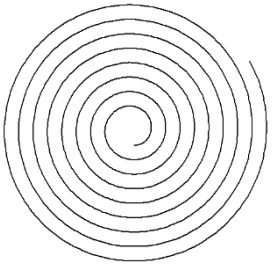
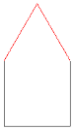
1NSI	Langages et programmation Variables et boucles	TP – **Variables et boucles**

Objectifs :

- ⇨ Découvrir la notion de variable et les mettre en œuvre
- ⇨ Connaitre les deux types de boucles utilisées en python



I – Des variables pour faire varier



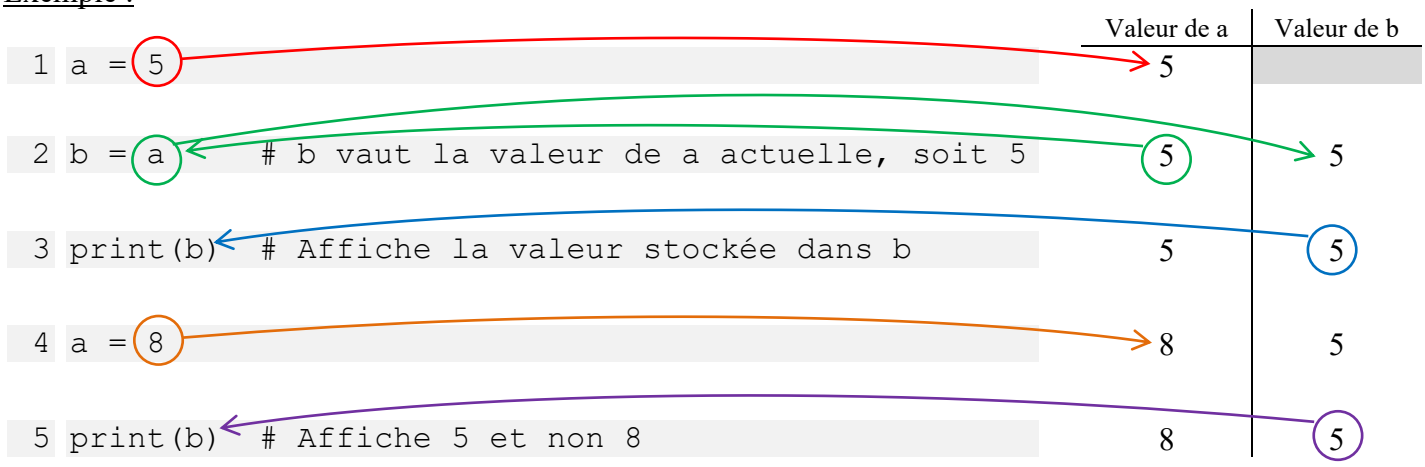
Reprenons le programme « tortue_v2.py » qui trace une belle maison carrée de 100 pixels. On souhaite maintenant faire la même maison en plus petit.

Question 1 :
On veut tracer une maison de 50 pixels au lieu de 100.
Combien de lignes du programme précédent doit-on modifier ? Faire les modifications et enregistrer le programme sous le nom « tortue_v4.py ». Vérifier qu'on obtient bien une maison de 50 pixels de large.

On souhaite pouvoir tracer des maisons ayant les largeurs suivantes : 200, 150, 100, 80, 60, 50, 40, 30 pixels.
On comprend bien qu'il est pénible de devoir écrire 8 fois le programme en changeant juste les valeurs des instructions `forward()`. Il serait plus intéressant de pouvoir faire varier cette grandeur en fonction de ce que l'on souhaite tracer dans notre programme.
Pour ce faire les langages de programmation utilisent la notion de *variable*. Une variable peut contenir une valeur qui peut changer au cours de l'exécution du programme.
En python une variable fait référence à un objet qui a un type bien défini parmi les différents types possibles (numérique, texte, image, liste, fichier …). Elle possède un nom que le programmeur choisi mais qui suit les contraintes vues dans le premier cours (lettres/chiffres et _, sensible à la casse, ne commence pas par un chiffre et distinct des mots-clés du langage).

