Apprendre à programmer avec Python

L5FL024 / LYOU008 - Groupe L

Séance 6 : boucles et variables

Marine DELABORDE

Sorbonne Nouvelle
université des cultures

CC BY NC ND

Informations générales

Rappels

- **Horaires** : le mardi de 17h à 19h (discord)
- **Modalités d'enseignement** : distanciel synchrone
- **Contact** : marine.delaborde@gmail.com
- **Ressources** :
  - icampus + http://www.linguist.univ-paris-diderot.fr/~amsili/Ens/L5FL024.php
  - discord https://discord.gg/37vxR6
- **Fiche d'informations :** https://framaforms.org/fiche-dinformations-1600359707
- **Fiche de présence:** https://lite.framacalc.org/9jbo-presence-python-gl
- **Modalités de contrôle des connaissances** :

  Quiz (4 meilleurs sur 5) 50% + Examen (dernière séance) 50%

Correction du quizz 2

- **Quiz 2/5 : 10 minutes**

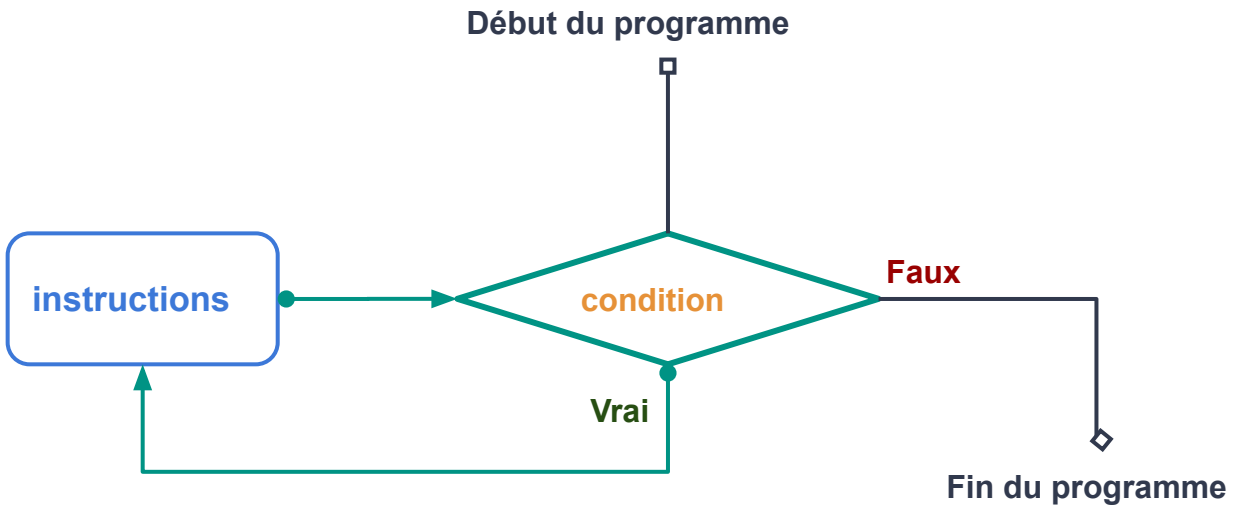
- **Les notes des 4 meilleurs quiz seront prises en compte pour représenter 50% de la note finale du cours**

exercices sur repl.it : https://repl.it/languages/python

Corrections sur :
http://www.linguist.univ-paris-diderot.fr/~amsili/Ens/L5FL024.php

Les boucles While en Python (rappel)

- **Répéter un bloc d'instructions, tant qu'une condition est vraie :**



Les boucles While en Python (rappel)

- **Répéter un bloc d'instructions, tant qu'une condition est vraie :**

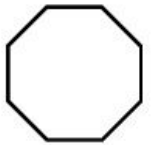
```
while condition:
        instructions
        instructions
        ...
```

Les boucles While en Python (rappel)

