INF1 – P21

Examen Médian - corrigé

1ère partie : scrabble *(6 points)*

Au jeu de Scrabble, les joueurs doivent former des mots à partir des 7 lettres dont ils disposent. On appellera `jeu` cet ensemble de 7 lettres dans la suite.

1. Écrire en Python une fonction booléenne `acceptable(mot, jeu)` qui retourne `True` si un mot est acceptable pour un jeu donné et `False` sinon.

   Indications :

   - Un mot est dit acceptable si on peut le former à partir des lettres du jeu, même si ce mot n'existe pas en français.
   - On considérera pour cela que `jeu` et `mot` sont constitués uniquement de lettres majuscules non accentuées.

   Exemples : ELEVE, WVB ou BLE sont des mots acceptables avec le jeu BEWEELV

   Conseils :

   - Pour écrire cette fonction, on pourra remarquer qu'un mot est acceptable si le nombre d'occurrences de chacune des lettres du mot est inférieur ou égal au nombre d'occurrences de chacune des lettres du jeu.
   - On pourra utiliser la méthode `count(car)` d'une chaine de caractères, qui retourne le nombre d'occurrences du caractère `car` dans cette chaine. Par exemple, si le mot est ELEVE, `mot.count('E')` retourne 3.

Deux versions :

```
def acceptable (mot,jeu):
    possible=True
    for lettre in mot:
        if jeu.count(lettre)<mot.count(lettre):
            possible = False
    return possible


def acceptable2 (mot,jeu):
    possible=True
    i=0
    while possible and i < len(mot):
        if jeu.count(mot[i])<mot.count(mot[i]):
            possible = False
        i=i+1
```

```
    return possible
```

2. Au scrabble, un nombre de points est associé à chaque lettre de la façon suivante :

1 point	EAINORSTUL
2 points	DMG
3 points	BCP
4 points	FHV
8 points	JQ
10 points	KWXYZ

Écrire une fonction `nbPoints(mot)` qui retourne le cumul du nombre de points des lettres du mot `mot`.

```
def nbPoints(mot):
    points = 0
    for lettre in mot :
        if lettre in 'EAINORSTUL':
            points = points + 1
        elif lettre in 'DMG':
            points = points + 2
        elif lettre in 'BCP':
            points = points + 3
        elif lettre in 'FHV':
            points = points + 4
        elif lettre in 'JQ':
             points = points + 8
        elif lettre in 'KWXYZ':
            points = points + 10
    return points
```

3. En utilisant les deux fonctions précédentes, écrire un programme qui :
   a. demande à l'utilisateur le jeu dont il dispose et le mot qu'il souhaite former à l'aide ce jeu,
   b. si le mot est acceptable, affiche le nombre de points qu'il rapporterait et sinon affiche qu'il n'est pas acceptable.

On supposera que l'utilisateur saisit le mot et le jeu en majuscules.

```
jeu=input("Entrez le jeu dont vous disposez : ")
mot=input("Entrez le mot que vous souhaitez former : ")
if acceptable2(mot, jeu):
    print("Avec le mot ", mot, " vous obtenez ", nbPoints(mot), "
                                                        points")
else:
    print("Le mot ", mot, " n'est pas acceptable")
```

---------- *{ prendre une nouvelle copie }* ----------

2ème partie : alternatives *(7 points)*

1. Un magasin de reprographie facture 20 centimes pièce les dix premières photocopies, 15 centimes les vingt suivantes et 5 centimes au-delà.

