





01

## DÉFINITION SÉQUENCE

Qu'est ce qu'une  
séquence Python?

03

## EXEMPLES D'UTILISATION

Comment manipuler  
une liste?

05

## MÉTHODES

Quelles sont les  
méthodes de liste?

02

## DEFINITION LISTE

Qu'est-ce qu'une liste?

04


## OPÉRATEURS

Quels sont les opérateurs  
de liste de base?

06

## FONCTIONS

Quelles sont les  
fonctions de liste?

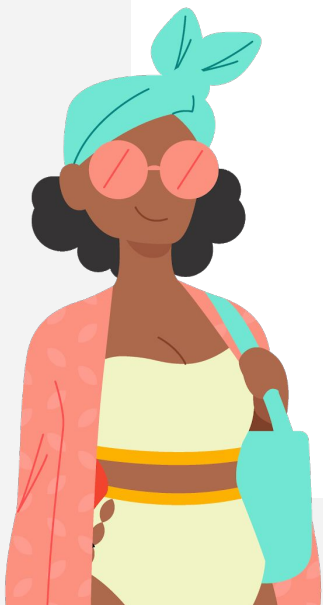




## DÉFINITION SÉQUENCE

La **structure de données principale** en Python est la **séquence**. Chaque élément d'une séquence possède un **index** qui définit sa **position** dans celle-ci. Le **premier index est zéro**, le **deuxième index est un**, et ainsi de suite.

Python possède **six types de séquences**, mais les plus courantes sont les **listes** et les **tuples**. Plusieurs **opérations** sont **communes** entre ces types dont, **l'indexation**, **le découpage**, **l'addition**, **la multiplication** et **la vérification de l'appartenance**. D'autres fonctions intégrées permettent de trouver la **longueur d'une séquence** et ses éléments les plus grands ou les plus petits.



A vibrant, stylized illustration of tropical foliage, including large green monstera leaves, pink and red palm fronds, and yellow ferns, set against a yellow background.

## DÉFINITION LISTE

La **liste** est un type de données très polyvalent qui permet de **stocker des valeurs séparées par des virgules entre crochets**. Les éléments d'une liste ne sont pas nécessairement du même type.

Pour accéder aux valeurs des listes, il suffit d'utiliser les crochets de découpage avec l'index pour obtenir la valeur disponible à cette position. Vous pouvez mettre à jour un ou plusieurs éléments de la liste en donnant la tranche à gauche de l'opérateur d'affectation. Pour supprimer un élément de la liste, vous pouvez utiliser l'instruction *"del"* si vous savez quel(s) élément(s) supprimer, ou la méthode *"remove()"* si vous ne le savez pas.





# EXEMPLES D'UTILISATION

Voici quelques exemples classiques de manipulation de listes :

- **Création** : Entre crochets, avec ou sans éléments de types potentiellement différents.
- **Accession** : La liste suivit d'une paire de crochets, contenant un index ou une chaîne d'indexes existants.
- **Mise à jour** : Remplace la valeur à la position défini par l'index de la liste par la nouvelle.
- **Suppression** : Si l'index n'est pas spécifié, la liste entière sera supprimée. S'il y a index, seulement la valeur à cette position sera retirée

```
list1 = ['apple', 'banana', 19, 20]
```

```
list2 = [1, 2, 3, 4, 5]
```

```
print("list1[0] : ", list1[0], "\nlist2[1:5] : ", list2[1:5])
```

```
print("list1[2] : ", list1[2])
```

```
list1[2] = 'cherry'
```

```
print("list1[2] : ", list1[2])
```

```
del list1[2]
```

```
print("del list1[2] : ", list1)
```

## # Resultats

```
list1[0] : apple
```

```
list2[1:5] : [2, 3, 4, 5]
```

```
list1[2] : 19
```

```
list1[2] : cherry
```

```
del list1[2] : ['apple', 'banana', 20]
```



# OPÉRATEURS

Les listes répondent aux opérateurs “+” et “\*” comme des chaînes de caractères. Ils signifient aussi **concaténation** et **répétition**, sauf que le résultat est une **nouvelle liste** et non une chaîne de caractères.

