



COMPLEXITÉ, PILES ET FILES

TP 11

Algorithmie et types de variables

- v1.0

Lycée La Martinière Monplaisir, 41 Rue Antoine Lumière, 69372 Lyon

1 Courbe et flocon de Koch

1.1 Module MyTurtle

Pour faire les dessins, on va utiliser un module dédié appelé **MyTurtle** : il permet de contrôler un stylo en précisant de combien on veut avancer et comment tourner. On vous donne ici une liste limitée des commandes (vous n'avez normalement pas besoin de plus), mais vous avez la liste complète en fin d'exercice.





<code>t.reset()</code>	Effacer le dessin et revenir à la position initiale
<code>t.forward(d)</code>	Avancer d'une distance d
<code>t.left(a)</code>	Tourner à gauche de a degrés
<code>t.right(a)</code>	Tourner à droite de a degrés
<code>t.show()</code>	Montrer le dessin

Le fichier correspondant `MyTurtle.py` ainsi qu'un fichier d'utilisation `Koch_etudiant.py` est dans votre espace de classe partagé. N'oubliez pas le raccourci `ctrl+shif+E` afin d'appeler le module contenu dans le même dossier.

1.2 Courbe de Koch

On crée la figure appelée courbe de Koch à partir d'un segment de droite, en modifiant récursivement chaque segment de droite de la façon suivante :

1. on divise le segment de droite en trois segments de longueurs égales
2. on construit un triangle équilatéral ayant pour base le segment médian de la première étape
3. on supprime le segment de droite qui était la base du triangle de la deuxième étape.

Étape	Dessin
1	
2	
3	
4	

La courbe de Koch est la limite des courbes obtenues, lorsqu'on répète indéfiniment les étapes mentionnées ci-devant. Cette courbe possède quelques propriétés intéressantes :

- elle possède une longueur infinie ;
- la surface délimitée par la courbe est cependant finie ;
- c'est une courbe continue, mais non dérivable en chacun de ses points.

Question 1 Écrire une fonction récursive `courbe_koch(d, n)` qui prend en argument deux entiers, d représentant la longueur initiale du segment, et qui trace la courbe de Koch obtenue après n itérations des étapes de modification.

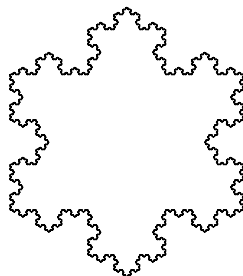
Question 2 Trouver la relation de récurrence correspondant à sa complexité temporelle.

Question 3 En déduire sa complexité.

1.3 Flocon de Koch

Le flocon de Koch s'obtient de la même façon que la fractale précédente, en partant d'un triangle équilatéral au lieu d'un segment de droite, et en effectuant les modifications en orientant les triangles vers l'extérieur.

Après quelques étapes, on obtient alors une figure du type ci-dessous :



Question 4 Écrire une fonction `flocon_koch(d, n)` qui dessine le flocon de Koch correspondant à la courbe de Koch de rang n .

1.4 Liste complète des fonctions du module MyTurtle

Module MyTurtle inspiré de turtle graphics

```
import MyTurtle as t
```

<code>t.reset()</code>	Effacer le dessin et revenir à la position initiale
<code>t.goto(x,y)</code>	Se déplacer au point de coordonnées (x,y) sans changer l'orientation
<code>t.forward(d)</code>	Avancer d'une distance d
<code>t.backward(d)</code>	Reculer d'une distance d
<code>t.penup()</code>	Lever le crayon (pour se déplacer sans dessiner)
<code>t.pendown()</code>	Baisser le crayon (pour dessiner)
<code>t.left(a)</code>	Tourner à gauche de a degrés
<code>t.right(a)</code>	Tourner à droite de a degrés
<code>t.show()</code>	Montrer le dessin

2 Somme dichotomique : une bonne idée ?

On souhaite calculer la somme de tous les éléments d'une liste de façon récursive et dichotomique : en effet, la somme de tous les éléments d'une liste est égale à la somme de la moitié droite plus la somme de la moitié gauche.

Question 1 Rappel de la complexité en temps et en espace d'une somme implémenter classiquement.

Question 2 Écrire une fonction récursive `somme(L)` qui renvoie la somme des éléments d'une liste en suivant l'algorithme décrit plus haut (on rappelle que la somme d'une liste vide est nulle).

Question 3 Montrer que cette implémentation n'est pas intéressante en étudiant sa complexité en temps et en espace.

3 Somme des éléments d'une pile

Un fichier de librairie `piles_files.py` ainsi qu'un fichier d'utilisation `somme_etudiant.py` sont dans votre espace de classe partagé. N'oubliez pas le raccourci `ctrl`+`↑`+`E` sous Pyzo afin d'appeler le module contenu dans le même dossier. Cette bibliothèque permet de définir des objets piles et des objets files :

- pour les piles :
 - ◊ `p = pile()` permet de créer une pile vide `p`,
 - ◊ `p.empiler(x)` permet d'empiler `x` dans `p`,
 - ◊ `x = p.depiler()` permet de retirer le sommet de `p` et de le mettre dans `x`,
 - ◊ `p.estVide()` permet de vérifier si une pile `p` est vide (renvoie un booléen);
- pour les files :
 - ◊ `f = file()` permet de créer une file vide `f`,
 - ◊ `f.enfiler(x)` permet de mettre en file `x` dans `f`,
 - ◊ `x = f.defiler()` permet de retirer l'élément en tête de la file `f` et de le mettre dans `x`,
 - ◊ `f.estVide()` permet de vérifier si une file `f` est vide (renvoie un booléen).

