Projet python : pavage par les trapèzes

0 Consignes de rendu

Ce projet est à rendre pour le **7 décembre**.

Les séances du **19 novembre, 26 novembre, 3 décembre** sont prévues pour travailler le projet.

Il se fait seul(e) ou à deux, mais il sera validé par une **soutenance orale individuelle** de 10 minutes (entretien avec votre professeur(e)).

Il sera à rendre sur pronote dans la partie "devoirs".

Les soutenances auront lieu la semaine du **13 décembre.**

Si vous avez plus d'un fichier python, mettez-les dans un dossier, et archivez l'ensemble dans une archive zip avant de le mettre sur pronote (une soumission par binôme).

Le respect des consignes compte pour une part importante de la note du projet, mais tout ce qui n'est pas explicitement obligatoire est libre (style, . . .).

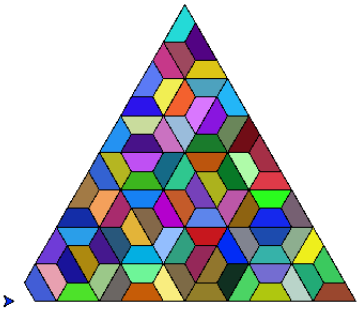
La lisibilité du code est également une part importante dans la notation : mettez des **commentaires**, *utilisez des boucles et des variables pour ne pas recopier des valeurs partout, mettez des* **commentaires**, *utilisez des fonctions si cela vous paraît pertinent, et surtout mettez des* **commentaires**. *Et n'oubliez pas de* **commenter** *votre code.*

1 Ce qu'il faut coder

Vous utiliserez la bibliothèque `turtle` afin de tracer une solution du problème du pavage du triangle par des petits trapèzes (voir fiche d'intro pour un exemple).

Le code doit être composé d'une ou plusieurs fonctions, bien documentées (la docstring !), et bien sûr bien commentées, et à la fin seulement un simple appel à la fonction « finale » avec l'argument n au choix (d'autres arguments sont possibles) doit permettre de tout dessiner (il n'est pas nécessaire de tracer le petit triangle noir).

Exemple : dans le code, j'appelle `dessine_solution(4)` et voici ce qui se dessine à l'écran :



Ici les petits trapèzes sont de couleur aléatoire afin de mieux les distinguer mais ce n'est pas obligatoire.

Conseils

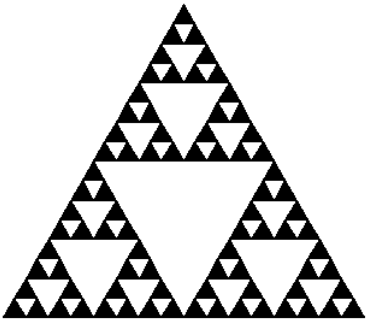
— Faites des schémas pour bien réfléchir à votre solution : comment va se déplacer la tortue ? Dans quel ordre va-t-on dessiner les trapèzes ?
— Spécifiez très très très soigneusement vos fonctions. Par exemple une fonction qui dessine un petit trapèze : où commence la tortue ? Où finit-elle ? Dans quelle orientation ? Où est le « trou » ? C'est encore plus important quand on utilise la récursivité !
— Pour le dessin de la solution, il peut s'avérer pertinent de faire revenir la tortue à son point de départ après avoir dessiné. Vous vous rendrez compte rapidement pourquoi. Remarque : il est possible, en `turtle`, de sauvegarder sa position et son orientation dans des variables puis de s'y replacer à la fin...
— Comme d'habitude, testez vos fonctions sur les cas simples d'abord ($n = 1$ puis $n = 2$, ...). Au delà de $n = 5$ ça devient très gros (et long à dessiner).
— Il peut être utile de rajouter un peu partout un paramètre (ex : a) qui est l'échelle, par exemple le nombre de pixels de côté d'un petit triangle de base. Ainsi on peut mettre un a plus petit si on dessine un cas avec n plus grand histoire que ça rentre dans l'écran.
— Il est plus pertinent de travailler en « relatif » (avancer, tourner) qu'en coordonnées absolues pour ce problème : déjà parce que les coordonnées sont compliquées à cause du pavage triangulaire, mais aussi parce que ça permet de mieux faire les appels récursifs.
— Commentez votre code. Encore et toujours.

2 Soutenances

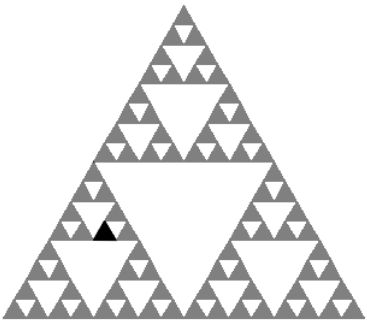
La soutenance individuelle se fera avec votre professeur(e), à son bureau, via des questions qui vous seront posées. Vous pourrez répondre en montrant votre code ou en l'exécutant (vous aurez à disposition l'ordinateur avec votre projet dessus), ou en écrivant au tableau pour illustrer votre réponse. Vous aurez probablement à expliquer comment la tortue se déplace pour les différents appels récursifs.

