

1 Le type str

1.1 Définition des chaînes de caractères

Définition 1.1 Une *chaîne de caractères* est une collection ordonnée (ou séquence) de caractères délimitée par des apostrophes ('), des guillemets ("), des triples apostrophes ('') ou des triples guillemets ('''').

En Python, le type de données associé aux chaînes de caractères est **str** pour *string*. Ce sont des données **non modifiables**.

Exemple 1.2 'Bonjour !' est une chaîne de caractères.

Remarque 1.3 Le choix des délimiteurs est le plus souvent imposé par le contenu de la chaîne de caractères. Par exemple la chaîne 'Bonjour !' ne nécessite que des apostrophes, par contre, la chaîne "Aujourd'hui" contenant une apostrophe nécessite des guillemets comme délimiteur.

Définition 1.4 (Longueur d'une chaîne de caractères) On appelle *longueur d'une chaîne de caractères* le nombre d'éléments (de caractères) qui la compose. Cette longueur peut-être obtenue à l'aide de la fonction `len`.

Exemple 1.5 L'instruction `len('Bonjour !')` renvoie 9.

Définition 1.6 Il existe plusieurs manières de définir une chaîne de caractères :

- ★ **Chaîne de caractères vide** : elle est définie par l'instruction '' ou l'instruction ""
- ★ **Chaîne de caractères par extension** : elle est définie en indiquant directement entre délimiteurs les éléments de la chaîne.
Exemple : la chaîne de caractères 'Bonjour !' est une chaîne de caractères définie par extension.
- ★ **Chaîne de caractère par transtypage** : elle est définie en appliquant la fonction `str` sur les données que l'on veut transtyper.
Exemple : l'instruction `str(123)` donne la chaîne de caractères '123'

Définition 1.7 (Chaînes de caractères particulières)

- ★ La chaîne de caractères '\n' permet de faire un saut de ligne.

Exemple : l'instruction `print('Bonjour.\nComment allez-vous ?')` produit l'affichage

```
Bonjour.
Comment allez-vous ?
```

- ★ La chaîne de caractères '\t' permet de faire une tabulation.

Exemple : l'instruction `print('Bonjour.\tComment allez-vous ?')` produit l'affichage

```
Bonjour.    Comment allez-vous ?
```

1.2 Indexation

- ★ Les éléments d'une chaîne de caractères étant ordonnés, ils sont repérés à l'aide d'un indice de la façon suivante :
 - le premier élément de la chaîne a pour indice 0 ;
 - le deuxième élément de la chaîne a pour indice 1 ;
 - le troisième élément de la chaîne a pour indice 2 ;
 - etc.

Ainsi, l'**indice d'un élément dans une chaîne de caractères est égal à la position - 1 de l'élément dans cette chaîne..**

- ★ L'élément d'indice `j` d'une chaîne de caractères `chaine` est défini par l'instruction `chaine[j]`.

Exemple : on considère la chaîne de caractères `chaine = 'Bonjour !'`. Alors :

- `chaine[0]` correspond à 'B' (élément d'indice 0);
- `chaine[1]` correspond à 'o' (élément d'indice 1);
- `chaine[7]` correspond à ' ' (élément d'indice 7);
- `chaine[8]` correspond à '!' (élément d'indice 8).

Remarque : si on veut accéder à un indice qui n'existe pas, on déclenche une erreur de type

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

c'est-à-dire que l'on a voulu utiliser un indice en dehors de la plage autorisée (de 0 jusqu'à `len - 1`).

Remarque 1.8 Les chaînes de caractères n'étant pas modifiables, des instructions du type `chaine = 'BonJour !'; chaine[3] = 'j'` renverront des erreurs du type

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

1.3 Opérateurs

Les deux opérateurs + et * sont compatibles avec les chaînes de caractères :

- ★ L'opérateur + est un **opérateur de concaténation**, c'est-à-dire qu'il permet de rassembler deux chaînes de caractères en une seule.
Exemple : l'instruction 'Bon' + 'jour' renvoie la chaîne de caractères 'Bonjour'.

- ★ L'opérateur * est un **opérateur de duplication**, c'est-à-dire qu'il renvoie une chaîne de caractères dont les éléments sont ceux d'une chaîne de caractères qui ont été dupliqués plusieurs fois.

