Chaînes de caractères

Bases de Python

1 Définition

Une **chaîne de caractères** (type `str`) est une séquence de caractères. C'est le type utilisé par Python pour représenter du texte : un prénom, une phrase, un mot de passe, une adresse e-mail, etc.

On crée une chaîne en plaçant du texte entre **guillemets** `"` ou **apostrophes** `'` :

```
prenom = "Alice"
ville = 'Paris'
print(f"{prenom} habite à {ville}")
```

```
Alice habite à Paris
```

Les deux notations sont équivalentes. Les guillemets doubles sont pratiques lorsque la chaîne contient une apostrophe, et inversement :

```
phrase1 = "L'informatique est passionnante"
phrase2 = 'Il a dit "bonjour" en entrant'
print(phrase1)
print(phrase2)
```

```
L'informatique est passionnante
Il a dit "bonjour" en entrant
```

Si une chaîne doit contenir à la fois des guillemets et des apostrophes, on utilise le **caractère d'échappement** `\` :

```
mixte = "Elle a dit : \"c'est formidable\""
print(mixte)
```

```
Elle a dit : "c'est formidable"
```

2 Longueur d'une chaîne

La fonction `len()` renvoie le nombre de caractères d'une chaîne :

```python
mot = "Python"
print(len(mot))       # 6

vide = ""
print(len(vide))      # 0
```

```python
phrase = "Bonjour !"
print(len(phrase))
```

Question.
Quel nombre s'affiche ?

Réponse.

```
9
```

Tous les caractères comptent, y compris l'espace et le point d'exclamation : B, o, n, j, o, u, r, l'espace et `!` font 9 caractères.

③ Lire un caractère avec son indice

Les caractères d'une chaîne sont numérotés à partir de **zéro** : chaque caractère a un **indice**. Les **indices négatifs** comptent à partir de la fin : `-1` désigne le dernier caractère, `-2` l'avant-dernier.

	P	y	t	h	o	n
indice	0	1	2	3	4	5
indice négatif	−6	−5	−4	−3	−2	−1

On lit un caractère en indiquant son indice entre crochets, après la chaîne :

```python
mot = "Python"
print(mot[0])    # premier caractère
print(mot[3])    # quatrième caractère
print(mot[-1])   # dernier caractère
print(mot[-2])   # avant-dernier
```

```
P
h
n
o
```

Question.
Que vaut `mot[2]` ?

Réponse.
`'t'` : l'indice `2` désigne le troisième caractère, puisque le premier a l'indice `0`.

④ Attention au dernier indice

Le mot `"Python"` contient 6 caractères : ses indices vont donc de `0` à `5`. Demander l'indice `6`, c'est chercher un caractère qui n'existe pas, et Python s'arrête avec une erreur :

```
mot = "Python"
print(mot[6])
```

```
IndexError: string index out of range
```

À retenir. Les indices valides d'une chaîne `s` vont de `0` à `len(s) - 1`. Pour lire le dernier caractère, le plus simple est d'écrire `s[-1]`.

⑤ Une chaîne ne se modifie pas

Les chaînes sont **immuables** : on ne peut pas modifier un caractère en place. Pour transformer `"Python"` en `"Jython"`, on essaie de remplacer le premier caractère :

```
mot = "Python"
mot[0] = "J"
```

```
TypeError: 'str' object does not support item assignment
```

Python refuse. Pour obtenir une chaîne modifiée, il faut en **construire une nouvelle** :

```
mot = "Python"
nouveau = "J"
for i in range(1, len(mot)):   # tous les caractères sauf le premier
    nouveau = nouveau + mot[i]
print(nouveau)
```

```
Jython
```

La boucle ajoute un à un, à la fin de `nouveau`, les caractères d'indice 1 à 5 de `mot`. La chaîne `mot`, elle, n'a pas changé.

⑥ Concaténation et répétition

L'opérateur `+` **concatène** (assemble) deux chaînes, et l'opérateur `*` **répète** une chaîne :

```python
prenom = "Ada"
nom = "Lovelace"
complet = prenom + " " + nom
print(complet)      # Ada Lovelace

tirets = "-" * 20
print(tirets)       # --------------------

motif = "ab" * 4
print(motif)        # abababab
```

```python
print("ha" * 3 + "!")
```

Question.
Qu'affiche cette ligne ?

Réponse.

```
hahaha!
```

Comme pour les nombres, la répétition `*` passe avant la concaténation `+` : Python calcule d'abord `"ha" * 3`, soit `"hahaha"`, puis ajoute `"!"`.

