**Séquences, Liste, Ensembles, Tuple**

**Séquences**

Une séquence n'est pas un type de données ; c'est une **catégorie de types de données** .

Voici plusieurs types de séquences en Python ; les trois suivants sont les plus importants.

ListsIl s'agit du type de séquence le plus polyvalent. Les éléments d'une liste peuvent être n'importe quel objet, et les listes sont mutables : elles peuvent être modifiées. Les éléments peuvent être réaffectés ou supprimés, et de nouveaux éléments peuvent être insérés.

Par exemple:

my\_name = "Jack"

hello = "Hello World"

my\_age = 27

my\_list = [my\_name, my\_age, hello]

Tuplessont comme des listes, mais elles sont immuables ; elles ne peuvent pas être modifiées.  
*Remarque* : les tuples utilisent des parenthèses.

my\_tuple = (1+3, 2.7, 'Thursday')

StringsIl s'agit d'un type particulier de séquence ne pouvant stocker que des caractères et possédant une notation spécifique. Cependant, toutes les opérations de séquence décrites ci-dessous peuvent également être appliquées aux chaînes de caractères.

my\_string = "I love cheese!"

**Séquences d'indexation**

Les séquences peuvent être indexables, ce qui signifie que chaque élément peut être récupéré avec son index : sequence\_name[index\_num].

Par exemple, en utilisant unlist

my\_name = "Jack"

hello = "Hello World"

my\_age = 27

my\_list = [my\_name, hello, my\_age]

print(my\_list[0]) # The result gives us "Jack"

Vous vous demandez peut-être pourquoi my\_list[0]me donne le premier élément de ma liste ? Notez que les index commencent à [0], qui correspond au premier élément de la liste, et index [1], nous donne le deuxième élément de notre liste, soit la hellovariable de valeur "Hello World".

Il existe d’autres méthodes d’indexation :

* Index négatifs, c'est-à-dire commençant de la fin vers le début ( sequence\_name[-index\_num]) ceci est un exemple utilisant un tuple:

my\_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)

print(my\_tuple[-2]) # The result gives us the number 8

* Les index de plage renvoient une liste d'un index à l'autre ( sequence\_name[start:end]), voici un exemple utilisant un list:

my\_list = [10, 20, 30, 40, 50, 60, 70, 80, 90]

print(my\_list[0:4]) # The result gives us [10, 20, 30, 40]

Lors de l'utilisation de plages, l'index de fin **n'est pas inclus**

**cordes**

En fait, strc'est une séquence ; c'est une suite de lettres.  
Vous pouvez indexer les lettres d'une chaîne grâce à la technique d'indexation.

>>> my\_name = "Rick"

>>> print(my\_name[0])

"R"

>>> print(my\_name[2])

"c"

>>> print(my\_name[-1])

"k"

>>> print(my\_name[1:3])

"ic"

**Listes**

Les listes permettent de stocker une collection d'éléments. Ces éléments peuvent être de tout type : entiers, chaînes, booléens ou même d'autres listes.  
Consultez la documentation officielle [**ici.**](https://docs.python.org/3/tutorial/datastructures.html#more-on-lists)

Une liste est une collection ordonnée et modifiable. Elle autorise les doublons.

Pour créer une liste, utilisez des crochets []et insérez des éléments séparés par une virgule ,. Les éléments peuvent être de tous types de données.

Par exemple:

>>> my\_hobbies = ["Eat", "Sleep", "Code"]

Pour accéder aux éléments, reportez-vous au numéro d'index.

>>> my\_hobbies[0]

"Eat"

>>> my\_hobbies[2]

"Code"

>>> my\_hobbies[-2]

"Sleep"

* **Modifier un élément**

La liste est modifiable ; son contenu peut donc être mis à jour. Pour modifier un élément d'une liste, reportez-vous à son numéro d'index et attribuez-lui une nouvelle valeur.

**Attention :** les index dans la liste commencent à 0 et non à 1.

>>> print(my\_hobbies)

["Eat", "Sleep", "Code"]

>>> my\_hobbies[1] = "Meditate"

>>> print(my\_hobbies)

["Eat", "Meditate", "Code"]

Attention ! Si vous essayez de faire référence à un index qui n'existe pas, votre programme plantera.IndexError

* **Ajouter un élément**

L'ajout d'un élément à une liste s'appelle l'ajout.

Pour ajouter un élément à une liste, utilisez la append()méthode.

>>> print(my\_hobbies)

["Eat", "Meditate", "Code"]

>>> my\_hobbies.append("Baseball")

>>> print(my\_hobbies)

["Eat", "Meditate", "Code", "Baseball"]

* **Supprimer un élément**

Il existe plusieurs méthodes pour supprimer des éléments d’une liste.

* Pour supprimer un élément spécifié, utilisez la removeméthode ; cette méthode supprime la première occurrence de l'élément.

