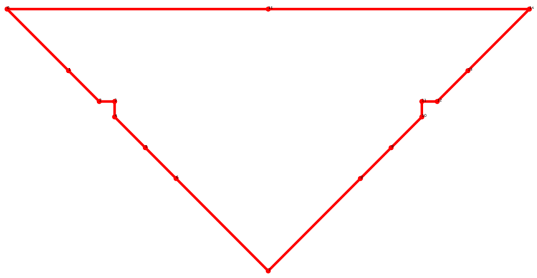
Bases de la programmation impérative

Ensimag 1A
Examen intermédiaire



1 Introduction

On s'intéresse dans cet exemple à un jeu de dessin pour enfant. Il s'agit d'un ensemble de points du plan, marqués et numérotés sur le papier. L'enfant doit relier avec un crayon les points dans l'ordre (et revenir au point de départ) pour former une image.

Nous nous proposons d'écrire une classe `EnsemblePoints` contenant un attribut `points` stockant dans l'ordre tous les points à relier d'une image. Cet attribut est un vecteur de points, chaque point étant un couple de deux flottants.

Nous vous demandons de compléter le code fourni.

Attention, la clarté du code ainsi que l'utilisation de pylint seront fortement pris en compte dans l'évaluation.

Nous vous fournissons un fichier principal *dessin.py* permettant de tester toutes les fonctionnalités demandées ainsi qu'un fichier *ensemble.py* à compléter.

2 À Implémenter

Note : Dans cet exercice, nous utiliserons différents **itérateurs**. Tous les **itérateurs** devront être écrits dans un nouveau module *iterateurs* que vous créerez.

1. Implémenter une fonction `cases` prenant en argument un itérable sur des données indexables (tuples ou vecteurs) ainsi qu'un argument entier `index` et renvoyant un **itérateur** sur tous les données numéro `index` de l'itérable d'entrée (à l'aide d'un générateur). Par exemple `for x in cases([(0, 1), (2, 3)], 0)` itérera sur 0 puis 2. `for x in cases([(0, 1), (2, 3)], 1)` itérera sur 1 puis 3.
2. Implémenter la méthode `coordonnees_max` de la classe `EnsemblePoints` renvoyant un couple composé de la coordonnée x maximale et de la coordonnée y maximale de l'ensemble.
3. Implémenter la méthode `svg_vide` de la classe `EnsemblePoints`. Cette méthode prend un nom de fichier en argument et sauvegarde un svg contenant les points numérotés

(juste les points, pas de segments). La taille de l'image devra être suffisamment grande pour contenir tous les points. On supposera que les points n'ont pas de coordonnées négatives et que le point de départ est toujours (10, 10). Vous pouvez lire le fichier *exemple_vide.svg* pour vous remémorer la syntaxe du *svg*.

Rappel Le code suivant permet l'écriture d'une ligne dans un fichier

```
1  with open("test.svg", "w") as fichier_svg:
2      fichier_svg.write("ligne\n")
```

4. Implémentez une fonction *couples* prenant en argument un itérable et renvoyant un itérateur sur tous les couples consécutifs de données (puis à la fin le couple dernière donnée, première donnée). Par exemple, `for x in couples([0, 1, 2])` itérera sur (0, 1) puis (1, 2) puis (2, 0). `for x in couples([0])` n'itérera sur rien.
5. Implémenter la méthode `eliminer_doublons` de la classe `EnsemblePoints`. Cette méthode ne prend aucun argument (en plus de l'objet) et modifie le vecteur de points stockés afin de s'assurer que deux points consécutifs ne sont jamais identiques. On s'assurera également que le dernier point sera différent du premier. Vous utiliserez la fonction `couples` de la question précédente.
6. Implémentez la méthode `svg_traits` prenant en argument un nom de fichier ainsi qu'un nombre de traits à dessiner et remplissant le fichier svg contenant les points numérotés (tous) ainsi qu'autant de segments que le nombre requis (en partant du point 0 et en suivant les points dans l'ordre).
7. On vous demande enfin une dernière méthode : la méthode `simplification`. Cette méthode prend en argument un nombre de points plus petit que le nombre actuel et modifie l'ensemble en supprimant les points les moins intéressants. Pour y arriver, on associe à chaque point une valeur flottante, son *intérêt*. On sélectionne le nombre de points voulus les plus intéressants (dont la valeur d'intérêt est la plus grande). Bien entendu, les points restants respectent encore l'ordre de départ. Par exemple pour en ensemble $[p, q, r, s, t, u, v]$ dont les points ont respectivement les intérêts $[1, 4, 5, 6, 3, 8, 9]$. Si on supprime deux points, on éliminera donc les points d'intérêts 1 et 3 soit p et t. On obtient alors $[q, r, s, u, v]$.

   L'intérêt d'un point est défini comme l'aire du triangle qu'il forme avec son point précédant et son point suivant. L'aire d'un triangle (p, q, r) est calculable avec la formule suivante :

$$|(p.x \times q.y - p.y \times q.x) + (q.x \times r.y - q.y \times r.x) + (r.x \times p.y - r.y \times p.x)|/2$$

Note : vous pourrez utiliser `sorted` qui permet de trier un vecteur (par exemple `v_trie = sorted([3, 1 ,2])`).