Types construits (Programmation 05)

I – Types construits

Les types construits sont les types qui permettent de stocker plusieurs autres types de données.

II – Tableaux statiques

1 - Définition d'un tableau statique

Un tableau statique est un tableau
1. comportant un nombre de cases fixé et
2. dont chaque case contient le même type.

Les cases sont identifiées par un numéro qu'on nomme souvent **indice** (index en anglais).
La première case est celle d'indice 0.

Indice	0	1	2
Elément	'A'	'B'	'C'

2 - Déclaration d'un tableau (crochets)

La syntaxe consiste à utiliser :
- [en ouverture,] en fermeture et
- des virgules pour séparer les valeurs.

```
>>> notes = [20, 8, 18, 12]
>>> type(notes)
<class 'list'>
```

3 - Lecture avec la syntaxe Python

L'accès à la valeur stockée dans une case consiste à taper **nom[indice]**.

```
Indice       0   1   2   3
>>> notes = [20, 8, 18, 12]
>>> notes[0]
20
>>> notes[1]
8
>>> notes[2]
18
>>> notes[4]
IndexError: list index out of range
```

4 - Déterminer la longueur d'un tableau

La longueur correspond au nombre de "cases" du tableau. Pour connaître le nombre de cases d'un tableau, on utilise la fonction native **len()**.

```
>>> notes = [20, 8, 18, 12]
>>> len(notes)
4
```

5 - Modifier une case d'un tableau

En Python, les tableaux ont un avantage majeur : on peut **modifier le contenu des cases** après la création. On dit que les tableaux statiques sont **muables**.

Pour cela, il suffit de faire une nouvelle affectation de valeur sur la case correspondante.

```
Indice       0   1  2   3
>>> notes = [20, 8, 18, 12]
>>> notes[1] = 11
>>> notes
[20, 11, 18, 12]
```

III – N-uplets / Tuples

1 - Définition d'un n-uplet

Un n-uplet est une séquence ordonnée d'éléments de n éléments où tous les éléments peuvent avoir un type différent, ou pas.
La première case est celle d'indice 0.

Indice	0	1	2
Elément	'A'	5	False

2 - Déclaration d'un n-uplet (parenthèses)

La syntaxe consiste à utiliser :
- (en ouverture,) en fermeture et
- des virgules pour séparer les valeurs.

```
>>> notes = (20, 8, 18, 12)
>>> type(notes)
<class 'tuple'>
```

3 - Lecture avec la syntaxe Python

On peut accéder à la valeur stockée dans une case en tapant simplement le nom de la variable-tuple suivi de **crochets** et de l'indice voulu.

```
Indice             0          1       2
>>> eleve = ("In Bordeland", "Alice", 18)
>>> eleve[0]
'In Bordeland'
>>> eleve[3]
IndexError: tuple index out of range
```

4 - Déterminer la longueur d'un n-uplet

Pour connaître le nombre de cases d'un n-uplet, on utilise la fonction native **len().**

```
>>> notes = (20, 8, 18, 12)
>>> len(notes)
4
```

5 - Modifier une case d'un tuple

En Python, le type tuple n'est pas modifiable après création. On dit que le type tuple est **immuable** : on ne peut pas modifier le contenu du tuple après création.

Imaginons qu'on veuille mettre à jour l'élève le jour de ses 19 ans :

```
>>> eleve = ("In Bordeland", "Alice", 18)
>>> eleve[2] = 19
TypeError: 'tuple' object does not support item assignment
```

Attention, immuable ne veut pas dire que c'est une constante. On peut modifier la variable dans sa totalité.

```
>>> eleve = ("In Bordeland", "Alice", 18)
>>> eleve = ("In Bordeland", "Alice", 19)
>>> eleve[2]
19
```

Immuable veut dire qu'on ne peut pas modifier une partie du contenu sans recréer tout le contenu.

IV – Dictionnaires

1 - Définition d'un dictionnaire

Les cases sont identifiées par un identifiant quelconque qu'on nomme une **clé** (key en anglais).
Voici un dictionnaire décrivant le contenu d'un sac exemple

Clé	"Cahier"	"Crayon"	"Blouse"
Valeur	3	"usé"	"neuve"

2 - Déclaration d'un dictionnaire (accolades)

- Les éléments délimitateurs sont les accolades {}
- On fournit un couple **cle: valeur**
- Chaque couple est séparé des autres par une virgule.

```
>>> sac = {"Cahier": 3, "Crayon": "usé", "Blouse": "neuve"}
>>> type(sac)
<class 'dict'>
```

3 - Lecture avec la syntaxe Python

```
>>> sac["Crayon"]
'usé'
>>> sac["dm pour aujourd'hui"]
KeyError: "dm pour aujourd'hui"
```

4 - Déterminer la longueur d'un dictionnaire

On peut utiliser la fonction native **len()** pour obtenir le **nombre de couples (clé, valeur)** enregistrés.

```
>>> sac = {"Cahier": 3, "Crayon": "usé", "Blouse": "neuve"}
>>> nbr = len(sac)
>>> nbr
3
```

→ le dictionnaire contient bien **3 couples** (clé, valeur), et pas 6 éléments indépendants.

5 - Rajouter ou modifier un couple (clé, valeur)

Le type dict de Python est **muable**.

On peut facilement rajouter des couples (clé, valeur) dans un dictionnaire, **contrairement aux tableaux statiques** dans lesquels le nombre de cases est fixé à la création.

Par contre, attention à la syntaxe : on utilise bien les crochets et pas des accolades !

```
>>> sac = {"Cahier": 3, "Crayon": "usé", "Blouse": "neuve"}
>>> sac
{"Cahier": 3, "Crayon": "usé", "Blouse": "neuve"}
>>> sac['paquet de biscuits'] = 12
>>> sac
{"Cahier": 3, "Crayon": "usé", "Blouse": "neuve", 'paquet de biscuits': 12}
>>> len(sac)
4
```

→ Le dictionnaire contient bien maintenant **4 couples** (clé, valeur).

Résumé des 3 conteneurs (à écrire sur votre propre feuille)

	Tableau statique	N-uplet	Dictionnaire
Création	**Crochets []**	**Parenthèses ()**	**Accolades {}**
Lecture	`nom[indice]`	`nom[indice]`	`nom[clé]`
Modification	Muable `nom[indice] = valeur`	Immuable	Muable `nom[clé] = valeur`
Avantage	Modification des valeurs Prend peu de place mémoire	Prend peu de place mémoire Types différents dans les cases	Modification des valeurs Rajout de couples Suppression de couples Types différents
Désavantage	Nombre de cases fixe ; pas de rajouts ou de suppression Même type dans les cases	Immuable donc pas de rajout ou de suppression	Place mémoire importante