1 Guess da Numba !

On va écrire le jeux de "Trouve le nombre!". Pour ce faire, on doit décomposer le déroulement du jeu.

Déroulement

Ce jeu va avoir pour étapes:

1. Sélection du niveau.
2. Générer un nombre aléatoire 'secret'
3. Boucle de jeu
    i. Demander de faire un choix
    ii. Vérifier si le nombre est identique au secret
        - **Si oui**: fin du jeu
        - **Si non**:
            - Si plus petit dire "C'est plus grand"
            - Si plus grand dire "C'est plus petit"
4. Afficher le score
5. Demander si on refait une partie
    - **Si oui**: on revient en 1
    - **Si non**: fin du programme

Nous avons eu là une approche *par-dessus* ou *top-down*, c'est la partie **Design** , mais maintenant que nous devons coder, nous ne pouvons coder la partie "Boucle Jeu" sans avoir déjà codé "Demander de faire un choix"... Coder est donc *par-dessous* ou *bottom-up*.

Boucle de jeu

Nous allons commencer par le coeur de notre programme, la boucle de jeu. Pour ce faire, comme indiqué précédemment nous allons devoir nous occuper de ce qu'il y a à l'intérieur de cette boucle avant. Je propose de voir comment on vérifie qu'une chaîne de caractères est un nombre entier.

to_integer

Appelons la fonctions qui suit `to_integer`. Elle prendra en entrée:

```
s -- une chaîne de caractères
a -- un minimum
b -- un maximum
```

Donc aura la forme:

```
def to_integer(s, a, b):
    # À toi de jouer !
```

Elle retournera soit un entier, si la chaîne est un entier entre `a` et `b`, sinon `None`.

Outils utile: `isdigit()`. C'est la fonction qui indique si une chaîne est un entier, ou plutôt, uniquement composée de chiffres.

```
>>> '3'.isdigit()
True

>>> 'A'.isdigit()
False

# N'aime pas les flottants, il ya un '.'
>>> '2.0'.isdigit()
False

# N'aime pas les '-'.
>>> '-1'.isdigit()
False
```

get_input

Maintenant, passons à la fonction qui va demander un nombre et voir si il est valide.

Je propose la fonction `get_input` qui aura pour entrée:

```
a -- un minimum
b -- un maximum
```

Et qui retourne un entier entre `a` et `b`.

Il faut en effet que la boucle retourne obligatoirement un entier valide. Donc nous allons 'boucler' tant que ce n'est pas le cas. Cette fonction aura donc la forme suivante:

```python
def get_input(a, b):
    while True:
        # Écrire la suite.
```

```
Pour tout ce qui concerne la gestion des entrées utilisateur,
se référer à la partie "Input/Output" du fichier python.md
```

Pour rappelle, on utilise les fonction `print` pour afficher, `input` pour demander une entrée et potentiellement `format` pour formater une chaîne de caractères.

Pour vérifier que l'entrée est correct, je propose d'utiliser la fonction `to_integer` précédente.

game_loop

Passons maintenant à la boucle de jeu en tant que telle. La fonction `game_loop` prendra en entrée:

```
a -- un minimum
b -- un maximum
```

Les bornes de la sélection du nombre secret, et ne renverra rien.

On se rappelle que la boucle de jeu fais ceci:

```
1. Demander de faire un choix
2. Vérifier si le nombre est identique au secret
    - Si oui: fin du jeu
    - Si non:
        - Si plus petit dire "C'est plus grand" et revenir à 2
        - Si plus grand dire "C'est plus petit" et revenir à 2
```

Mais avant de lancer la boucle nous devons générer le nombre aléatoire, nous allons utiliser la fonction `randint` du module `random`:

```python
from random import randint # À placer en haut du fichier.

secret = randint(0, 100) # Nombre entier aléatoire entre 0 et 100
# 0 <= a <= 100
```

La fonction `game_loop` aura donc la forme suivante:

```python
def game_loop(a, b):
    # Initialiser le nombre secret et autres variables utiles.

    while True:
        # Faire ce qui est indiqué ci-dessus

    # Efface le terminal (voir plus bas).
    clear_terminal()

    # Le joueur a gagné, lui afficher un petit message gratifiant
    # avec le nombre secret et le nombre de coups qu'il/elle a mit
    # pour le trouver.
```

Je mets à disposition la fonction suivante (à copier-coller) qui permet d'effacer le terminal.

```python
import os # À écrire en haut du fichier

def clear_terminal():
    """
    Efface le terminal
    """
    # Pas grand chose à en dire, faut lire la documentation
    # pour savoir ça (ou faire un recherche en ligne), rien
    # qui ne soir nécessaire pour le moment.
    os.system('cls' if os.name == 'nt' else 'clear')
```

Boucle principale

Maintenant que l'on a la boucle de jeu, il nous faut écrire la boucle principale, ou plutôt celle du menu. Elle va faire ce qui suit:

1. Demander le niveau au joueur
2. Lancer une partie
3. Demander si le joueur veut en en refaire une.

Pour la `3.`, je propose de faire une nouvelle fonction (ENCORE? - Oui.)

get_yes_no

Elle ne prend rien en entrée et renvoie un booléen (`True` | `False`). Elle pose simplement la question 'Oui | Non ?' ou 'O|N ?', et traduit l'entrée utilisateur en `True` ou `False`.

Elle aura donc la forme suivante:

```
def get_yes_no():
    while True:
        # Écrire la suite.
```

menu_loop

Elle ne prendra aucune entrée et ne renverra rien. Elle aura donc la forme suivante:

```
def game_loop()
    # Ce que l'on veut
    while True:
        #1
        #2
        #3
    # Dire 'Au-revoir!'.
```

Derniers mots

```
Et franchement... Voilà.
```