- **Répéter un bloc d'instructions, tant qu'une condition est vraie :**

```
compteur = 0 #initialisation de la variable "compteur" (c'est comme une constante dont la valeur va changer)

while compteur < 8: #tant que la variable "compteur" est strictement inférieure à 8
        fd(20) #trace un trait de 20 pixels
        rt(45) #pivote le curseur de 45 degrés vers la droite
        compteur = compteur + 1 #incrémente la variable "compteur" de 1 à chaque boucle
```

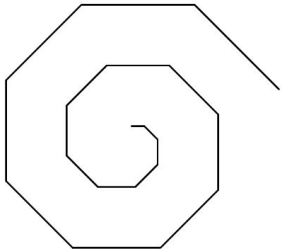


Les boucles While en Python

- **Répéter un bloc d'instructions, tant qu'une condition est vraie :**

```
longueur = 10 #initialisation de la variable "compteur" à 10

while longueur < 100: #tant que la variable "longueur" est strictement inférieure à 100
        fd(longueur) #trace un trait égal à la valeur de la variable "longueur"
        rt(45) #pivote le curseur de 45 degrés vers la droite
        longueur = longueur + 5 #incrémente la variable "longueur" de 5 à chaque boucle
```



Les boucles While en Python

Pour plus tard : https://fr.wikibooks.org/wiki/Programmation_Python/Op%C3%A9rateurs

Opérateurs de comparaison	
<	inférieur
>	supérieur
==	égal
<=	inférieur ou égal
>=	supérieur ou égal
!=	différent
is (not)	même objet (ou pas)
and / or	ET / OU

Opérateurs mathématiques	
+	addition
-	soustraction
*	multiplication
**	puissance
/	division

Opérateurs d'affectation augmentés	
+=	addition
-=	soustraction
*=	multiplication
**=	puissance
/=	division

Variable : définition (rappel)

- **Variable** : espace de stockage pour une valeur
  - Ex : compteur = 1
    - **Opérateur d'affectation**
  - Une variable peut être **utilisée** plusieurs fois
  - La valeur stockée peut **varier** au cours du programme
    - Ex : compteur = compteur + 1
  - **Pour l'utilisateur** : une information sur laquelle il y a une étiquette → nom de la variable
  - **Pour l'ordinateur** : une information stockée dans la mémoire vive (RAM) → adresse mémoire

Variable : le nom (rappel)

- **Nom de variable** : identificateur **unique**
  - Peut contenir des lettres, chiffres et underscore (tiret du bas)
  - Ne peut pas commencer par un chiffre
  - Différenciation entre majuscules et minuscules
  - **Conseils** :
    - un nom qui décrit plutôt le rôle que la fonction (ex : `somme` plutôt que `nombre`)
    - éviter les caractères accentués (même si ok dans python 3)
    - attention au l minuscule qui peut se confondre avec i majuscule
    - attention aux mots réservés de python 3 : `and del from None True as elif global nonlocal try assert else if not while break except import or with class False in pass yield continue finally is raise def for lambda return`
    - La norme PEP 8 décrit les conventions de nommage : https://www.python.org/dev/peps/pep-0008/

Variable : le type (rappel)

- **Type de variable** : information sur le contenu de la variable (sa nature)

  → indique à l'interpréteur python comment manipuler cette information

  - **int** : nombre entier (*integer*)
  - **float** : nombre décimal
  - **str** : chaîne de caractère (*string*)
  - **bool** : booléen (`True` ou `False`)
  - …
  - **Typage dynamique de Python** : il reconnaît certains types automatiquement (int, float) mais pour les strings il faut l'aider avec des guillemets.
  - Le **type** de la variable peut **changer**, il dépend de la valeur qui lui est attribuée :
    - il correspond toujours au type de sa dernière affectation
    - La fonction `type(variable)` retourne le type de la variable
    - La fonction `str(variable)` change le type de la variable en string

Variable : déclaration (rappel)

- **Initialisation = déclaration** :
  - Pas besoin de **déclarer** une variable pour dire qu'elle existe (contrairement à d'autres langages)
  - En python il suffit **d'attribuer une valeur** à une variable pour la **déclarer**
  - Ex :

```
x = 2.5 #création de la variable x initialisée à 2.5 (x est donc de type "float")
```

