TP n°1 : Tuples

Instruction générale : hormis pour les exercices corrigés collectivement, vous ferez valider votre travail par l'enseignant.

1 Tuple

Un **tuple** ou **p-uplet** est une collection immuable de valeurs. Cette collection est **indexée**, i.e. qu'à chaque valeur est attribuée sa position dans la collection. Il s'agit d'un objet de type **construit**.

En Python, les tuples sont de type `tuple` en Python et sont définis en séparant les valeurs par une virgule et en entourant la collection de valeur par des parenthèses.

Exemple :

```
a=(3,89,-4) # Création d'un tuple à 3 éléments
b=() # Création d'un tuple vide
```

Remarques : Lorsqu'il n'y a pas d'ambiguïtés, Python reconnaît un tuple sans les parenthèses.

```
>>> a = 3, 4, 5
>>> type(a)
< class 'tuple' >
>>> b = 3,
>>> type(b)
< class 'tuple' >
```

On peut mettre un tuple dans un tuple, il est alors nécessaire d'utiliser les parenthèses.

```
>>> t = 3, 4, (2,1), 5
>>> type(t)
< class 'tuple' >
>>> print(t)
(3,4,(2,1),5)
```

Exercice 1. Dans la console Python, créer une variable `t1` étant un tuple contenant les trois nombres 5, 3 et 8 dans cet ordre.

2 Opérations sur les tuples

2.1 Position et longueur

Comme pour les listes, il est possible d'obtenir la longueur d'un tuple grâce à la commande `len(tuple)` et la valeur d'un élément du tuple grâce à la syntaxe `tuple[numéro élément]`.

Exemple :

```
>>> a=(3,89,-4)
>>> a[0] # Première valeur (en position 0)
3
>>> a[2] # Troisième valeur (en position 2)
-4
>>> len(a) # Longueur de a
3
>>> a[-1] # Avant-dernière valeur (en position "-1")
-4
```

Exercice 2. Dans la console Python, créer la variable `t2=(1,)`. Que renvoie les commandes suivantes ? Interpréter.

1. `t2[0]` ;
2. `t2[1]` ;
3. `t2[1]=2` ;
4. `t2=(1,2)`.

2.2 Appartenance

Il est possible de tester directement l'appartenance d'un élément ou d'accéder à celui-ci grâce à la commande `in`.

```
>>> a=(3,89,-4)
>>> 3 in a
True
>>> 2 in a
False
```

Exercice 3. Écrire un programme en Python affichant tous les éléments d'un tuple.

Exercice 4. Écrire un programme en Python récupérant tous les éléments d'un tuple pour les stocker dans une liste.

Exercice 5. Écrire une fonction testant l'appartenance d'un élément à un tuple sans utiliser `in`. Quelle est la complexité de cet algorithme ? Autrement dit, quel est l'ordre de grandeur du nombre d'opérations effectuées par le programme en fonction de la taille du tuple ?

2.3 Création par concaténation

On peut créer un tuple en concaténant d'autres tuples grâce l'opérateur `+`.

Exemple :

```
>>> a=(3,89,-4)
>>> c=(2,)
>>> d = a + c
>>> print(d)
(3,89,-4,2)
```

Exercice 6. Dans la console Python, créer une variable `t3` en concaténant les tuples `t1` et `t2` puis une variable `t3` en concaténant les tuples `t2` et `t1` (dans l'ordre inverse). Tester l'égalité entre **t3** et `t4`, que constate-t-on ?

Exercice 7. Dans la console Python, créer deux variables `a=(1,2)` et `b=(3)` puis les concaténer. Que constatez-vous ? Même question avec `b=(3,)`. Qu'en déduisez-vous ?

Exercice 8. Créer un tuple contenant les entiers de 1 à 100.

2.4 Création par répétition

On peut créer un tuple par répétition d'un tuple avec l'opérateur $*$.

```
>>> a=(1,2)
>>> e=a*3
>>> print(e)
(1,2,1,2,1,2)
```

Exercice 9. Créer un tuple contenant cent fois la valeur 1 puis un autre alternant dix fois les valeurs 1, 2 et 3.

Exercice 10. [Damier] En s'inspirant du code ci-dessous, écrire un programme Python affichant un damier de taille quelconque. Cette dernière devra être demandée à l'utilisateur.

```
import matplotlib.pyplot as plt

damier=((0,1),(1,0))

plt.matshow(damier)
plt.show()
```

Indication : il peut être judicieux de distinguer les tailles paires et impaires.

3 Tuples et résultats de fonctions

Il est possible d'affecter les multiples résultats d'une fonction à un tuple comme on peut le voir dans l'exemple ci-dessous.

```
def coordFonctionCarre(x):
    return (x, x**2)  # Ici la fonction renvoie un tuple

x,y = coordFonctionCarre(5)
```

Exercice 11. [Milieu] Écrire une fonction prenant en entrée les coordonnées de deux points sous forme de tuples et renvoyant les coordonnées de leur milieu, toujours sous forme de tuple. Les points seront de dimension quelconque et la fonction devra s'assurer qu'ils ont bien la même (on ne peut pas calculer le milieu de $(1\,;2)$ et $(1\,;2\,;3)$ par exemple).