En fait, les listes répondent à toutes les **opérations de séquence générale** que nous avons utilisées sur les chaînes de caractères dans le chapitre précédent.

```
print("len([0, 1, 2]) : ", len([0, 1, 2]))
print("[1, 2, 3] + [4, 5, 6] : ", [1, 2, 3] + [4, 5, 6])
print("['apple'] * 3 : ", ['apple'] * 3)
print("3 in [1, 2, 3] : ", 3 in [1, 2, 3])
```

```
for x in ["apple", "banana", "cherry"]:
    print(x)
```

**Resultats :**

```
len([0, 1, 2]) : 3 # Longueur
[1, 2, 3] + [4, 5, 6] : [1, 2, 3, 4, 5, 6] # Concaténation
['apple'] * 3 : ['apple', 'apple', 'apple'] # Répétition
3 in [1, 2, 3] : True # Appartenance
apple # Itération 1
banana # Itération 2
cherry # Itération 3
```



# OPÉRATEURS

Comme les listes sont des **séquences**, l'**indexation** et le **découpage** fonctionnent de la même manière pour les listes que pour les chaînes de caractères.

- Le premier index est 0.
- Un index négatif, indique qu'il faut compter à partir de la droite.
- "x:y" permet d'indiquer une **section**. Si x est **absent**, la section commence avec le **premier élément** et si y est **absent**, la fin de la section est le **dernier élément** de la liste.

```
fruits = ['Mango', 'Pineapple', 'Papaya']
```

```
print('fruits[2] : ', fruits[2])  
print('fruits[-2] : ', fruits[-2])  
print('fruits[1:] : ', fruits[1:])  
print('fruits[:2] : ', fruits[:2])  
print('fruits[:] : ', fruits[:])
```

**Resultats :**

```
fruits[2] : Papaya  
fruits[-2] : Pineapple  
fruits[1:] : ['Pineapple', 'Papaya']  
fruits[:2] : ['Mango', 'Pineapple']  
fruits[:] : ['Mango', 'Pineapple', 'Papaya']
```



# MÉTHODES

Voici quelques méthodes pour manipuler les listes.

## **append(obj)**

Permet d'ajouter un élément à la liste.

## **count(obj)**

Compte le nombre de fois qu'un objet est présent dans la liste.

## **extend(seq)**

Ajoute le contenu d'une séquence à la liste.

```
listA = [123, 'apple', 'banana']  
listB = [456, 'cherry']
```

```
listA.append(123)  
print("Updated List : ", listA)  
print("Count for 123 : ", listA.count(123))  
print("Count for apple: ", listA.count('apple'))  
listA.extend(listB)  
print("Extended List : ", listA)
```

## **Resultats :**

```
Updated List : [123, 'apple', 'banana', 123]  
Count for 123 : 2  
Count for apple : 1  
Extended List : [123, 'apple', 'banana', 123, 456, 'cherry']
```





# MÉTHODES

## **index(obj)**

Permet d'obtenir l'index le plus bas de la liste où apparaît l'élément donné.

## **insert(index, obj)**

Insère l'objet donné dans la liste à l'index précisé.

## **pop(obj=list[-1])**

Supprime et retourne l'élément donné ou le dernier élément par défaut de la liste.

```
listA = [123, 'apple', 'banana']
```

```
print("Index for apple : ", listA.index('apple'))  
print("Index for banana: ", listA.index('banana'))  
listA.insert(3, 456)  
print("Final List : ", listA)  
print("List A : ", listA.pop())  
print("List A : ", listA)
```

## **Resultats :**

```
Index for apple: 1  
Index for banana: 2  
Final List : [123, 'apple', 'banana', 456]  
List A : 456  
List A : [123, 'apple', 'banana']
```



# MÉTHODES

## **remove(obj)**

Supprime l'élément donné de la liste.

## **reverse()**

Inverse les objets de la liste en place.

## **sort([func])**

Trie les objets de la liste, utilise la fonction de comparaison si elle est donnée.

```
listA = [123, 'apple', 'banana', 456]  
listB = ['pineapple', 'apple', 'mango', 'banana', 'cherry']
```

```
listA.remove('apple')  
print("List : ", listA)  
listA.reverse()  
print("List : ", listA)  
listA.sort()  
print("List : ", listA)
```

## **Resultats :**

```
List : [123, 'banana', 456]  
List : [456, 'banana', 123]  
List : ['apple', 'banana', 'cherry', 'mango', 'pineapple']
```



# FONCTIONS

## **len(list)**

Donne le nombre d'éléments contenus dans la liste.

## **max(list)**

Retourne l'élément de la liste ayant une valeur maximale.

## **min(list)**

Retourne l'élément de la liste ayant une valeur minimale.

