A vibrant tropical illustration on a solid yellow background. It features a toucan with a large orange and blue beak and a white throat patch on the left. In the center is a whole pineapple with a green crown. Below the pineapple is a slice of watermelon with red flesh and black seeds. The scene is surrounded by various stylized tropical leaves in shades of pink, teal, blue, and yellow. The text 'LES CHAÎNES DE CARACTERES' is written in bold white capital letters across the center, partially overlapping the pineapple and watermelon.

# LES CHAÎNES DE CARACTERES



01

## DÉFINITION

Qu'est ce qu'une string  
ou chaine de caractères?

02

## CARACTÈRES D' ÉCHAPPEMENT

Qu'est-ce qu'un caractère  
d'échappement?

03

## OPÉRATEURS

Quels sont les différents  
opérateurs de strings?

04


## OPÉRATEURS DE FORMATAGE

Comment formater  
une string?

05

## MÉTHODES INTÉGRÉES

Présentation des  
différentes méthodes.





## DÉFINITION

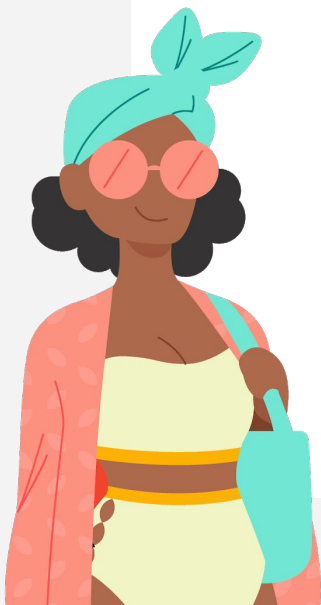
Les chaînes de caractères ou strings sont créées en mettant des caractères entre guillemets. Python traite les guillemets simples de la même manière que les guillemets doubles. Pour accéder à une sous-chaîne, il suffit d'utiliser l'index ou les indexes entre crochets pour obtenir votre sous-chaîne.

```
var = 'Hello World!'
```

```
print("var[0]: ", var[0])  
print("var[6:11]: ", var[6:11])  
print(var[:6] + "Mango")
```

Resultats :

```
var[0]: H  
var[6:11]: World  
Hello Mango
```



# CARACTÈRES D'ÉCHAPPEMENT



Un caractère d'échappement est un caractère qui déclenche une interprétation alternative du ou des caractères qui le suivent.

Le caractère d'échappement peut :

- retirer sa valeur spéciale à un caractère
- ajouter une valeur spéciale à un caractère ordinaire

Il est interprété de la même manière dans une string entre guillemets simples ou doubles.

`\n` : saut de ligne  
`\t` : tabulation  
`\b` : "backspace" (suppression du caractère précédent)  
`\'` : « ' », ne ferme pas la string  
`\"` : « " », ne ferme pas la string  
`\\` : "\"

Pour conserver le symbole "\" dans une chaîne, nous pouvons aussi utiliser une "raw string" (chaîne brute) en préfixant le premier guillemet avec un "r".

```
str1 = r"I love all the following fruits : apple, banana, cherry, mango and pineapple."  
str2 = "I love all the fruits but I don't like the following vegetables : \nasparagus \nturnip"
```



# OPÉRATEURS

## Appartenance : "in"

Retourne vrai si un caractère existe dans la chaîne donnée.

Exemple : "a" in "apple" donne VRAI

## Non Appartenance : "not in"

Retourne vrai si un caractère n'existe pas dans la chaîne donnée.

Exemple : "i" not in "apple" donne VRAI

## Formatage : "%"

Effectue le formatage des chaînes.  
(voir section suivante)

## Concaténation : "+"

Ajoute des valeurs de chaque côté de l'opérateur.

Exemple : "pine" + "apple" donne "pineapple"

## Répétition : "\*"

Crée de nouvelles chaînes, en concaténant plusieurs copies de la même chaîne.

Exemple : 3 \* "pine" donne "pinepinepine"

## Tranche : "[]"

Donne le caractère de l'index donné.

Exemple : "apple"[0] donne "a"

## Tranche de plage : "[ : ]"

Donne les caractères de la plage donnée.

