NSI : Leçon 3, les tuples.

1. Définition et fonctions et méthodes.

A. Définition

Un tuple en Python est une collection ordonnée d'objets séparés par des virgules. Ces objets peuvent être de types différents.

Les tuples comme les châines de caractères sont immutables : on ne peut modifier leur contenu.

Ils sont souvent utilisés pour traiter des données que l'on veut préserver de toute modification lors de leurs exploitations informatiques.

Ils permettent un accès en mémoire plus rapide que d'autres types de collections d'objets.

```
>>> a=(2,3,4,"aze",7.8) # exemple de tuple possédant 5 éléments séparés par un ,
>>>type(a)
<class 'tuple' >
>>> b=((1,2),(3,4)) # exemple de tuple contenant des tuples
>>> c=() # tuple vide
>>> d=(2,) # tuple ayant un seul élément
>>>>>> a=5,8 # ceci est un tuple
>>> type(a)
<class 'tuple'>
```

B. Fonctions et méthodes utiles sur les tuples

```
>>> a=(10,21,11,10,5,6,7,11,10,31) # un tuple ne contenant que des int
>>> len(a) # renvoie la longueur du tuple, c'est à dire son nombre d'éléments
10
>>> max(a) # renvoie la plus grande valeur
31
>>> min(a) # renvoie la plus petite valeur
5
>>> a.index(10) # renvoie le 1er index du tuple contenant la valeur 10
0
>>> a.index(21)
1
>>> a.count(10) # renvoie le nombre de fois où la valeur 10 apparait dans le tuple
3
>>> 10 in a # permet de déterminer si une valeur appartient au tuple
True
>>> 15 in a
False
```

C. Accéder aux éléments d'un tuple.

On peut accèder aux valeurs contenues dans un tuple en recourant à leur indice comme dans les chaînes de caractères.

```
>>> a=(10,23,"a",17)
>>> a[0]
10
>>> a[1]
23
>>> a[len(a)-1]
17
>>> a[-2]
'a'
>>> a[:2]
(10, 23)
>>> a[2:]
('a', 17)
```

Pour des tuples contenant des tuples, ce qui est assez fréquent en Python :

```
>>> a=((1,2,3),(4,5,6))
>>> a[0]
(1, 2, 3)
>>> a[0][0]
1
>>> a[1][2]
6
```

On peut parcourir également un tuple avec une boucle for :

```
def parcourir(mon_tuple):
    for valeur in mon_tuple:
        print(valeur)
```

donnera :

```
>>> a=(1,2,3)
>>> parcourir(a)
1
2
3
>>> a=((1,2),"aze",5)
>>> parcourir(a)
(1, 2)
aze
5
```

D. Ajouter un élément à un tuple.

Il n'existe pas de méthodes pour ajouter ou enlever un élément à un tuile. On peut cependant utiliser des méthodes de concétanation et d'affectation pour obtenir un résultat similaire :

```
>>> a=(1,2,3,4)
>>> a=a+(5,)
>>> a
(1, 2, 3, 4, 5)
>>> a=a[1:] # méthode des slices
>>> a
(2, 3, 4, 5)
>>> a=a[:1]+(1,)+a[1:]
>>> a
(2, 1, 3, 4, 5)
```

2. Pour s'entraîner.

Question 1

On définit un tuple t rempli de 0. Ce tuple est un tuple de tuples, tous les sous-tuples étant de même longueur :

```
tuple=((0,0,0,0,0,...,0),
           (0,0,0,0,0,...,0),
           (0,0,0,0,0,...,0),
           (0,0,0,0,0,...,0),
           (0,0,0,0,0,...,0),
           ....
           (0,0,0,0,0,...,0)
           )
```

Déterminer l'instruction qui permet de déterminer le nombre de 0 contenus dans ce tuple :

Question 2

Ecrire une fonction permettant de transformer un temps donné en (h,mn,s) en s:

```
def temps_secondes(temp):
  '''
  retourne le temps en secondes correspondant au temps ( h ,mn, s )
  : temp : (h,mn,s) tuple
  : return : temps en secondes, int

  >>>temps_secondes((1,10,5))
  4205
  '''
```

Question 3

Ecrire une fonction permettant de transformer un temps donné en secondes en (h,mn,s) :

```
def temps_h_mn_s(temp):
  '''
  retourne le temps donné en secondes en temps ( h ,mn, s )
  : temp : temps en secondes, int
  : return : (h,mn,s) tuple

  >>>temps_h_mn_s(4205)
  (1,10,5)
  '''
```

Question 4

Après les résultats d'une course, on doit traiter des données qui sont sous la forme suivante :

```
donnees=(
  # ( nom du candidat, temps sous la forme (h,mn,s))
    ("Paul",(2,3,45)),
    ("Sandrine",(1,10,5)),
    ("Jacques",(2,0,15))
    )
```

1. Ecrire une fonction permettant de transformer les données pour créer un tuple de nouvelles données sous la forme :

   ```
   def temps_en_s(var):
     '''
     transforme les temps des coureurs en temps en secondes
     : return : tuple

     >>>temps_en_s(donnees)
      (('Paul', 7425), ('Sandrine', 4205), ('Jacques', 7215))
      '''
   ```

2. Ecrire une fonction permettant d'obtenir sous la forme d'un tuple l'ensemble des temps mis par les candidats :

```
def liste_temps(var):
    ''' retourne les temps sous la forme d'un tuple:

    >>>course=(('Paul', 7425), ('Sandrine', 7102), ('Jacques', 7215))
    >>>liste_temps(course)
    (7425,7102,7215)
    '''
```

3. Ecrire une fonction permettant d'obtenir la valeur minimale et l'indice correspondant d'un tuple d'entiers :

```
def val_min(var):
  '''
  retourne la valeur minimale et l'indice correspondant

  >>>A=(2,3,6,2,1,6)
  >>>val_min(A)
  (1,4)

  '''
```

4. Ecrire une fonction permettant d'obtenir le nom du candidat ayant fini la course en premier :

```
def gagnant(var):
  '''
  retourne le nom du gagnant de la course
  : return : str

  >>> gagnant(donnees)
  'Sandrine'
  '''
```