Pour créer une variable ou simplement changer sa valeur, on utilise l'opérateur d'affectation '='. Il modifie la valeur de la variable au moment où il est évalué. **Il ne correspond donc pas au « = » des mathématiques**.

Exemple :



Question 2 :

1) Modifier le programme précédent pour qu'il trace une maison de `taille` pixels où `taille` est une variable qu'on définira d'abord à 200, puis 150, …
2) On veut maintenant que le programme demande directement à l'utilisateur la taille de la maison à tracer. Pour ce faire on peut utiliser la fonction `input()` (voir encadré ci-après).

La fonction `input()` :

Cette fonction met en pause le programme le temps que l'utilisateur rentre quelque chose au clavier et valide par « entrée ». La fonction renvoie alors une variable de type chaîne de caractère contenant ce que l'utilisateur a tapé et le programme reprend son exécution.
On peut donner en argument à la fonction une chaîne de caractère qu'elle affichera (comme avec `print()`) juste avant le prompt pour préciser à l'utilisateur ce que l'on attend de lui.
Exemple : `prenom = input('Quel est votre nom ? ')`

NB : La variable retournée est toujours de type chaîne. Si on veut récupérer un entier il faut utiliser la fonction `int()` qui converti la chaîne donnée en argument en nombre entier.
Ex : `n = int(input('nombre de répétitions ? '))`

Remarque : On peut modifier le contenu d'une variable en utilisant son contenu précédent dans le calcul pour déterminer son nouveau contenu. Ainsi `x = x + 1` permet d'augmenter la valeur de la variable `x` de 1.

II – Les boucles

On cherche maintenant à représenter 10 maisons à l'écran. Comme on l'a vu précédemment, il n'est pas pratique de copier-coller le code 10 fois, on va chercher plutôt une méthode qui nous permette d'écrire le code une seule fois et de demander à l'ordinateur de répéter l'opération.

Les langages de programmation proposent plusieurs façons de réaliser des répétitions (on parle plutôt de « boucles »).
Python propose deux mécanismes que nous allons voir ici.

1) La boucle while

Il s'agit d'une boucle dont le nombre d'itérations (de répétitions) n'est pas connu à l'avance. Sa structure est la suivante :

```
while condition :
      Bloc
      d'instruction
      à
      répéter
suite du programme
```

On remarque tout de suite que le bloc d'instruction à répéter est indenté par rapport au début de ligne. Toutes les lignes de ce bloc d'instruction doivent commencer à la même colonne et cette colonne doit se situer après (décalé vers la droite) celle de l'instruction `while`. On peut obtenir cette indentation en utilisant la touche tabulation (Tab, juste au-dessus de la touche verrouillage majuscule) ou avec des espaces.
Cette présentation (l'indentation) doit impérativement être respectée en python[1] car c'est la seule façon pour l'interpréteur de délimiter les blocs de programme.

La boucle va s'exécuter tant que la condition est vérifiée. Nous verrons en détail ce que cela signifie, mais pour l'instant on se contentera de tests d'inégalité (ex : `a > 20`) ou d'égalité avec l'opérateur « `==` » (ex : `b == 15` signifie que l'on teste si la variable b vaut 15).

[1] D'autres langages utilisent d'autres mécanismes comme les accolades {} en c ou java ce qui est plus explicite mais alourdi la présentation du code.

Question 3 :

1) Sans l'exécuter, déterminer ce que fait le programme ci-contre :

2) Combien de fois le code de la boucle est-il exécuté ? Que vaut la variable `pas` à la fin du programme ?

3) Copier-coller le programme dans Thonny (ou ouvrir le programme « question3.py » et vérifier vos réponses en l'exécutant pas à pas.

```
from turtle import *
reset()
pas = 5
while pas < 200 :
    forward(pas)
    left(30)
    pas = pas * 2
print(pas)
```

Question 4 :

Ecrire un programme « repetition.py » qui affiche les nombres de 0 à 10 en utilisant une boucle `while`.

