Comment utiliser Split en Python

1. Objectifs

- Être capable d'utiliser la méthode Split en Python.

2. Présentation

- Les chaînes en Python sont immuables, ce qui signifie qu'une fois créées, elles ne peuvent pas être modifiées.
- Cependant, nous pouvons faire une copie de la chaîne et effectuer diverses opérations dessus. Pour cela, Python fournit une classe distincte "**str**" pour la gestion et les manipulations des chaînes. Une de ces opérations consiste à diviser une grande chaîne en morceaux ou sous-chaînes plus petits.
- La classe Python "**str**" fournit une fonction intégrée `split()` pour faciliter cette opération de fractionnement sur les chaînes.
- La fonction `split ()` est l'opposé de la concaténation qui concatène de petites chaînes pour former une grande chaîne, tandis que `split()` est utilisé pour diviser une grande chaîne en sous-chaînes plus petites.
- Elle retourne le nombre d'éléments d'une chaîne de caractères ou d'une liste.
- C'est l'opposé de la concaténation qui fusionne ou combine des chaînes en une seule.
- **`split()`** divise ou décompose une chaîne et ajoute les données à un tableau de chaînes à l'aide d'un séparateur défini.
- La méthode `.split()` découpe une chaîne de caractères en plusieurs éléments appelés **champs**, en utilisant comme séparateur n'importe quelle combinaison "d'espace(s) blanc(s)".
- `split()` : convertit une chaîne en une liste de sous-chaînes. On peut choisir le caractère séparateur en le fournissant comme argument, sinon c’est un espace par défaut.

apcpedagogie.com



3. ## Syntaxe de la fonction split()

   - La syntaxe de la fonction split() est la suivante:

```
str.split (séparateur, maxsplit)
```

4. ## Paramètres de la méthode split()

   - La méthode `split()` prend deux paramètres:
     - **séparateur**: est un délimiteur. La chaîne se divise à ce séparateur spécifié. Si n'est pas fourni, tout espace blanc est un séparateur.
     - **maxsplit**: C'est un nombre qui nous dit de scinder la chaîne en un nombre maximum de fois. S'il n'est pas fourni, il n'y a pas de limite.

   1. ### Le séparateur de champs

      - Il indique à Python où casser la chaîne. Il fonctionne comme un délimiteur et la chaîne est séparée en fonction du séparateur prédéfini.
      - Il est **facultatif**, ce qui signifie que si le séparateur n'est pas spécifié dans split, la fonction utilise un espace blanc comme séparateur par défaut.
      - Cependant, si le séparateur est spécifié en tant que valeur de chaîne, la sortie sera une chaîne vide.

apcpedagogie.com

1. **Espace blanc**

   - Un espace blanc (whitespace en anglais) correspond aux caractères qui sont invisibles à l'œil, mais qui occupent de l'espace dans un texte. Les espaces blancs les plus classiques sont l'espace, la tabulation et le retour à la ligne.
   - on peut écrire `chaine.split()`. Par **exemple**, `' a b c '.split()` donne `['a', 'b', 'c']`.
   - `split` se comporte différemment sans argument ou avec une chaîne ayant **un espace**:
   - `' a b'.split(' ')`: renvoie `['', 'a', '', 'b']`
   - `' a b'.split()`: renvoie `['a', 'b']`: les chaines vides sont éliminés !

2. **Tabulation**

   - `chaine.split('\t')` : splitte sur les tabulations, et se comporte bien par défaut si à la fin de la chaîne, il y a des tabulations consécutives

```
a_string = "a\tb\tc\td\te"
print(a_string)
PRODUCTION
a□b□c□d□e

split_string = a_string.split("\t")

print(split_string)
PRODUCTION
['a', 'b', 'c', 'd', 'e']
```

5. **Fractionnement de la chaîne à la première occurrence**

   - Lorsque nous divisons une chaîne en fonction de la première occurrence d'un caractère, il en résulte deux sous-chaînes : la première sous-chaîne contient les caractères **avant le séparateur** et la deuxième sous-chaîne contient le **caractère après le séparateur**.
   - Un exemple de fractionnement d'une chaîne à la première occurrence d'un caractère est illustré ci-dessous :

```
str = "abcabc"
print(str.split('c'))
```

apcpedagogie.com

```
#La sortie du code ci-dessus est la suivante :
['ab', 'ab', '']
```

6. Ici, nous avons déclaré **str** avec une valeur de chaîne "abcabc" . La fonction `split ()` est implémentée avec le séparateur comme "**c**" et la valeur **maxsplit** est prise comme **1**. Chaque fois que le programme rencontre "**c**" dans la chaîne, il sépare la chaîne en deux sous-chaînes - la première chaîne contient des caractères avant "**c**" et la seconde contient des caractères après "**c**".