Exemple : l'instruction 'bon'*2 renvoie la chaîne de caractères 'bonbon' constituée de l'élément 'bon' dupliqué deux fois.

1.4 Tests

- ★ On peut tester si deux chaînes de caractères sont égales ou différentes grâce aux opérateurs de comparaison == et !=
Exemples :

- L'instruction 'Bonjour' == 'Bonjour' renvoie True (les deux chaînes sont égales).
- L'instruction 'Bonjour' != 'Bonjour' renvoie donc False.
- L'instruction 'Bonjour' == 'bonjour' renvoie False (les deux chaînes ne sont pas égales).
- L'instruction 'Bonjour' != 'bonjour' renvoie donc True.

- ★ On peut également comparer deux chaînes de caractères avec les opérateurs de comparaison >, >=, < et <=
Exemples :

- L'instruction 'arbre' < 'bonbon' renvoie True.
- L'instruction 'arbre' > 'bonbon' renvoie donc False.
- L'instruction 'a' < 'A' renvoie False.
- L'instruction 'a' > 'A' renvoie donc True.

- ★ On peut tester si une chaîne de caractères appartient à une autre chaîne de caractères ou non grâce aux opérateurs in et not in
Exemples :

- L'instruction 'Bon' in 'Bonjour' renvoie True ('Bon' apparaît dans la chaîne).
- L'instruction 'Bon' not in 'Bonjour' renvoie donc False.
- L'instruction 'bon' in 'Bonjour' renvoie False ('bon' n'apparaît pas dans la chaîne).
- L'instruction 'bon' not in 'Bonjour' renvoie donc True.

1.5 Parcours de chaînes de caractères

Une chaîne de caractères est une structure qui peut être parcourue afin d'obtenir des informations sur chacun de ses éléments.

Comme pour les listes et les tuples, il existe deux types de parcours : le **parcours par indice** et le **parcours par élément**.

Définition 1.9 On donne ci-dessous la syntaxe Python pour chacun des deux parcours pour une chaîne de caractères chaine :

Parcours par indice :

```
for i in range(len(chaine)):
    bloc d'instructions
```

Parcours par élément :

```
for car in chaine:
    bloc d'instructions
```

Remarque 1.10 Une chaîne de caractères n'étant pas modifiable, ces deux parcours ne permettent d'accéder aux éléments qu'en lecture.

Exemple 1.11 Le tableau ci-dessous donne le code Python permettant d'afficher tous les éléments de la chaîne de caractères chaine définie à la première ligne suivant ces deux types de parcours :

Parcours par indice :

```
chaine = 'Bonjour !'
for i in range(len(chaine)):
    print(chaine[i])
```

Parcours par élément :

```
chaine = 'Bonjour !'
for car in chaine:
    print(car)
```

L'exécution de ces deux codes donne le même affichage :

```
B  
o  
n  
j  
o  
u  
r  
!
```

1.6 Fonctions et méthodes agissant sur les chaînes de caractères

1.6.1 Les fonctions `input` et `print`

« La fonction `input`. » La fonction `input` permet de communiquer avec un utilisateur et de récupérer sa saisie sous forme d'une chaîne de caractères.

Remarque 1.12

- ★ La fonction `input` est une instruction bloquante, c'est-à-dire que le programme attend une réponse de l'utilisateur avant de poursuivre.
- ★ La fonction `input` admet une chaîne de caractères comme paramètre optionnel ; cette chaîne est affichée à l'écran afin de donner des indications à l'utilisateur.
- ★ La fonction `input` renvoie **toujours** une chaîne de caractères. On peut donc lui appliquer les méthodes des chaînes de caractères ou des fonctions permettant le transtypage des données.

Pour stocker le résultat saisi par l'utilisateur, il faut obligatoirement procéder à une affectation.

Exemple 1.13

- ★ L'instruction :

```
x = input("Saisissez votre nom : ")
```

affiche à l'écran la chaîne de caractères :

```
Saisissez votre nom :
```

puis attend que l'utilisateur saisisse des données. Le résultat de la saisie est affecté à la variable `x` qui est alors une chaîne de caractères.

- ★ L'instruction :

affiche à l'écran la chaîne de caractères :

```
n = int(input("Saisissez un nombre entier : "))
```

```
Saisissez un nombre entier :
```

puis attend que l'utilisateur saisisse des données. La chaîne saisie est alors transtypée en entier avec la fonction `int()` puis est affectée à la variable `n`. Attention, si le transtypage n'est pas valide, une exception est levée et le programme s'arrête...

