TP3 - Les chaînes de caractères

M. Tellene

1 Création de l'environnement de travail

Avant de commencer le TP, vous allez créer votre environnement de travail. Pour ce faire vous allez dans votre dossier personnel, puis « apprentissage_python ». Une fois arrivé, créer un dossier « TP3 ». C'est dans ce dossier qu'il faudra mettre tous les exercices du TP.

2 Les chaînes de caractères, qu'est-ce que c'est ?

Une chaîne de caractères (de type string : `str`) est une suite de lettres et/ou de chiffres et/ou de caractères spéciaux. En bref, une chaîne de caractères est une suite de caractères. Pour être reconnue, une chaîne de caractères peut être :

— délimitée par des apostrophes « '...' » (simple quote) :

```
>>> chaine = 'salut à tous'
>>> type(chaine)
<class 'str'>
>>> chaine
'salut à tous'
```

— délimitée par des guillemets « "..." » (double quote) :

```
>>> chaine = "salut à tous"
>>> type(chaine)
<class 'str'>
>>> chaine
'salut à tous'
```

Le langage Python ne fait pas de différence entre ces deux possibilités puisqu'il affiche ensuite dans l'interpréteur avec simple quote dans les deux cas.

Il est également possible de délimiter par des triples guillemets (« """..""" »), ceci autorisent le passage à la ligne lors de l'affichage, mais **cette syntaxe joue un rôle particulier pour les fonctions** que nous verrons plus tard. Par conséquent, **il est très déconseillé d'utiliser les triples guillemets pour faire des chaînes de caractères**.

3 Opération sur les chaînes de caractères

Il est possible d'effectuer des opérations sur les chaînes de caractères au même titre qu'il est possible de faire des opérations sur les nombres.

3.1 La concaténation de chaînes

La concaténation est l'action de mettre bout à bout au moins deux chaînes de caractères. En Python, il est possible de mettre de faire des concaténation de chaînes de caractère avec l'opérateur « + ». En quelque sorte, nous ajoutons une chaînes de caractères à une autre chaîne.

1. Créer un fichier **concatenation.py**
2. Écrire un programme qui concatène 2 chaînes de caractères. Pour ce faire vous créerez un première variable `morceau1` qui sera égale à « Salut ». Vous créerez une deuxième variable `morceau2` qui sera égale à « ça va ? ». Le programme devra concaténer `morceau1` et `morceau2` dans un variable `phrase`, puis afficher le résultat à l'écran
3. Marquer en commentaire du programme le résultat produit lors de l'affichage de la variable `phrase`. En Python un commentaire s'exprime avec le symbole « # ». L'exemple ci-dessous montre un commentaire :

```
# je suis un commentaire Python
# blablabla un autre commentaire
```

   Un commentaire est du code qui n'est pas exécuté par Python

4. Créer maintenant une nouvelle variable `sephra` qui sera égale à la concaténation de `morceau2` et `morceau1` **(attention à l'ordre des variables lors de la concaténation)** . Le programme devra afficher la variable `sephra`
5. Marquer en commentaire du programme le résultat produit lors de l'affichage de la variable `sephra`
6. Quelle est la différence entre les deux affichages ? Comment cela s'explique-t-il ? .....

   .....

   .....

   .....

3.2 Petite mise en garde

Nous pouvons concaténer des chaînes, mais **attention à ne pas mélanger les types** - en tout cas pour la concaténation.

1. Créer un fichier **concatenation_erreur.py**
2. Écrire le programme suivant :

```
morceau1 = "Mon chiffre préféré est le "
morceau2 = 2
phrase = morceau1 + morceau2
print(phrase)
```

3. Que ce passe-t-il quand le programme est exécuté ? .....
4. Que pouvez-vous conclure ? .....

   Il est possible de parer à ce problème avec l'instruction `str(...)`. Pour ce faire, remplacer la ligne

```
morceau2 = 2
```

   par

```
morceau2 = str(2)
```

5. Exécuter le programme et indique en commentaire ce qui s'affiche dans la console

3.3 La répétition de chaînes

Nous avons vu qu'il était possible de concaténer les chaînes de caractères. Nous allons prendre l'exemple suivant :

1. Créer un fichier**repetition_chaine.py**
2. Écrire le programme suivant :

```
morceau1 = "salut à tous !"
morceau2 = "salut à tous !"
morceau3 = "salut à tous !"
phrase = morceau1 + morceau2 + morceau3
print(phrase)
```

3. Si nous exécutons le programme nous aurons comme résultat :

```
'salut à tous !salut à tous !salut à tous !'
```

   Vous pouvez constater que le programme suivant concatène 3 fois **la même chaîne de caractères**. Ceci n'est pas très efficace. En effet, il serait plus judicieux de répéter la chaîne de caractères 3 fois au lieu de la concaténer 3 fois. En Python, cela possible grâce à l'opérateur « * ».

4. Modifier le programme précédent pour qu'il fasse la même chose, mais en utilisant la répétition de chaînes

3.4 Longueur de chaîne

Dans cette partie, nous allons nous intéresser à une opération de base sur les chaînes de caractères **très pratique** : la récupération de taille (ou longueur). En Python, pour récupérer la taille d'une chaîne de caractères on utilise **la fonction** `len(...)`.

