BEN HAMOUDA Amel
PEMODJO Marylin
TD a TP2

Rapport du projet d'informatique

Projet : Solveur pour « Des chiffres et des lettres »

Tables des matières

I. Présentation du jeu :

Le déroulement du jeu est le suivant : deux candidats s'opposent sur plusieurs types d'épreuves pour obtenir le meilleur score possible (140 au maximum).
Voici la liste des épreuves :

- *Le Compte est bon*
- *Le Mot le plus long*
- *Les duels*
- *Le sprint final*

Ce qui va nous intéresser sont le compte est bon et le mot le plus.

- Le compte est bon :

Le but de cette épreuve est d'obtenir un nombre à partir d'opérations élémentaires (Addition, Soustraction, Multiplication, Division) sur des entiers naturels, en partant de nombres tirés au hasard.

- Le mot le plus long :

Dans cette épreuve, les deux candidats décident chacun leur tour s'ils souhaitent une voyelle ou une consonne soit tirée au sort, jusqu'à obtenir 10 lettres. Le règlement impose qu'il y ai au moins deux voyelles dans le tirage de lettres.
Le but est de trouver le mot le plus long possible en utilisant les lettres qui ont été tirées.

II. Objectif du projet :

Le projet a pour objectif la création d'un programme permettant de trouver les solutions du « compte est bon » et du « mot le plus long »avec plusieurs conditions. Telles que les fonctions imposées sur le sujet et pour certaines fonctions on ne peut pas avoir de complexité quadratique ou plus.

L'objectif du projet était en soit assez simple. Il fallait créer un solveur à partir des compétences que nous avions acquises en cours (TD, TP et Amphithéâtre) et en suivant les consignes qui étaient indiquées sur le sujet qui nous a été fourni.

III. Elaboration du projet :

A. Répartition du travail

Pour la répartition de travail, on a choisi de travailler ensemble, mais chacune a fait une fonction pour les lettres et pour les chiffres on a tout fait ensemble. On peut quand même dire que l'on a tout fait ensemble puisqu'à chaque fois que l'une d'entre nous finissait une fonction, elle l'envoyait à l'autre.

B. Organisation des programmes

Le programme est scindé en deux. Un fichier contenant tous les programmes correspondant aux fonctions pour l'épreuve « le mot le plus long », ainsi qu'un deuxième fichier qui contient toutes les fonctions pour réaliser l'épreuve « le compte est bon ».

Pour chaque épreuve, le fichier est composé d'un programme principal qui permet d'afficher un résultat particulier lorsque la ligne de commande python3 lettres.py ou python3 chiffres.py est effectuée.

Dans les deux cas, aucune invite de saisie, pas d'annonce des résultats n'est affiché.

C. Rôle des fonctions

La plupart des fonctions ont étaient créées sans difficulté particulières, tel que les fonctions pour l'épreuve « le mot le plus long ».

1. Le mot le plus long

- La fonction ***mot_possible()*** prend en argument un *tirage* composé de lettres et un *mot*. Elle a pour but de renvoyer *True* si le mot peut être composé avec le tirage et *False* dans le cas contraire.

  Pour réaliser cette fonction, on réalise trois tests :
  - Si la longueur du mot est supérieure au tirage,
  - Si le nombre d'occurrence de chaque lettre du *mot* est identique à celui du *tirage,*
  - Si toutes les lettres du *mot* sont dans le tirage.

  Si l'un des tests est vérifié alors on renvoie *False ;* dans le cas contraire, on renvoie *True.*

- La fonction ***tous_mots_possibles()*** prend une *liste* et un *tirage* en argument. Elle renvoie une liste qui contient uniquement les mots de la *liste* donnée en argument qui peuvent être formés à partir du *tirage.*

  Après avoir inséré chaque lignes du fichier (dico) dans une liste, on parcourt cette même liste à l'aide d'une boucle for, en regardant, pour chaque mot de la liste, si ce mot peut être formé par le *tirage*. Si c'est le cas, on l'ajoute à une autre liste. Une fois la liste de mots parcourue entièrement, on renvoie la liste contenant tous les mots qui peuvent être crée par le *tirage*.

- La fonction ***tri_liste()*** prend une *liste* en argument. Son objectif est de trier cette liste par ordre croissant et de la renvoyer ensuite. Pour cela, on utilise la méthode tri bulle vu en cours.

- Pour le **programme principal**, la chaîne entrée par l'utilisateur est récupérée puis transformée en liste. On fait ensuite appelle à la fonction ***tous_mots_possibles()*** sur cette liste. Puis on affiche le résultat trié.

2. Le compte est bon

- La fonction ***operations_possibles()*** doit renvoyer une liste de tuples (x, op, y, res) où *x* et *y* sont des valeurs du tirage, *op* le caractère correspondant à une opération (+, -, *ou/) et *res* est le résultat.