   Écrire un algorithme qui demande à l'utilisateur le nombre de photocopies à effectuer puis affiche le montant correspondant en centimes.

```
Constantes :
        P1=20 ; P2=15 ; P3=5
Variables :
        prix : reel (ou entier)
        N : entier
Début
      Afficher ('Quel est le nombre de photocopies ? :')
      Lire N

      Si N ≤10 Alors
          prix ←P1 * N

      Sinon Si N ≤30
          prix ← P1*10 + P2 * (N-10)

      Sinon
           prix ← P1 * 10 + P2 * 20 + P3 * (N-30)
      Fsi

      Afficher ('Le montant à payer est:', prix)
 Fin
```

2. Une fréquence cardiaque normale au repos se situe entre 50 et 90 bpm (battements par minutes). Durant l'activité physique, la fréquence cardiaque normale peut augmenter jusqu'à 120 bpm.

   Écrire un algorithme permettant de déterminer si une fréquence cardiaque donnée est normale.
   L'utilisateur saisira au clavier la fréquence cardiaque (HR) et indiquera si la mesure a été faite dans le cadre d'une activité physique ou non (variable booléenne activite)

   Exemples :

   Si l'utilisateur saisit HR = 110 et activite = True
   => L'algorithme affiche : Fréquence cardiaque normale
   Si l'utilisateur saisit HR = 95 et activite = False

=> L'algorithme affiche : Fréquence cardiaque anormale

Variables :

HR : entier
activite : bool

Début

Afficher ('Saisir une fréquence cardiaque')
Lire HR

Afficher ('Est-ce que la personne est en activité True/False ?')
Lire activite

Si HR < 90 et HR > 50:
    Afficher ('Votre fréquence cardiaque est normale')

Sinon si 90<= HR <120 et activite == True
    Afficher ('Votre fréquence cardiaque est normale')
Sinon
    Afficher('Votre fréquence cardiaque est anormale')
Fsi

Fin

3. On considère le programme Python suivant, qui décrit une procédure de filtrage de valeurs RR, RR correspondant à la durée (en millisecondes) entre deux pics R d'un signal électrique du cœur. RR0, RR1, RR2 et RR3 sont 4 intervalles successifs d'un signal cardiaque.

```
RR0 = float(input('RR0'))
RR1 = float(input('RR1'))
RR2 = float(input('RR2'))
RR3 = float(input('RR3'))

RR_f = RR2 + RR3
e1 = abs(RR_f-RR1)/RR1

print(f' e1 = {e1}')

if e1 < 0.40:
    print(f'RR_f = RR2 + RR3 = {RR_f}')
    print(f'RR2 est fusionnée avec la valeur suivante, RR3')

else:
    RR_f= RR1 + RR2
    e2 = abs(RR_f-RR0)/RR0
```

```
    print(f' e2 = {e2}')

    if e2 < 0.40:
        print(f'RR_f = RR2 + RR1 = {RR_f}')
        print(f'RR2 est fusionnée avec la valeur précédente, RR1')
    else:
        print('Les deux erreurs sont supérieures à 40%.')
        print('Il est donc impossible de fusionner les valeurs.')
```

Quel est l'affichage obtenu si l'utilisateur saisit les valeurs suivantes ?

a) RR0 = 0.4, RR1 = 0.4, RR2 = 0.2, RR3 = 0.7
b) RR0 = 0.4, RR1 = 0.6, RR2 = 0.22, RR3 = 0.35

a) E1 = 1.25
E2 = 0.5
Impossible de fusionner les valeurs RR

b) E1 = 0.050000000000000044 (0.05 accepté)
RRj = RR2 + RR3 = 0.57

---------- ***{ prendre une nouvelle copie }*** ----------

3ème partie : lancés de dés *(7 points)*

Dans les jeux de plateau, les joueurs doivent souvent lancer un ou deux dés pour déterminer le joueur qui commencera à jouer. Par exemple, le premier joueur à faire un 6 ou un double 6 commence à jouer. Nous allons simuler ce(s) lancé(s) de dé.

1. Dans un premier temps, on souhaite simplement simuler les lancés d'un seul dé sans déterminer le joueur qui fait un 6 en premier.

