Les tuples et les dictionnaires sont deux autre types construits, similaire aux tableaux mais avec des propriétés différentes.

Partie I : Les tuples (CQFR !)

Un tuple est un **ensemble d'éléments qui ne peuvent pas être modifiés** (on dit que le tuple est un objet *immutable*), dont chaque élément est **indexé**.

On peut se représenter un tuple comme une armoire dont les niveaux sont indexés (numérotés), mais qui est fermée à clef (on ne peut pas modifier le contenu).

La syntaxe pour définir un tuple *T* est :
`T = (élément, élément,..., élément, élément)` ou juste `T = élément, élément,..., élément, élément`
Chaque élément du tuple peut être retrouvé grâce à son indice : `T[indice]`

La plupart des commandes utilisées pour les tuples sont identiques à celles des tableaux. Par exemple, il est possible de répéter un tuple, de concaténer plusieurs tuples ou de vérifier l'appartenant d'un élément à un tuple.

Exemples : Avec T1 = "a", "b" et T2 = "c", "d"
- *T1*2 répètera 2 fois le tuple et on aura alors ("a", "b", "a", "b").*
- *T1 + T2 ajoutera les éléments de T2 à la suite de T1 et on aura alors ("a", "b", "c", "d")*
- *element in Tuple vérifira si l'élément existe dans le tuple*

MAIS, contrairement aux tableaux, les éléments d'un tuple ne sont pas modifiables.
Exemple : Si on fait T[0]=Z, un message d'erreur s'affiche

Grace aux tuples, une fonction peut renvoyer plusieurs valeurs, et il est possible d'assigner en une ligne la valeur de chaque élément d'un tuple à une variable : `Var1, Var2 = (élément, élément)`

Exemple : Dans l'algorithme ci-dessous, la fonction TestTuple prend en argument un entier n et renvoi un tuple contenant deux éléments : n+1 et n-1.
On assigne ensuite à des variables Inc et Dec le résultat de la fonction pour un entier donné (ici, 5).

```
def FONCTION Operations(n, un entier)
    RENVOYER n+1, n-1
Inc, Dec = Operations (5)
```

A. Pour débuter ensemble

I.1. Le code suivant, qui est censé prendre en argument un entier *n* et afficher le tuple *Addition*, donne un message d'erreur. Expliquer pourquoi et corriger le code.

```
1 | def Operations(n):
2 |     return n+1, n-1
3 | #on ajoute 2 aux résultats
4 | Addition = Operations(2) + 2
5 | print(Addition)
```

B. Exercices pour s'entrainer

Pour chacun des exercices suivants, écrire le code qui réalise la tâche demandée puis le tester.

I.2. Écrire une fonction `PlusFois` prenant en arguments deux entiers *a* et *b* et renvoyant leur somme et leur produit.
Ajouter, en dehors de la fonction, l'affichage du résultat de la somme divisée par le produit pour $a=1$ et $b=2$.

I.3. Écrire une fonction `LongueurCotes` prenant en arguments les coordonnées des sommets A, B et C d'un triangle sous forme de 3 tuples (chaque tuple correspondant à (x, y) pour chaque sommet) et renvoyant un tuple contenant les distances AB, BC et AC entre chaque points du triangle.

Aide : La distance entre un point A = (x_A, y_A) et un point B = (x_B, y_B) est donnée par $AB=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$
La √ est une fonction nommée `sqrt` *qui se trouve dans la bibliothèque* `math`.

Partie II : Les dictionnaires (CQFR !)

Un dictionnaire est un **ensemble d'entrées associant une clef et une valeur**, sans indexation. Les clefs sont des nombres ou des chaines de caractères, et les valeurs peuvent être de tout type.

On peut se représenter un dictionnaire comme une commode avec des tiroirs : chaque tiroir (clef) contient une valeur.

Un dictionnaire *D* se défini sous la forme :

D = {clef :valeur, clef :valeur,..., clef :valeur, clef :valeur}

Pour créer un dictionnaire, on crée la variable puis on ajoute une valeur en l'associant à une clef :

```
D={}
D[clef]=valeur
```

Chaque valeur du dictionnaire peut être retrouvée grâce à sa clef : `D[clef]`

Pour supprimer une entrée, on utilise : `del D[clef]`

Pour vérifier si une clef existe dans un dictionnaire, on peut utiliser : `if clef in D`

Il est possible de parcourir un dictionnaire à l'aide d'une boucle for. Ce parcours peut se faire selon les clés ou les valeurs :

- `for i in dico.keys():` Parcourir les clefs
- `for elem in dico.values():` Parcourir les valeurs

A. Pour débuter ensemble

II.1. Que va afficher le code suivant ?

```
1 Liste_courses = {}
2 Liste_courses["pommes"] = 3
3 Liste_courses["bananes"] = 6
4 print("Je vais acheter ", Liste_courses["pommes"]," pommes et, " Liste_courses["bananes"]," bananes")
5 Liste_courses["pommes"] = Liste_courses["pommes"] + 2
6 Liste_courses["bananes"] = Liste_courses["bananes"] / 2
7 print("Je vais acheter ", Liste_courses["pommes"]," pommes et ", Liste_courses["bananes"]," bananes")
```

B. Exercices pour s'entrainer

II.2. On considère le contenu d'un dressing, représenté par un dictionnaire :

`dressing = {"pantalons": 3, "pulls": 4, "tee-shirts": 8}`

a. Ajouter 5 chemises au dictionnaire `dressing`.

b. Écrire une procédure (une fonction sans `return`) `Ajout` qui prend en argument un type d'habit et qui augmente son nombre de 1 dans le dictionnaire.

c. Améliorer la fonction pour qu'elle crée une nouvelle entrée si le type d'habit n'existe pas comme clé.

II.3. a. Écrire une fonction `IncremDico` prenant en argument un dictionnaire et une clef, et qui :

- incrémente la valeur de l'entrée si la clef existe déjà dans le dictionnaire
- crée l'entrée avec la valeur initialisée à 1 dans le cas contraire.

La fonction renvoie alors le dictionnaire modifié.

b. À l'issue d'une élection à scrutin uninominal (où on ne peut voter que pour une personne), on récupère un tableau contenant tous les noms inscrits sur les bulletins trouvés dans l'urne. On veut déterminer le vainqueur de l'élection, en un seul parcours de l'urne, mais on ne connait ni le nombre de candidats, ni leur nom.

En utilisant un la fonction `IncremDico` écrite précédemment, écrire une fonction `Depouillement` prenant en argument la liste des votes et qui renvoie le nom du vainqueur.

II.4. À l'aide d'un parcours du dictionnaire**,** créer le dictionnaire «inversé» `dico_inverse`, c'est-à-dire en échangeant clés et valeurs, du dictionnaire suivant :

`dico = {'L': 'S', 'C': 'D', 'M': 'V', 'R': 'M', 'CA': 'T', 'GR': 'EP', 'HE': 'AD', 'CH': 'MOD'}`