# Labo 6

## Chaines de caractères

## Boucles dans chaines de caractères

# Objectifs de ce laboratoire

Exercices avec:

- Variables de type chaines de caractères
- Boucles dans chaines de caractères

# Chaines de caractères

```
>>> s = 'bonjour'
>>> type(s)
<class 'str'>
```

- La séquence **\n** dans une chaine provoque un saut à une nouvelle ligne.
- La séquence **\'** permet d'insérer une apostrophe dans une chaine délimitée par des apostrophes.
- De la même manière, la séquence **\"** permet d'insérer des guillemets dans une chaine délimitée elle-même par des guillemets.
- Rappelons encore ici que la casse est significative dans les noms de variables (il faut respecter scrupuleusement le choix initial de majuscules ou minuscules).

# Concaténation, répétition, in

- Les chaines peuvent être *concaténées* avec l'operateur **+** et *répétées* avec l'operateur *****

```
>>> n = 'abc' + 'def' # concaténation
>>> m = 'zut ! ' * 4 # répétition
>>> print(n, m)
abcdef zut ! zut ! zut ! zut !
```

- L'instruction **in** peut être utilisée indépendamment de **for**, pour *vérifier si un élément donné fait partie ou non d'une séquence*.

```
>>> 'a' in 'abba'
True
```

# Triple quotes

- Pour insérer plus aisément des caractères spéciaux dans une chaine, sans faire usage de l'*antislash*, ou pour faire accepter l'*antislash* lui-même dans la chaine, on peut encore délimiter la chaine à l'aide de *triples guillemets* ou de *triples apostrophes* :

```
>>> s = """aa
    bb
    cc
    """
>>> s
'aa\n    bb\n    cc\n    '
```

# Exercice 1

En utilisant l'interpréteur Python, affectez la valeur de type chaine de caractères 'bon' à une variable s1, 'mauvais' a une variable s2 et 'fou' a une variable s3.

Écrivez des expressions Python avec les variable s1, s2, et s3 pour:

a) 'ou' est contenu en s3
b) un espace n'est pas contenu en s1
c) la concaténation de s1, s2, et s3
d) l'espace est contenu dans la concaténation de s1, s2, et s3
e) la concaténation de 10 copies de s3
f) Le nombre total de caractères dans la concaténation de s1, s2, et s3

# Indiçage, extraction, longueur

- Les chaines sont des *séquences* de caractères. Chacun de ceux-ci occupe une place précise dans la séquence. Les éléments d'une séquence sont *indicés* (ou numérotés) *à partir de zéro*.
- Si l'index est négatif, il est considéré par rapport à la fin de la chaine. -1 désignera le dernier caractère, -2 l'avant dernier, etc.

```
>>> nom = 'Cedric'
>>> print(nom[1], nom[3], nom[5])
e r c
>>> print (nom[-1], nom[-2], nom[-4], nom[-6])
Cid
>>> print(len(nom))
6
```

# Extraction de fragments de chaînes

- *Slicing* (découpage en tranches) indique entre crochets les indices correspondant au début et à la fin de la tranche que l'on souhaite extraire:

```
>>> ch = "Juliette"
>>> print(ch[0:3])
Jul
>>> print(ch[:3]) # les 3 premiers caractères
Jul
>>> print(ch[3:]) # tout ce qui suit les 3
                  # premiers caractères
iette
```

# Exercice 2

1. En utilisant l'interpréteur Python, créez une variable nommée aha et affectez-lui la valeur 'abcdefgh' .

2. Écrivez des expressions Python (dans l' interpréteur) en utilisant la variable aha qui seront évaluées à:

a) `'abcd'`
b) `'def'`
c) `'h'`
d) `'fg'`
e) `'defgh'`
f) `'fgh'`
g) `'adg'`
h) `'bd'`

# Methodes pour les chaines de caracteres (str)

On ne peut pas modifier les chaines de caractères directement.

| Usage | Explication |
|---|---|
| `s.capitalize()` | retourne une copie de `s` qui commence avec majuscule |
| `s.count(cible)` | retourne le nombre de fois la valeur de `cible` arrive en `s` |
| `s.find(cible)` | retourne le index de la première occurrence de `cible` en `s` |
| `s.lower()` | retourne une copie de `s` en minuscules |
| `s.replace(vieux, nouveau)` | retourne une copie de `s` avec `vieux` remplacé par `nouveau` (toutes les occurrences) |
| `s.split(sep)` | retourne une liste de sous-chaines (fragments) de `s`, délimité par `sep` |
| `s.strip()` | retourne une copie de `s` sans des espaces au début et a la fin |
| `s.upper()` | retourne une copie de `s` en majuscules |

# Exercice 3

Copiez cette expression dans l' interpréteur Python:

s = ''' En 1815, M. Charles-François-Bienvenu Myriel était évêque de Digne. C'était un vieillard d'environ soixante-quinze ans ; il occupait le siège de Digne depuis 1806. … '''

(Le début du roman Les misérables par Vctor Hugo.)

Effectuez les exercices suivantes dans l'interpréteur:

(a) Créez une copie de s, nommé nS, avec les caractères . , ; et \n remplacées par des espaces.

(b) Effacez les espaces qui sont au début ou a la fin de nS (et affectez le nouveau chaine a la même variable nS).

(c) Changez tous les caractères de nS en minuscules (et nomme zle nouveau chaine nS).

(d) Calculez le nombre des fois nS contient 'de'.

(e) Changez tous les sous-chaine était a est (et nommez le nouveau chaine nS).

# Exercice 4

- Écrivez une fonction Python nommée `compte` qui va calculer le nombre d'occurrences d'une caractère c dans une chaine s. Essayez deux version: avec la méthode `count` de la class `str` et sans cette méthode (en utilisant une boucle while ou for).

- Développez la partie principale du programme qui obtient de l'utilisateur une chaine de caractères nommé s, et appelle la fonction deux fois pour calculer le nombre de 'a', et le nombre de 'de la'. La dernière partie doit être:

```
print(compte(s,'a'))
```

# Exercice 5

- Écrivez une fonction Python nommé `espaces` qui prend une chaine de caractères s et retourne  une autre chaine avec des espaces insérés entre les lettres voisines. N'utilise pas `print` dans la fonction. La chaine retournée ne devrait pas avoir d'espace a la fin.

- Testez la fonction avec un programme principal, ou dans l'interpréteur. Par exemple:

```
>>> espaces(‘important')
'i m p o r t a n t'
```

# Exercice 6

- Écrivez une fonction Python nommée coder prend une chaine de caractères s et retourne une autre chaine codée. Le code est calculé par prendre chaque paire de lettres consécutives en échangeant l'ordre dans la paire (espaces, ponctuation, etc. sont traités comme les lettres).
- Testez votre fonction avec un programme principal, ou dans l'interpréteur. Par exemple:

```
>>> codez('message secret')
'emssga eesrcte'
>>> coder('Message')
‘eMssgae'
```