Remarque importante :

Ce type de boucle est très dangereux, car si les instructions à l'intérieur du bloc ne changent pas les termes de la condition de poursuite, on peut se retrouver avec le cauchemar du programmeur : une boucle infinie !

2) La boucle for

Ce type de boucle permet de répéter un bloc d'instruction un nombre de fois fixé à l'avance.
Pour préciser le nombre de répétitions, on va utiliser la fonction built-in `range()`.

`range(n)` génère une liste de tous les entiers de 0 à n-1 (il y a donc **n** valeurs)

Exemple : `range(5)` génère la liste `[0, 1, 2, 3, 4]`.

La structure d'une boucle for est la suivante :

```
for variable in range(n) :
      Bloc
      d'instruction
      à
      répéter
suite du programme
```

On remarque que le bloc à répéter est, là aussi, indenté par rapport au reste du code.
`variable` est le nom de la variable qui va recevoir les valeurs successives renvoyées par `range`. On essaye de choisir un nom de variable assez explicite, mais les programmeurs utilisent assez souvent le nom de variable `i` (comme **i**ndex) car il est assez court.

Question 5 :

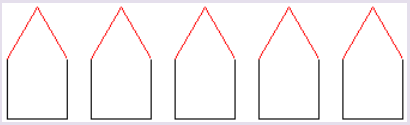
1) Sans l'exécuter, déterminer ce que fait le programme suivant :

```
for nombre in range(7):
    print("Nombre examiné à cette itération :", nombre)
print("La boucle est terminée")
```

2) Combien de fois le code de la boucle est-il exécuté ? Que vaut la variable `nombre` à la fin du programme ?

3) Copier-coller le programme dans Thonny (ou ouvrir le programme « question5.py » et vérifier vos réponses en l'exécutant pas à pas. On observera notamment avec la fenêtre variable (menu View/Variables) les changements de valeur de la variable `nombre`.

Question 6 :



1) Ouvrir le programme « tortue_v4.py » et le sauvegarder sous le nom « tortue_v5.py ».

On souhaite modifier le programme pour qu'il dessine une rangée de 5 maisons de 50 pixels de large espacées chacune de 20 pixels. On pourra commencer le tracé en (-150, 100) pour centrer le tracé et utiliser la fonction `speed()` pour augmenter la vitesse de tracé (voir l'aide de la commande pour davantage de précision).

2) Quel type de boucle (`while` ou `for`) vaut-il mieux utiliser ? Pourquoi ?
3) Réaliser le programme avec une boucle `for` (« tortue_v5_for.py »), puis avec une boucle `while` (« tortue_v5_while.py »).
4) Faire en sorte que le programme représente maintenant 2 rangées de 5 maisons, la première avec des maisons de 50 pixels espacées de 20 et la seconde avec des maisons de 40 pixels espacées de 30 pixels. Ne pas hésiter à rajouter l'instruction `speed(0)` en début de programme pour accélérer le tracé.
5) Si vous aviez copié-collé le code de la première boucle à la question précédente, essayez de modifier votre code pour réaliser le tracé demandé en utilisant deux boucles imbriquées (le code pour dessiner la maison ne doit donc apparaître qu'une seule fois dans votre programme).

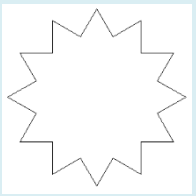
PROLONGEMENT :

```
Quelle table voulez-vous afficher ? 7
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
```

Q1 : Ecrire un programme qui demande un nombre entier puis affiche la table de multiplication correspondant à ce nombre.

Aide :

La fonction `range()` peut aussi s'écrire avec deux arguments : `range(a,b)` renvoie tous les entiers de a à b-1. Exemple : `range(2, 8)` donne `[2, 3, 4, 5, 6, 7]`



Q2 : Ecrire un programme qui trace une étoile à 12 branches comme la figure ci-contre. La taille des pointes doit être réglable.

Q3 : Modifier le programme précédent pour qu'il permette de tracer une étoile avec un nombre de branches et une taille que l'on demandera à l'utilisateur (voir figures ci-dessous).