   Proposer un programme Python qui remplit un tableau avec les valeurs des lancés. Par exemple, s'il y a 5 joueurs, le tableau pourrait contenir les valeurs suivantes : `[4, 3, 2, 6, 1]`

   On utilisera pour cela la fonction `random.randint(1, 6)`. Cette fonction permet de retourner un entier entre 1 et 6.

```
import random
DIM = 5
t = ['-']*DIM
print(t)
for i in range(DIM):
    t[i] = random.randint(1, 6)
print(t)
```

ou

```
DIM = 5
t = []
for i in range(DIM):
    t.append(random.randint(1, 6))
print(t)
```

2. On souhaite maintenant que les joueurs lancent successivement un dé jusqu'à ce que l'un d'eux fasse un 6.

   Proposer un programme Python qui affichera un tableau contenant le lancé de chaque joueur, puis le numéro du premier joueur à faire 6. Pour reprendre l'exemple 1, on s'arrêtera au joueur 4 et on affichera *:*

```
[4, 3, 2, 6]
Le premier joueur à faire 6 est le joueur 4
```

   Si aucun joueur n'obtient 6, le programme affichera le tableau contenant les valeurs obtenues et signalera qu'aucun joueur n'a obtenu 6.

```
t = []
i = 0
b6 = False
while (b6 == False) and i < DIM:
    t.append(random.randint(1, 6))
    b6 = t[i] == 6
    i += 1
print(t)
if b6:
    print(f'Le premier joueur est le joueur {i}')
else:
    print("Personne n'a fait de 6")
```

3. Modifier ce programme (en Python ou sous forme d'algorithme), pour que :
   a. Si personne ne fait 6, on relance des tours de lancés de dé jusqu'au premier 6.
   b. Le programme affiche le numéro du premier joueur à obtenir 6 et le tour pendant lequel il l'a obtenu.

   Par exemple, l'affichage pourrait être :

```
==== Tour 1 ====
[3, 1, 5, 5, 1]
==== Tour 2 ====
[2, 6]
Le joueur qui a fait un 6 en premier est le joueur 2 au tour 2
```

```
def Tour():
    t = []
    i = 0
    b6 = False
    while (b6 == False) and i < DIM:
        t.append(random.randint(1, 6))
        b6 = t[i] == 6
        i += 1
    return t
```

```
def LancesDes():
    b6 = False
    tour = 0
    while (b6 == False):
        tour += 1
        print(f'==== Tour {tour} ====')
        t = Tour()
        b6 = t[len(t)-1] == 6
        print(t)
    if b6:
        print(f'Le 1er joueur à faire un 6 est le joueur {i} au
tour {tour}')
```

4. Les joueurs lancent maintenant 2 dés jusqu'à ce que l'un d'eux obtienne un double 6. Par exemple, au 1er lancé, si le joueur 1 fait 4 et 1 et le joueur 2 fait 2 et 4, le tableau se remplit ainsi : [[4, 1], [2, 4]…].
   Proposer un algorithme, sur le principe de la question 3, qui déterminera le 1er joueur à faire un double 6.
   Par exemple, l'affichage pourra être :

```
==== Tour 1 ====
[[4, 1], [2, 4], [3, 2], [2, 1], [4, 5]]
==== Tour 2 ====
[[4, 3], [2, 6], [2, 5], [6, 6]]
Le joueur qui a fait un double 6 en premier est le joueur 4 au
tour 2
```

```
def Double6_1Tour():
    DIM = 5
    i = 0
    bDouble6 = False
    t = []
    while bDouble6 == False and i < DIM:
        lance = []
        for j in range(0, 2):
            lance.append(random.randint(1, 6))
        t.append(lance)
        bDouble6 = (t[i][0] == 6) and (t[i][1] == 6)
        i += 1
    print(t)
    if bDouble6:
        print(f'Le 1er joueur à faire un 6 est le joueur {i}')
    else:
        print("Personne n'a fait de double 6")

Double6_1Tour()
```