## **list(seq)**

Convertit un tuple en liste.

```
list1, list2 = ['mango', 'apple', 'cherry'], [123, 'banana']  
tuple1 = (123, 'apple', 'banana', 'cherry')
```

```
print("First list length : ", len(list1))  
print("Second list length : ", len(list2))  
print("Max value element : ", max(list1))  
print("Min value element : ", min(list1))  
listA = list(tuple1)  
print("List elements : ", listA)
```

## **Resultats :**

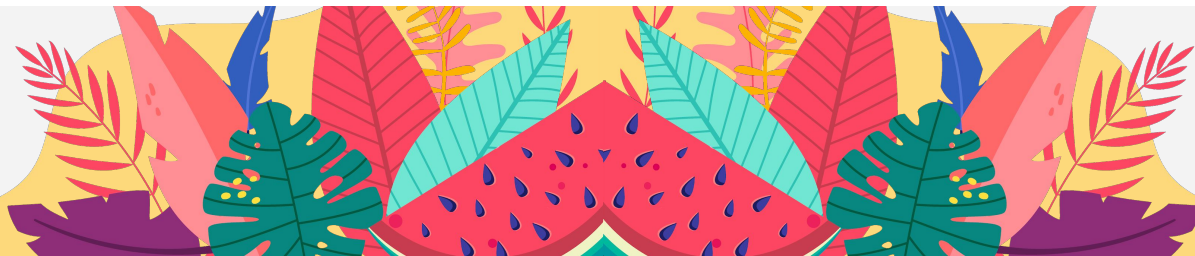
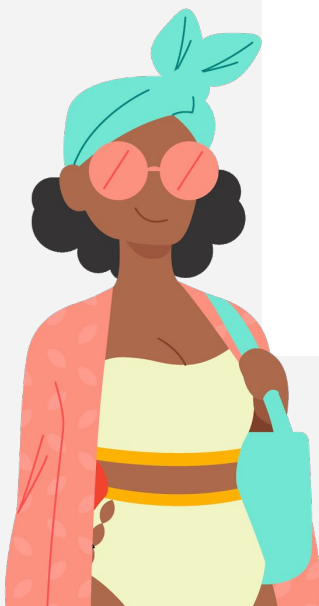
```
First list length : 3  
Second list length : 2  
Max value element : mango  
Min value element : apple  
List elements : [123, 'apple', 'banana', 'cherry']
```

The background of the slide is a vibrant, colorful pattern of various tropical leaves and plants. The leaves are in shades of green, yellow, orange, red, and purple. Some leaves are large and broad, while others are smaller and more delicate. The pattern is dense and covers the entire slide, with a white rectangular area in the center for the text.

# **EXERCICES**

# EXERCICES

1. Créez une liste contenant les entiers suivants : -1, 987, 452, 136
2. Additionnez tous les éléments de la liste précédente.
3. Affichez l'entier le plus grand et le plus petit de la liste.
4. Ajoutez à votre liste, la somme de tous les éléments de la liste et la somme de l'élément le plus grand et le plus petit de la liste.
5. Combien d'éléments possède la liste précédente?



# EXERCICES

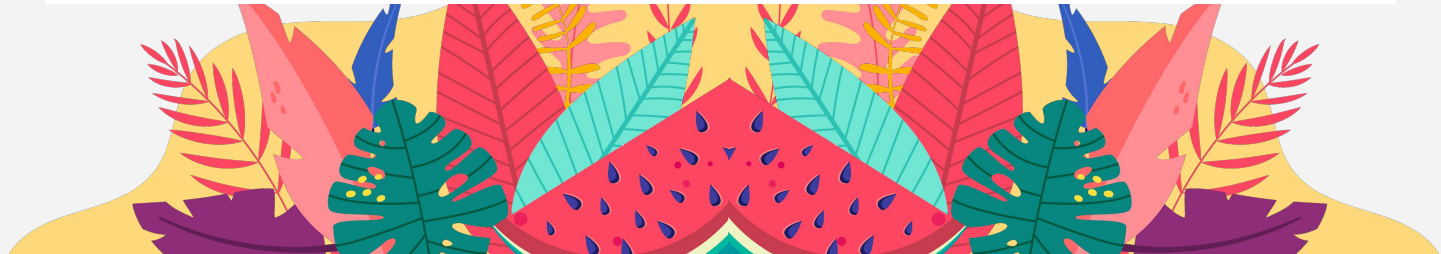
6. Triez la liste précédente dans l'ordre décroissant.

7. Supprimez la liste précédente.

8. Créez une nouvelle liste nommée "colors" contenant les éléments suivants et affichez chaque élément de la liste précédé par son index.