  On utilise deux compteurs, le premier qui parcourra la liste une seule fois et qui mettra fin à la boucle lorsque la longueur de la boucle sera atteinte et le second qui parcourra également la liste de façon répétée.

  On effectue alors différents tests après avoir vérifié que chaque valeur est non nulle :
  - Addition : si l'addition entre deux éléments est non nulle, alors on ajoute le tuple correspondant à la liste.
  - Multiplication : on vérifie au préalable que l'on ne multiplie pas par 1. Si ce n'est pas le cas, on ajoute le tuple correspondant.
  - Si la valeur à la position du premier compteur *x* est supérieure à celle du deuxième compteur *y*, alors on s'occupe que de la soustraction et la division :
    - Si le résultat de *x – y* n'est pas nul, alors on ajoute le tuple correspondant.
    - Si y est différent de 1, alors on vérifie si la division de *x* par *y* est bien un entier. Si c'est le cas et que le résultat n'est pas nul, alors on ajoute le tuple correspondant.

- Si la valeur à la position du premier compteur est inférieure à celle du deuxième compteur, alors on s'occupe également que de la soustraction et de la division :
    - Si le résultat de *y - x* n'est pas nul, alors on ajoute le tuple correspondant.
    - Si x est différent de 1 alors on vérifie si la division de *y* par *x* est bien un entier. Si c'est le cas et que le résultat n'est pas nul, alors on ajoute le tuple correspondant.

Tous ces tests sont effectués tant que la liste n'a pas été totalement parcourue. Une fois fait, on renvoie la liste qui contient tous les tuples.

- La fonction ***suites_possible()*** prend en argument un *tirage* et doit renvoyer une liste de couple *(suite_d_operations, nombres)*. Cette fonction a pour but de renvoyer toutes les opérations possibles à partir d'un *tirage* avec le *tirage* modifié.
  Pour cette fonction, on a eu beaucoup de difficultés puisque qu'il nous a été imposé de faire cette fonction récursivement. Pour cela, on a créé une liste qui correspond à toutes les opérations possibles sur le *tirage* et une liste vide qui prendra le résultat à renvoyer.
  On commence par donner la condition d'arrêt, si le tirage contient au moins deux nombres, le programme s'arrête.
  Sinon, on regarde pour chaque opération possible et on a créé un nouveau tirage qui prend au départ le *tirage*, puis on calcule la liste de nombre obtenue en effectuant l'opération à l'aide de deux boucles for pour vérifier si les valeurs du tirage sont dans l'opération.
  Ensuite, on cherche toutes les suites possibles à partir de cette nouvelle liste, puis on déduit la liste des suites possibles depuis le tirage courant sans oublier d'inclure la suite déjà obtenue.

- La fonction ***solutions()*** prend en argument deux éléments *tirage* et *cible*. Cette fonction a pour but de trouver toutes les opérations possibles à partir d'un *tirage* qui ont pour *cible* comme résultat.
  Pour cela, on a créé une liste qui correspond à toutes les opérations possibles sur le *tirage.* On parcourt cette liste avec une boucle for. Pour chaque élément de la liste (des tuples), on accède au premier élément du tuple, puis on vérifie si le dernier nombre du dernier tuple qui est le résultat des opérations précédents est identique à la *cible.*
  Si c'est le cas, alors on ajoute à une autre liste (vide au départ) la liste des tuples. Pour le trie de la liste, on utilise la méthode de tri bulle vu en cours.

- Pour le **programme principal**, on récupère la chaîne que l'utilisateur entre.
  Grâce à une boucle, on transforme cette chaîne en liste, ensuite on récupère le résultat que l'utilisateur veut obtenir. On a créé une liste qui sera renvoyée par la fonction ***solutions(),*** utilisé avec la chaîne entrée par l'utilisateur, transformée en liste et le résultat voulu.
  Avec une boucle for, on parcourt chaque élément de la liste et on les ajoute au fur et à mesure à une chaine de caractère en ajoutant « = », « » ou « \n » en fonction de l'élément. Une fois la chaîne remplie, on l'affiche.

D. Difficultés rencontrées

La seule difficulté rencontrée est la fonction ***suites_possibles()*** dans chiffres.py, puisque l'on devait la faire récursivement. Nous devions insérer le tirage modifié à chaque nouvelle opération, sans oublier d'afficher dans une même liste toutes les opérations effectuer pour arriver à un certain tirage et ajouter à la fin de cette liste le tirage modifier.

C'est la seule fonction qui nous a pris autant de temps, puisqu'on a travaillé dessus durant plusieurs jours.

