

Sciences Numériques et Technologie

Ressources dans <https://philippereneviergonin.github.io/snt/>

Philippe.Renevier-Gonin@ac-grenoble.fr

Les listes / tableaux

- Version « simplifiées »
- Une liste est un ensemble de valeurs dans une même « variable »
- Les valeurs d'une liste sont numérotées de 0 à (taille-1) où taille est la longueur de la liste (le nombre d'éléments qu'elle contient)

Liste en python

- `[]` est une liste vide
- `tableau = ["a", "b", "c", "d", "e"]`
- `tableau` est une liste de 5 éléments.
- Son premier élément à l'indice 0. `tableau[0]` est "a"
- Son dernier élément à l'indice 4 ($\text{taille}-1==5-1==4$). `tableau[4]` est "e"
- On peut ajouter des éléments à la fin d'un tableau avec « append » :
après `tableau.append("f")` `tableau` devient `["a", "b", "c", "d", "e", "f"]`
- Une liste de python peut contenir n'importe quel type de variables/données (int, str, une autre liste, etc.)

Parcours d'une liste

- On peut parcourir les valeurs d'une liste sans savoir leur position dans la liste.

Dans l'exemple suivant, valeur va prendre à tour de rôle toutes les valeurs comprises dans tableau

```
for valeur in tableau:  
    print(valeur)
```

- On peut parcourir les valeurs en se basant sur les index :

Dans l'exemple suivant, index va prendre toutes les valeurs de 0 à la longueur de la liste donnée par len(tableau)

Note : len est pour **length**

```
for index in range(0, len(tableau)):  
    print(tableau[index])
```

Les Fonctions

- Structuration du code (isolation)
 - Qui pourront être appelées à différents moments
 - Qui pourront être ré-utilisées
 - C'est possible avec d'autres approches que nous ne verrons pas (découpage en plusieurs fichiers, programmation orientée objet, programmation fonctionnelle, etc.)
- Gain en clarté et en lecture
- Factorisation (non-duplication) du code

Qu'est qu'une fonction ?

- C'est une partie de code délimitée et nommée que l'on peut appeler
- Avantages :
 - Non ré-écriture du code à chaque utilisation.
Exemple : à partir des données gps et des photos lors d'une randonnée, vous voulez envoyer un « post » sur différents réseaux sociaux pour chaque faits « marquants » :
 - Le début (avec une photo s'il y a)
 - Pour chaque photo
 - La fin (avec une photo s'il y a)
 - Pour chaque « faits marquants », c'est le même code pour poster un message
 - Structure le code en isolant des parties de code
 - Le nom de la fonction doit permettre de savoir ce qu'elle fait
 - Le contenu de la fonction est « inconnu » du code qui l'appel
 - Potentiellement ré-utilisation (c.f. les fonctions des modules importés comme requests et csv)

Une fonction en python

- Fonctions prédéfinies
 - Usage : `nom_de_la_fonction()` ou `nom_de_la_fonction(paramètre1, paramètre2)` (etc.)
 - Dans python
Exemples : `print`, `input`, etc.
 - Dans des bibliothèques
- Définir ses propres fonctions
 - Mot-clé : `def`
 - Un nom unique
 - Puis on utilise comme les fonctions prédéfinies

Une fonction en Python

Mot-clé **def** qui annonce la définition d'une fonction

Le nom de la fonction qui détermine comme elle sera appelée. Le nom doit expliquer ce qu'elle fait

Paramètre de la fonction. Il peut ne pas y en avoir : ()

```
def formatage_nb_allumettes(nbAll) :
```

Les « : » sont obligatoires

```
    formatage = ""
```

```
    if nbAll == 0:
```

```
        formatage = "zéro allumette"
```

```
    elif nbAll == 1:
```

```
        formatage = "une seule allumette"
```

```
    else:
```

```
        formatage = str(nbAll)+" allumettes"
```

```
    return formatage
```

« corps » de la fonction : c'est le code qu'elle contient. Ce peut être long, court, avec des boucles, des « if », des appels à d'autres fonctions, etc. C'est l'indentation qui détermine la fin de la fonction

La fonction peut retourner une valeur : print ne retourne rien, input retourne une string. C'est marqué par le mot-clé **return**

Paramètres et variables locales

- Une variable définie dans une fonction (ex: formattage dans le transparent d'avant) n'existe que dans la fonction
- Une variable définie dans une fonction est qualifiée de « **locale** »
 - Par opposition, une variable définie en dehors des fonctions, comme dans le bloc « `if __name__ == '__main__':` », ces variables sont dites « **globales** » car définies partout
- Un paramètre nommé n'existe que dans la fonction
- **Attention** : une variable locale ou un paramètre d'une fonction peuvent avoir le même nom qu'une autre variable définie « ailleurs » (par exemple dans le « `if __name__ == '__main__':` »), mais ce n'est pas la même « variable », ce sont des homonymes.

Fonction avec ou sans valeur de retour

- Une fonction peut ne pas retourner de valeur
 - Cas d'une fonction qui ne fait que des « print » par exemple
 - Ou qui ne modifie que des variables « globales », etc.
- Pour retourner une valeur (n'importe quelle valeur) : **return**
 - Le code dans la fonction après le return n'est pas exécuté
 - Plusieurs « return » sont possibles, pour couvrir différent cas
- Récupération d'une valeur retournée : `coupOrdi = ordi_fort(nbAllumettes)`
 - `coupOrdi` : la variable qui reçoit la valeur retournée
 - `ordi_fort()` est la fonction appelée
 - `nbAllumettes` est la valeur passée en paramètre
 - Si jamais la fonction ne retourne rien, la variable `coupOrdi` vaudra « None »