Devoir Maison n°2 d'Informatique

pour Jeudi 5 Décembre 2024

Exercice 1 : jouer au Mastermind

Voici un jeu très classique : l'ordinateur a en mémoire un mot de 4 lettres. Le joueur doit le deviner en proposant des mots à l'ordinateur, qui lui indique à chaque fois le nombre de lettres bien placées et le nombre de lettres correctes mais mal placées.

1) Version Allégée

Commençons par mettre au point un premier programme qui teste le nombre de lettres de l'entrée, puis indique le nombre de lettres bien placées. Voici les consignes :

❶ Le mot à deviner sera stocké dans une variable `mot` en début de programme.

❷ Tant que le mot proposé ne possède pas quatre lettres, l'ordinateur réitère la demande.

❸ Il compte ensuite les lettres bien placées et renvoie leur nombre. Il faudra bien entendu que le programme soit insensible aux majuscules comme aux minuscules.

❹ On recommence jusqu'à ce que le joueur trouve le bon mot.

❺ Pour éviter les crises de nerfs, il faut prévoir une clause de sortie : si l'utilisateur rentre le mot STOP, le programme se termine.

Testez votre programme, en prenant comme mot à deviner LAVE et en proposant en entrée LOUP, PALE, LEVA.

2) Version Kouign Amann

Il faut améliorer l'algorithme précédent afin de déterminer le nombre de lettres correctes mais mal placées. C'est plus délicat, surtout quand la même lettre se répète.

❶ Le plus simple est d'utiliser une liste `l_m` contenant les lettres du mot à trouver, et une autre liste `l_e` contenant celles du mot fourni en entrée.

❷ Quand on trouve une lettre bien placée, on l'élimine simultanément des deux listes.

❸ Ensuite, les lettres de `l_e` qui appartiennent à `l_m` sont des lettres correctes mais mal placées. Il faudra les éliminer de `l_m` au fur et à mesure, pour éviter les problèmes engendrés par les répétitions de lettres.

Modifier votre programme en conséquence. Pour vous assurer qu'il fonctionne correctement, vous pouvez prendre comme mot à deviner CHAT et proposer en entrée TATA puis SHAH.

Indications :

① *Se servir de boucles* `while` *pour réitérer les demandes.*

② *L'instruction* `s.upper()` *permet de passer en majuscules une chaîne de caractères* `s`*.*

③ *Utiliser des compréhensions de liste pour créer les listes* `l_m` *et* `l_e`*.*

④ *On pourra employer la commande* `liste.remove(lettre)` *afin de supprimer la lettre* `lettre` *de la liste* `liste`*.*

Exercice 2 : écriture d'un entier en base b

Pour convertir un entier n en base b, on effectue des divisions euclidiennes successives par b en reprenant les quotients, les restes donnant les chiffres de la représentation de n en base b écrits **de droite à gauche**. On s'arrête alors lorsque l'on obtient un quotient nul.

Remarque : cet algorithme a été exposé dans le chapitre 2 de votre excellent cours d'informatique.

Partie A : préliminaires

1) Tapez dans un console `iPython` les instructions suivantes :

```
a=3    str(a)    str(100)    str(4)+str(21)    str(21)+str(4)
```

Qu'observe-t-on ? Que fait donc la commande `str()` ?

2) On dispose de deux variables `a=8` (type `int`) et `b='251'` (type `str`).
Quelle instruction faut-il taper pour obtenir `c='8251'` à l'aide de `a` et `b` ?

Partie B : Un exemple pour se faire la main

1) On va commencer par s'amuser à convertir 13 en base 2 « à la main ».

❶ On définit initialement `n=13` et `resultat=''`.

❷ À chaque étape, on détermine le quotient `q` et le reste `r` de la division de `n` par 2.

❸ Le chiffre `r` est ajouté **à gauche** de la chaîne de caractères `resultat`.

❹ La valeur de `q` est ensuite affectée à la variable `n`, et l'on recommence.

Recopier et compléter le superbe tableau ci-dessous :

	n	q	r	resultat
Initialisation	13			''
Étape 1	13 6	6	1	 '1'
Étape 2	6 3	3	0	 '01'
Étape 3				
Étape 4				

Vous pouvez recommencer avec un nombre un peu plus compliqué et une autre base si vous n'êtes pas complètement à l'aise avec le fonctionnement de l'algorithme ... car il va falloir réussir à le programmer dans les parties suivantes !

2) Voyons maintenant comment programmer cela. Que faut-il taper pour :
   (a) stocker le quotient et le reste de la division euclidienne de `n` par 2 dans deux variables `q` et `r` ?
   (b) ajouter le chiffre `r` **à gauche** de la chaîne `resultat` ?
3) Pour réitérer ces différentes opérations, on va devoir utiliser une boucle `while`.
   Quelle sera la condition d'arrêt ? *Pensez à regarder le tableau ...*
4) Écrire alors un programme `Python` qui utilise les variables `n`, `q`, `r`, `resultat` et qui renvoie à la fin l'écriture de 13 en base 2.

Partie C : Généralisation du programme

Modifier le programme précédent afin qu'il :

❶ demande un nombre entier `n` à l'utilisateur ;

❷ demande la base `b` à l'utilisateur ;

❸ affiche l'écriture du nombre `n` en base `b`.