- Les variables de type string en python -

Une variable de type str (pour string) est une chaîne de caractères.
C'est donc une séquence de caractères quelconques dans un ordre précis.

Un caractère peut aussi bien être une lettre qu'un chiffre, qu'un signe de ponctuation ou qu'un caractère non imprimable (retour à la ligne, tabulation, etc.).

>>> help(str) pour afficher la documentation complète de la classe **string** dans la console.

1) Déclaration d'une chaîne de caractères :

```
# déclaration d'une variable de type str

# chaînes de caractères vides
texte1 = str()
texte2 = ''
texte3 = ""
texte4 = """"""

# exemples de variables de type string
texte5 = 'une chaîne de 27 caractères'
texte6 = "0123456789"
texte7 = """un
texte
sur
cinq
lignes."""
```

Dans la console :

```
>>> texte7
'un\ntexte\nsur\nquatre\nlignes.'
>>> print(texte7)
un
texte
sur
quatre
lignes.
```

\n correspond à un retour à la ligne dans une chaîne de caractères.

```
# chaînes de caractères avec apostrophe
texte8 = 'L\'été s\'en est allé...'
texte9 = "Et l'hiver arrive !"

# chaînes de caractères avec guillemets
texte10 = 'un petit "bonjour" pour l\'exemple'
texte11 = "et le même \"bonjour\" ici"

# chaînes de caractères avec tabulation \t ou retour à la ligne \n
texte12 = 'nom\tprénom'
texte13 = "nom\nprénom"
```

```
>>> texte8
"L'été s'en est allé..."
>>> print(texte8)
L'été s'en est allé...
```

```
>>> texte12
'nom\tprénom'
>>> print(texte12)
nom prénom
```

```
>>> texte13
'nom\nprénom'
>>> print(texte13)
nom
prénom
```

Le caractère d'échappement **** est utilisé avant d'insérer des caractères qui doivent être interprété par Python de manière particulière, et non pas comme un simple caractère supplémentaire de la chaîne de caractères.

2) Opérations sur les chaînes de caractères :

```
30 # concaténation de chaînes de caractères : opérateur +
31 mot1 = 'abcdef'
32 mot2 = "12345"
33 mot3 = mot1 + mot2
34 mot4 = mot2 + " " + mot1
35 mot5 = mot4 + '\t' + mot4
```

```
>>> mot3
'abcdef12345'
>>> mot4
'12345 abcdef'
>>> print(mot5)
12345 abcdef    12345 abcdef
```

```
37 # opérateur *
38 mot6 ='abc123'
39 mot7 = mot6 * 3
```

```
>>> mot7
'abc123abc123abc123'
```

3) Longueur d'une chaîne de caractères : *fonction* len(chaine)

```
>>> exemple = "abc\t123"
>>> len(exemple)
7
```

Le caractère tabulation **\t** compte pour un seul caractère dans la longueur de la chaîne.

4) Accéder à un caractère à partir de sa position dans la chaîne : chaine[index]

```
>>> exemple[0]
'a'
>>> exemple[-1]
'3'
>>> exemple[3]
'\t'
>>> exemple[-5]
'c'
>>> exemple[len(exemple) - 1]
'3'
```

le premier caractère est à la position d'index 0

le dernier caractère, d'index -1

le 4ème caractère, d'index 3

le 5ème caractère en partant de la fin, d'index -5

le dernier caractère encore, d'index len(chaine) - 1

5) Obtenir la position d'un caractère donné : *méthode* chaine.index('caracteres')

```
>>> exemple
'abc\t123'
>>> exemple.index("b")
1
>>> exemple.index("3")
6
```

6) Parcourir une chaîne de caractères :

```
>>> mot = "Hello !"
>>> for k in range(len(mot)):
...     print(mot[k])
...
H
e
l
l
o

!
```

```
>>> for lettre in mot:
...     print(lettre)
...
H
e
l
l
o

!
```

Exercice n° 1 :

- Écrire une fonction qui prend en argument une chaîne de caractères, et renvoie une nouvelle chaîne de caractère, qui est l'envers de celle donnée en argument.
- Écrire une fonction qui prend une chaîne de caractères en argument, et renvoie True si cette chaîne est un palindrome (un mot qui se lit de la même manière dans les deux sens), et False sinon.

7) Parties de chaînes de caractères : les *slices* chaine[deb : fin]

```
>>> essai = "0123456789"
>>> essai[2:]
'23456789'
```
du 3ème au dernier caractère

```
>>> essai[:5]
'01234'
```
du 1er au 5ème caractère, celui d'index 4

```
>>> essai[2:5]
'234'
```
du 3ème au 5ème caractère, de l'index 2 à l'index 4

```
>>> essai[-5:]
'56789'
```
les 5 derniers caractères

```
>>> essai[:-2]
'01234567'
```
tous les caractères, sauf les 2 derniers

```
>>> essai[-5:-2]
'567'
```
tous les caractères, sauf les 5 premiers et les 2 derniers

```
>>> essai[2:8:2]
'246'
```
du 3ème au 8ème caractère, par **pas** d'indice 2

```
>>> essai[8:2:-1]
'876543'
```
du 8ème au 3ème caractère, en reculant de 1

```
>>> essai[-2:-5:-1]
'876'
```
du 2ème au 4ème caractère en partant de la fin

8) Les variables de type **str** sont dites **non mutables**.

Une fois la variable chaîne de caractères déclarée, il n'est plus possible de la modifier.

Exemples :

Il est impossible de modifier un caractère dans une variable de type **str** :

```
>>> phrase = "On continue !"
>>> phrase[2]
' '
>>> phrase[2] = "\t"
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Le fait de changer les lettres en majuscules ne modifie pas la chaîne de départ :

***méthodes* .upper()** et **.lower()** :

```
>>> phrase
'On continue !'
>>> phrase.upper()
'ON CONTINUE !'
>>> phrase
'On continue !'
>>> phrase.lower()
'on continue !'
```

Pour obtenir la chaîne de caractères modifiée, il faut utiliser une nouvelle variable :

***méthode* replace(car1, car2)** :

```
>>> phrase
'On continue !'
>>> autre = phrase.replace("n", "$")
>>> autre
'O$ co$ti$ue !'
>>> phrase
'On continue !'
```

Exercice n° 2 :

Écrire une fonction qui prend une chaîne de caractères en argument, et renvoie cette chaîne de caractères avec une majuscule en chaque début de mot, quelle que soit la forme de la chaîne de caractères entrée.

9) L'opérateur is et la fonction id() :

10) Quelques méthodes du type str :