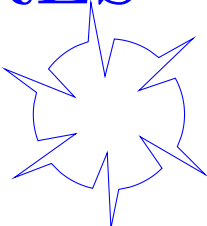


CHAPITRE
<h1>8</h1>
<h1>DICTIONNAIRES</h1>


## Table des matières

---

1	Introduction .....	2
2	Les p-uplets ou tuple... ..	2
2.1	Création d'un tuple .....	2
2.2	Opérations : .....	2
2.3	Appartenance : .....	3
2.4	Utilisation des indices : .....	3
3	Dictionnaires :.....	5
3.1	Création : .....	5
3.2	Accès aux éléments : .....	5
3.2.1	Accès en lecture .....	5
3.2.2	Accès en écriture .....	5
3.3	Accès à tous les éléments d'un dictionnaire .....	6
3.4	Appartenance .....	6
4	Exercices .....	7

# 1 Introduction

---

Nous avons découvert plusieurs types de données dans des contextes variés :

- des variables de type `int` pour stocker les entiers ;
- des variables de type `float` pour représenter des nombres décimaux ;
- des variables de type `string` qui contiennent des chaînes de caractères et qui se déclarent à l'aide de guillemets ou apostrophes.

Ces types simples ne suffisent pour garder en mémoire un grand nombre de valeurs comme dans le traitement de données par exemple : ceci amène la définition de `liste` de type `list` qui a fait l'objet d'une leçon particulière.

Mais il existe d'autres objets appelés **types construits** que nous allons définir ici.

## 2 Les p-uplets ou tuple...

---

Un objet de type **tuple** est une **suite ordonnée d'éléments** qui peuvent être chacun de n'importe quel type. On parlera indifféremment de tuple ou p-uplet. Ces types sont **non modifiables** ; on dit qu'ils sont **non mutables**, comme les chaînes de caractères d'ailleurs. Et c'est la différence profonde avec les listes qui elles sont modifiables...

### 2.1 Création d'un tuple

Pour créer un p-uplet non vide, on écrit  $n$  valeurs **séparées** par des **virgules**. Par exemple :

```
1 t1 = "a", "b", "c", 3 #tuple à 4 éléments
2 t2 = (2, 5) #tuple à 2 éléments
3 t3 = "c", (4, -1) #tuple à 2 éléments
```

Autres possibilités : utiliser des parenthèses :

```
1 t4 = (1, 2, 3, 10) #tuples de 4 éléments
2 t5 = (False, True) #tuples à 2 éléments
```

### 2.2 Opérations :

Il existe deux opérateurs de concaténations qui s'utilisent comme sur les chaînes de caractères.

---

```
>>> t1 = "a","b"
>>> t2 = "c","d"
>>> t1 + t2
('a', 'b', 'c', 'd')
>>> 3*t1
('a', 'b', 'a', 'b', 'a', 'b')
```

---

## 2.3 Appartenance :

Pour tester l'appartenance d'un élément à un tuple, on utilise l'opérateur `in` :

---

```
>>> t = "a", "b", "c"
>>> "a" in t
True
>>> "d" in t
False
```

---

## 2.4 Utilisation des indices :

Les indices permettent d'accéder aux différents éléments d'un tuple. La syntaxe `t[i]` permet d'accéder à l'élément d'indice `i` du tuple `t`. L'indice `i` peut prendre des valeurs entières entres 0 et  $n - 1$  où  $n$  est la longueur du tuple. Cette longueur s'obtient avec la fonction `len`. Par exemple :

---

```
>>> t = "a", 1, "b", 2, "c", 3
>>> len(t)
6
>>> t[2]
'b'
>>> t[-1]
3
>>> tp1 = (5,4), (-1,0)
>>> tp1[0]
(5, 4)
>>> tp1[1][0]
-1
```

---

Remarquons enfin qu'on ne peut pas modifier un tuple :

