des forums pour me renseigner sur ces trois aspects et fait quelques mini-projets pour m’entraîner avant de les intégrer au programme principal.

*Problème d’affichage :* pour chaque étape de l’expérience j’avais défini un écran Tkinter, qui était détruit lorsqu’on n’en avait plus besoin. Après avoir intégré au programme l’attente dans l’affichage des listes, j’ai été confrontée à un problème : IDLE effectuait tout le programme (dont les parties où on attend plusieurs secondes), PUIS affichait les écrans (puisque la commande fenetre.mainloop() se trouve à la fin).
J’ai réglé ce problème en créant un seul écran, sur lequel se plaçaient successivement plusieurs cadres, un par étape de l’expérience. Tous les cadres sont créés lors de l’ouverture du programme, mais sont caché. Pour passer d’un cadre à un autre, plutôt que de faire pack et destroy comme précédemment avec les fenêtres, les boutons de changement d’étape appellent une fonction qui fait cacher le cadre 1 et afficher le cadre 2 (avec pack et pack.forget ). De plus, toutes les parties demandant au programme d’attendre sont maintenant dans des fonctions, qui sont appelées seulement par des boutons (comme ça le programme n’attend pas à des moments inopportuns).

*Problème d’attente :* après avoir modifié le programme pour avoir des cadres plutôt que des fenêtres, et mis les parties d’attente dans des fonctions, j’ai eu un nouveau problème : la méthode .sleep ne fonctionnait pas comme je m’y attendais, elle bloquait tout le programme de manière permanente plutôt que de le mettre en pause quelques secondes. J’ai donc dû apprendre la méthode .after, qui était plus adaptée au fonctionnement de mon programme, mais plus compliquée. En effet, j’avais du mal à savoir à quel objet l’intégrer, et je ne savais pas qu’il fallait ajouter un .update pour qu’elle soit effective. La partie d’affichage mot par mot des listes m’a donc pris beaucoup de temps, mais je ne pouvais pas simplifier cette partie et contourner le problème, car le fait que les mots soient lus un par un et pas en bloc est le centre même du projet.

*Problème de restitution :* la restitution des mots retenus via un widget entry a nécessité d’apprendre à utiliser les objets StringVar, que je ne connaissais pas. J’avais commencé par utiliser des variables string normales, mais cela posait problème car seul l’entry pouvait modifier le string, et pas le contraire : on mettait le contenu de l’entry dans le string avant de l’ajouter à une liste, mais l’entry restait plein. Avec une StringVar, on peut récupérer le contenu de l’entry  et l’ajouter à la liste, puis effacer le contenu de la StringVar ; comme elle est liée à l’entry, ce dernier se vide aussi et on peut donc y entrer une nouvelle valeur.

*Stockage des données :* cette partie ne m’a pas posé de problèmes majeurs, à part le fait que j’ai dû apprendre à me servir des bases de données. L’intérêt d’utiliser une base donnée pour stocker les résultats, plutôt qu’un simple tableau dans le programme, est qu’on conserve les données lorsque le programme est fermé. De plus, pour le traitement des résultats, on peut demander à récupérer des données répondant à certains critères (par exemple : les résultats de toutes les personnes de moins de 20 ans), et l’ordinateur les trouve tout seul ; plutôt que de devoir écrire soi-même une fonction servant à trier la base de donnée avant d’afficher les résultats. On peut également récupérer la moyenne ou la médiane d’une liste de valeurs (mais je n’ai pas eu besoin de cette option)

*Histogrammes :* pour tracer les histogrammes j’ai dû exprimer les coordonnées de chaque rectangle en fonction du résultat (nombre de mots retenus) et de l’effectif pour ce résultat. Je n’ai pas eu de problème avec l’ordonnée (l’effectif), mais j’ai eu du mal à trouver une manière d’exprimer les abscisses en fonction du nombre de mots retenus, sans avoir de décalage ou de rectangles superposés (puisque je devais mettre trois rectangles pour chaque résultat : le test 1, le test 2 et un espace blanc). J’ai fini par faire une boucle appelant une fonction qui traçait les trois rectangles à la suite, plutôt que un seul rectangle à la fois comme j’avais initialement prévu ; ainsi, les coordonnées du deuxième rectangle et de l’espace blanc dépendent de celles du premier rectangle, et pas des résultats seuls (ce qui aurait été ingérable).

**Organisation du temps de travail :**

J’ai travaillé majoritairement chez moi car cela me permettait de travailler pendant plus longtemps sans m’interrompre. J’ai commencé par préparer une petite présentation du projet que j’ai proposée à mon professeur : le but de l’application, l’enchaînement des étapes et une maquette des écrans. Ensuite j’ai écrit les principales fonctions permettant de passer le test : la création des listes, la lecture des listes, la restitution des mots retenus, la comparaison des mots retenus avec ceux lus, et l’enregistrement des résultats. Chaque fonction était écrite dans des fichiers séparés. J’ai fait cela surtout pendant les vacances et les week-ends, et j’ai mis à profit les séances d’ISN pour demander de l’aide à mon professeur, ou rédiger des parties de code qui ne nécessitaient pas de recherche ou de brouillons.
Ensuite, j’ai bâti le programme principal : j’ai mis toutes les fonctions dans le même fichier, téléchargé les modules (Tkinter, randrange, etc), et défini les écrans ; cette partie devait être faite après avoir créé les fonctions, puisque les boutons et labels appellent les fonctions. Néanmoins j’ai dû modifier les fonctions au fur et à mesure, pour les adapter au fonctionnement de Tkinter (ce que je n’avais pas pensé à faire dès le départ).
J’ai fait la partie d’exploitation des résultats en dernier, car elle n’est pas nécessaire au fonctionnement du reste de l’application, et nécessite des compétences différentes de celles mises en œuvre pour la partie expérience : une base de donnée SQLite et les dessins graphiques de Tkinter.

**Ressources documentaires**

J’ai utilisé des tutoriels sur Internet, en particulier les sites apprendre-python.com, stackoverflow.com, et le manuel de Python et de Tkinter.

Tkinter :
<http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel>
<http://fsincere.free.fr/isn/python/cours_python_tkinter.php>

Gestion de temps :
<http://stackoverflow.com/questions/25753632/tkinter-how-to-use-after-method>

SQLite :
<http://apprendre-python.com/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite>
<http://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html>
<http://www.sqlfacile.com/apprendre_bases_de_donnees/count_et_group_by_1>

J’ai également reçu de l’aide de mon professeur d’ISN, Monsieur Tamby, en particulier pour le problème d’attente du programme (la méthode .after) et pour la création de l‘histogramme ; et de ma mère informaticienne pour gérer la base de donner SQLite, découvrir l’IDE PyCharm, et pour l’organisation générale du programme principal et l’optimisation des méthodes de programmation.

**Analyse de la dimension sociétale**

**Auto-évaluation finale individuelle**

J’ai travaillé seule sur ce projet, il n’y a donc pas eu d’aspect gestion d’équipe/des compétences de chacun.
J’ai appris à utiliser différents outils de Python, en particulier l’interaction graphique avec l’utilisateur à travers de Tkinter, et la gestion de données avec SQLite. Mais j’ai surtout développé une méthode de travail en informatique, car mener un projet long comme celui-ci demande plus d’organisation et de planification que les TDs faits en classe.