TD n°6
Les tuples

L'utilisation du terme anglais **tuple** , abréviation de quin-tuple/sex-tuple/..., est courante dans des ouvrages de programmation.
En français, on parle plutôt de **n-uplet** : doublet, triplet...

Pour créer un tuple , on place les **items séparés par des virgules entre parenthèses** :

```
>>> mon_tuple = (5, 8, 6, 9)
>>> mon_tuple
(5, 8, 6, 9)
>>> type(mon_tuple)
<class 'tuple'>
```
On constate que la variable mon_tuple est bien de type tuple.

Dans le code ci-dessus, la variable mon_tuple référence un tuple qui est constitué des entiers 5, 8, 6 et 9.

Remarque : on peut créer un tuple sans mettre de parenthèses mais il est conseillé de les ajouter afin d'améliorer la lisibilité :

```
>>> mon_tuple = 5, 8, 6, 9
>>> mon_tuple
(5, 8, 6, 9)
```

Un tuple ne contient pas forcément que des nombres entiers, il peut aussi contenir des nombres décimaux, des chaînes de caractères, des booléens... voir même un mélange de ces différents types de variables.

```
>>> tuple = (1, 6, 9, "Bonjour", False)
>>> tuple
(1, 6, 9, 'Bonjour', False)
```

Et pourquoi pas des tuples de tuples...

```
>>> t1 = 1, 6, 8
>>> t2 = 5, 8, 10
>>> tuple = (t1, t2)
>>> tuple
((1, 6, 8), (5, 8, 10))
```

Un tuple est **itérable** : chaque élément du tuple possède un indice (ou index).
On peut ainsi accéder facilement à l'élément de son choix dans un tuple , le principe est exactement le même que pour les listes ou chaînes de caractères :

```
>>> mon_tuple = (5, 8, 6, 9)
>>> mon_tuple[1]
8
>>> mon_tuple[-1]
9
>>> mon_tuple[1:]
(8, 6, 9)
```

On peut tester l'appartenance à un tuple avec **in**

```
>>> mon_tuple = (5, 8, 6, 9)
>>> 5 in mon_tuple
True
>>> 12 in mon_tuple
False
```

Un **tuple** permet l'affectation multiple de valeurs :

```
>>> a, b, c, d = (5, 8, 6, 9)
>>> a
5
>>> b
8
>>> c
6
>>> d
9
```

Grâce au **tuple**, une fonction peut renvoyer plusieurs valeurs :

```
def add(a, b):
    c = a + b
    return (a, b, c)
```

```
>>> mon_tuple = add(5, 8)
(5, 8, 13)
```

Un tuple est **immutable** (non mutable) : il n'est pas possible de modifier un tuple après sa création.

```
>>> mon_tuple = (5, 8, 6, 9)
>>> mon_tuple[2]
6
>>> mon_tuple[2] = 4
```
Essayer modifier un tuple existant et l'interpréteur Python vous renvoit une erreur.

```
Traceback (most recent call last):
File "<pyshell>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Comparatif :

	str	***list***	***tuple***
Mutabilité	Non	Oui	Non
Not. indicielle	Oui	Oui	Oui
Fonction *len*	Oui	Oui	Oui
Opérateur *in*	Oui	Oui	Oui
Opérateur +	Oui	Oui	Oui
Itération *for*	Oui	Oui	Oui
Méthode *index*	Oui	Oui	Oui
Méthode *count*	Oui	Oui	Oui
Méthode *copy*	Non	Oui	Non

Exercice 1

Ecrire dans le langage Python une fonction dont le paramètre est un entier et qui renvoie 1 tuple formé des 3 entiers suivants.

Exercice 2

Ecrire dans le langage Python une fonction dont les paramètres sont 2 tuples représentant les coordonnées de 2 points et qui retourne un tuple formé des coordonnées du milieu des 2 points.

Exercice 3

Ecrire dans le langage Python une fonction dont les paramètres sont 3 nombres et retourne les 2 plus grands sous forme de tuple.

Exercice 4

Ecrire dans le langage Python une fonction dont le paramètre est un tuple représentant une liste de nombres et qui retourne la moyenne de ces nombres.