Écrire dans **la console** le programme suivant :

```
>>> chaine = "Python c'est cool"
>>> len(chaine)
```

La console devrait vous afficher « 17 ». Ce nombre correspond au nombre de caractères (espaces compris) qui composent `chaine`, si vous n'êtes pas sûr vous pouvez vérifier.

3.5 Extraction de caractères

Dans cette dernière partie, nous allons parler de l'extraction de caractère. Une extraction de caractère consiste à récupérer un caractère précis dans une chaîne de caractères. En Python, nous allons utiliser les crochets « [...] ».

1. Écrire dans **la console** la ligne suivante

```
>>> alphabet = "abcdefghijklmnopqrstuvwxyz"
```

2. Compléter le tableau suivant :

Commande à taper dans la console	Résultat
`len(alphabet)`	
`alphabet[0]`	
`alphabet[1]`	
`alphabet[25]`	
`alphabet[26]`	
`alphabet[-1]`	
`alphabet[len(alphabet)]`	
`alphabet[-len(alphabet)]`	
`alphabet[len(alphabet)-1]`	
`alphabet[len(alphabet)//2]`	

3. À votre avis, pourquoi `alphabet[26]` et `alphabet[len(alphabet)]` produisent des erreurs ?

.....

.....

.....

4 C'est à vous !

EXERCICE 1

Pour cet exercice, créer un fichier **repetition_2.py**

Écrire un programme qui initialise une variable `phrase` avec une chaîne de caractères et une variable `x` avec un nombre. Le programme doit afficher `phrase` `x` fois.

EXERCICE 2

Pour cet exercice, il faudra modifier le fichier **repetition_2.py**

Modifier le programme précédent pour qu'il la chaîne de caractères contienne des retours à la ligne.
Pour vous aider, consulter le site https://www.freecodecamp.org/news/python-new-line-and-how-to-python-print-

Exemple de rendu :

```
phrase = "NSI"
x = 3

#Résultat du programme :

'NSI'
'NSI'
'NSI'
```

EXERCICE 3

Pour cet exercice, il faudra créer un fichier **calcul_longueur.py**

Écrire un programme qui initialise une variable avec une chaîne de caractères. Le programme devra afficher :

- la longueur de la chaîne
- la moitié de la longueur de la chaîne

Exemple de rendu :

```
phrase = "Salut à toutes et à tous les amis !"

#Résultat du programme :

La phrase possede 38 caractères
Cette phrase est composée de deux morceaux de 19 caractères
```

EXERCICE 4

Pour cet exercice, il faudra créer un fichier **longueur_pair_impair.py**

Écrire un programme qui initialise une variable avec une chaîne de caractères.

- Si la phrase possède un nombre de caractères pair alors le programme devra afficher « La phrase possède un nombre de caractères pair »
- Sinon il devra afficher « La phrase possède un nombre de caractères impair »

EXERCICE 5

Pour cet exercice, il faudra créer un fichier **transformation.py**

Écrire un programme qui initialise une variable avec une chaîne de caractère (`mot`) et une variable contenant une lettre (`lettre`). Le programme doit afficher le `mot` en ajoutant `lettre` au début et à la fin de celui-ci.

EXERCICE 6

Pour cet exercice, il faudra créer un fichier **contient_e.py**

Écrire un programme qui initialise une variable avec une chaîne de caractères.

- Si la chaîne possède le lettre « e », le programme devra afficher « La phrase contient un e »
- Sinon il devra afficher « La phrase ne contient pas de e »

Pour savoir si un caractère est dans une chaîne de caractères, on utilise la structure suivante :

```
if lettre_a_tester in chaine_de_caracteres:
    #si lettre_a_tester est dans chaine_de_caractères alors on va là
else:
    #si ce n'est pas le cas on va là
```

EXERCICE 7

Pour cet exercice, il faudra créer un fichier **deux_chaines_une.py**

Écrire un programme qui fait une chaîne de caractères à partir de 2. Pour ce faire, le programme devra initialiser deux variables contenant une chaîne de caractères (`mot1` et `mot2` par exemple). La chaîne de caractères finale devra être égale au deux premiers caractères de `mot1` et au deux derniers caractères de `mot2`.

EXERCICE 8

Pour cet exercice, il faudra modifier le fichier **deux_chaines_une.py**

Vous avez peut-être remarqué que votre programme ne gère pas les petites chaînes de caractères. Tester le programme **deux_chaines_une.py** avec *un* et *a*.

La mot formé, ne correspond pas à ce que nous avons demandé, car la variable `mot2` contient le chaîne de caractères *a*. On ne peut donc pas avoir les deux derniers caractères. Nous allons donc modifier le programme pour qu'il gère les petites chaînes.

Le programme devra, si un des deux mots contient moins de deux caractères, afficher « Désolé la construction du mot est impossible ».

RÉSUMÉ :

- une chaîne de caractères Python est délimitée par '...' ou "..."
- on peut **concaténer** deux chaînes de caractères en une avec **l'opérateur « + »**
- il n'est pas possible de concaténer une chaîne de caractères avec un `int` ou un `float`
- on peut **répéter** un certain nombre de fois une chaîne de caractères avec **l'opérateur « * »**
- la fonction Python `len(chaine)` permet de récupérer le nombre de caractères contenu dans `chaine`
- on peut **extraire** un ou des caractères d'une chaîne en utilisant les **[...]**