Exemple : "apple"[1:3] donne "pp"



# OPÉRATEURS DE FORMATAGE

Python fournit 3 manières différentes de formater des chaînes de caractères :

- **Opérateur “%”** : permet de définir un texte à trous, et ensuite de dire quoi mettre dans les trous. C'est une logique de template.  
*%s pour string, %d pour int, %f pour float ...*
- **Méthode .format()** : permet de définir un ordre d'insertion et accepte tous les itérables. Il est également possible de nommer les trous.
- **f-strings** : combine les avantages de .format() et %, sans les inconvénients.

```
fruit = "cherry"  
number = 6
```

```
str1 = "I picked %s %s" % (number, fruit)  
str2 = "I picked {} {}".format(*[number, fruit])  
str3 = "I picked {0} {1}".format(number, fruit)  
str4 = "I picked {number} {fruit}".format(number=3 ,  
                                          fruit=apple)  
str5 = f"I picked {number:.2f} {fruit.upper()}"
```

**# Résultats :**

```
# str1, str2, str3 et str4 sont identiques.  
print(str1) donne : I picked 6 cherry  
print(str4) donne : I picked 3 apple  
print(str5) donne : I picked 6.00 CHERRY
```



# MÉTHODES INTÉGRÉES

Voici quelques méthodes pour manipuler les strings.  
Pour les curieux [la liste complète](#).

## **capitalize()**

Met en majuscule la première lettre de la chaîne.

## **decode(encoding='UTF-8',errors='strict')**

Décode la chaîne en utilisant le codec enregistré pour l'encodage.

## **encode(encoding='UTF-8',errors='strict')**

Renvoie la version codée de la chaîne de caractères ; en cas d'erreur, la valeur par défaut est ValueError.

```
str1 = "this apple is very juicy!"  
print("Capitalized string : ", str1.capitalize())
```

```
str1 = str1.encode(encoding="ascii", errors="strict")  
print("Decoded string: " + str1.decode('ascii', 'strict'))
```

```
str2 = 'Möngö'  
print("The string is : ", string)  
string_utf = string.encode()  
print("The encoded version is : ", string_utf)
```

## **# Résultats :**

Capitalized string : This apple is very juicy!

Decoded string: this apple is very juicy!

The string is : Möngö

The encoded version is : b'M\xc3\xb6ng\xc3\xb6'



# MÉTHODES INTÉGRÉES

## **count(str, beg=0, end=len(string))**

Compte combien de fois la chaîne se produit dans une chaîne ou une sous-chaîne de chaîne, si le début de l'index et la fin de l'index sont donnés.

## **endswith(suffixe, beg=0, end=len(string))**

Détermine si une chaîne ou une sous-chaîne de chaîne se termine par un suffixe ; renvoie vrai si oui et faux sinon.

```
str = "this apple is very juicy!"  
char = "i"  
print("str.count(char , 4, 20) : ",  
      str.count(char , 4, 20))
```

```
suffix = "juicy!"  
print(str.endswith(suffix))  
print(str.endswith(suffix, 20))  
suffix = "is"  
print(str.endswith(suffix, 2, 4))  
print(str.endswith(suffix, 2, 6))
```

## **Résultats :**

```
str.count(sub, 4, 20) : 1  
True  
False  
True  
False
```





# MÉTHODES INTÉGRÉES

**find(str, beg=0 end=len(string))**

Détermine si str se trouve dans une chaîne, renvoie index si trouvé et -1 sinon.

**index(str, beg=0, end=len(string))**

Même chose que find(), mais soulève une exception si str n'est pas trouvé.

**isalnum()**

Vrai si la chaîne a au moins 1 caractère et si tous les caractères sont alphanumériques, faux sinon.

```
str1 = "this apple is very juicy!"
```

```
str2 = "is"
```

```
str3 = "this2020"
```

```
print("find", str1.find(str2), "index", str1.index(str2))
```

```
print("find", str1.find(str2, 10), "index", str1.index(str2, 10))
```

```
print("find", str1.find(str2, 40), "index", str1.index(str2, 40))
```

```
print(str1, str1.isalnum())
```

```
print(str3, str3.isalnum())
```

**Résultats :**

```
find, 2, index, 2
```

```
find, 11, index, 11
```

```
find, -1, index, ValueError: substring not found
```

```
this apple is very juicy!, False
```

```
this2020, True
```



# MÉTHODES INTÉGRÉES

## **isalpha()**

Vrai si la chaîne a au moins 1 caractère et si tous les caractères sont alphabétiques, faux sinon.

