Tuples - Exercices

Exercice 1

On considère le tuple `'ingredients'` suivant :

```
ingredients = ('farine', 'oeufs', 'lait', 'sucre')
```

1. Que valent les instructions `'ingredients[1]'` et `'ingredients[4]'` ?
2. Quelle instruction renvoie le nombre d'ingrédients présents dans ce tuple ?

Exercice 2

Vous avez vu en classe de seconde la propriété suivante :

Soient $A(x_A, y_A)$ et $B(x_B, y_B)$ deux points dans un repère orthogonal. Le milieu I du segment $[AB]$ a pour coordonnées : $x_I = \frac{x_A + x_B}{2}$ et $y_I = \frac{y_A + y_B}{2}$.

Un point peut être représenté par un *tuple* de deux valeurs, la première est l'abscisse et la seconde l'ordonnée du point.

1. On considère dans un repère les points $A(2;3)$ et $B(4;-1)$. Quelles instructions permettent d'accéder à l'abscisse x_A du point A et à l'ordonnée y_B du point B ?

```
A = (2, 3)
B = (4, -1)
# à compléter
xA = ...
yB = ...
```

2. On note I le milieu du segment $[AB]$. Complétez les instructions suivantes pour stocker respectivement dans les variables x_I et y_I l'abscisse et l'ordonnée de I (il s'agit de retranscrire les formules données plus haut).

```
A = (2, 3)
B = (4, -1)
# à compléter
xI = ...
yI = ...
```

3. Écrivez une fonction `milieu(A, B)` qui prend en paramètres deux tuples *A* et *B* de 2 valeurs, et qui renvoie un tuple correspond aux coordonnées du milieu du segment $[AB]$.

```
A = (2, 3)
B = (4, -1)
print(milieu(A, B))
```

```
(3.0, 1.0)
```

Exercice 4 : Distance dans le plan

On rappelle que dans un repère orthonormé, la distance entre deux points $A(x_A, y_A)$ et $B(x_B, y_B)$ est égale à : $AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$

1. Écrivez une fonction Python `distance(A, B)` qui renvoie la distance entre les deux tuples `A` et `B` (représentant deux points d'un repère orthonormé).

```
# pour utiliser la fonction sqrt qui correspond à la fonction racine carrée
from math import sqrt

...

print(distance((1,1), (-2,-3)))
```

```
5.0
```

Exercice 5 : Conversion de durées

1. Écrire une fonction temps_vers_secondes(s) qui prend en paramètre un tuple de la forme (heures, minutes, secondes) et renvoie ce temps converti en seconde.

```
t = (1, 23, 45) # 1H, 23 mins, 45 secs
print(temps_vers_secondes(t))
```

```
5025
```

2. Écrire une fonction seconde_vers_temps(s) qui prend en paramètre une durée en secondes et qui renvoie le résultat dans un tuple de la forme (heures, minutes, secondes).

```
print(seconde_vers_temps(5025))
```

```
(1, 23, 45)
```

3. Vérifier que les égalités suivantes sont vraies.

```
t = (1, 23, 45) # 1H, 23 mins, 45 secs
print(seconde_vers_temps(temps_vers_secondes(t)) == t)
print(temps_vers_secondes(seconde_vers_temps(5025)) == 5025)
```

```
True
True
```

Exercice 6 : Parcours

Écrire la fonction `somme(t)` qui prend en paramètre un tuple `t` contenant des valeurs numériques et qui renvoie la somme de ces valeurs. La taille du tuple passé en paramètre peut être quelconque.

```
print(somme((1, 2, 4, 8, 16, 32)))
print(somme((1, -1, 2, -2)))
```

```
63
0
```

Exercice 7 : Morpion

On décide de représenter une grille de morpion de taille 3 x 3 par un tuple contenant 9 valeurs :

1	2	3
4	5	6
7	8	9

```
Grille = (1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Les valeurs du tuple peuvent être prise parmi les valeurs suivantes :

- `'_'` pour indiquer que la case n'a pas été jouée.
- `'x'` pour indiquer que le joueur 1 a joué sur la case.
- `'o'` pour indiquer que le joueur 2 a joué sur la case.

Voici un exemple de tuple représentant une grille gagnante (horizontalement) pour le joueur 1 :

```
g0 = ('x', 'x', 'x', 'o', '_', 'o', 'o', '_', '_')
```

On se limite par la suite aux grilles gagnantes verticales.

1. Donner 2 grilles gagnantes `g1` et `g2` pour le joueurs 1.

2. Donner 2 grilles gagnantes `g2` et `g3` pour le joueurs 2.

3. Donner 2 grilles `g4` et `g5` sans gagnant (sans victoire ni verticale, ni horizontale, ni diagonale).

4. Écrire la fonction `victoire_verticale(g)` qui prend en paramètre un tuple représentant une grille et qui retourne `'x'`, `'o'` ou `'_'` si la grille est gagnante pour le joueur 1, le joueur 2 ou aucun des joueurs respectivement.
   Tester votre fonction avec les grilles des questions 1, 2 et 3.

```
g6 = ('x', '_', 'o', '_', '_', 'o', 'x', '_', '_')
v = victoire_verticale(g6)
print(v)
```

```
_
```

5. Écrire une fonction `initialisation_grille()` qui retourne une grille vide.
6. Écrire une fonction `jouable(g, n)` qui retourne `True` si la case `n` (en suivant la numérotation vue précédemment) de la grille `g` n'a pas encore été jouée et `False` sinon.

```
g6 = ('x', '_', 'o', '_', '_', 'o', 'x', '_', '_')
print(jouable(g6, 1))
print(jouable(g6, 4))
```

```
False
True
```

7. Écrire une fonction `jouer(g, n, j)` qui retourne une copie de la grille `g` en modifiant la case `n` avec la valeur `j`.
   Aucune vérification n'est à faire pour savoir si la case a déjà été jouée ou non.

```
g6 = ('x', '_', 'o', '_', '_', 'o', 'x', '_', '_')
g7 = jouer(g6, 4, 'o')
print(g7)
```

```
('x', '_', 'o', 'o', '_', 'o', 'x', '_', '_')
```