## Caractères et chaines de caractères

On appelle caractère tout symbole qui peut être écrit comme par exemple :

- Les lettres de l'alphabet latin : abcd…xyzABCD…XYZ
- Les chiffres décimaux : 0123456789
- Les symboles de ponctuation (y compris l'espace) : . , ; : ! ?
- Les symboles de parenthèse : () [] {}
- Et bien d'autres caractères encore comme les lettres accentuées à û É é è À… et les lettres d'autres alphabets : α , β,ش, д, ф, خ ,ב ,א, …

Une chaîne de caractères (string en anglais) est une séquence de caractères, c'est-à-dire des caractères qui se suivent les uns derrières les autres.

Une chaîne de caractères peut ne contenir aucun caractère : on l'appelle alors chaîne vide.

Un caractère n'est plus qu'une chaine de caractère de longueur 1

### Les Chaînes de Caractères en Python

Les chaînes de caractères (*strings* ) sont un type de données fondamental en Python(**str**). Elles sont utilisées pour représenter du texte et sont immuables, ce qui signifie qu'une fois créées, elles ne peuvent pas être modifiées.

### Déclaration d'une Chaîne de Caractères

En Python, les chaînes de caractères peuvent être déclarées en utilisant des guillemets simples (') ou doubles (").

```
Chaine1 = 'bonjour'
Chaine2 = "bonjour"
```

Les deux déclarations ci-dessus sont équivalentes.

Pour les chaînes multilignes, vous pouvez utiliser des triples guillemets (''' ou """).

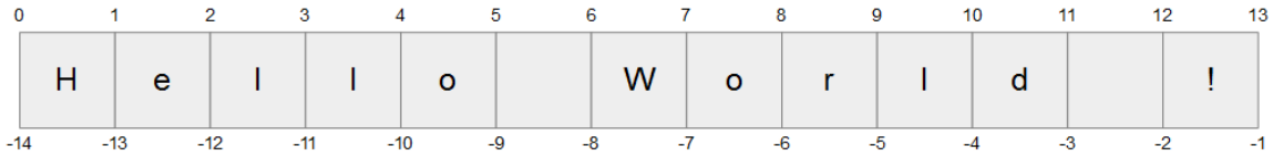
```
Chaine_multilingne = """ceci est une chaine
qui s'étend sur
plusieurs lignes."""
```

## *Représentation d'une chaine*

Une chaine de caractère est représentée sous la forme d'un tableau chaque caractère est rangée dans une case et identifié par son indice (ou index).

**indices positifs →** **0** à **N-1**

**indices négatifs →** **-1** à **-N**




## *Longueur d'une Chaîne*

La fonction **len()** permet de connaître la longueur d'une chaîne (le nombre de caractères).

```
chaine = "Python"
print(len(chaine))        # Affiche 6
print(len('bon jour'))    # Affiche 8
```

## *Formatage de Chaînes* (voir chapitre 3)

## *Caractères d'Échappement* (voir chapitre 3)

## *Accès aux Caractères d'une Chaîne*

On peut accéder aux caractères d'une chaîne par leurs indices (ou index) dans la chaîne, et extraire des sous-chaînes d'une chaîne.

Pour accéder à un caractère bien déterminé, on utilise le nom de la variable qui contient la chaîne, et on lui accole entre deux crochets ( **[ ]** ) l'index numérique qui correspond à la position du caractère dans la chaîne.

L'opérateur d'indiçage (ou d'indexation) ( **[ ]** ) permet aussi de sélectionner des sous-chaines selon leurs indices. On appelle cette technique le ***slicing*** (« **découpage en tranches** »).

### Syntaxe:

Soit **Ch** est une chaine de caractères.

| | |
|---|---|
| `Ch[i]` | **Accès au caractère d'indice i** |
| `Ch[-1]` ou `Ch[len(Ch)-1]` | **Accès au dernier caractère** |
| `Ch[i:j]` ou `Ch[i:j:1]` | **Extraire la sous-chaîne de l'élément i (inclus) à l'élément j (exclu) pas par défaut égal 1** |
| `Ch[i:j:k]` | **Extraire la sous-chaîne de l'élément i (inclus) à l'élément j (exclu) pas égal k** |
| `Ch[i:]` | **Extraire la sous-chaîne depuis l'élément i (inclus) jusqu'à la fin de la chaine.** |
| `Ch[:j]` | **Extraire la sous-chaîne de début indice 0 jusqu'à l'indice j (exclu).** |
| `Ch[:]` | **Extraire la sous-chaîne de début jusqu'à la fin de la chaine** |

*Exemple :*

```
chaine = "Python"
print(chaine[0])      # Affiche 'P'
print(chaine[-1])     # Affiche 'n' (accès depuis la fin)
```

## *Parcourir une chaine*

Ce qui arrive souvent, c'est que nous devons traiter la chaîne entière caractère par caractère du début à la fin pour effectuer n'importe quelle opération sur chaque chaîne. Nous appelons cette opération un **parcours**.