## **isdigit()**

Vrai si la chaîne ne contient que des chiffres et faux sinon.

## **islower()**

Vrai si la chaîne a au moins 1 caractère en majuscule et si tous les caractères en majuscule sont en minuscules, faux sinon.

```
str1 = "this apple is very juicy!"  
str2 = "apple"  
str3 = "12345"  
str4 = "this is very JUICY!"
```

```
print(str1, str1.isalpha(), str2, str2.isalpha())  
print(str1, str1.isdigit(), str3, str3.isdigit())  
print(str1, str1.islower())  
print(str4, str4.islower())
```

## **Résultats :**

```
this apple is very juicy!, False, apple, True  
this apple is very juicy!, False, 12345, True  
this apple is very juicy!, True,  
this is very JUICY!, False
```



# MÉTHODES INTÉGRÉES

## **istitle()**

Vrai si la chaîne est "titrée" et faux sinon.

## **isupper()**

Vrai si tous les caractères sont en majuscule et faux sinon.

## **join(seq)**

Fusionne les éléments d'une séquence en une string, avec une string de séparation.

## **len(string)**

Retourne la longueur de la chaîne.

```
str1 = "This Apple Is Very Juicy!"  
str2 = "THIS APPLE IS VERY JUICY!"  
s = "-";  
seq = ("a", "b", "c")
```

```
print("istitle() : " + str1, str1.istitle())  
print("istitle() : " + str2, str2.istitle())  
print("isupper() : " + str1, str1.isupper())  
print("isupper() : " + str2, str2.isupper())  
print(s.join( seq ))
```

## **Résultats :**

```
istitle() : This Apple Is Very Juicy!, True  
istitle() : THIS APPLE IS VERY JUICY!, False  
isupper() : This Apple Is Very Juicy!, False  
isupper() : THIS APPLE IS VERY JUICY!, True  
a-b-c
```



# MÉTHODES INTÉGRÉES

## **lower()**

Convertit toutes les lettres majuscules d'une chaîne de caractères en minuscules.

## **max(str) (min(str))**

Retourne le caractère alphabétique maximum (minimum) de la chaîne de caractères str.

## **replace(old, new [, max])**

Remplace toutes les occurrences de l'ancienne chaîne par les nouvelles en fonction du nombre maximum donné.

```
str1 = "THIS APPLE IS VERY JUICY!"  
str2 = "This apple is very juicy!"
```

```
print("lower() : " + str1.lower())  
print("Max character : " + max(str1))  
print("Min character : " + min(str1))  
print(str2.replace("is", "was"))  
print(str2.replace("is", "was", 1))
```

## **Résultats :**

```
lower() : This apple is very juicy!  
Max character : Y  
Min character : "  
Thwas apple was very juicy!  
Thwas apple is very juicy!
```



# MÉTHODES INTÉGRÉES

**split(str="", num=string.count(str))**

Divise la chaîne en fonction du délimiteur et renvoie une liste de sous-chaînes, en fonction du nombre indiqué.

**splitlines( num=string.count('\n'))**

Divise la chaîne à chaque nouvelle ligne et renvoie une liste de sous-chaînes, en fonction du nombre indiqué.

**startswith(str, beg=0,end=len(string))**

Détermine si une chaîne ou une sous-chaîne de chaîne commence par une sous-chaîne de chaîne, renvoie vrai si oui et faux sinon.

```
str = "Line1-Apple \nLine2-Banana \nLine3-Cherry"
```

```
print(str.split())  
print(str.split(' ', 1))  
print(str.splitlines())  
print(str.splitlines( 3 ))  
print((str.startswith('Line'))  
print(str.startswith('ne', 2, 4))
```

**Résultats :**

```
['Line1-Apple', 'Line2-Banana', 'Line3-Cherry']  
['Line1-Apple', '\nLine2-Banana \nLine3-Cherry']  
['Line1-Apple ', 'Line2-Banana ', 'Line3-Cherry']  
['Line1-Apple \n', 'Line2-Banana \n', 'Line3-Cherry']  
True  
True
```

# MÉTHODES INTÉGRÉES



## **strip([chars])**

Supprime tous les espaces au début et à la fin de la chaîne ou le caractère spécifié.