---

```
>>> t = (5, 10, 15)
>>> t[1] = 0 #tentative pour changer le 10 en 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

---



Les tuples ne diffèrent des listes que dans leur impossibilité de les modifier, comme les chaînes de caractères.

Sinon, l'interface est quasiment la même!

### Exercice n°1

Écrire un programme Python qui a partir de deux tuples représentant les coordonnées de deux points, retourne un tuple formé des coordonnées du milieu de ces deux points. Par exemple si les deux tuples sont (5,2) et (-1,7) alors le programme affiche le tuple (2,4.5)

### Exercice n°2

Un serveur reçoit un tuple de deux chaînes de caractères représentant des nombres. Il doit renvoyer au client la somme de ces deux nombres. Par exemple, il reçoit ("10", "8") et doit renvoyer "18". Écrire un programme qui permet de réaliser cette tâche.

## 3 Dictionnaires :

---

Les éléments d'une liste sont repérés par des indices 0,1,2,... Une différence essentielle avec les dictionnaires est que les indices sont remplacés par des objets de types `str`, `int`, `float`, `tuple`, ... On les appelle des **clés** et à chaque clé correspond une **valeur**. On parle aussi de **tableau associatif** pour désigner un dictionnaire : à un élément quelconque on lui associe une valeur !

### 3.1 Création :

Les éléments d'un dictionnaire sont des couples **clés-valeurs** ; il est défini avec des accolades, les différents couples étant séparés par des virgules. La clé et sa valeur sont séparés par deux points: .

```
1 dico = {"A": 0, "B": 1, "C": 2}
2 dico_age = {"Jean": 34, "Sacha": 15, "Nathalie": 24}
```

On peut aussi construire un dictionnaire par compréhension :

```
1 alphabet = {chr(65 + i) : i for i in range(26) }
```

### 3.2 Accès aux éléments :

#### 3.2.1 Accès en lecture

Pour accéder aux clés ou aux valeurs ,on utilise les méthodes `keys` et `values` ; la méthode `items()` permet l'affichage de tous les couples (clé,valeurs).

---

```
>>> dico_age = {"Jean":34, "Sacha":15 ,"Nathalie":24}
>>> dico_age.keys()
dict_keys(['Jean', 'Sacha', 'Nathalie'])
>>> dico_age.values()
dict_values([34, 15, 24])
>>> dico_age.items()
dict_items([('Jean', 34), ('Sacha', 15), ('Nathalie', 24)])
```

---



Les objets obtenus sont des itérables qu'on peut parcourir à l'aide de boucles!

L'accès à une valeur particulière s'obtient en précisant la clé :`dico_age["Jean"]` a la valeur 34.

#### 3.2.2 Accès en écriture

Nous pouvons modifier une valeur par affectation :

---

```
>>> dico_age = {"Jean": 34, "Sacha": 15, "Nathalie": 24}
>>> dico_age["Jean"] = 35
>>> dico_age["Arthur"] = 12
>>> dico_age
{'Jean': 35, 'Sacha': 15, 'Nathalie': 24, 'Arthur': 12}
```

---

---

### 3.3 Accès à tous les éléments d'un dictionnaire

Un dictionnaire est un objet itérable que l'on peut parcourir à l'aide d'une boucle :

```
1 atp = {"Sinner":11830, "Zverev": 7915, "Alcaraz": 7010, "Fritz": 5100,  
→ "Medvedev":5030}
```



Comment ajouter un couple au dictionnaire ?

Par exemple, le sixième au classement est Ruud avec 4255 points. Pour l'ajouter au dictionnaire, il suffit de faire :

```
1 atp["Ruud"] = 4255
```

qui a pour effet d'ajouter au dictionnaire `atp` la *clé* "Ruud" avec la *valeur* 4255.



Comment afficher toutes les clés ?

À l'aide d'une boucle :

```
1 for cle in atp:  
2     print(cle)
```

La variable `cle` parcourt toutes les clés.



Comment afficher toutes les valeurs ?

Idem, à l'aide d'une boucle :

```
1 for cle in atp:  
2     print(atp[cle])
```



Comment modifier les valeurs de toutes les clés ?

Par exemple, je veux ajouter 10 points au classement à tout le monde :

```
1 for cle in atp:  
2     atp[cle] = atp[cle] + 10
```

### 3.4 Appartenance

Nous pouvons tester l'appartenance à un dictionnaire :

---

