TP d'informatique : jeu du pendu

PCSI 2021 – 2022

Le but du TP est de programmer un jeu du pendu. Un utilisateur doit deviner un mot et propose des lettres. Si la lettre n'appartient pas au mot, on lui compte une erreur, sinon on lui affiche le mot avec la lettre au bon endroit.

Vous aurez besoin de la fonction « input » de python qui

- Prend en argument une chaine de caractère et l'affiche à l'écran,
  Attend que l'utilisateur tape une chaine de caractère sur le clavier puis tape entrée,
- Renvoie la chaine de caractère tapée par l'utilisateur.

Testez input dans quelques cas simples pour vous familiariser avec son utilisation.

Il est conseillé de définir des étapes simples. Par exemple on pourra commencer avec le même mot à deviner à chaque fois, puis si le programme marche, choisir un mot aléatoirement parmi une liste.

Essayez de définir vous même les étapes et d'être autonome. N'hésitez pas à faire des fonctions pour rendre votre programme plus clair.

Si vous bloquez, une proposition de démarche est présente dans les pages suivantes.

Il faudra afficher des étoiles à la place des lettres non devinée et les lettres à leur « bonne position » si elles ont déjà été proposée. Par exemple si le mot à deviner est `bonjour`, le programme devra afficher :
`*******` au début lorsqu'aucune lettre n'a été proposée
`*o**ou*` si les lettres o, u et z ont été proposées.

- Enregistrez régulièrement votre travail et lorsque vous avez une version qui marche, faites une sauvegarde avec un autre nom.

Une fois qu'un pendu simple marche avec un seul mot, on suggère les améliorations suivantes (dans l'ordre qui vous plait)

1. mesurer le nombre d'erreur et le donner à l'utilisateur.
2. choisir un mot aléatoirement (parmi une liste prédéfinie de mots rentrées dans le programme ou dans un fichier). On pourra regarder le module `random`.
3. mesurer le temps mis par l'utilisateur pour trouver. On pourra utiliser le module `time`.
4. proposer de jouer en boucle tant que l'utilisateur ne dit pas qu'il veut arrêter.
5. afficher un pendu qui se dessine au fur et à mesure des erreurs.
6. ... Toutes les autres bonnes idées sont les bienvenues.

```
      _______________
      | /           |
      |/            0
      |            /|\
      |            / \
      |
     /|     PERDU ...
===========================
```

Ne pas tourner la page (sauf si vous bloquez vraiment)

I Apprivoiser input

1. tester la fonction input avec par exemple l'instruction `r = input("Entre du texte ici puis appuie sur entrée : ")`. Que contient la variable r ? Testez dans le cas où vous tapez "1", puis faites $3 * r$. Comment manipuler des nombres ?
2. programmer un jeu où l'utilisateur doit deviner un nombre entre 1 et 100 que vous aurez préalablement défini. Tant que l'utilisateur ne trouve pas lui reposer la question en lui indiquant si le nombre est trop grand ou trop petit. (Attention au type des objets : une conversion est nécessaire vu qu'input renvoie une chaine.)

II Gérer les lettres et les *

Une difficulté de ce jeu en python est que les chaines ne sont pas mutables. Il faut malgré tout gérer l'affichage du mot avec les étoiles qui change en fonction des lettres rentrées.
Il y a plusieurs solutions, à vous de choisir la votre. On peut citer par exemple :

— Mémoriser le mot avec des étoiles dans une liste, et lors de l'affichage concaténer les différents éléments de la liste. Par exemple si `mot_secret = "informatique"` alors on peut créer
`liste_secret = ['i','n','f','o','r','m','a','t','i','q','u','e']`
`liste_trouvee = ['*','*','*','*','*','*','*','*','*','*','*','*']`
puis changer la liste trouvée en fonction les lettres proposée, par exemple si l'utilisateur propose la lettre i, il faudra changer la liste en
`liste_trouvee = ['i','*','*','*','*','*','*','*','i','*','*','*']` et on s'arrête lorsque les deux listes sont égales.

— On peut aussi mémoriser les lettres proposée dans une liste et reconstituer le mot trouvé à chaque nouvelle lettre proposée :
`mot_secret = "informatique"`
`lettre_proposee = ["i","f"]`
alors je parcours le mot informatique lettre par lettre et si la lettre est dans la liste proposée alors j'ajoute la lettre sinon j'ajoute une étoile : cela formerait le mot
`mot_trouve = "i*f*****i***"` et on s'arrête lorsque `mot_secret = mot_trouve`

— On peut aussi modifier le mot en utilisant la concaténation comme cela a été vu en cours pour corriger une faute d'orthographe.

Dans tous les cas il faudra modifier un mot (celui avec les "*") en fonction d'un autre (le mot secret). Il est donc conseillé de parcourir le mot secret en utilisant les indices (range).

III Condition d'arrêt

Il faut continuer le jeu tant que l'utilisateur n'a pas gagné ou alors que l'on a atteint un nombre de lettre proposée fausse tel qu'il a perdu. C'est donc une bonne idée d'utiliser une boucle while en s'arrêtant si l'une ou l'autre des conditions est vérifiée.

Ne pas tourner la page (sauf si vous bloquez vraiment)

IV Démarche détaillée si vous n'y arrivez vraiment pas tout(e) seul(le)

On va d'abord écrire le contenu de la boucle puis le mettre dans une boucle tant que.

1. Avant la boucle

Avant la boucle, définir une variable `secret` contenant le mot secret (sous forme d'une liste de lettre) et une variable `affiche` qui sera le mot affiché et qui contient pour le moment une liste d'étoile (autant que de lettres dans le mot secret, comme décrit dans la première proposition dans le II).
Définir une variable `nombre_erreur` qui comptera le nombre de lettres proposées ne faisant pas partie du mot secret.

2. Ce que l'on mettra dans la boucle

Ne pas mettre de boucle pour le moment.

1. demander à l'utilisateur d'entrer une lettre et l'enregistrer dans une variable.
2. tester si la lettre appartient ou non au mot secret puis
   (a) si elle n'appartient pas au mot, incrémenter `nombre_erreur` de 1 et indiquer à l'utilisateur qu'il s'est trompé.
   (b) si elle appartient au mot, parcourir le mot à l'aide d'une boucle utilisant les indices (range)
      i. pour chaque indice tester si la lettre de secret est égale à la lettre rentrée, si c'est le cas, modifier `affiche`, sinon passer à la lettre suivante (ne rien faire).
3. Afficher le contenu de `affiche` (dans un premier temps sans faire attention à la présentation, dans un deuxième temps en le présentant correctement pour ne plus avoir les "[]" et les "," dus à la structure de liste).

Tester cette partie du programme (si ce n'est pas déjà fait).

3. Le mettre dans la boucle

Indenter tout ce qui a été écrit à l'étape précédente pour l'inclure dans une boucle while qui s'arrête lorsque l'utilisateur a trouvé le bon mot ou a fait un « trop grand » nombre d'erreur.