## **title()**

Retourne tous les mots commençant par des majuscules et le reste par des minuscules.

## **upper()**

Convertit les lettres minuscules d'une chaîne de caractères en lettres majuscules.

## **zfill (width)**

Renvoie la chaîne originale complétée à gauche par des zéros en fonction du nombre donné.

```
str1 = "000This apple is very juicy!000"  
str2 = "197"
```

```
print("str1.strip() : ", str1.strip('O'))  
print("str1.title() : ", str1[3:-3].title())  
print("str1.upper() : ", str1[3:-3].upper())  
print("str2.zfill() : ", str2.zfill(4))  
print("str1.zfill() : ", str1[3:-3].zfill(30))
```

## **Résultats :**

```
str1.strip() : This apple is very juicy!  
str1.title() : This Apple Is Very Juicy!  
str1.upper() : THIS APPLE IS VERY JUICY!  
str2.zfill() : 0197  
str1.zfill() : 000000This apple is very juicy!
```

The background of the slide is a vibrant, colorful pattern of various tropical leaves and foliage. The leaves are in shades of green, yellow, orange, red, and purple, creating a dense and lively border around the central text. Some leaves are large and detailed, while others are smaller and more stylized. The overall effect is a tropical and energetic aesthetic.

# **EXERCICES**



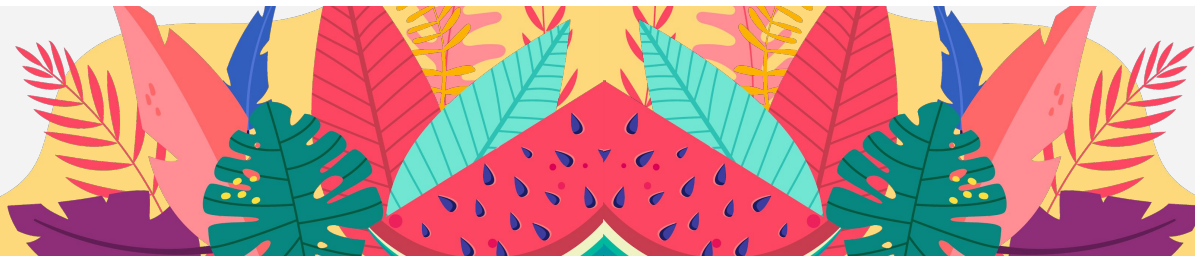
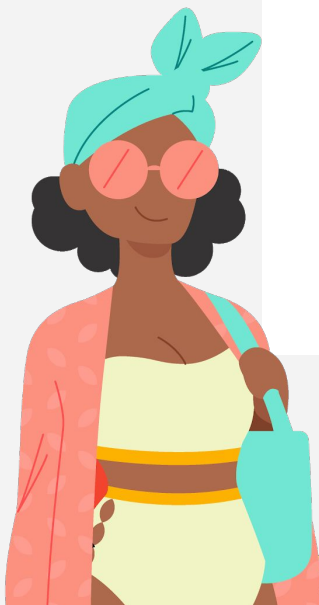
# EXERCICES

**1. Créez une variable *paragraphe* en lui assignant le texte suivant :**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam lobortis, tortor at porttitor ultricies, ipsum turpis scelerisque erat, ut bibendum mauris lectus vitae purus. Vivamus sem nulla, interdum a sapien vitae, mollis dictum velit. Aenean porttitor eu dolor ut vulputate. Vivamus nisi leo, elementum eu accumsan a, facilisis ut sem. Cras interdum scelerisque euismod. Integer mattis pharetra aliquam. Vestibulum molestie semper tortor a condimentum. Maecenas id est nisi.

**2. Extraire les 2 premiers mots du paragraphe dans une variable *words*.**

**3. Complétez la phrase suivante en utilisant la variable *words* et l'opérateur de formatage :**  
Le ... est une suite de mots sans signification.