Attention. on ne peut pas concaténer une chaîne et un nombre directement.

```python
age = 17
print("J'ai " + age + " ans")
```

```
TypeError: can only concatenate str (not "int") to str
```

Il faut d'abord convertir le nombre en chaîne avec `str()`, ou utiliser une f-string :

```python
age = 17
print("J'ai " + str(age) + " ans")  # correct
```

```
J'ai 17 ans
```

7 Les f-strings

Les **f-strings** (chaînes formatées, depuis Python 3.6) permettent d'insérer des variables et des expressions directement dans une chaîne, en les plaçant entre accolades `{}` :

```python
prenom = "Alice"
age = 17
print(f"{prenom} a {age} ans")
print(f"Dans 10 ans, {prenom} aura {age + 10} ans")
```

```
Alice a 17 ans
Dans 10 ans, Alice aura 27 ans
```

On peut aussi contrôler le **format d'affichage** :

```python
import math

# Arrondir un flottant
print(f"Pi vaut environ {math.pi:.2f}")    # 3.14

# Aligner à droite sur 10 caractères
print(f"{'Alice':>10}")                     #      Alice

# Afficher un pourcentage
ratio = 0.85
print(f"Réussite : {ratio:.0%}")           # Réussite : 85%
```

```
Pi vaut environ 3.14
     Alice
Réussite : 85%
```

⑧ Caractères spéciaux

Certains caractères ne sont pas directement saisissables. On les écrit à l'aide de **séquences d'échappement** commençant par \ :

Séquence	Signification
\n	Retour à la ligne
\t	Tabulation
\\	Le caractère \ lui-même
\'	Apostrophe
\"	Guillemet

```python
print("Ligne 1\nLigne 2")
print("Nom\tPrénom\tÂge")
print("Alice\tDupont\t17")
```

```
Ligne 1
Ligne 2
Nom Prénom  Âge
Alice   Dupont  17
```

9 Chaînes multilignes

Pour écrire une chaîne sur plusieurs lignes, on utilise les **triples guillemets** `"""` ou `'''` :

```
poeme = """Ceci est un texte
qui s'étend sur
plusieurs lignes."""
print(poeme)
```

```
Ceci est un texte
qui s'étend sur
plusieurs lignes.
```

10 Chercher un texte dans un autre

L'opérateur `in` teste si une sous-chaîne est présente dans une chaîne. Il répond par un booléen.

```
phrase = "Bienvenue en NSI"
print("NSI" in phrase)
print("nsi" in phrase)
```

Question.
Que renvoient ces deux tests ?

Réponse.

```
True
False
```

`"NSI"` figure bien dans la phrase. En revanche, `"nsi"` en minuscules n'y figure pas : le test distingue les majuscules des minuscules (on dit qu'il est **sensible à la casse**).

La sous-chaîne peut aussi se trouver au milieu d'un mot :

```
phrase = "Bienvenue en NSI"
print("venu" in phrase)     # True
print("z" not in phrase)    # True
```

11 À retenir

1. Une chaîne de caractères s'écrit entre guillemets ou entre apostrophes ; `len()` donne son nombre de caractères, espaces compris.
2. Chaque caractère a un indice, qui commence à `0` ; les indices négatifs comptent à partir de la fin, `-1` désignant le dernier caractère.

3. Les indices valides vont de `0` à `len(s) - 1` : au-delà, Python affiche une erreur `IndexError`.
4. Une chaîne est immuable : on ne peut pas modifier un caractère, on construit une nouvelle chaîne.
5. `+` assemble deux chaînes, `*` répète une chaîne ; on ne peut pas ajouter un nombre à une chaîne sans le convertir avec `str()`.
6. Une f-string insère des valeurs dans une chaîne : `f"{prenom} a {age} ans"`.
7. `in` teste si un texte figure dans un autre, en distinguant majuscules et minuscules.

12 Résumé

Opération	Syntaxe	Exemple
Création	`"..."` ou `'...'`	`"Bonjour"`
Longueur	`len(s)`	`len("NSI")` → `3`
Accès	`s[i]`	`"NSI"[0]` → `"N"`
Immuabilité	on ne peut pas modifier `s[i]`	construire une nouvelle chaîne
Concaténation	`s1 + s2`	`"Bon" + "jour"` → `"Bonjour"`
Répétition	`s * n`	`"ab" * 3` → `"ababab"`
Appartenance	`x in s`	`"a" in "abc"` → `True`
f-string	`f"...{var}..."`	`f"J'ai {age} ans"`

La suite. Les méthodes de manipulation des chaînes (parcours, recherche, découpage, etc.) sont présentées dans le chapitre suivant : 07 - Chaînes : méthodes.

Exercices

Exercice 1 *Accès et longueur* **(3 points)** Soit la chaîne `mot = "algorithme"`.

1. Combien de caractères contient-elle ?
2. Quel est le caractère d'indice 0 ? D'indice 4 ? D'indice –1 ?
3. Que produit `mot[10]` ? Pourquoi ?

Exercice 2 *Concaténation et f-strings* **(3 points)**

1. Écrire un programme qui demande le prénom et le nom de l'utilisateur, puis affiche son nom complet et son nombre de caractères.

   Exemple : prénom `"ada"`, nom `"lovelace"` → affiche `ada lovelace (12 caractères)`.

2. Écrire un programme qui demande un nombre décimal et l'affiche arrondi à deux chiffres après la virgule en utilisant une f-string.