Initiation à la programmation – TP8 – 20 ou 21 octobre 2020

Ce TP est à terminer après la séance en présentiel.

Exercice : Jeu de mémoire type "Simmon"

Vous allez écrire en Python un programme pour un petit jeu de mémoire type "Simmon". L'ordinateur propose des chaînes composées de 0 1 et 2 de plus en plus longues (on rajoute un caractère à chaque tour), quand l'utilisateur a mémorisé la chaîne il appuie sur une touche, le programme saute 50 lignes (pour que la chaîne ne soit plus visible à l'écran) et l'utilisateur doit recopier la chaîne de mémoire. Le programme lui affiche bravo et continue ou alors s'arrête en cas d'erreur en indiquant le nombre de réponses justes.

1. Èrire la fonction saut(n) qui étant donné un entier n effectue un saut de n lignes à l'écran.
2. Èrire la fonction alea() qui renvoie un caractère au hasard parmi "0", "1" et "2".
3. Èrire la fonction affiche(ch) qui affiche successivement chacun des caractères de la chaine de caractères ch séparés par des espaces et sans aller à la ligne.

```
affiche("01112001")
0 1 1 1 2 0 0 1
```

4. Èrire la fonction SansEspaces(ch) qui étant donnée une chaîne de caractères renvoie la chaîne recopiée sans les espaces.

```
SansEspaces("  un ex emple ici  ")
"unexempleici"
```

5. Èrire la fonction de jeu. Le programme doit fabriquer au fur et à mesure une chaîne , l'afficher, faire le saut de ligne demander et juger la réponse. La chaîne construite ne contiendra pas d'espaces, la chaîne saisie par l'utilisateur pourra en contenir ou non on les supprimera avec la fonction SupprimeEspaces. Le jeu s'arrête quand l'utilisateur s'est trompé et le programme renvoie le nombre de bonnes réponses de l'utilisateur.

   Exemple de déroulement du programme (sans les sauts de lignes pour plus de visibilité).

```
1
vous avez memorise? tapez sur une touche pour continuer



votre reponse
1
Bravo on continue

1  2
vous avez memorise? tapez sur une touche pour continuer
```

```
votre reponse
1 2
Bravo on continue


1  2  0
vous avez memorise? tapez sur une touche pour continuer



votre reponse
120
Bravo on continue



1  2  0  0
vous avez memorise? tapez sur une touche pour continuer



votre reponse
1200
Bravo on continue



1  2  0  0  0
vous avez memorise? tapez sur une touche pour continuer



votre reponse
120000
dommage c'est perdu au bout de 4 coups gagnants
4
>>>
```

6. Enchainement de parties : écrire un programme qui permette d'enchaîner les parties tant que l'utilisateur le souhaite : à la fin d'une partie on affichera le record en cours (nombre de coups gagnants maximum des parties déjà jouées) et on demander à l'utilisateur s'il veut continuer.
7. Et ensuite ? on peut proposer des variantes (plus de caractères, challenges divers .... soyez imaginatifs !)

Exercice : Jeu des allumettes

Il s'agit de réaliser un programme en python, pour jouer au jeu des allumettes ou jeu de Nim . Ce jeu très connu et dont il existe de nombreuses variantes se jouera de la manière suivante : on dispose d'un certain nombre d'allumettes. Chacun son tour, un des deux joueurs prend 1 , 2 ou 3 allumettes selon son choix. Celui qui est obligé de prendre la dernière allumette perd la partie.
Il est demandé de faire un programme pour jouer à ce jeu avec deux joueurs "humains".

1. Écrire la fonction affiche(n) : cette fonction qui prend en paramètre un entier n doit afficher sur une ligne n étoiles séparées par des espaces puis aller à la ligne.
2. Écrire la fonction min(a,b) qui prend en paramètre deux entiers renvoie le plus petit des deux.
3. Écrire la fonction saisie(a,b) qui prend en paramètre deux entiers , demande à l'utilisateur de saisir un entier entre a et b (compris) et recommence tant que la réponse n'est pas satisfaisante. Si la réponse est satisfaisante, l'entier est renvoyé. On supposera que l'utilisateur saisie effectivement un entier et on ne testera que le fait qu'il soit compris entre a et b.
4. On fera ensuite le programme principal. On jouera avec 21 allumettes au départ. Le programme devra afficher le joueur dont c'est le tour, l'état du jeu, etc.. comme sur l'exemple ci-dessous. On appellera les joueurs joueur1 et joueur2. A tout moment un joueur doit choisir comme nombre d'allumettes un nombre entre 1 et le minimum de nb-1 et de 3 si nb est le nombre d'allumettes restantes.