# EXERCICES

4. Relevez le nombre de fois où le mot 'vitae' est employé dans le paragraphe.
5. Extraire la première phrase du paragraphe contenant uniquement des caractères alpha numériques, dans une variable *sentence*.
6. Si la variable *words* n'est pas un titre; alors modifiez la pour qu'elle le soit.
7. Passer en majuscule tous les caractères de la variable *words*.
8. Extraire toutes les phrases du paragraphe dans une variable *sentences*.
9. Fusionnez tous les éléments de la variable *sentences* entre eux, séparés par le caractère suivant : '|'
10. Remplacez toutes les occurrences de la lettre 'e' par la lettre 'o' dans le paragraphe donné précédemment.



The background of the slide is a vibrant, colorful pattern of various tropical leaves and foliage. The leaves are in shades of green, yellow, orange, red, and purple, creating a dense and lively border around the central text. Some leaves are large and detailed, while others are smaller and more stylized. The overall effect is a tropical, summery aesthetic.

# **SOLUTIONS**

# CORRECTION

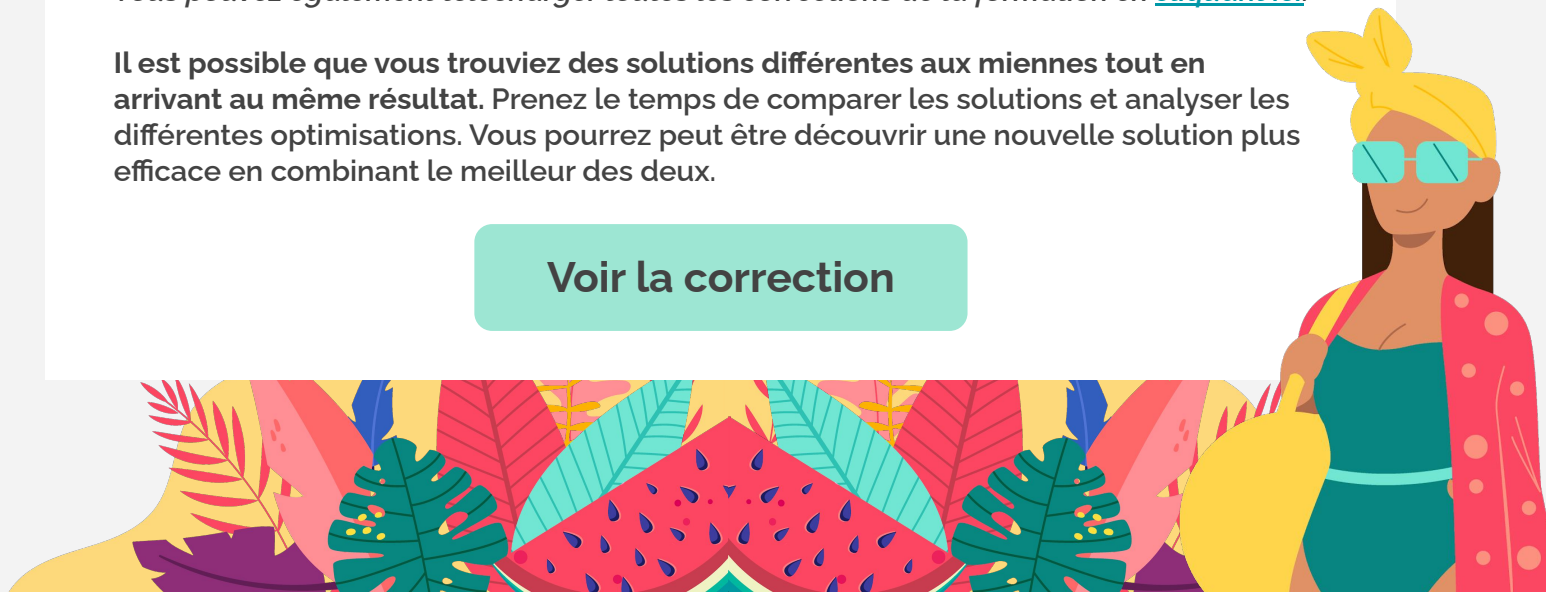
Pour visualiser la correction du chapitre cliquer sur le bouton ci-dessous.

*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 chaînes de  
caractères avec Python!**

A decorative border of various tropical leaves in vibrant colors like red, orange, yellow, green, and purple surrounds the central white area. The leaves include Monstera, palm, and other exotic foliage.

# CRÉDITS

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