Le séparateur décimal est le point et non la virgule

Récapitulatif (rappel)

- **Variable** = une *étiquette* que l'on colle sur un objet pour lui donner un *nom*
- **Stocker** un objet dans une variable = permettre à python de le **manipuler** (il se souvient de lui)
- Le **type** d'un objet = décrit sa nature (à quoi il ressemble, comment l'utiliser, etc.)
  - **int** = nombres entiers
  - **float** = nombres à virgule
  - **str** = chaînes de caractères (*texte*)
  - **bool** = vrai ou faux

Les **fonctions** sont des *actions* que l'on peut invoquer pour les faire exécuter par Python.

Affectation multiple

```
a, b = 6, 8
```

équivaut à

```
a = 6
b = 8
```

Permuter les valeurs des variables :

```
a = 6
b = 8
a, b = b, a
```

Portée des variables

- **Variable locale** : ne peut être utilisée que dans la fonction où elle est définie.
  - On ne peut pas l'appeler en dehors de cette fonction.
- **Variable globale** : peut être utilisée dans tout le programme.
  - Elle est accessible en lecture dans les fonctions (la fonction peut l'utiliser mais pas la modifier - sauf avec le mot-clé `global`).
- La portée d'une variable dépend de l'endroit où elle est définie.

Fonctions (précisions)

- **Afficher** le résultat :

```
def ma_fonction(a, b):
    c = a + b
    print(c)

print(ma_fonction(6, 8) + 2) #erreur
```

- **Retourner** le résultat :

```
def ma_fonction(a, b):
    c = a + b
    return(c)

print(ma_fonction(6, 8) + 2) #ok
```

Exercices (feuille n°5) : correction

1. Stocker le résultat du calcul suivant dans une variable et “imprimer” son résultat dans le terminal : 10/4
2. Stocker 59 dans une variable et 23 dans une autre.
    2.1. Afficher la somme de ces deux valeurs dans le terminal.
    2.2. Afficher le type de la somme dans le terminal.
3. Faire la même chose avec “py” et “thon”.
4. Faire une boucle while qui permet d’incrémenter une variable age d’un “an” à chaque passage de boucle, en partant de 7 jusqu’à 77. Chaque valeur devra s’afficher dans le terminal. À la fin de la boucle, afficher “Fin.”
5. Afficher “J’ai 20 ans” dans le terminal depuis deux variables : une pour le texte et une pour l’âge[1].
6. Faire une boucle while qui permet de soustraire 1 “jour” par passage de boucle à une variable jour à chaque passage de boucle, en partant de 30 jusqu’à 1. Chaque valeur devra s’afficher dans le terminal selon la forme suivante : “Il reste ?? jour(s) avant les vacances”. À la fin de la boucle, afficher “C’est les vacances !.”

1. Si un message d’erreur s’affiche, lisez-le bien pour essayer de comprendre d’où vient l’erreur.

Exercices (feuille n°6)

1. Créer 3 variables `x`, `y` et `z` qui valent respectivement 16, 3 et 289. Créer une boucle while qui ajoute `y` à `x` tant que `x` est inférieur à `z`.

   Combien y a-t-il eu de tour de boucle ? Afficher le compte dans le terminal à l'aide d'un compteur.

2. Avec une affectation multiple, définir les variables `a`, `b`, et `c` valant respectivement 4, 3 et 120. Créer une boucle while qui donne à `a` sa valeur multipliée par `b`, et qui donne à `c` sa valeur moins `b`, tant que `a` vaut moins que 300 et que `c` est différent de 0.

   Afficher les valeurs de `a`, `b`, et `c` à chaque tour de boucle.

3. Afficher la table de 5 dans le terminal à l'aide d'une boucle while et d'un compteur (afficher les 10 calculs et leurs résultats).

4. Un écrivain envoie un document de 64 pages complètes à son éditeur. Leur contrat stipule que chaque page doit contenir 22 lignes et que chaque ligne coûte 43 cents. Calculer la valeur de ce document, en vous aidant d'une boucle while et d'un compteur.