7. **Fractionnement d'une chaîne avec plusieurs délimiteurs**

   - Vous pouvez fractionner une chaîne à l'aide de plusieurs délimiteurs en mettant différents caractères comme séparateur dans la fonction Fractionner.
   - Un délimiteur est un ou plusieurs caractères dans une séquence qui sont utilisés pour désigner les limites entre les régions d'un texte. Une virgule ("**,**") ou un deux-points ("**:**") est un exemple de délimiteur. Une chaîne avec plusieurs délimiteurs peut être fractionnée à l'aide de la fonction `re.split ()`.
   - Un exemple de fractionnement d'une chaîne avec plusieurs délimiteurs est illustré ci-dessous :

```
import re
str = 'Python\nest; un*langage\ntrès\nfacile'
print(re.split('; |, |\*|\n',str))
#La sortie du code ci-dessus est la suivante :
['Python', 'est', 'un', 'langage', 'très', 'facile']
```

   - Dans l'exemple ci-dessus, nous importons le module intégré **re** qui importe les bibliothèques et les fonctions de "**Regular Expressions**". La variable **str** est déclarée avec une chaîne avec plusieurs délimiteurs comme une nouvelle ligne (**\n**) , un point-virgule (**;**) ou un astérisque (*****).
   - La fonction `re.split ()` est implémentée avec différents délimiteurs comme séparateur et la sortie est une liste de chaînes à l'exclusion des délimiteurs.

8. **Autres séparateurs**

   - Il est possible de modifier le séparateur de champs, par exemple :

```
>>> animaux = "girafe:tigre:singe::souris"
>>> animaux.split(":")
['girafe', 'tigre', 'singe', '', 'souris']
```

    - Dans cet exemple, le séparateur est un seul caractères " : " (et non pas une combinaison de un ou plusieurs :) conduisant ainsi à une chaîne vide entre singe et souris.

9. Vous pouvez également diviser n'importe quelle chaîne avec un caractère dièse (**#**) comme délimiteur. La fonction `Split ()` prend un hachage (**#**) comme séparateur, puis divise la chaîne au point où un hachage est trouvé. Le résultat est une liste de sous-chaînes.

```
str = 'Python#est#un#langage#très#facile'
print(str.split('#'))
# La sortie du code ci-dessus est la suivante :
['Python', 'est', 'un', 'langage', 'très', 'facile']
```

10. La variable **str** est déclarée avec une chaîne avec des caractères dièse (**#**) entre eux. La fonction Split est exécutée avec un hachage comme séparateur.
11. La fonction divise la chaîne partout où elle trouve un hachage (**#**) et le résultat est une liste de sous-chaînes à l'exclusion du caractère de hachage.

- ## maxsplit

    - La méthode `split ()` prend également un paramètre `maxsplit` qui définit le nombre de fractionnements qui doivent avoir lieu. Il spécifie le nombre de fois que la chaîne peut être divisée.
    - c'est un **paramètre facultatif** et sa valeur par défaut est -1, ce qui indique qu'il n'y a pas de limite au nombre de fois qu'une chaîne peut être divisée, autrement il n'y a pas de limite sur le nombre de fractionnements
    - Si **maxsplit** n'est pas défini dans la fonction split, la chaîne entière est analysée et Python la sépare chaque fois qu'un délimiteur est rencontré.
    - Il est également intéressant d'indiquer à `.split()` le **nombre de fois** qu'on souhaite découper la chaîne de caractères avec l'argument`maxsplit`

```
>>> animaux = "girafe tigre singe souris"
>>> animaux.split(maxsplit=1)
['girafe', 'tigre singe souris']
>>> animaux.split(maxsplit=2)
['girafe', 'tigre', 'singe souris']
```

- ## La méthode `rsplit ()`

    - La syntaxe de la fonction rsplit() est la suivante:

```
str.rsplit (séparateur, maxsplit)
```

- La méthode `rsplit ()` sépare la chaîne de la droite au séparateur spécifié et renvoie une liste de chaînes.
- La seule différence entre `split()` et `rsplit()` est lorsque le paramètre **maxsplit** est spécifié.
- Si le paramètre **maxsplit** est spécifié, la méthode `rsplit()` commence à diviser une chaîne à partir du côté droit (à partir du dernier caractère), tandis que la méthode `split()` commence à diviser à partir du côté gauche (à partir du premier caractère).
- Si aucun `maxsplit` n'est spécifié, cette méthode retournera le même que la méthode`split()`.

```
chaine = "pomme, banane, cerise"
nouvChaine = chaine.rsplit(", ")
print(nouvChaine)
Production
['pomme', 'banane', 'cerise']
```

> - *Si **maxsplit** est spécifié, `rsplit()` compte les divisions à partir de l'extrémité droite, tandis que `split ()` les compte à partir de la gauche. Sinon, ils se comportent tous deux exactement de la même manière.*
> - *La méthode `rsplit()` lèvera le **ValueError** si un séparateur est une chaîne vide **''**.*

- Vous pouvez regarder la vidéo suivante:

- ## Applications

1. **App**
   - Écrire un programme python pour demander à l'utilisateur d'entrer son prénom et son nom en réponse à une seule invite.
   - Le programme doit utiliser la méthode `split ()` pour extraire le prénom et le nom , puis attribue chaque partie (prenom et nom) à une variable différente.
   - Pour cet exercice, vous pouvez supposer que l'utilisateur a un seul prénom et un seul nom de famille.

   - **Solution**

     - Essayez de faire l'exercice de votre côté avant de regarder la Solution !

apcpedagogie.com

2. **App**

Cours Python

Source : https://apcpedagogie.com/comment-utiliser-split-en-python · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com