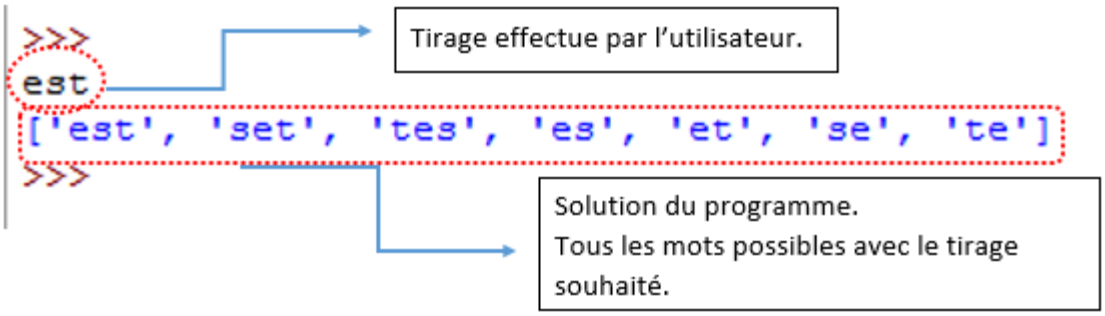
Pour surmonter cette difficulté, on a dû demander de l'aide à l'un des professeurs de TD par mail et échanger quelque idée avec des étudiants.

IV. Manuel utilisateur :

A. Le mot le plus long

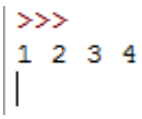
Il faut vérifier que vous avez le fichier dico.txt.

Pour commencer, il suffit d'ouvrir le fichier contenant le programme et de le lancer.
Lorsqu'on le lance, rien n'apparaît. C'est à ce moment-là où il faut taper le tirage souhaité sans espace.
Et là, on voit apparaître la liste des solutions par rapport au tirage entrée au début.

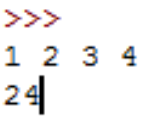


B. Le compte est bon

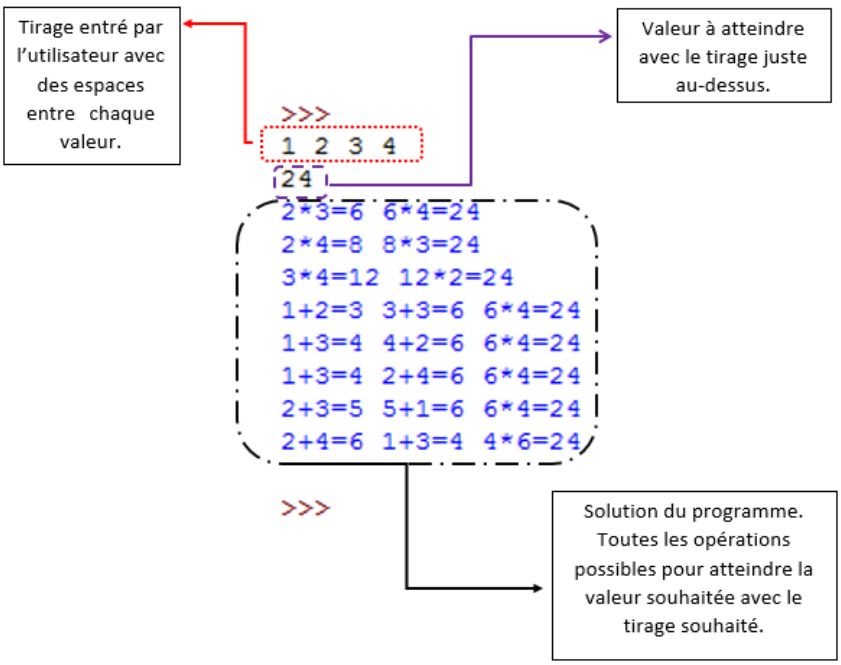
Pour commencer, il suffit d'ouvrir le fichier contenant le programme et de le lancer.
Lorsqu'on le lance, rien n'apparait. C'est à ce moment-là qu'il faut taper le tirage souhaité avec un espace après chaque valeur.
Après avoir fini de taper le tirage souhaiter, appuyer sur ENTRER (sur le clavier).



Pour la deuxième fois, rien n'apparaît. C'est à ce moment-là qu'il faut tapez le résultat souhaité (cible).



Après, tapez sur ENTRER comme précédemment.
Et là, apparaît toutes les opérations possibles pour arriver au résultat souhaité.



V. Conclusion :

Comme vous avez pu le constater tout au long de ce rapport, la seule difficulté rencontrée pendant la réalisation de ce projet, en a fait plus ou moins l'intérêt.

En effet, on a pu apprendre de nombreuses notions et appréhender des techniques vues en cours.

Néanmoins, nous ne sommes pas parvenues à faire les améliorations (demander) par manque de temps.

VI. Annexe :

Fichiers à trouver dans le rendu :

- lettres.py
- chiffres.py
- dico.txt
- rapport.pdf