On peut utiliser la structure répétitive for ou while pour parcourir caractère par caractère une chaîne.

*Exemple1 :* utilisation des indices avec la fonction **range()** et **len()**

```
ch = 'bon jour'
taille = len(ch)
#afficher tous les caracteres de ch
for i in range(taille):
    print(ch[i])
```

```
b
o
n

j
o
u
r
```

*Example2:* utilisation des caractères au lieu des indices.

```
ch = 'bon jour'
taille = len(ch)
#afficher tous les caracteres de ch
for car in ch:          # ou for car in 'bon jour' :
    print(car)
```

```
b
o
n

j
o
u
r
```

*Example3:* Afficher des caractères avec leur indice par **enumarate()**.

```
ch = 'bon jour'

taille = len(ch)

#afficher tous les caracteres de ch

for i,car in enumerate(ch):

    print('index : ' ,i, '    caractere : ' ,car)
```

```
index :  0    caractere :  b
index :  1    caractere :  o
index :  2    caractere :  n
index :  3    caractere :
index :  4    caractere :  j
index :  5    caractere :  o
index :  6    caractere :  u
index :  7    caractere :  r
```

## *Les chaînes sont non modifiables*

Comme mentionné précédemment, les chaînes de caractères sont immuables. Cela signifie que vous ne pouvez pas modifier une chaîne existante, En d'autres termes, vous ne pouvez pas utiliser l'opérateur **[]** sur le côté gauche d'une instruction d'affectation. mais vous pouvez en créer une nouvelle.

```
chaine = "Python"

chaine[0] = 'J'              # Cela générerait une erreur


nouvelle_chaine = 'J' + chaine[1: ]

print(nouvelle_chaine)      # Affiche "Jython"
```

## *Les opérations sur les chaines*

Les opérations à appliquer sur une chaine de caractères

| | |
|---|---|
| + | La concaténation de deux chaines |
| * | La duplication d'une chaine (Répétition) |
| in ou not in | Test d'appartenance renvoie True ou False |
| >,<,<=,>=,!=, == | Opérateurs de comparaisons renvoie True ou False |
| is ou is not | Opérateur d'identité renvoie True ou False |

### *Concaténation et répétition* (voir chapitre 2)

### *Appartenance d'un élément à une chaîne*

Vous pouvez vérifier si une sous-chaîne est présente dans une chaîne en utilisant l'instruction **in** ou **not in**.

```
chaine = "Bonjour le Monde"
print("Monde" in chaine)  # Affiche True
print("Python" in chaine) # Affiche False
print("Python" not in chaine) # Affiche True
```

### *Comparaison de Chaînes*

Les chaînes peuvent être comparées en utilisant les opérateurs de comparaison (==, !=, <, >, etc.). La comparaison est effectuée caractère par caractère en fonction de leur valeur Unicode.

```
print("abc" == "abc")  # Affiche True
print("abc" != "def")  # Affiche True
print("a" > "b")       # Affiche False
```

## *Slicing (Tranches)*

L'opérateur d'indexation ( **[]** ) vous permet également de sélectionner une sous-chaîne en fonction de son index. Cette technique s'appelle slicing (tranchage).

Le slicing permet d'extraire une sous-chaîne à partir d'une chaîne.

Dans la plage de [**n**, **m**], le **n**ième caractère est inclus, mais le **m**ième caractère n'est pas inclus.

L'extraction de parties d'une chaîne de caractères en Python se fait en utilisant la syntaxe [**début** : **fin** : **pas**], où :

**début** : est l'indice de départ de l'extraction.

**fin** : est l'indice de fin (non inclus).

**pas** : est l'incrément.

*Exemple :*

```
chaine = "bon jour"
print(chaine[1 : 6])   # Affiche 'on jo' (de l'indice 1 à 5)
print(chaine[ : 5])    # Affiche 'bon j' (du début à l'indice 4)
print(chaine[2 : ])    # Affiche 'n jour' (de l'indice 2 à la fin)
print(chaine[0:5:2]) # Affiche 'bnj'  (de l'indice 0 à  4 extrait chaque deuxième caractère)
print(chaine[ : : 3]) # Affiche   'b  j'  (du debut a la fin extrait chaque troisième caractère)
```

## *Méthodes Courantes sur les Chaînes*

Python propose de nombreuses méthodes pour manipuler les chaînes de caractères. Voici quelques-unes des plus couramment utilisées : (**chn** est une chaine de caractères).

