# Python - Données en Liste et string

Les listes et chaines de caractères (types list et str) sont des types construits, des séquences:
- on peut les traiter par partie.
- Les éléments y sont rangés dans un certain ordre.
- Leur position est repérée par un indice.

Voir fiche et TP sur les indices

**FOR ET IN**
Les constructions *for* et *in* de Python sont extrêmement utiles. Nous allons les utiliser pour la première fois avec des listes. La construction *pour* -- for var in list -- permet d'examiner facilement chaque élément d'une liste (ou d'une autre collection

```
squares = [1, 4, 9, 16]
  sum = 0
  for num in squares:
    sum += num
  print(sum)
```

**FOR ET IN RANGE()**
in range va créer un itérable numérique. Cet itérable peut être utilisé pour placer une fonction de l'indice utilisé pour parcourir les éléments de liste:

```
squares = [1, 4, 9, 16]
for i in range(len(square)):
    print(i,':',squares[i])  -> 0:1, 1:4, 2:9, 3:16
    print(i,squares[3-i]) -> 0:16, 1:9, 2:4, 3:1
    print(i,square[i%2]) -> 0:1, 1:4, 2:1, 3:4
```

**FONCTIONS ET MÉTHODES ASSOCIÉES AUX LISTES**

**Fonctions**
len  longueur de la liste
min, max element de valeur min ou max dans la liste
sorted nouvelle liste triée à partir d'une copie de celle d'origine (alors que sort la trie en place)
list créé une liste à partir d'un itérable

**Méthodes: (voir tableau ci-contre)**

| Method | Description |
|---|---|
| append() | Adds an element at the end of the list |
| clear() | Removes all the elements from the list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| extend() | Add the elements of a list (or any iterable), to the end of the current list |
| index() | Returns the index of the first element with the specified value |
| insert() | Adds an element at the specified position |
| pop() | Removes the element at the specified position |
| remove() | Removes the first item with the specified value |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

**FONCTIONS ET MÉTHODES ASSOCIÉES AUX STR**

Seule la fonction `len` sera utilisée avec le type STR

Parmi les méthodes de listes, seules les méthodes `count` et `index` sont communes avec le type STRING.

Le type STRING possède aussi les méthodes **`find`**`, join, replace et` **`split`**`. (en gras, celles vue en TP).`

La différence vient du fait que list est un type mutable (modifiable en partie), alors que str ne l'est pas.

**ALGORITHMES UTILES SUR LES STR (SUR FICHE)**

fonction va compter le nombre de fois où le `caractere` est présent dans `mot`.

```
def occurence(mot,caractere):
    s = 0
    for c in mot:
        if c == caractere:
            ... # a completer
    return ...

>>> occurence('lustucru')
3
```

la fonction `binaire_decimal` calcule la valeur décimale d'un nombre binaire mis en paramètre.

```
b = "10001111"
N = int(b[0])*128 + …
>>> binaire_decimal("10001000")
136
```

fonction `permute` qui change tous les bits d'un nombre binaire sur 1 octet:

```
def permute(nombre):
    nombre2 = ""
    for c in nombre:
        if c == '1':
            ... # a completer
        else:
            ...
    return ...

permute("10100011")
>>> permute("10100011")
"01011100"
```

renverse qui prend pour paramètre mot et qui retourne le mot à l'envers:

```
def renverse(mot):
    mot2=""
    for c in mot:
        mot2 = c + mot2
    return mot2

>>> renverse('LEON')
NOEL
```

`ajoute1` qui ajoute 1 au nombre binaire sur 1 octet. Cette fonction aura pour paramètre `nombre`, une chaine de caractères avec 8 bits. Utiliser la fonction `renverse` pour commencer l'addition sur le chiffre le plus à gauche de `nombre`. Cette fonction devra ne tiendra pas compte de l'eventuel depassement:

```
def ajoute1(nombre):
    retenue = 1
    n_renverse = renverse(nombre)
    nombre2 = ""
    for c in n_renverse:
        if retenue == 1 and c == '0':
            nombre2 = '1' + nombre2
            retenue = 0
        elif retenue == 1 and c == '1':
            nombre2 = '0' + nombre2
            retenue = 1
        else:
            nombre2 = c + nombre2
        print(nombre2)
    return nombre2

>>> ajoute1("11010111")
"11011000"
>>> ajoute1("11111111")
0
```

# Listes - TP python n°5

à partir de la liste:
s = ['lundi', 'mardi',  'mercredi']

1. Tester compléter le tableau:

| proposition | résultat/commentaire |
|---|---|
| s[0] | |
| s[1] | |
| s[2] = "jeudi" | |
| s[4] = "samedi" | erreur de type: … … |

2. *Question a:* Comment modifie-t-on la liste `['lundi', 'mardi',  'mercredi']` pour obtenir `['lundi', 'mardi', 'jeudi']`?

3. Pourquoi l'instruction `s[4] = "samedi"` génère t-elle une erreur?

4. Tester les propositions et compléter à partir de la nouvelle liste:
s = ['lundi', 'mardi',  'mercredi', 'jeudi', 'vendredi']

| proposition | résultat/commentaire |
|---|---|
| s[1:] | |
| s[1:4] | |
| s[:4] | |
| s[1:-1] | |
| s[0:-1] | |

5. *Question b:* Que retourne la proposition `s[1:]`? Découpe t-elle la liste à partir du premier élément, du 2e élément, ou bien retourne t-elle la liste entière?

6. *Question c:* Que vaut t à la fin du script `t[2] = t[2] + 5`? La valeur 5 est-elle ajoutée à chaque élément de la liste, ou bien à un seul élément?

7. à partir de la liste:
s = ['lundi', 'mardi',  'mercredi']

Compléter:

| proposition | résultat/commentaire |
|---|---|
| len(s) | |
| s.append('jeudi') | |

| | |
|---|---|
| len(s) | |
| s.append('vendredi') | |
| len(s) | |
| s.pop() | |

7. *Question e:* Pourquoi la valeur renvoyée par `len(s)` évolue t-elle au cours de l'exercice?

8. *Question f:* Y-a-t-il une différence entre
   * l'opérateur `+`appliqué à une chaine de caractères
   ```
   debut = "20"
   fin = "22"
   debut + fin
   ```
   * l'opérateur `+` appliqué à une liste?
   ```
   debut = [2,0]
   fin = [2,2]
   debut + fin
   ```

9. Compléter le script pour afficher l'élément de rang 3:
   ```
   n = 3
   semaine = ['Lundi', 'Mardi','Mercredi','Jeudi',...]
   print(…)
   ```

à partir de la liste suivante:
```
matrice  = [[1,2,3], [4,5,6],  [7,8,9],  [10,11,12]]
```

10. *Question g:* Recopier et compléter le tableau:

| proposition | valeur |
|---|---|
| matrice[0] | retourne le premier élément de `matrice` |
| matrice[1] | … |
| matrice[1][2] | |
| matrice[2][1] | |
| matrice[3][0] | |

*11. Question h: remplir le tableau*

| proposition | valeur |
|---|---|
| matrice[.. ][.. ] | 2 |
| matrice[. ][. ] | 4 |
| matrice[.. ][.. ] | 12 |

12. *Question j:* Quelle instruction permet d'obtenir une diagonale de 'X' pour la matrice:
```
tictactoe = [['X', 'O',  'O'],
             ['O', 'O',  'O'],
             ['O', 'O',  'X']]
```