Algorithmique et Programmation 1 – TD - TP 8
DES PETITS JEUX
CORRECTION

Exercice 1 - Mastermind
Dans cet exercice, on se propose d'écrire un programme permettant de jouer au *Mastermind*.

On rappelle le principe de ce jeu. Un joueur (ici, la machine) choisit une combinaison secrète de 5 couleurs parmi 8 couleurs disponibles (on représentera durant tout l'exercice les couleurs par les entiers 0, 1, 2, 3, 4, 5, 6, 7). Une couleur peut apparaître plusieurs fois dans une combinaison. L'adversaire de ce joueur a 10 tentatives pour deviner cette combinaison sachant qu'après chaque proposition, il bénéficie d'une indication : le premier joueur examine sa proposition et lui dit combien de pions colorés sont bien placés et combien sont présents dans la combinaison secrète, mais mal placés.

Ainsi dans l'exemple suivant :
- combinaison secrète : [4, 2, 6, 0, 5]
- combinaison proposée : [1, 6, 4, 0, 5]

le joueur qui devine se verra répondre "2 pion(s) bien placé(s) et 2 pions mal placé(s)", correspondant aux couleurs 0 et 5 pour les bien placées, et aux couleurs 6 et 4 pour les mal placées.

Voici un deuxième exemple pour bien fixer les idées :
- combinaison secrète : [1, 1, 6, 0, 3]
- combinaison proposée : [1, 6, 4, 1, 1]

le joueur qui devine se verra répondre "1 pion(s) bien placé(s) et 2 pion(s) mal placé(s)", correspondant à la couleur 1 en position 0 bien placée, et les couleurs 1 (position 3 ou 4) et 6 (position 1).

1. Écrire une fonction `nb_bien_places(proposition, solution)` qui renvoie le nombre de pions bien placés dans la liste `proposition` et fonction de la liste `solution`

```
def nb_bien_places(proposition, solution) :
    """List x List --> Int
    Retourne le nombre d'éléments placés en même position dans la proposition et
    dans la solution."""
    nb = 0
    for i in range(len(proposition)):
        if proposition[i] == solution[i] :
            nb = nb +1
    return nb
```

2. Écrire une fonction `nb_bien_places_couleur(proposition, solution, couleur)` qui renvoie le nombre de pions de la couleur `couleur` bien placés dans la liste `proposition` et fonction de la liste `solution`

```
def nb_bien_places_couleur(proposition, solution, couleur) :
    """List x List x Int --> Int
    Retourne le nombre de fois que la couleur est dans la meme position dans
    la proposition et la solution."""
    nb = 0
    for i in range(len(proposition)):
        if proposition[i] == couleur and solution[i] == couleur :
            nb = nb +1
    return nb
```

3. Écrire une fonction `compte_occurences(liste)` qui renvoie une liste de taille 8 dont la i-ème case contient le nombre d'occurrences de la valeur i dans la liste d'entiers `liste`

```
def compte_occurences(liste) :
    """List --> List
    Retourne une liste de longueur 8, dont la i-ème case contient
    le nombre d'occurences de la valeur i dans la liste liste"""

    liste_nb_occur = [0, 0, 0, 0, 0, 0, 0, 0]
    for i in range(len(liste)):
         k = liste[i]
         liste_nb_occur[k] = liste_nb_occur[k] + 1
    return liste_nb_occur
```

4. On veut désormais écrire une fonction permettant de compter le nombre de pions mal placés en se basant sur l'observation suivante : pour une couleur donnée, si elle apparaît x fois dans la proposition et y fois dans la solution secrète, alors le nombre de pions mal placés de cette couleur plus le nombre de pions bien placés de cette couleur est égal au minimum de x et de y.
En déduire une fonction `nb_couleurs_mal_placees(proposition, solution)` qui renvoie le nombre de couleurs mal placées.
On pourra écrire une fonction `min_int(x, y)` qui renvoie le minimum des deux entiers x et y donnés en paramètre.

```
def min_int(x, y) :
    """ Int x Int --> Int
    Retourne le minimum entre x et y"""
    if x < y :
        return x
    else :
        return y

def nb_couleurs_mal_placees(proposition, solution) :
    """List x List --> Int
    Retourne le nombre de couleurs mal placées dans la proposition
    par rapport à la solution"""

    mal_places = 0
    l_nb_occur_prop = compte_occurences(proposition)
    l_nb_occur_sol = compte_occurences(solution)

    for i in range(8) :
        bien_places = nb_bien_places_couleur(proposition, solution, i)
        nb_min = min_int(l_nb_occur_prop[i], l_nb_occur_sol[i])
        mal_places = mal_places + nb_min - bien_places
    return mal_places
```

5. Écrire une fonction `generation_solution()` qui renvoie une liste aléatoire de longueur 5 constitué de valeurs de 0 à 7.

```
def generation_solution() :
    """None --> List
    Génère une liste de 5 entiers de 0 à 7"""
    liste = [0,0,0,0,0]
    for i in range(5):
        liste[i] = random.randrange(8)
    return liste
```

