



Les tuples et les listes



1. les tuples

Un objet de type tuple (p-uplet), est une suite ordonnée de p éléments qui peuvent être chacun de n'importe quel type.

Code Python	Résultat affiché dans la console
<pre>a=(1,2) print (type(a)) print(a)</pre>	<pre><class 'tuple'> (1, 2)</pre>
<pre>a=(1,'bonjour') b=a*2 print(b)</pre>	<pre>(1, 'bonjour', 1, 'bonjour')</pre>
<pre>tuple_1 = (5, 2, 25, 56) tuple_2 = ("jack","tony") tuple_3 = tuple_1 + tuple_2 print(tuple_3)</pre>	<pre>(5, 2, 25, 56, 'jack', 'tony')</pre>
<pre>couleur=(255,128,64) rouge=couleur[0] vert=couleur[1] bleu=couleur[2] print(rouge,vert,bleu)</pre>	<pre>255 128 64</pre>
<pre>tuple=('a',1,5.6,45,'toto') longueur=len(tuple) print("longueur du tuple",longueur) for i in range(0,longueur): print(tuple[i])</pre>	<pre>longueur du tuple 5 a 1 5.6 45 toto</pre>
<pre>tuple=('a',1,5.6,45,'toto') longueur=len(tuple) print("longueur du tuple",longueur) for valeur in tuple: print(valeur)</pre>	<pre>longueur du tuple 5 a 1 5.6 45 toto</pre>

Les parenthèses d'un tuple ne sont pas obligatoires mais conseillées.

Code Python	Résultat affiché dans la console
<code>a=(1,2) # ou a=1,2 a[0]=0</code>	<code>TypeError: 'tuple' object does not support item assignment</code>

Un tuple est **non mutable**, c'est-à-dire qu'il ne peut pas être modifié après sa création.

Code Python	Résultat affiché dans la console
<code>a=(1,2) b=(0,a[1]) print(a,b)</code>	<code>(1, 2) (0, 2)</code>

A quoi sert un tuple ?

Le tuple permet une affectation multiple.

Code Python	Résultat affiché dans la console
<code>couleur=(255,128,64) rouge=couleur[0] vert=couleur[1] bleu=couleur[2] print(rouge,vert,bleu)</code>	<code>255 128 64</code>
<code>couleur=(255,128,64) #emballage rouge,vert,bleu=couleur #déballage print(rouge,vert,bleu)</code>	<code>255 128 64</code>

Un tuple permet également de renvoyer plusieurs valeurs lors d'un appel d'une fonction.

Code Python	Résultat affiché dans la console
<code>def donne_moi_ton_nom(): tuple=("Roger", "Federer") return tuple print(donne_moi_ton_nom())</code>	<code>('Roger', 'Federer')</code>

Prochainement, nous verrons les tuples nommés.

<https://www.youtube.com/watch?v=n54Q-lccZEs>

2. Les listes à une dimension

Un objet de type list, que nous appelons une liste, est un ensemble ordonné d'éléments avec des indices pour les repérer. Les éléments d'une liste sont séparés par des virgules et entourés de crochets. Une liste est **mutable**, c'est-à-dire que l'on peut changer le contenu de la liste.

Code Python	Résultat affiché dans la console
<pre>lst=[45,12,4,78] print (type(lst)) print(lst)</pre>	<pre><class 'list'> [45, 12, 4, 78]</pre>
<pre>lst=[45,12,4,78] lst[1]=0 print(lst)</pre>	<pre>[45, 0, 4, 78]</pre>
<pre>lst=[45,12,4,78] longueur=len(lst) print("longueur de la liste",longueur) for i in range(0,longueur): print(lst[i])</pre>	<pre>longueur de la liste 4 45 12 4 78</pre>
<pre>lst=[45,12,4,78] for valeur in lst: print(valeur)</pre>	<pre>45 12 4 78</pre>

Initialisation de liste

Code Python	Résultat affiché dans la console
<pre>lst=[0] *10 print(lst) lst[2]=5 print(lst)</pre>	<pre>[0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [0, 0, 5, 0, 0, 0, 0, 0, 0, 0]</pre>
<pre>lst=[i for i in range(0,10)] print(lst)</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]</pre>
<pre>lst=[i*2 for i in range(0,10)] print(lst)</pre>	<pre>[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]</pre>
<pre>#ajout d'un élément dans la liste lst=[i for i in range(0,10)] print(lst) lst.append(50) print(lst)</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 50]</pre>

Accès à plusieurs données dans une liste.

