1. Présentation du projet :

Le but de ce projet est de créer un programme capable de transcrire un fichier normé, représentant un L-System, en un programme capable de le dessiner.

Ce projet servira d’évaluation finale du module Python.

1. Caractéristique de Projet

Langage de programmation : Python

Entrée : fichier norme sous la forme :

* Axiome : chaine de caratere
* Regle : liste de symbole associé à une chaine de caractère associé de remplacement
* Taille : nombre
* Angle nombre
* Niveau : nombre

L’ordre des règles, sera considéré comme sans importance. Néanmoins chacune des règle excepté ‘regle’ ne doit pas apparaître plusieurs fois. De plus si un symbole accepte plusieurs règle de remplacement

Sortie : Le programme de sortie doit être un Python et être directement exécutable

1. Organisation :

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.

1. Conception

Déroulement basique :

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

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 préférons utiliser les mot français qui seront les même dans le fichier d’entrée.

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.

1. Développement

A/ Programme basique

Dans sa version basique le programme n’a besoin que de 3 fonctionnalité : Une qui charge le L-System

Dans un premier une qui charge le L-System a partir d’un fichier :

Ajout fonction : <nom de la function>

Paramètres : <liste params>

Fonctionnement : <explication fonctionnement>

Retour : <type de retour + explication>

[<Annexes>]

Puis une qui generele le L-System au niveau voulue

Ajout fonction : <nom de la function>

Paramètres : <liste params>

Fonctionnement : <explication fonctionnement>

Retour : <type de retour + explication>

[<Annexes>]

Et enfin une qui convertie l’axiome vers du code python :

Ajout Fonction : LSystemToPythonCode

Paramètres : l’axiome du L-System (on aurait pue prendre le L-System en entier aussi), le nom du fichier dans lequel écrire le code python

Fonctionnement : Elle ouvre le fichier en ecriture et écris le instruction necessaire au bon fonctionnement du programme. Puis elle boucle sur l’axiome et cherche l’instruction associé au symbole dans un dictionnaire appeler actions qui contient soit une chaine contenant l’instruction soit une liste de ces chaine.

Retour : rien

Action switcher (placeholder)

Replace placeholder

Photo exemple action

B/ Programme étendu au vérification:

A cette étape du développement, nous essayons d’ajouter tout les vérification nécessaire au programme pour éviter les erreurs et gérer celle-ci lorsque qu’on ne peut les éviter.

Ainsi durant le chargement du L-System nous vérifions si la règles est valide et après le chargement nous vérifions leur la valeur

Ajout fonction : check

Modification fn

En cas problème important le programme s’arrête et affiche l’erreur pour éviter que le programme plante par la suite.

C/ Interaction du programme

Le but ici est de régler le problème par une interaction avec l’utilisateur

Ajout de fonction :

D/ lecture des paramettre en entrée

1. Acquis

A/ With block

???

B/ No swith block -> switcher via dict

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.

C/ Input with specifique type

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

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.

Dans un premier temps, nous nous somme concentrer 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. Le programme ne pouvait alors que lire un fichier spécifique, pas forcement valide, et le traduisait dans un nouveau fichier définis en code python.

A cette instant le programme était composé de seulement 3 fonctions :

* LSystemToPythonCode : qui transcrit le l’axiome en ligne de code permettant de la dessiner a l’interieur d’un fichier
* Generate\_L\_System\_by\_Level: qui permet de generer le l’axiome du L-System au niveau voulu
* getInput : permettantde charger le donné du fichier

Ensuite nous avons ajouter la vérification du fichier

Deroulement programme

1. Demander fichier input
2. Ouverture + verif ok
3. Verif fichier + erreur
4. Creation du L-System au niveau X
5. Fermer fichier
6. Demander fichier output
7. Ouverture + verif ok
8. Generation du programme
9. Fermer fichier
10. Exécution du fichier

Fonction nécessaires :

* Mémoriser position stylo
* Retourenr a la position du style
* Verif fichier input
* Lecture fichier
* Creation du L-ssytem niveau X
* Convertion L System -> Python

On utilise les même norme pour les caractères

On ne peut pas utiliser replace car remplace tout d’un coup et peut replacer un remplacement

Erreur fichier ?

Test unitaire