« La fonction `print`. » Pour afficher à l'écran une donnée, on utilise la fonction `print`

Exemple 1.14

Les instructions :

affichent à l'écran :

```
print("Bonjour")
x = "Bonsoir" ; print(x)
y = 25 ; print(y)
```

```
Bonjour
Bonsoir
25
```

Remarque 1.15

- ★ La fonction `print` procède automatiquement à un transtypage des données en chaîne de caractères.
- ★ Par défaut, la fonction `print` renvoie à la ligne. Ainsi, lorsqu'on l'utilise plusieurs fois de suite, les affichages successifs se font sur des lignes différentes.
- ★ La fonction `print` admet un paramètre optionnel `end` qui indique la façon dont se termine la chaîne de caractères affichée. Par défaut, on a `end="\n"`, d'où le passage à la ligne. En modifiant la valeur de ce paramètre, on peut en particulier effectuer des affichages multiples sur une même ligne

Exemple 1.16

Les instructions :

```
print("Bonjour.", end=" ")
print("Comment allez-vous ?")
```

permettent d'afficher les deux chaînes sur une même ligne :

Bonjour. Comment allez-vous ?

Remarque 1.17 La chaîne de caractères qui est affichée à l'écran ne peut bien sûr pas être réutilisée ultérieurement. On réservera donc cette fonction aux procédures de débogage.

1.6.2 Les méthodes `split` et `join`

Une chaîne de caractères n'étant pas modifiable, il est relativement difficile de la manipuler (par exemple, si on veut modifier certains de ses caractères). Pour contourner ce problème, une méthode consiste à transformer la chaîne en liste, à effectuer diverses manipulations sur celle-ci, puis à la retransformer en chaîne par concaténation.

«**La fonction de transtypage `list`.** Une première technique de transformation consiste à appliquer la fonction de transtypage `list()` à la chaîne de caractères que l'on veut manipuler. On obtient alors une liste dont les éléments correspondent aux caractères de la chaîne.

Exemple 1.18 L'instruction `list("Bonjour")` renvoie la liste `["B", "o", "n", "j", "o", "u", "r"]`.

Cette technique de transformation a bien sûr son avantage, mais a également l'inconvénient que toute la chaîne est décomposée caractères par caractères. En particulier, si la chaîne contient des mots (= groupes de caractères séparés les uns des autres par des espaces), il est impossible de les conserver tels quels. Pour contourner ce problème, on utilise une méthode particulière appelée `split`.

«**La méthode `split`.** Si `chaine` est une chaîne de caractères, l'instruction `chaine.split(sep)` renvoie une liste de chaînes de caractères obtenues en découplant la chaîne de caractères originale `chaine` en fonction du séparateur `sep` indiqué. Si le séparateur `sep` n'est pas spécifié, le découpage est effectué suivant les blancs (espaces, tabulations, fins de ligne, fin de fichier). En particulier, les caractères de fin de ligne ou de fin de fichier sont également supprimés. Pour plus de détails, on pourra consulter la [documentation officielle](#).

Exemple 1.19 On considère les instructions suivantes :

```
1 texte = "Bonjour.\nComment allez-vous ?"
2 print(texte)
3 texte.split()
4 texte.split("o")
```

La deuxième ligne affiche le texte `texte` sur deux lignes :

Bonjour.
Comment allez-vous ?

La troisième ligne renvoie la liste `["Bonjour.", "Comment", "allez-vous", "?"]`.

La quatrième ligne renvoie la liste `["B", "n", "ur.\nC", "mment allez-v", "us ?"]`.

Une fois les modifications effectuées sur la liste, il ne reste plus qu'à concaténer ses éléments en une nouvelle chaîne. Pour cela, on utilise une autre méthode particulière appelée `join`.

«**La méthode `join`.** Si `L` est une liste de chaînes de caractères et si `sep` est une chaîne de caractères (éventuellement vide), l'instruction `sep.join(L)` renvoie une chaîne de caractères obtenue par concaténation, suivant le séparateur `sep`, des éléments de la liste `L`.

