Structures de données : listes, dictionnaires et tuples ¶

Listes

Une liste est un ensemble d'éléments de n'importe quel type.

In [4]:
```
liste = [2, 4.0, "chaîne", [2, 3]]
```

Une liste vide est une liste valide.

In [23]:
```
vide = []
```

On peut accéder aux éléments d'une liste grâce à l'opérateur d'indexation.

In [6]:
```
print(liste[1])
```

```
4.0
```

Les listes peuvent être modifiées.

In [7]:
```
liste[1] = 5.0
print(liste)
```

```
[2, 5.0, 'chaîne', [2, 3]]
```

Les listes peuvent être parcourues avec une boucle for.

In [8]:
```
for élément in liste:
    print(élément)
```

```
2
5.0
chaîne
[2, 3]
```

On peut aussi utiliser une boucle for pour modifier tous le éléments d'une liste.

In [9]:
```
nombres = [1, 2, 3]
for i in range(len(nombres)):
    nombres[i] = nombres[i] + 1
print(nombres)
```

```
[2, 3, 4]
```

L'opérateur + concatène des listes, et l'opérateur * répète une liste

In [10]:
```
a = [1, 2]
b = [3, 4]
print(a + b)
print(a * 2)
```

```
[1, 2, 3, 4]
[1, 2, 1, 2]
```

L'opérateur de tranche permet d'accéder à un partie de la liste.

In [11]:
```
l = [1, 2, 3, 4, 5, 6]
print(l[1:3])
print(l[:3])
print(l[2:])
```

```
[2, 3]
[1, 2, 3]
[3, 4, 5, 6]
```

L'opérateur de tranche peut être utilisé pour modifier plusieurs éléments à la fois

In [18]:
```
l[1:3] = [2.0, 3.0]
print(l)
```

```
[1, 2.0, 3.0, 4, 5, 6]
```

La méthode append ajoute un nouvel élément, et extend ajoute plusieurs éléments à la fois

In [20]:
```
l.append(7)
print(l)
l.extend([8, 9])
print(l)
```

```
[1, 2.0, 3.0, 4, 5, 6, 7]
[1, 2.0, 3.0, 4, 5, 6, 7, 8, 9]
```

La méthode sort trie les éléments par ordre croissant.

In [21]:
```
désordre = [2, 4, 3, 5, 1]
désordre.sort()
print(désordre)
```

```
[1, 2, 3, 4, 5]
```

Un map est une fonction qui crée une nouvelle liste en modifiant les éléments d'une autre.

In [27]:
```
lettres = ["a", "b", "c"]
def majuscule(l):
    res = []
    for e in l:
        res.append(e.capitalize())
    return res
LETTRES = majuscule(lettres)
print(LETTRES)
print(lettres)
```

```
['A', 'B', 'C']
['a', 'b', 'c']
```

Un filtre est une fonction qui crée une nouvelle liste à partir de certains éléments d'une autre.

In [28]:
```
exemple = ["a", "B", "C", "d", "E"]
def tri_maj(l):
    res = []
    for e in l:
        if e.isupper():
            res.append(e)
    return res
trié = tri_maj(exemple)
print(trié)
print(exemple)
```

```
['B', 'C', 'E']
['a', 'B', 'C', 'd', 'E']
```

La méthode pop supprime un élément et renvoie sa valeur, tandis que l'opérateur del supprime tout simplement un ou plusieurs éléments. Tous deux prennent comme argument l'indice de l'élément.

In [33]:
```
n = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(n.pop(8))
print(n)
del n[7]
print(n)
del n[5:7]
print(n)
```

```
9
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5]
```

La méthode remove supprime un élément en prenant comme argument sa valeur.

In [34]:
```
n.remove(5)
print(n)
```

```
[1, 2, 3, 4]
```

La fonction list transforme une chaîne de caractères en liste.

In [35]:
```
c = "caractères"
print(list(c))
```

```
['c', 'a', 'r', 'a', 'c', 't', 'è', 'r', 'e', 's']
```

Dictionnaires

Un dictionnaire est comme une liste, mais les indices (appelés clés) peuvent être de tous types. Les éléments sont des couples clé/valeur.

In [2]:
```
fr_en = {"un":"one", "deux":"two", "trois":"three"}
```

Quand un dictionnaire est imprimé, l'ordres des éléments est aléatoire.

In [37]:
```
print(fr_en)
```

```
{'deux': 'two', 'un': 'one', 'trois': 'three'}
```

L'indice d'un élément n'est pas un nombre comme pour les listes mais sa clé.

In [38]:
```
print(fr_en["deux"])
```

```
two
```

L'opérateur in renvoie True si l'élément apparaît comme une clé.

In [3]:
```
print("un" in fr_en)
print("one" in fr_en)
```

```
True
False
```

La méthode values crée une liste composée des valeurs des éléments.

In [4]:
```
print(fr_en.values())
```

```
dict_values(['one', 'three', 'two'])
```

Tuples

Un tuple est une suite d'éléments. Il peut s'écrire avec ou sans parenthèses.

In [5]:
```
t1 = "a", "b", "c"
t2 = ("d", "e", "f")
```

Un tuple d'un seul élément doit contenir une virgule finale.

In [7]:
```
a = "a",
print(type(a))
b = "b"
print(type(b))
```

```
<class 'tuple'>
<class 'str'>
```

La différence avec une liste est qu'un tuple ne peut pas être modifié.

In [8]:
```
t1[0] = "A"
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-8-4660a97433f0> in <module>()
----> 1 t1[0] = "A"

TypeError: 'tuple' object does not support item assignment
```

In [ ]:
```
1
```