6. Écrire maintenant un programme qui simule le jeu, c'est-à-dire qu'il laisse à l'utilisateur dix tentatives pour trouver la solution en lui donnant après chaque tentative le nombre de bien placés et de mal placés. Si le joueur a trouvé la bonne solution, la boucle doit s'arrêter en affichant un message de victoire, et si le joueur échoue après dix tentatives, le programme doit afficher un message d'échec en lui donnant la combinaison secrète. A chaque tour, l'utilisateur entrera sa combinaison en tapant les 5 chiffres successivement.

```
solution = generation_solution()
trouve = False
compteur = 1

while compteur <= 10 and not(trouve) :
    proposition = [0,0,0,0,0]
    print("Tentative", compteur)
    print("Proposez 5 couleurs (chiffres de 0 à 7):")
    for i in range(5) :
        print("Couleur", i+1)
        proposition[i] = int(input())
    print("Votre proposition est la suivante :", proposition)
    c_bien_placees = nb_bien_places(proposition, solution)
    c_mal_placees = nb_couleurs_mal_placees(proposition, solution)
    if c_bien_placees == 5:
        trouve = True
    else :
        print(c_bien_placees, "pion(s) bien placé(s) et",
        c_mal_placees, "pion(s) mal placé(s)")
        compteur = compteur + 1

if trouve :
    print("Félicitations, vous avez gagné! la solution était", solution)
else :
    print("Vous avez perdu. La solution était", solution)
```

Exercice 2 - Jeu du pendu
Dans cet exercice, on se propose d'écrire un programme permettant de jouer au *jeu du pendu*.

Le fichier `donnees.py` contient des données utiles au programme du pendu (variable définissant le nombre de coups par partie, liste de 1360 mots de plus de 5 lettres).

Ecrire un programme contenant les fonctions suivantes :

1. Une fonction permettant de récupérer les lettres tapées au clavier par le joueur
2. Une fonction permettant de choisir aléatoirement un mot à partir de la liste des mots disponibles
3. Une fonction permettant d'écrire le mot à deviner, en affichant uniquement les lettres déjà trouvées et le symbole '-' s'il apparait dans le mot. Par exemple : `*a*lle-**a***`
4. Le programme principal, qui permet de jouer au pendu. Le déroulé peut, par exemple, être de ce type là :

```
Mot à trouver ****** (encore 8 chance(s))
Tapez une lettre: a
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver ****** (encore 7 chance(s))
Tapez une lettre: e
Bien joué.
Mot à trouver **e**e (encore 7 chance(s))
Tapez une lettre: p
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver **e**e (encore 6 chance(s))
Tapez une lettre: i
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver **e**e (encore 5 chance(s))
Tapez une lettre: e
Vous avez déjà choisi cette lettre.
Mot à trouver **e**e (encore 5 chance(s))
Tapez une lettre: r
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver **e**e (encore 4 chance(s))
Tapez une lettre: t
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver **e**e (encore 3 chance(s))
Tapez une lettre: m
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver **e**e (encore 2 chance(s))
Tapez une lettre: l
Bien joué.
Mot à trouver *le**e (encore 2 chance(s))
Tapez une lettre: b
... non, cette lettre ne se trouve pas dans le mot...
Mot à trouver *le**e (encore 1 chance(s))
Tapez une lettre: c
Bien joué.
Mot à trouver *lec*e (encore 1 chance(s))
Tapez une lettre: f
Bien joué.
Mot à trouver flec*e (encore 1 chance(s))
Tapez une lettre: h
Bien joué.
Félicitations ! Vous avez trouvé le mot fleche
```

```
from donnees import *
from random import choice

#Fonction permettant de récupérer une lettre tapée par l'utilisateur
def recup_lettre():
    """ None --> Str
    Cette fonction récupère une lettre saisie par
    l'utilisateur."""
    lettre = input("Tapez une lettre: ")
    lettre = lettre.lower()
    if len(lettre)>1 or not lettre.isalpha():
        print("Vous n'avez pas saisi une lettre valide.")
        return recup_lettre()
    else:
        return lettre

# Fonction permettant de choisir un mot
def choisir_mot():
    """None --> Str
    Cette fonction renvoie le mot choisi dans la liste des mots
    liste_mots."""
    return choice(liste_mots)

#Fonction permettant d'afficher uniquement les lettres déjà trouvées d'un mot
def recup_mot_masque(mot_complet, lettres_trouvees):
    """Str x List --> Str
    Retourne le mot d'origine avec des * remplaçant les lettres que l'on
    n'a pas encore trouvées."""

    mot_masque = ""
    for lettre in mot_complet:
        if lettre in lettres_trouvees or lettre == "-":
            mot_masque = mot_masque + lettre
        else:
            mot_masque = mot_masque + "*"
    return mot_masque

#Programme principal
mot_a_trouver = choisir_mot()
lettres_trouvees = []
mot_trouve = recup_mot_masque(mot_a_trouver, lettres_trouvees)
nb_chances = nb_coups
while mot_a_trouver!=mot_trouve and nb_chances>0:
    print("Mot à trouver", mot_trouve,"(encore", nb_chances, "chance(s))")
    lettre = recup_lettre()
    if lettre in lettres_trouvees: # La lettre a déjà été choisie
        print("Vous avez déjà choisi cette lettre.")
    elif lettre in mot_a_trouver: # La lettre est dans le mot à trouver
        lettres_trouvees.append(lettre)
        print("Bien joué.")
    else:
        nb_chances = nb_chances - 1
        print("... non, cette lettre ne se trouve pas dans le mot...")
    mot_trouve = recup_mot_masque(mot_a_trouver, lettres_trouvees)

# A-t-on trouvé le mot ou nos chances sont-elles épuisées ?
if mot_a_trouver==mot_trouve:
    print("Félicitations ! Vous avez trouvé le mot", mot_a_trouver)
else:
    print("PENDU !!! Vous avez perdu. Le mot à trouver était", mot_a_trouver)
```