Exemple 1.20 On considère les instructions suivantes :

```
1 mots = ["Rouge", "Vert", "Bleu"]
2 " ".join(mots)
3 " et ".join(mots)
4 "".join(mots)
```

La ligne 2 renvoie la chaîne "Rouge Vert Bleu".
La ligne 3 renvoie la chaîne "Rouge et Vert et Bleu".
La ligne 4 renvoie la chaîne "RougeVertBleu".

1.6.3 Autres méthodes

Il existe un grand nombre d'autres méthodes agissant sur les chaînes de caractères. On ne les présente pas toutes ici. La liste complète peut être obtenue avec l'instruction `help(str)`.

« Pour compter, trouver et modifier des caractères. »

- ★ `chaine.count(s)` renvoie le nombre de fois où la sous-chaîne de caractères `s` (éventuellement composée d'un seul caractère) apparaît dans la chaîne de caractères `chaine`.
- ★ `chaine.find(s)` renvoie l'indice du début de la première occurrence de la sous-chaîne `s` (éventuellement composée d'un seul caractère) dans la chaîne de caractères `chaine` quand `s` est présente dans `chaine`; elle renvoie `-1` sinon.
- ★ `chaine.index(s)` renvoie l'indice du début de la première occurrence de la sous-chaîne `s` (éventuellement composée d'un seul caractère) dans la chaîne de caractères `chaine` quand `s` est présente dans `chaine`; elle renvoie une erreur sinon : (`ValueError: substring not found`).
- ★ `chaine.replace(s1,s2)` renvoie une copie de la chaîne de caractères `chaine` dans laquelle toutes les sous-chaînes de caractères `s1` ont été remplacées par la sous-chaîne de caractères `s2` (elle renvoie une copie de `chaine` si `s1` n'est pas présente dans `s2`).

Exemple 1.21 Soit la chaîne de caractères `chaine = 'Première NSI'`. Alors :

- ★ `chaine.count('m')` renvoie 1 et `chaine.count('re')` renvoie 2.
- ★ `chaine.find('e')` renvoie 2 et `chaine.find('mi')` renvoie 3.
- ★ `chaine.index('z')` renvoie `ValueError: substring not found` et `chaine.index('re')` renvoie 1.
- ★ `chaine.replace('e', 'z')` renvoie la chaîne de caractères `'Przmièrz NSI'`.

« Pour gérer les majuscules et les minuscules »

- ★ `chaine.capitalize()` renvoie une copie de la chaîne `chaine` avec le premier caractère en majuscule.
- ★ `chaine.isupper()` renvoie `True` si tous les caractères de la chaîne `chaine` sont en majuscules et `False` sinon.
- ★ `chaine.upper()` renvoie une copie de la chaîne `chaine` avec tous les caractères en majuscules.
- ★ `chaine.islower()` renvoie `True` si tous les caractères de la chaîne `chaine` sont en minuscules et `False` sinon.
- ★ `chaine.lower()` renvoie une copie de la chaîne `chaine` avec tous les caractères en minuscules.
- ★ `chaine.swapcase()` renvoie une copie de la chaîne `chaine` dans laquelle les caractères en majuscules ont été convertis en minuscules et inversement.

Noter que toutes ces méthodes renvoient systématiquement une copie de la chaîne initiale. En particulier, la chaîne `chaine` n'est pas modifiée, sauf si on lui réaffecte la copie.

Exemple 1.22 On considère les instructions suivantes :

```
chaine = "bonjour"
chaine.capitalize(); chaine
chaine = chaine.upper(); chaine
chaine.islower()
```

La deuxième ligne renvoie respectivement les chaînes "Bonjour" et "bonjour".

La troisième ligne renvoie la chaînes "BONJOUR" et la dernière ligne renvoie `False`.

1.7 Formatage de texte

1.7.1 Les f-strings

Les f-strings permettent d'insérer des variables dans les chaînes de caractères et de les mettre en forme. Pour les utiliser, il suffit de mettre un `f` devant la chaîne de caractères et, pour insérer la valeur d'une variable dans cette chaîne, il suffit de mettre la variable entre accolades. Si il n'y a pas de variable à substituer, il n'est pas nécessaire de mettre le `f` devant.

Exemple 1.23 On considère les instructions suivantes :

```
prenom = "Jean"
nom = "Dupont"
age = "18"
f"Bonjour. Je m'appelle {prenom} {nom} et j'ai {age} ans."
```

La dernière ligne renvoie la chaîne de caractères "Bonjour. Je m'appelle Jean Dupont et j'ai 18 ans.".