```
>>> dico_age = {"Jean":34,"Sacha":15,"Nathalie":24}
>>> "Jean" in dico_age
True
>>> 20 in dico_age.values()
False
>>> ("Sacha", 15) in dico_age.items()
True
```

---

## 4 Exercices

---

### Exercice n°3

- ① Qu'obtient-on si on tape `type(animaux)` dans une console python ?
- ② Qu'obtient-on si on tape `animaux[1]` dans une console python ?
- ③ De quel type sont les éléments de la liste `animaux` ?
- ④ Donner un programme qui ajoute le perroquet Jano qui a 3 ans.
- ⑤ Donner un programme qui affiche les noms de tous les animaux.
- ⑥ Donner un programme qui affiche les noms des chats.

```
>>> animaux = [{"Nom":"philo", "Animal":"chien", "Age":2}
...           ,{"Nom":"zeus", "Animal":"chat", "Age":6}
...           ,{"Nom":"pedro", "Animal":"pigeon", "Age":1}
...           ,{"Nom":"gaia", "Animal":"chat", "Age":7}
...           ]
```

#### Exercice n°4

On considère le dictionnaire ci-dessous où les clés sont des noms de participant à un jeu et les valeurs sont les scores obtenus. Dans chaque cas écrire les programmes qui permettent de réaliser les événements suivants.

- ① Ajouter 1 à chaque valeur de score ;
- ② Ajouter au dictionnaire le score de Bruno qui a obtenu 17 ;
- ③ Afficher toutes les clés ;
- ④ Afficher toutes les valeurs ;
- ⑤ Afficher le nom de celui qui a le plus gros score.

```
>>> participant = {"Jean": 34, "Sacha": 15, "Nathalie": 24, "Patrice": 52}
```



### Exercice n°5

On considère la liste `chiffres` ci-dessous. À l'aide d'un boucle, écrire le programme qui permet à partir de cette liste de construire le dictionnaire `dico` suivant qui compte le nombre de lettre de chaque chiffre :

```
1 chiffres = ['zéro', 'un', 'deux', 'trois', 'quatre', 'cinq', 'six', 'sept', 'huit',  
↪ 'neuf']  
2 ...  
3 ...  
4 dico = {'zéro': 4, 'un': 2, 'deux': 4, 'trois': 5, 'quatre': 6, 'cinq': 4, 'six':  
↪ 3, 'sept': 4, 'huit': 4, 'neuf': 4}
```

### Exercice n°6

Un mot ou un texte étant proposé, donner les instructions Python qui permettent de construire un dictionnaire où les clés sont les caractères composants le mot ou le texte et les valeurs sont le nombre d'occurrences de la clé correspondante.

Par exemple :

```
1 texte = "bonjour, tout le monde!"  
2 compte = {'b':1, 'o':4, 'n':2, 'j':1, 'u':2, 'r':1, ',':1, ' ':3, 't':2, 'l':1,  
↪ 'e':2, 'm':1, 'd':1, '!':1}
```

### Exercice n°7

On considère le dictionnaire Python ci-dessous. Quelle instruction Python permet de :

- ❶ afficher le numéro de téléphone de Piotr.
- ❷ savoir si Fanny est enregistrée dans le répertoire ?
- ❸ de modifier le numéro de Patrick qui finit par un 9 et pas un 8 ?
- ❹ ajouter Raoul dont le numéro 0712151618 ?

```
1 repertoire = {"Patrick" : "0612345678",  
2             "Marie " : "0687654321",  
3             "Hanae " : "0765432198",  
4             "Piotr " : "0777666555"}
```

### Exercice n°8

Étant donnée un dictionnaire python dont les clés sont les noms des élèves et les valeurs sont les listes des notes.

Par exemple, `note = {"Aladin": [12, 15 , 17] , "Nathalie" : [15, 13 , 16] , "Robert": [13, 15 , 11] }`.

- ❶ Écrire un programme qui ajoute 1 point à la deuxième note note du devoir
- ❷ Écrire un programme qui remplace les listes de notes par leur moyenne.