Question 1 Écrire une fonction `somme0(p)` qui prend en entrée une pile et renvoie la somme de ses éléments.

Question 2 Écrire une fonction `somme1(p)` qui prend en entrée une pile et renvoie la somme de ses éléments, mais en conservant la pile `p` intacte (elle devra être reconstruite à la fin). Si on a besoin d'une structure de donnée intermédiaire, on utilisera une pile.

4 Valeur maximale dans une file

Un fichier d'utilisation `maxFile_etudiant.py` est dans votre espace de classe partagé.

Question 1 Écrire une fonction `maxFile0(f)` qui prend en entrée une file et renvoie l'élément maximum de la file (supposée non vide).

Question 2 Écrire une fonction `maxFile1(f)` qui prend en entrée une file et renvoie l'élément maximum de la file en gardant la file intacte après la recherche et sans utiliser de structure de données intermédiaires.

L'idée est de « marquer » la position de référence puis de faire défiler les éléments en les rempilant jusqu'à retomber sur le marquage. La fonction `affichageFile(f)` fait quelque chose de similaire et vous pouvez donc vous en inspirer.

5 Notation Polonaise Inverse

La notation polonaise inverse (NPI) (en anglais RPN pour *Reverse Polish Notation*), également connue sous le nom de notation post-fixée, permet d'écrire de façon non ambiguë les formules arithmétiques sans utiliser de parenthèses. Elle a par exemple été utilisée par les premières générations de calculatrices scientifiques Hewlett-Packard, dont la HP-35.



FIGURE 1 – La HP-35 (1972) : première calculatrice scientifique de poche

Par exemple, l'expression « $3 \times (4 + 7)$ » peut s'écrire en NPI sous la forme « 4 ent 7 + 3 \times », ou encore sous la forme « 3 ent 4 ent 7 + \times ».

On va ici mettre en place une évaluation d'expression NPI. On va se contenter ici que des opérateurs binaires de base : la multiplication ('*'), l'addition ('+'), la soustraction ('-') et la division ('/').

Ainsi les entrées seront stockées par une file contenant les caractères des opérations et les opérandes. Le résultat des opérations et le stockage des résultats se feront avec une pile. L'idée consiste à :

- créer une pile vide `p` ;

- désenfiler l'expression NPI de la gauche vers la droite ;
- empiler dans `p` chaque nombre rencontré ;
- lorsqu'un élément rencontré est un opérateur,
 - ◊ on dépile depuis `p` les 2 opérandes,
 - ◊ on effectue le calcul,
 - ◊ on empile le résultat dans `p`.

Pour évaluer le calcul, on va utiliser la fonction `eval`.

Question 1 Évaluer dans la console ou dans un script les expressions suivantes `eval("3+5")`, `eval('3'+''+''+5')` et `a = 3; b = 5; eval("a"+"+"+"b")`.



Attention

La commande `eval` est très dangereuse : il rend un code sensible aux attaques par injection (exécution d'un code extérieur non contrôlé) quand la chaîne de caractères vient de l'extérieur.

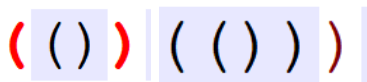
Question 2 Écrire une fonction `evalNPI(expr)` qui prend en entrée une dèque et montre dans la console le résultat des opérations une par une.

Question 3 Vérifier que `evalNPI(deque([8, 4, '/']))` affiche bien 2.0 dans la console.

Question 4 Vérifier que `evalNPI(deque([32, 7, "+", 17, 4, "+", 5, "*", 2, "*", "+"]))` affiche bien 39, 21, 105, 210 et 249 dans la console.

6 Expression bien parenthésée

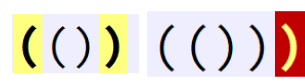
Dans un éditeur l'ajout d'une parenthèse fermante surnuméraire est signalé (mise en surbrillance). La construction d'un vérificateur de parenthèses repose sur l'utilisation de piles.



(a) Éditeur généraliste Notepad++



(b) Éditeur Python Pyzo



(c) Éditeur L^AT_EX TeXstudio

FIGURE 1 – Expressions bien et mal parenthésées dans différents éditeurs

Nous allons considérer 3 types de ponctuations symétriques :

- les parenthèses (et) ;
- les crochets [et] ;
- les accolades { et } ;

On souhaite savoir reconnaître 4 cas différents :

- fermeture sans ouverture préalable ;
- fermeture par le mauvais type de parenthèse ;
- une ouverture sans fermeture ;
- l'expression est bien parenthésée.

Nous allons commencer par nous occuper dans un premier temps uniquement d'un seul type de ponctuations : les parenthèses.

Question 1 Écrire une fonction `verifPar0(texte)` qui prend en entrée une chaîne de caractères et renvoie :

- le nombre d'ouvertures non-fermées ;
- -1 dès qu'on rencontre une fermeture sans ouverture préalable.

Une pile sera initialisée au début de la fonction : on ajoutera l'ouverture dans la pile qu'on dépilera lorsqu'il y a fermeture de la parenthèse avec le même type.

Question 2 Tester la fonction avec "`((()))`", "`((()())`" et "`((()()))`".

Nous ajoutons ici les autres types de ponctuations : il faut donc vérifier si la fermeture correspond bien à l'ouverture dans ce cas-là.

Question 3 Écrire une fonction `verifPar1(texte)` qui prend en entrée une file remplie de caractère et renvoie :

- le nombre d'ouvertures non-fermées ;
- -1 dès qu'on rencontre une fermeture sans ouverture préalable ;
- c dès qu'on rencontre une erreur de type au caractère c.

Question 4 Tester la fonction avec "`{[(())]}`", "`{[(())]`" et "`{[(())]}`".