Chaines de caractères (hors programme)

Opération (rappel)

Pour les chaînes de caractères, deux opérations sont possibles, l'addition et la multiplication :

```
>>> chaine = "Salut"
>>> chaine + " Python"
'Salut Python'
>>> chaine * 3
'SalutSalutSalut'
```

- L'opérateur d'addition `+` concatène (assemble) deux chaînes de caractères.
- L'opérateur de multiplication `*` entre un nombre entier et une chaîne de caractères duplique (répète) plusieurs fois une chaîne de caractères.

Attention aux opérations illicites :

```
>>> "toto" * 1.3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'float'
>>> "toto" + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Accès aux éléments (lettres)

Comme pour les p-uplets et tableaux, on peut accéder à un caractère d'une chaîne de caractères par sa position :

```
>>> animaux = "girafe tigre"
>>> len(animaux)
12
>>> animaux[3]
'a'
```

Ou à une partie de la chaîne (tranches) :

```
>>> animaux[0:4]
'gira'  >>> animaux[9:]
'gre'   >>> animaux[:-2]
'girafe tig'
```

Modifier une chaîne

Le type `str` est immuable, on ne peut pas modifier un des éléments d'une chaîne de caractère :

```
>>> animaux[4] = "F"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Par conséquent, pour modifier une chaîne de caractères, il faut écraser l'ancienne.

Caractères spéciaux

Il existe certains caractères spéciaux très pratiques, par exemple :

- `\n` pour le retour à la ligne,
- `\t` produit une tabulation,
- `\` (caractère d'échappement) par exemple dans `\'` ou `\"` permet d'écrire des guillemets simples ou doubles sans que ceux-ci ne soient pas confondus avec les guillemets de de la chaîne de caractères.

```
>>> print("Un retour à la ligne\npuis une tabulation\t puis un guillemet\"")
Un retour à la ligne
puis une tabulation     puis un guillemet"
>>> print('J\'affiche un guillemet simple')
J'affiche un guillemet simple
>>> print("Un brin d'ADN")
Un brin d'ADN
>>> print('Python est un "super" langage de programmation')
Python est un "super" langage de programmation
```

Méthodes associées aux chaînes de caractères

Voici quelques méthodes spécifiques aux objets de type str :

- `chaine.upper()` cConvertit la chaîne en majuscule.

  ```
  >>> x = "girafe"
  >>> x.upper()
  'GIRAFE'
  ```

- `chaine.lower()` convertit la chaîne en minuscule.

  ```
  >>> 'TIGRE'.lower()
  'tigre'
  ```

- `chaine.split(sep)` découpe une chaîne de caractères en utilisant le séparateur sep (ou blanc si il n'est pas précisé).

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

- `chaine.find(sub)` renvoie la position d'une sous-chaîne dans une chaine de caractères ou -1 si elle n'est pas trouvée.

```
>>> animal = "girafe"
>>> animal.find("i")
1
>>> animal.find("afe")
3
>>> animal.find("z")
-1
```

- `chaine.replace(x, y)` substitue une sous chaîne x par une autre y dans une chaîne de caractères.

```
>>> animaux = "girafe tigre"
>>> animaux.replace("tigre", "singe")
'girafe singe'
>>> animaux.replace("i", "o")
'gorafe togre'
```

- `chaine.count(x)` renvoie le nombre d'occurrence d'une sous chaîne x dans une chaîne de caractères.

```
>>> animaux.count("i")
2
>>> animaux.count("tigre")
1
```

Ces méthodes n'altèrent pas la chaîne de caractères de départ mais renvoient une nouvelle chaîne de caractère.

` Conversion de type (cast)

La conversion d'un tableau de chaînes de caractères en une chaîne de caractères est un peu particulière puisqu'elle fait appel à la méthode `.join()`.

```
>>> seq = ["A", "T", "G", "A", "T"]
>>> "-".join(seq)
'A-T-G-A-T'
```

Les éléments de la liste initiale sont concaténés les uns à la suite des autres et intercalés par un séparateur qui peut être une chaîne vide `""` pour concaténer plusieurs chaînes ensembles.

PEP 8

Préférer l'utilisation de `.join()` plutôt que `+` pour concaténer des chaines .

```
>>> "".join(seq)
'ATGAT'
```