>>> print(my\_hobbies)

["Eat", "Meditate", "Code"]

>>> my\_hobbies.remove("Meditate")

>>> print(my\_hobbies)

["Eat", "Code"]

* Pour supprimer un élément par son index, utilisez la popméthode ; cette méthode supprime l'élément à l'index donné et le renvoie.

>>> print(my\_hobbies)

["Eat", "Code"]

>>> my\_hobbies.pop(0)

>>> print(my\_hobbies)

["Code"]

* **Ajout de deux listes**

L'ajout de deux listes les concatènera.

>>> my\_hobbies = ["Food", "Code"]

>>> additional\_hobbies = ["Sport", "More code"]

>>> my\_hobbies + additional\_hobbies

["Food", "Code", "Sport", "More code"]

**Fonctions de liste**

La liste contient plusieurs fonctions intégrées que vous pouvez utiliser.

* **Longueur**  
  L'utilisation de la len()fonction permet de récupérer le nombre d'éléments contenus dans une liste donnée.

fruits = ["apple","pear", "banana", "melon"]

len(fruits)

>> 4

* **Somme**  
  Si vous avez une liste de nombres, vous pouvez obtenir à nouveau la somme de tous les nombres via une fonction.

numbers = [3, 12, 1, -4]

sum(numbers)

>> 12

* **Tri**  
  Il existe différentes manières de trier les éléments d'une liste, sorted()vous permet de le faire directement.

Si vous avez des numéros, une liste sera triée du plus petit au plus grand.

numbers = [3, 5, 1, 2]

sorted(numbers)

>> [1, 2, 3, 5]

Si vous avez des chaînes, elles seront triées par ordre alphabétique.

letters = ['d', 'a', 'g', 'b']

sorted(letters)

>> ['a', 'b', 'd', 'g']

Cela ne fonctionnera pas si vous avez des valeurs mixtes.

mixed = [3, 'd', 1, 'a']

sorted(mixed)

>> error!

**Plus de méthodes de liste**

food = ['spam', 'eggs', 'ham']

food.append('sushi')

print(food)

>> ['spam', 'eggs', 'ham', 'sushi']

food.insert(0, 'beans')

print(food)

>> ['beans', 'spam', 'eggs', 'ham', 'sushi']

food.extend(['bread', 'water'])

print(food)

>> ['beans', 'spam', 'eggs', 'ham', 'sushi', 'bread', 'water']

**En savoir plus sur les listes**[**ici**](https://www.w3schools.com/python/python_lists.asp)

**Exercice**

Étant donné cette liste :

liste1 = [5, 10, 15, 20, 25, 50, 20]

Recherchez la valeur 20 dans la liste et, si elle est présente, remplacez-la par 200. Mettez à jour uniquement la première occurrence d'une valeur

**Astuce : Regardez la méthode d'index**

Expected output:

list1 = [5, 10, 15, 200, 25, 50, 20]

**Tuples**

Les tuples sont des listes immuables, ce qui signifie que les éléments **ne peuvent pas être modifiés** .

Pour créer un tuple, utilisez des parenthèses :

>>> my\_tuple = (5,6,7)

L'avantage des tuples est qu'ils peuvent être décompressés, ce qui signifie que chaque valeur ira à une variable :

>>> a, b, c = my\_tuple

>>> print(a)

5

>>> print(b)

6

>>> print(c)

7

**Exercice**

Décompressez le tuple suivant en 4 variables

a\_tuple = (10, 20, 30, 40)

Expected output:

a\_tuple = (10, 20, 30, 40)

# Your code

print(a) # should print 10

print(b) # should print 20

print(c) # should print 30

print(d) # should print 40

**Ensembles**

**Important!!**

Listset tuplessont des types de données Python standard qui stockent des valeurs dans un fichier sequence.

**Les ensembles et les dictionnaires n'ont pas de notion d'indice d'élément** et ne peuvent donc pas être considérés comme des suites. En mathématiques, de manière informelle, une suite est une liste ordonnée d'objets (ou d'événements). Comme un ensemble, elle contient des membres (également appelés éléments ou termes).

SetsLes ensembles sont un autre type de données Python standard qui stocke également des valeurs. La principale différence réside dans le fait que, contrairement aux listes ou aux tuples :

* ne peut pas avoir plusieurs occurrences du même élément
* et stocker des valeurs non ordonnées

Nous avons l'habitude Setsde créer une liste sans doublons.

Pour créer un ensemble, utilisez la set()fonction

Pour ajouter un élément à un ensemble, utilisezset.add(element)

>>> my\_set = set()

>>> my\_set.add("Rick")

>>> my\_set.add("Morty")

>>> my\_set.add("Rick")

>>> print(my\_set)

{"Rick", "Morty"}

>>> this\_set = {"banana", "apple", "cherry", "apple"}

>>> print(this\_set)

{'banana', 'cherry', 'apple'}

# the set named this\_set is unordered