Remarque 1.24 On obtient le même résultat avec la variable `age = 18` qui est un entier. Autrement dit, les f-strings effectuent automatiquement un transtypage pour inclure la donnée dans la chaîne de caractères.

1.7.2 La méthode `format`

Cette méthode permet de formater des chaînes de caractères en y incluant des valeurs issues de variables.

Exemple 1.25 On considère les instructions suivantes :

```
prenom = "Jean"
nom = "Dupont"
age = "18"
"Bonjour. Je m'appelle {0} {1} et j'ai {2} ans.".format(prenom, nom, age)
```

La dernière ligne renvoie la chaîne de caractères "Bonjour. Je m'appelle Jean Dupont et j'ai 18 ans.".

Pour utiliser cette méthode, il suffit donc de passer en argument les valeurs que l'on souhaite intégrer dans la chaîne et de signaler dans la chaîne le numéro correspondant entre accolades.

Remarque 1.26 Comme pour les f-strings, on obtient le même résultat avec la variable `age = 18` qui est un entier. Autrement dit, la méthode `format` effectue automatiquement un transtypage pour inclure la donnée dans la chaîne de caractères.

1.7.3 L'opérateur %

Une autre technique de formatage des chaînes consiste à utiliser l'opérateur `%` afin de définir des « trous » dans la chaîne, puis de les combler avec les données d'un tuple. Par exemple, les instructions

```
prenom = "Jean"
nom = "Dupont"
age = "18"
"Bonjour. Je m'appelle %s %s et j'ai %s ans." % (prenom, nom, age)
```

renvoie la même chaîne de caractères que dans l'exemple 1.23. L'opérateur `%s` indique à Python qu'il doit convertir en `str` les données qui proviennent du tuple. En particulier, si `age = 18` est un entier, cette méthode est encore fonctionnelle. Noter également que les données du tuple sont intégrées dans la chaîne dans l'ordre dans lequel elles ont été écrites.

1.8 Exercices

Exercice 1.27 Compléter le Notebook *NSI Première Partie 1 Chapitre 8 Textes*.

Exercice 1.28 (QCM)

1. On considère la chaîne de caractères `chaine = 'Vive la NSI'`. Que renvoie l'instruction `chaine[2]` ?
 - (a) 'V'
 - (b) 'v'
 - (c) rien
 - (d) une erreur

2. On considère la liste `chaine = ['Vive', 'la', 'NSI']`. Que renvoie l'instruction `chaine[11]` ?
 - (a) 'I'
 - (b) la longueur
 - (c) rien
 - (d) une erreur

3. On considère la liste `chaine = ['Vive', 'la', 'NSI']`. Parmi les affirmations suivantes, lesquelles sont vraies ?
 - (a) l'instruction `len(chaine)` renvoie 11
 - (b) l'instruction `chaine[0] = 'v'` modifie la chaîne en 'vive la NSI'
 - (c) l'instruction `chaine.append('!')` modifie la chaîne en 'Vive la NSI!'
 - (d) l'instruction `chaine + '!'` renvoie la chaîne 'Vive la NSI!'

4. On considère la fonction suivante :

```
def nb_voyelle(chaine):
    ''' chaine est une chaîne de caractères '''
    cpt = 0
    for car in chaine:
        if car in 'àââéêëîîôöûüÿAEIOUY':
            cpt += 1
    return cpt
```

Parmi les affirmations suivantes, lesquelles sont vraies ?

- (a) la fonction nb_voyelle renvoie le nombre de voyelles présentes dans la chaîne de caractères chaine
- (b) l'instruction nb_voyelle('Vive la NSI') renvoie 4
- (c) l'instruction nb_voyelle('') renvoie une erreur
- (d) l'instruction nb_voyelle('BBB') renvoie 3

5. On considère les instructions suivantes :

```
mots = 'Vive les NSI'.split()
print('Il y a {1} mots dans la phrase et elle commence par {0}'.format(..., ...))
```

Parmi les affirmations suivantes, lesquelles sont vraies ?

- (a) mots est une chaîne de caractères
- (b) mots est une liste
- (c) Pour que la phrase ait un sens, il faut écrire format(3, 'Vive')
- (d) Pour que la phrase ait un sens, il faut écrire format(mots[0], len(mots))