Projet Python : L-System
Compte Rendu

Brisset Joachim Kemgne Jules-Antoine Hopsore Theo

11 janvier 2021

Table des matières

I Présentation

Ce projet est l'évaluation finale du module python. Ce projet aura pour but de réalisé un programme capable, de a partir d'un fichier d'entrée normé représentant un L-System, générer un code permettant de le dessiner.

II Caractéristique de Projet

Langage de programmation Le programme finale devra être rédiger en langage Python étant donné que ce projet est l'évaluation finale de ce module.

Entrée le fichier d'entrée devra être sous la forme suivante :

```
axiome=   <chaine de caractère>
regles=   <symbole> = <chaine de caractère>
                    ⋮
          <symbole> = <chaine de caractère>
taille=   <nombre>
angle=    <nombre>
niveau=   <nombre>
```

L'ordre de règles du fichier n'est pas considéré important. Cependant la présence et l'unicité des règles excepté "regles" est primordial. Il en vas de même si dans la règle "regles", un même symbole apparait plusieurs fois.

Sortie Le fichier de sortie doit être en langage Python pour pouvoir être exécuter immédiatement après.

III Organisation

Git Pour avoir un travail collaboratif efficace utilisons git, un logiciel qui permet de facilité le travail collaboratif sur les projets de programmation. Il permet a chacun de modifier le code sans trop se soucier des problèmes de modification simultanée, de garder une trace de toute nos modification du code et ainsi de revenir à un code antérieur si nécessaire, de travailler à part sur de nouvelle fonctionnalité en toute simplicité et revenir travailler sur le projet principal sans soucie.

Le projet sera hébergé sur GitHub en version public.

Afin de facilité la vie a certain qui n'ont pas le courage d'apprendre le bash de git, nous utilisons gitKraken, une application permettant une utilisation intuitive de git a l'aide d'une interface graphique.

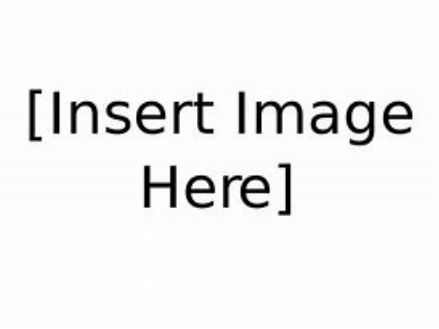

Figure 1 – screenshot de l'application GitKraken

Processus de développement Pour s'assurer que le programme fonctionnera convenablement nous nous sommes concentré tout d'abord sur le fonctionnement du programme sous sa forme la plus basique. C'est-à-dire que nous ne nous attardions pas sur les vérifications, interactivité du programme et tout autre ajout. A ce stade le programme ne peut que lire un fichier spécifique considéré valide et créer le code dans un autre fichier donné. Puis nous étendrons le programme en ajouter les vérification, l'interactivité et toutes autre fonctionnalités.

Planning de travail Pour être efficace dans le réalisation du projet nous avons commencé par réfléchir ensemble à la conception du programme pour ne pas trop revenir sur nos morceau de code précédent. Puis chaque semaine nous faisions un point sur l'avancement du projet aidions certain à finir le travail de la semaine précédente et enfin établissions le travail de chaque membre pour la semaine

IV Conception

Programme basique Voici ci-dessous le processus du programme basique :

Chacune de ces étapes fera office d'un fonction a part entière

Modèle du L-System Le l-System est représenter par un dictionnaire avec les index suivant : axiome, regles, taille, angle, niveau. Généralement en programmation nous ecrivons en anglais, cependant pour facilité le programme et la compréhension nous préférerons utiliser les mots français qui seront les même que dans le fichier d'entrée.

```
lsystem = {"axiome":None, "regles":{}, "angle":None, "taille":None, "niveau":None}  # default L-System
```

FIGURE 2 – modélisation en code Python du L-System

Structure Le projet n'étant pas très complexe nous avons décidé de rassembler l'intégralité programme dans un seul fichier, nommé *l-system.py*, et de le décomposé en plusieurs fonction effectuant chacune une tâche particulière.

V Développement

V.1 Programme basique

Dans sa version basique, le programme n'est pas très compliqué, on n'y distingue que trois fonctionnalités importantes :

En premier lieu, le chargement du L-System par la lecture d'un fichier :

Puis ensuite la croissance du L-System jusqu'au niveau souhaité :

Et enfin la transcription du L-System en un code python capable de le dessiner. Cette fonctionnalité se fait au travers de l'appel de la function *LSystemToPythonCode()* qui prend en paramètre l'axiome a dessiner (on aurait pu prendre le L-System en entier) et le nom du fichier de sortie.

Les instructions associées a leur symbole sont stocké dans un dictionnaire :

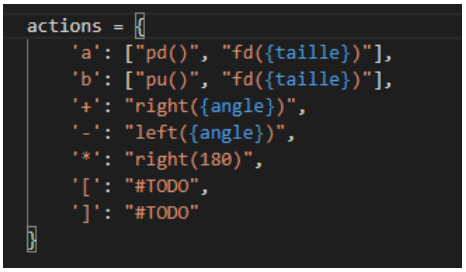

FIGURE 3 – pas de titre

La notation *{taille}* et *{angle}* sont appelé placeholder, ce sont des variable qui seront remplacer par autre chose plus tard et notamment par leur valeur dans ce programme.

Par la suite la fonction na plus qu'a écrire les ligne obligatoire tel que *from turtle import ** et pour chaque symbol qu'elle rencontre dans l'axiome elle cherche les instruction associé et les écrit dans le fichier et finit son exécution

V.2 Vérification et Interactivité

V.3 Argument de commande

VI Acquis

with syntax ???

dict switcher Durant le projet nous avons voulu utilisé un block switch pour pourvoir déterminer quelle instruction sont associé a un symbole néanmoins cette syntax n'existe pas en python (bien qu'on aurait put utilise 'if – elif'). Ce problème nous a amener a entrevoir une nouvelle utilisation des dictionnaire comme switcher grâce a « switcher['case'] ». De plus utilisé un dictionnaire de cette manière permet d'avoir un switch dynamique (qui n'est pas fixe dans le programme) et ainsi de pouvoir le modifier pendant l'execution.

type of input On a appris a bien gérer le type d'entrée voulus a l'aide de la fonction input et du block try : except ValueError :