4TPM205U: Algorithmique des tableaux: feuille 7

Récursivité

Travaux dirigés sur machine

Exercice 1.

1. Écrivez une version récursive de la fonction factorielle $n! = 1 \times 2 \times \cdots \times n$.
2. Dans cet exercice vous allez utiliser pythontutor pour visualiser le déroulement de votre fonction. Ouvrez l'url `https://services.emi.u-bordeaux.fr/pythontutor/visualize-initinfo.html`. Copiez-collez votre fonction factorielle dans la fenêtre d'exécution. Ajoutez une ligne d'appel de la fonction pour `n` valant 4. Cliquez sur Lancer exécution, puis plusieurs fois sur le bouton Avant pour exécuter pas à pas la fonction. Observez et interprétez l'évolution de la **pile d'exécution**.

Module Turtle et Courbes Fractales

Dans les exercices 2, 3 et l'exercice complémentaire 1, on utilise le module `turtle` de python. Ce module permet de dessiner des segments de droites, en déplaçant un stylo (appelé "tortue" en raison de sa rapidité de déplacement) sur une fenêtre. Pour pouvoir utiliser ce module, vous devez écrire :

```
from turtle import *
```

ou (ce qui est plus propre mais demande d'ajouter "`turtle.`" devant chaque nom de fonction)

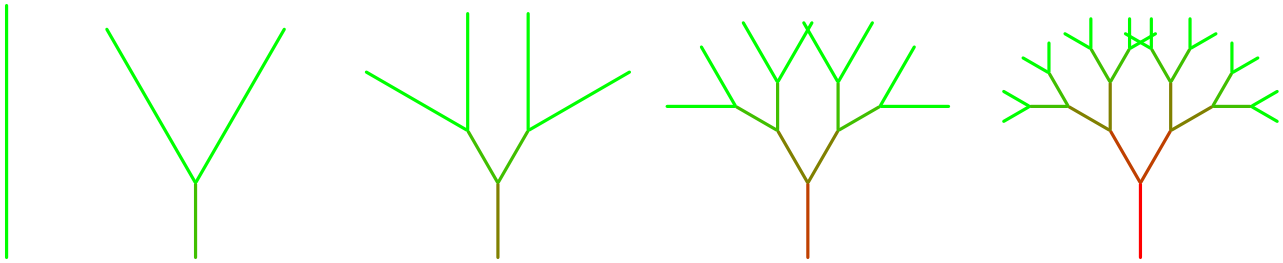
```
import turtle
```

Plusieurs fonctions deviennent alors disponibles, dont les suivantes :

- `Screen()`, ou `turtle.Screen()` : affiche la fenêtre de dessin.
- `clear()` : efface les dessins effectués.
- `reset()` : idem, repositionne la tortue et remet les variables à leur valeur par défaut.
- `up()` : lève le stylo, et `down()` : abaisse le stylo.
- `forward(d)`, `backward(d)` : avance ou recule la tortue de `d` pixels à partir de sa position courante. Si le stylo est baissé, un segment de droite sera dessiné.
- `goto(x,y)` : déplace la tortue sur un segment de droite depuis sa position courante jusqu'à la position (`x`, `y`). Si le stylo est baissé, ce segment de droite sera dessiné.
- `left(a)`, `right(a)` : fait tourner la tortue sur la gauche ou la droite de l'angle `a` (en degrés).
- `setheading(a)` : positionne la tortue à l'angle `a`. Si l'angle est 0, la tortue regarde vers la droite.
- `speed('fastest')` : augmente la vitesse.

Observation : Notez que par défaut le stylo est baissé et la tortue est orientée vers la droite (parallèle à l'axe des abscisses).

Exercice 2. On veut dessiner des arbres obtenus récursivement, de la façon suivante :



L'arbre de rang 0 (à gauche) est réduit à un segment vertical. Pour $n \geqslant 1$, l'arbre de rang n est composé d'un segment vertical et de deux arbres de rang $n-1$ commençant chacun en haut du tronc, et inclinés d'un angle donné (par exemple $30°$) symétriquement par rapport au tronc.

1. Écrivez une fonction `arbre(n, longueur, angle, ratio)` qui dessine l'arbre de rang n. La longueur (de la base du tronc aux extrémités des branches) est donnée par le second argument. Le troisième argument est l'angle des sous-arbres par rapport au tronc. Le dernier argument est le ratio entre la longueur du tronc et la longueur totale.
2. Combien d'appels récursifs sont-ils effectués par cette fonction ?

Observation : Il est conseillé de faire un `reset()` avant le premier appel à la fonction `arbre` afin de remettre la tortue à sa position initiale.

