Sciences Numériques et Technologie

Ressources dans https://github.com/PhilippeRenevierGonin/snt

Philippe.Renevier@ac-grenoble.fr

elif : instructions conditionnelles (if) avec plusieurs alternatives

- Exemple : une mention au bac
- Plus de 2 alternatives pour le « même test »
- Une notation raccourcie « elif » pour éviter « else if »
  - Pas de décalage de l'indentation
  - Alors que les conditions sont « au même niveau »

Il y a 5 cas : [16,20] , [14-16[, [12-14[, [10-12[et si la note est inférieure à 10

elif par l'exemple « mention au bac »

Le « if » commence « normalement », par le 1er cas

```
if (note >= 16):
    print("c'est une mention TB")
elif (note >= 14):
    print("c'est une mention B")
elif (note >= 12):
    print("c'est une mention AB")
elif (note >= 10):
    print("c'est une mention passable / sans mention")
else:
    print("malheureusement, le bac n'est pas obtenu")
```

Ici on regarde le 2e cas : entre 14 et 16 (exclu)
Ce test n'est fait que si le premier est faux
S'il est vrai, on sera bien entre 14 et 16

Ici on regarde le 3e cas : entre 12 et 14 (exclu)
Ce test n'est fait que si les deux premiers sont faux
S'il est vrai, on sera bien entre 12 et 14

Et ainsi de suite : si tous les tests précédents sont faux, ce test-là est effectué…

Finalement, il est possible de terminer le « if » « normalement », avec un « sinon » pour le cas où toutes les conditions précédentes sont fausses

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 le code à chaque appel.
    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. la fonction get du module requests)

Une fonction en python

- Foncions 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 : ( )

Les « : » sont obligatoires

```
def formattage_nb_allumettes(nbAll) :
    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 »