9. Supprimez l'élément en 1ere et 5eme position de la liste.

10. Renommez le dernier élément de la liste précédente par "Bleu".





The background of the entire image is a dense, colorful pattern of various tropical leaves and foliage. The leaves are rendered in a flat, stylized manner with bold outlines. Colors include bright red, teal, yellow, orange, and purple. Some leaves have detailed vein patterns, while others are solid colors. The pattern is symmetrical, framing a central white rectangular area.

# **SOLUTIONS**

# CORRECTION

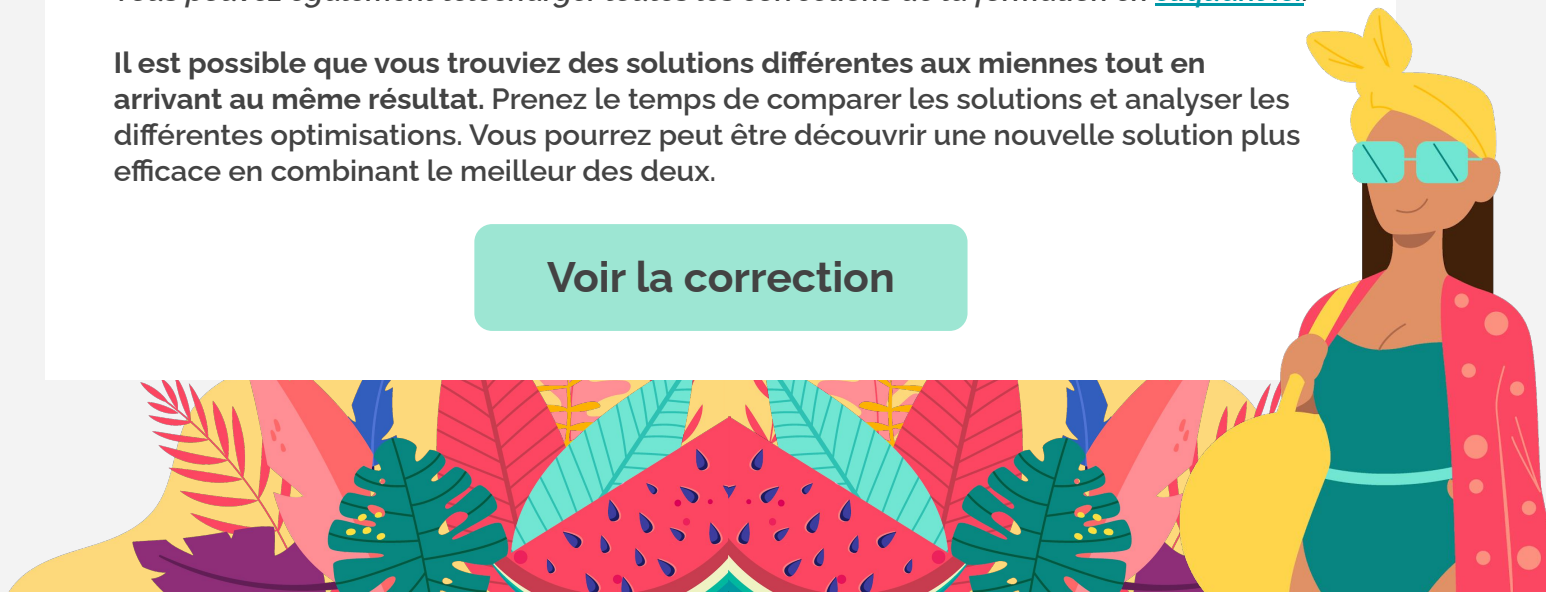
Pour visualiser la correction du chapitre cliquer sur le bouton ci-dessous.

*Le fichier de la correction s'ouvrira dans un nouvel onglet de votre navigateur préféré.  
Pour cela vous devez avoir accès à une connexion Internet.*

*Vous pouvez également télécharger toutes les corrections de la formation en [cliquant ici](#).*

**Il est possible que vous trouviez des solutions différentes aux miennes tout en arrivant au même résultat.** Prenez le temps de comparer les solutions et analyser les différentes optimisations. Vous pourrez peut être découvrir une nouvelle solution plus efficace en combinant le meilleur des deux.

**Voir la correction**







**Félicitation vous avez terminé  
le chapitre sur les listes avec  
Python!**

A decorative border of various tropical leaves in vibrant colors like red, orange, yellow, green, and purple surrounds the central white area.

# CRÉDITS

- Modèle de la présentation par [Slidesgo](#)
- Icônes par [Flaticon](#)
- Images et infographies par [Freepik](#)