   Exemple de jeu :

```
* * * * * * * * * * * * * * * * * * * * *
joueur 1 combien d'allumettes, saisir un entier entre 1 et 3: 6
saisir un entier entre 1 et 3:  3

* * * * * * * * * * * * * * * * * *
joueur 2 combien d'allumettes, saisir un entier entre 1 et 3: 3

* * * * * * * * * * * * * * * *
joueur 1 combien d'allumettes, saisir un entier entre 1 et 3: 3

* * * * * * * * * * * * *
joueur 2 combien d'allumettes, saisir un entier entre 1 et 3: 3

* * * * * * * * * *
joueur 1 combien d'allumettes, saisir un entier entre 1 et 3 :3

* * * * * *
joueur 2 combien d'allumettes, saisir un entier entre 1 et 3: 3

* * *
joueur 1 combien d'allumettes, saisir un entier entre 1 et 3: 1

* *
joueur 2 combien d'allumettes, saisir un entier entre 1 et 2: 3

mauvaise saisie , saisir un entier entre 1 et 2: 1

*
victoire du joueur 2
```

5. Bonus : faire une version où un joueur humain joue contre l'ordinateur.

Exercice : Exercices sur les mots (Python)

Cet exercice porte sur des listes de mots, chaque mot étant lui même une chaîne de caractères. On rappelle que len(chaîne) renvoie le nombre de caractères de la chaîne. Un exemple d'utilisation pour toutes les fonctions est donné à la fin de cet énoncé.

1. Écrire la fonction `compte_mots(n, listemots)` qui prend en argument une liste de mots et un entier et compte puis renvoie le nombre de mots de n lettres ou plus dans la liste.
2. Écrire la fonction `commence_par(carac, listemots)` qui prend en argument une liste de mots et un caractère carac et renvoie la liste des mots commençant par le caractère carac.
3. Écrire la fonction motpluslong(listemots) qui étant donnée une liste de mots listemots, renvoie un des mots qui a le plus de lettres dans la liste(si les mots les plus longs de la liste comportent 8 lettres, on renverra n'importe quel mot de 8 lettres)
4. Écrire la fonction petitelongueur(listemots) qui renvoie la longueur des mots les plus petits de listemots.
5. Écrire la fonction pluspetitsmots(listemots) qui étant donnée une liste de mots listemots, renvoie la liste de tous les mots qui ont le moins de lettres dans la liste de mots.
6. Écrire la fonction `recopie(mot)` qui prend en argument une chaîne de caractères mot , qui affiche ce mot et demande à l'utilisateur de le recopier jusqu'à ce que la recopie soit exacte. À ce moment-là le programme doit renvoyer le nombre d'essais c'est-à-dire le nombre de mots saisis. A chaque tentative le programme affichera c'est bien ou ce n'est pas correct.
7. Utiliser cette fonction pour écrire la fonction `test(listemots)` qui prend en argument une liste de mots et va demander à l'utilisateur de recopier successivement chaque mot jusqu'à ce ce que la recopie soit exacte. À la fin du programme, le nombre de mots demandés et le nombre total d'essais (et donc le nombre de fautes qui est la différence des deux) devra être affiché.
8. On va maintenant améliorer cet entraînement. On va d'une part limiter à 5 essais maximum le nombre d'essais pour chaque mot. Par ailleurs pour éviter des fautes liées à des espaces ou des majuscules/minuscules on va supprimer tous les espaces saisis et mettre les mots saisis en minuscules.
   — Ecrire la fonction supprime_avant(chaine) qui renvoie la chine obtenue en supprimant tous les espaces au début de la chaine
   — Ecrire la fonction supprime_apres(chaine) qui renvoie la chine obtenue en supprimant tous les espaces à la fin de la chaine
   — Modifier le programme .

```
>>> listemots=["citron", "orange", "carambole", "banane",
"ananas","kiwi", "poire"]
>>> compte_mots(6, listemots)
5
>>> commence_par("c", listemots)
['citron', 'carambole']
>>>motpluslong(["ananas", "pomme", "poire", "carambole", "banane"])
'carambole'
>>>motpluslong(["ananas", "pomme", "poire", "kiwi", "banane"])
'ananas'
>>>petitelongueur(["ananas", "pomme", "poire", "carambole", "banane"])
5
>>>>>> pluspetitsmots(["ananas", "pomme", "poire", "carambole", "banane"])
['pomme', 'poire']
>>>pluspetitsmots(["ananas", "pomme", "poire", "kiwi", "banane"])
['kiwi']


>>> recopie("orange")
recopie ce mot   orange
orenge
ce n'est pas correct
recopie ce mot   orange
```

```
oranje
ce n'est pas correct
recopie ce mot   orange
orange
3
>>> listemots=["citron", "orange"]
>>> test(listemots)
recopie ce mot   citron
citron
c'est bien
recopie ce mot   orange
orange
c'est bien
nombre de mots de l'exercice  2
nombre d'essais en tout  2
bravo, aucune faute
>>> listemots=["citron", "orange"]
>>> test(listemots)
recopie ce mot   citron
sitron
ce n'est pas correct
recopie ce mot   citron
cotroen
ce n'est pas correct
recopie ce mot   citron
citron
c'est bien
recopie ce mot   orange
orange
c'est bien
nombre de mots de l'exercice  2
nombre d'essais en tout  4
 vous avez fait  2 fautes
```