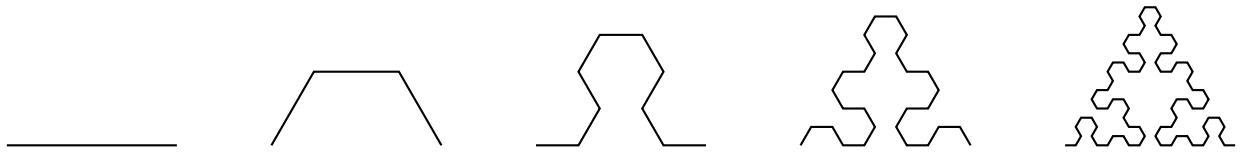
Exercice 3. Les flocons de Koch sont obtenus en répétant récursivement un motif. Les premiers flocons sont les suivants :

Le flocon de rang 0 (figure de gauche) est simplement un segment. Pour $n \geqslant 1$, le flocon de rang n est obtenu en juxtaposant 4 flocons de rang $n-1$: le premier horizontal, le second incliné de $60°$, le troisième incliné de $-60°$ et le quatrième également horizontal.

1. Écrivez une fonction `koch(n, longueur)` qui dessine le flocon de rang n, dont la longueur de l'extrémité gauche à l'extrémité droite est donnée par le second paramètre.
2. Combien d'appels récursifs sont-ils effectués par cette fonction ?

Observation : N'oubliez pas de faire un `reset()` avant chaque appel de la fonction `koch`.

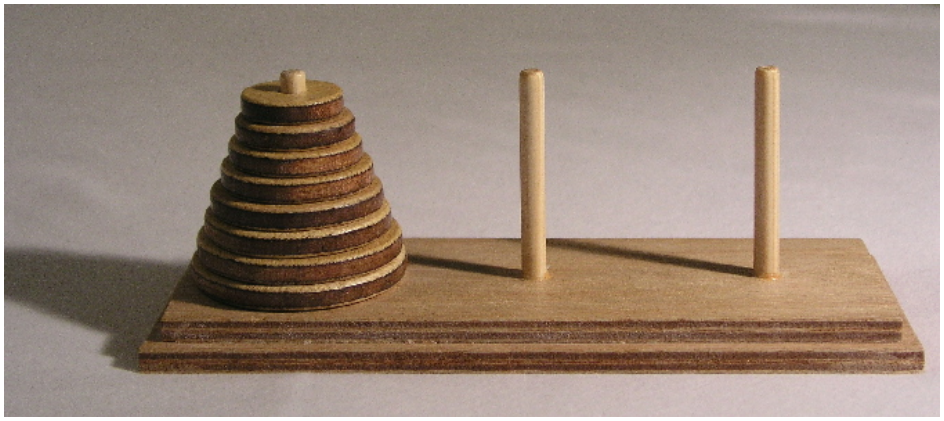
Exercice complémentaire 1. Même exercice pour le triangle de Sierpinski dont les premiers rangs sont représentés sur la figure suivante. Testez la fonction pour l'itération 8.



Exercice complémentaire 2. Écrire une version récursive de la fonction fibonacci définie dans la feuille TD#1. Vous pouvez télécharger le fichier trace.py permettant de tracer l'exécution de quelques fonctions récursives dont on vous a demandé l'écriture. Utilisez les fonctions fournies dans ce fichier pour visualiser les appels récursifs lors de l'exécution de ces fonctions (attention à ne pas choisir de trop grandes valeurs pour les arguments). Quel est le rôle de la variable `nbAppels` ?

Exercice complémentaire 3. Le jeu des tours de Hanoï se joue avec 3 piquets, appelés `d` (comme départ), `a` (comme arrivée), et `i` (comme intermédiaire), et n disques de tailles différentes que l'on met sur ces piquets.

On part de la configuration où les n disques sont disposés sur le piquet `d` de départ, comme sur la photo suivante (avec $n = 8$).



Crédit photo : https://fr.wikipedia.org/wiki/Tours_de_Hano%C3%AF#/media/File:Tower_of_Hanoi.jpeg

L'objectif est de déplacer les disques du piquet `d` vers le piquet `a`, en déplaçant un seul disque à chaque étape. La contrainte à respecter est qu'on n'a pas le droit de placer un disque sur un disque plus petit que lui.

Via l'applet `http://jeux.prise2tete.fr/tour-de-hanoi/tour-de-hanoi.php` vous pouvez vous exercer au jeu des tours de Hanoï.

1. En supposant que vous savez déplacer les $(n-1)$ disques les plus petits d'un piquet vers un autre en utilisant une suite de mouvements, comment faire pour déplacer les n disques du piquet `d` vers le piquet `a` ?
2. Écrire une fonction récursive permettant de résoudre le jeu, en affichant (par l'instruction `print`) la suite des déplacements.
3. Combien de déplacements de disques effectue cet algorithme ?