1. Utilisation comme retour multiple dans les fonctions

Une fonction peut renvoyer plusieurs données rangées dans un tuple de la taille nécessaire.

Ex 1 fonction qui renvoie le résultat de la recherche dans une liste sous la forme d'un booléen et l'indice de l'élément trouvé.

```
def find(data, element):
    i = 0
    while i < len(data):
        if data[i] == element:
            return True, i
        i += 1
    return False, None

valeurs = [1, 2, 3, 4]

print(find(valeurs, 2))
print(find(valeurs, 6))
```

Résultat :

(True, 1)

(False, None)

Ex2 fonction qui renvoie le périmètre et la surface d'un même disque

```
import math

def cercle(rayon):
    p = 2*math.pi*rayon
    s = math.pi * math.pow(rayon,2)
    return p,s  #retourne un tuple

rayon = float(input("Veuillez donner le rayon de votre disque : "))
#le retour est rangé dans un tuple de même taille pour être exploité
perimetre,surface = cercle(rayon)
print("Périmètre disque : {0:.2f} Surface disque : {1:.2f}".format(peri-
metre,surface))
```

Résultat :

Veuillez donner le rayon de votre disque : 10

Périmètre disque : 62.83 Surface disque : 314.16

2. Les tuples ou n-uplets nommés

Pour accéder à un élément d'un tuple, il faut utiliser son indice. Dans le cas de tuples hétérogènes, ce n'est pas toujours très intuitif. Prenons l'exemple d'un article d'un magasin :

Ex :

```
item = 5449000000996, "Coca-Cola 33cl", 0.70
print("ref:",item[0],"désignation : ",item[1],"prix : ",item[2])
```

Pour rendre plus explicite qu'on a un tuple hétérogène dont chaque élément représente un *attribut* différent, on peut utiliser un *tuple nommé*. Pour cela, il faut avant tout déclarer un nouveau type de tuple nommé, avec `namedtuple`, en déclarant un nom et une liste d'attributs.

`collections` — Types de données de conteneurs

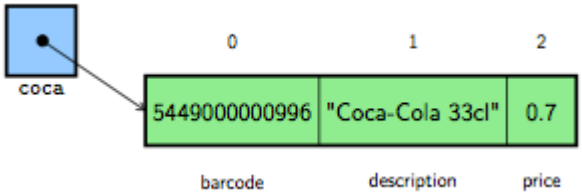
`namedtuple()` Fonction permettant de créer des sous-classes de `tuple` avec des champs nommés

```
from collections import namedtuple
Item = namedtuple('Item', ['ref', 'designation', 'prix'])
```

Une fois le nouveau type de tuple nommé créé, on peut créer des tuples nommés comme suit :

```
coca = Item(5449000000996, "Coca-Cola 33cl", 0.70)
```

La variable `coca` contient un tuple nommé.



L'avantage apporté par les tuples nommés est qu'on leur nom avec l'*opérateur d'accès* (.). On fait donc le nom de l'attribut, séparé par un point.

```
print(coca.designation)
PrixPackdeSix = 6 * coca.prix
print("prix du pack de six : {:.2f}".format(PrixPackdeSix))
```

```
Coca-Cola 33cl
prix du pack de six : 4.20
```

Toutes les opérations précédemment vues sur les tuples sont également applicables sur les tuples nommés. On peut, par exemple, écrire les instructions suivantes :

```
print(coca)
print(len(coca))
print(coca[1])
print(coca[1:3])
```

```
Item(ref=5449000000996, désignation='Coca-Cola 33cl', prix=0.7)
3
Coca-Cola 33cl
```