- chn.**upper()** : Convertit la chaîne en majuscules.
- chn.**lower()** : Convertit la chaîne en minuscules.
- chn.**capitalize ()** : Convertir la première lettre de la chaîne en majuscule.
- chn.**strip()** : Supprime les espaces en début et fin de chaîne.
- chn.**replace(old, new)** : Remplace toutes les occurrences de **old** par **new**.
- chn.**split(sep)** : Divise la chaîne en une liste de sous-chaînes en utilisant sep comme séparateur.
- 'separ'.**join(iterable)** : Joint les éléments d'un itérable (comme une liste) en une seule chaîne, avec **separ** comme séparateur.
- **max(**chn**)** : Renvoie le caractère alphabétique maximal de la chaîne chn.
- **min(**chn**)** : Renvoie le caractère alphabétique minimal de la chaîne chn.

Vous pouvez obtenir la liste complète de toutes les méthodes à l'aide de la fonction intégrée **dir(' ')**.

*Exemple :*

```
ma_chaine = "  Bonjour le Monde  "
print(ma_chaine.strip())                # Affiche "Bonjour le Monde"
print(ma_chaine.lower())                # Affiche "  bonjour le monde  "
print(ma_chaine.split())                # Affiche ['Bonjour', 'le', 'Monde']
print(ma_chaine.replace("le", "un"))    # Affiche "  Bonjour un Monde  "
print("-" .join ( ["2023", "10", "05"] ) )      # Affiche "2023-10-05"
```

## *Méthodes de Recherche*

- chn.**find(sub)** : Retourne l'indice de la première occurrence de sub dans la chaîne, ou -1 si non trouvé.
- chn.**index(sub)** : Similaire à find(), mais lève une exception si sub n'est pas trouvé.
- chn.**count(sub)** : Retourne le nombre d'occurrences de sub dans la chaîne.

*Exemple :*

```
salutation = "Bonjour le Monde"
print(salutation.find("le"))        # Affiche 8
print(salutation.index("le"))       # Affiche 8
print(salutation.count("o"))        # Affiche 3
```

## *Méthodes de Vérification*

- chn.**isalpha()** : Retourne True si tous les caractères sont alphabétiques.
- chn **isdigit()** : Retourne True si tous les caractères sont des chiffres.
- chn.**isalnum()** : Retourne True si tous les caractères sont alphanumériques.
- chn.**isspace()** : Retourne True si tous les caractères sont des espaces.

*Exemple :*

```
print("abc".isalpha())        # Affiche True
print("123".isdigit())        # Affiche True
print("abc123".isalnum())     # Affiche True
print("abc/123".isalnum())    # Affiche False
print("   ".isspace())        # Affiche True
print("abc123".isspace())     # Affiche False
```

## *Exemples Pratiques*

*Exemple 1 :* Inverser une chaîne

```
chaine = "Python"
chaine_inverse = chaine[ : : -1]
print(chaine_inverse)    # Affiche "nohtyP"
```

*Exemple 2 :* Compter les mots dans une chaîne

```
chaine = "Bonjour le Monde"
nombre_mots = len(chaine.split())
print(nombre_mots)  # Affiche 3
```

*Exemple 3 :* Remplacer des mots dans une chaîne

```
chaine = "Bonjour le Monde"
nouvelle_chaine = chaine.replace("Monde", "Python")
print(nouvelle_chaine)   # Affiche "Bonjour le Python"
```

## *Conclusion*

Les chaînes de caractères sont un outil puissant en Python, et leur manipulation est essentielle pour de nombreuses tâches de programmation. En maîtrisant les concepts et les méthodes présentés dans ce cours, vous serez en mesure de manipuler efficacement du texte en Python.

# *Exercices*

## *Exercice 1*

On considère la chaîne de caractère **a = "Bonjour"**. Donner le résultat de chacune des instructions ci-dessous (si la syntaxe est incorrecte, répondre « erreur »).

1. Print(a[0])
2. Print(a[3] )
3. Print(a[-3] )
4. Print(len(a) )
5. Print(a[-8:-3] )
6. Print(a[0:4] )
7. Print(a[1:5:2] )
8. Print(a[0]+a[-1] )
9. Print(a[0]+4*a[-1] )
10. Print( 'B' in a )
11. Print( 'b' in a )
12. Print(a[1]=='o')
13. Print(a[1]='B')

## *Exercice 2*

Écrire un programme qui affiche le nombre de voyelles existantes dans une phrase que vous avez entrée.

## *Exercice 3*

Ecrire un script qui détermine le nombre de lettres d'un mot que vous avez entré.

## *Exercice 4*

Ecrire un programme qui détermine le nombre de « e » contenu dans une phrase que vous avez entrée.

## *Exercice 5*

Ecrire un programme permettant de saisir un mot et afficher le à l'envers.

Exemple : Asri devient irsA.

## *Exercice 6*

Ecrire un script qui écrit votre prénom avec une étoile entre chaque lettre.

Exemple: INSFP → I*N*S*F*P