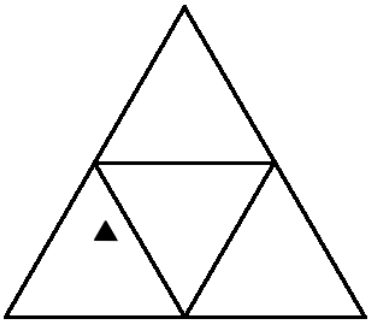
Projet python : pavage avancé

Pour celles et ceux qui veulent aller plus loin...
Remarque : il est conseillé de savoir déjà résoudre le problème « simple ».
De toutes façons vous pourrez réutiliser pas mal de votre travail pour cette version donc ce ne sera pas perdu.
Il est possible de placer le « trou » ailleurs que dans un coin. En fait, si on le place à l'un de ces emplacements (en noir), le pavage fonctionne :



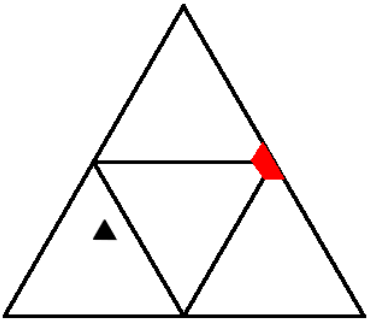
(ici pour $n = 4$)
Pourquoi ? Choisissons un de ces petits triangles, et découpons encore le triangle en 4 triangles comme tout à l'heure.





On regarde où est le trou : il est dans le triangle en bas à gauche.

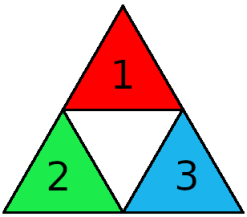
On peut alors aussi placer le petit trapèze dans les trois triangles qui n'ont pas de trou :



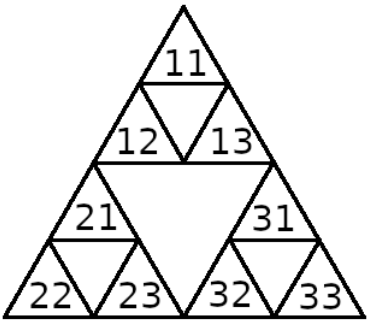
Et utiliser encore la récursivité. Le fait qu'on ait placé le triangle à cet endroit et pas n'importe où permet de s'assurer qu'au cas de base (voir ci-dessous) le « trou » se retrouvera dans un des trois cas ci-dessous, qui fonctionnent bien.

Ici on voit que si le trou est un de ces 3 triangles, ça marche. Si c'était le triangle du milieu par contre, non.

Bon, ça c'est la théorie, mais comment on fait en turtle ?
La première chose est de trouver un système de « coordonnées » pour repérer le trou. Une fois qu'on a trouvé ce système, on peut alors envisager de tirer un trou au hasard, puis de passer ces coordonnées en argument à la fonction qui dessine la solution. Il faudra ensuite, pour chaque appel récursif, bien recalculer la position du trou...
Voici un système de coordonnées un peu étrange mais qui marche bien (libre à vous d'en trouver un autre si vous préférez).
On choisit d'appeler 1, 2 et 3[1] les trois grands "triangles" dans cet ordre. Si $n = 1$ alors les coordonnées de chaque triangle sont simplement 1, 2 et 3.



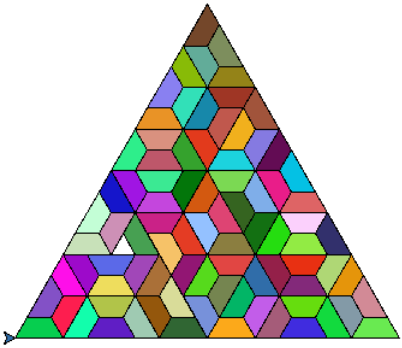
Si $n = 2$, on met un premier chiffre (entre 1 et 3) pour noter le grand « triangle », puis un deuxième chiffre (entre 1 et 3) pour noter la position dans chaque « petit triangle ».



[1]. vous pouvez aussi les appeler A, B et C ou 0, 1 et 2, bref

Et ainsi de suite. Dans l'exemple ci-dessus, le codage des coordonnées serait 2132 (voyez-vous pourquoi ?).

Et donc votre mission, si vous l'acceptez, est d'écrire la fonction qui dessine la solution, prenant en argument la position du trou. Exemple pour le positionnement ci-dessus :



(vous pourrez rajouter une fonction qui génère un trou aléatoire).

Bon courage ! :)