I Procédure de rendu

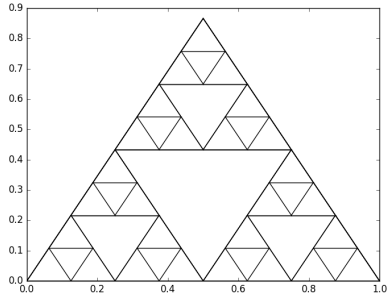
Une seule manière de rendre ce DM : se connecter au site de la classe et déposer votre fichier en cliquant à l'endroit adéquat.
Je ne prendrai pas en compte les fichiers envoyés par mail. Par contre, vous pouvez tout à fait poser vos questions sur le site ou par mail.
J'attends en plus une copie par élève contenant les informations indiquées dans l'énoncé.

Tester vos fonctions

C'est une étape indispensable. N'oublier par de tester plusieurs cas.

II Graphique

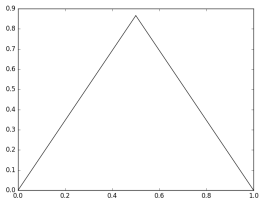
Dans ce premier exercice, le but est de reproduire la figure suivante :

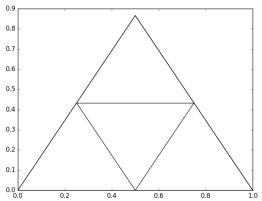


C'est le résultat de l'appel à

```
sierpinsky(4)
```

Voici également le résultat pour des valeurs de n égales à 1 et 2





Dans un premier temps, décrire sur la copie comment passer du cas $n = 1$ au cas $n = 2$. De manière plus générale, décrire comment construire l'étape n en fonction de l'étape $n - 1$. On pourra faire un schéma et donner des noms aux points importants.
Dans un deuxième temps, compléter la fonction `sierpinsky_rec` qui utilise `trace_ligne` pour les tracés.

III Un cavalier qui surgit hors de la nuit

3.1 Echiquier, cavalier

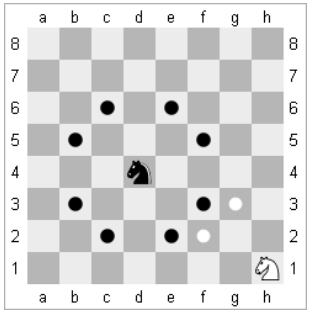
Un échiquier est une grille carrée comportant 64 cases. Il est donc de côté 8. Nous modéliserons un échiquier par une matrice (une liste de listes) et une case sera dite vide si elle contient le nombre 0. Nous utiliserons les indices python pour localiser les cases des échiquiers, c'est la dire que la case en haut à gauche est d'indice $(0, 0)$.

Exercice 1
Compléter la fonction `echiquier_vide`. Tester avec

```
e = echiquier_vide(2)
e[0][0] = 1
```

puis vérifier que seule la première case a été modifiée.

Sur un échiquier, le cavalier se déplace en L : deux cases horizontales et une verticale, ou deux cases verticales et une horizontale.



Le cavalier noir peut se déplacer sur les cases marquées d'un point noir. Idem pour le blanc.

Exercice 2
On se donne un cavalier en position $(i, j) \in [\![0, n-1]\!]^2$ sur un échiquier carré de taille n. Expliquer sur la copie sur quelles cases il peut se déplacer, puis compléter la fonction `cases_accessibles`

3.2 Parcours complet

On cherche à savoir si un cavalier placé au hasard sur un échiquier peut se déplacer successivement sur toutes les cases sans passer deux fois au même endroit.
Notre convention sera de placer un 1 dans notre matrice/échiquier à chaque fois que le cavalier passe par une case.

Exercice 3
Compléter les fonctions `fini` et `cases_disponibles`. Vous expliquerez vos algorithmes sur la copie.

Exercice 4
On considère un cavalier placé en position (i, j) sur un échiquier $n \times n$. En admettant que l'on sait répondre à la question "existe-t-il un parcours complet du cavalier" pour un échiquier de taille $n \times n$ dont 2 cases ont été visitées, expliquer sur la copie comment répondre à la question initiale.
Nous avons ainsi une procédure récursive. Quels est son cas d'arrêt ? Dans ce cas, à quel condition avons-nous trouvé un parcours complet ?
Compléter la fonction `parcours`. Avant le ou les appels récursifs, on veillera à modifier l'échiquier, et à annuler éventuellement la modification après.
Pour les tests on prendra $n = 4, 5, 6$ et on devra obtenir faux, vrai, vrai.

IV Bonus

Comment modifier la fonction précédente pour obtenir plutôt la liste des déplacements ? le nombre de parcours complets possibles ?