Code Python	Résultat affiché dans la console
<pre>#affiche les derniers éléments en partant de l'indice 2 lst=[i for i in range(0,10)] print(lst) print(lst[2:])</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [2, 3, 4, 5, 6, 7, 8, 9]</pre>
<pre>#affiche les 3 premiers éléments lst=[i for i in range(0,10)] print(lst) print(lst[:3])</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2]</pre>
<pre>#affiche le 2ème élément en partant de la fin lst=[i for i in range(0,10)] print(lst) print(lst[-2])</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 8</pre>
<pre>#affiche les éléments entre l'indice 2 et l'indice 6 (exclu) lst=[i for i in range(0,10)] print(lst) print(lst[2:6])</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [2, 3, 4, 5]</pre>

Supprimer un élément

Code Python	Résultat affiché dans la console
<pre>lst=[i for i in range(0,10)] print(lst) lst.remove(5) print(lst)</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 6, 7, 8, 9]</pre>

Insérer un élément

Code Python	Résultat affiché dans la console
<pre>lst=[i for i in range(0,10)] print(lst) lst.insert(7,12) print(lst)</pre>	<pre>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 12, 7, 8, 9]</pre> <p style="text-align: right;">Indice 7</p>

<https://www.youtube.com/watch?v=AFRdL2hge0o>

3. Les listes à deux dimensions

Code Python	Résultat affiché dans la console
<pre>lst=[[1,2,3], [4,5,6], [7,8,9]] print(lst)</pre>	<pre>[[1, 2, 3], [4, 5, 6], [7, 8, 9]]</pre>
<pre>#accès à un élément du tableau lst=[[1,2,3], [4,5,6], [7,8,9]] ligne=1 colonne=2 print(lst[ligne][colonne])</pre>	<pre>6</pre>
<pre>#affichage sous forme de tableau lst=[[1,2,3], [4,5,6], [7,8,9]] for ligne in range(0,3): for colonne in range(0,3): print(lst[ligne][colonne],end=' ') print()</pre>	<pre>1 2 3 4 5 6 7 8 9</pre>
<pre>#initialisation d'un tableau à 2 dimensions ligne = 3 colonne = 4 lst = [[0] * ligne for i in range(colonne)] lst[1][1]=2 print(lst)</pre>	<pre>[[0, 0, 0], [0, 2, 0], [0, 0, 0], [0, 0, 0]]</pre>

Listes et tuples

- Liste** → suite indexée et modifiable d'éléments de tout type Attention : l'indexation commence à 0

NomListe = [élément1 , élément2 , élément3,...]
NomListe[i] → élément d'index i (lecture ou écriture)
NomListe[0] → premier élément
NomListe[-1] → dernier élément

- Principales fonctions :

Longueur : *len()* → renvoie le nombre d'éléments
Ajout : *NomListe.append(x)* → ajoute x en fin de liste
Insertion : *NomListe.insert(i, x)* → insert x à l'index i
Suppression : *NomListe.pop()* → supprime le dernier élément
NomListe.pop(i) → supprime élément d'indexe i

```
L=[5,8,"Julie"] # Liste de 3 éléments
a=L[0] # Lecture: a vaut 5
L[1] =10 # Ecriture: L vaut [5,10,"Julie"]
b=L[-1] # b vaut 'Julie'
x= len(L) # x vaut 3
c=L[3] # erreur dépassement
L.append(3) # L vaut [5,10,'Julie',3]
L.insert(2,"P") # L vaut [5,10,'P','Julie',3]
L.pop() # L vaut [5,10,'P',Julie']
L.pop(2) # L vaut [5,10,'Julie']
```

- Tuples** → un tuple est une liste non modifiable
NomTuple = (élément1 , élément2 , élément3,...)
NomTuple[i] → élément d'indexe i en lecture seule
- Liste de listes** → une liste peut contenir des listes !!
NomListe[i][j] désigne l'élément d'index j de la liste d'index i

```
T=(4,8,10)
x=T[0] # x vaut 4
T[0]=5 # erreur, T non modifiable
```

```
Tab=[[4,2,9] , [0,7,8]]
x=Tab[0] # x vaut [4,2,1] : index 0 de Tab
y=Tab[0][2] # y vaut 9 : index 2 de Tab[0]
```

Annexe

Visualiser un code pas à pas graphiquement

<http://pythontutor.com/visualize.html#mode=edit>

Code Python	Résultat affiché dans la console
<pre>liste=[5,6,7,8,9,4] somme=0 for n in range(len(liste)): somme=somme+liste[n] print(somme)</pre>	39

Write code in Python 3.6

```

1 liste=[5,6,7,8,9,4]
2 somme=0
3 for n in range(len(liste)):
4     somme=somme+liste[n]
5 print(somme)
6 |

```

Help improve this tool by completing a [short user survey](#)

Visualize Execution Live Programming Mode

Déroulement de l'exécution pas à pas

Python 3.6

```

1 liste=[5,6,7,8,9,4]
2 somme=0
→ 3 for n in range(len(liste)):
→ 4     somme=somme+liste[n]
5 print(somme)

```

[Edit this code](#)

→ line that has just executed
→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

Step 9 of 16

Print output (drag lower right corner to resize)

Frames

Global frame	
liste	
somme	18
n	2

Objects

list					
0	1	2	3	4	5
5	6	7	8	9	4