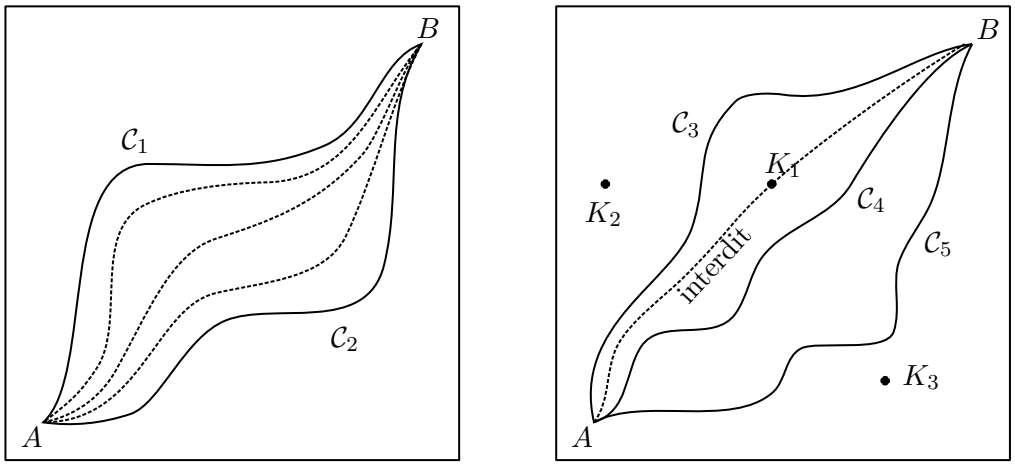
# Devoir d'informatique numéro 0

À traiter pendant les vacances d'été

Considérons deux chemins $\mathcal{C}_1$ et $\mathcal{C}_2$ tracés dans le plan entre les mêmes extrémités $A$ et $B$ (partie gauche de la figure). Il est en général possible de passer de l'un à l'autre par déformation continue et on exprime cette propriété en disant que les deux chemins sont *homotopes*. La situation se complique si on interdit aux courbes intermédiaires, tracées en pointillés, de franchir certains points $K_i$. Par exemple, sur la partie droite de la figure, n'importe quelle déformation continue amenant $\mathcal{C}_3$ sur $\mathcal{C}_4$ entraînerait le passage par $K_1$ et ces deux chemins *ne sont donc pas homotopes*. Au contraire, rien n'entrave le passage de $\mathcal{C}_4$ à $\mathcal{C}_5$ qui demeurent donc homotopes malgré la présence de « trous » en $K_1$, $K_2$ et $K_3$.



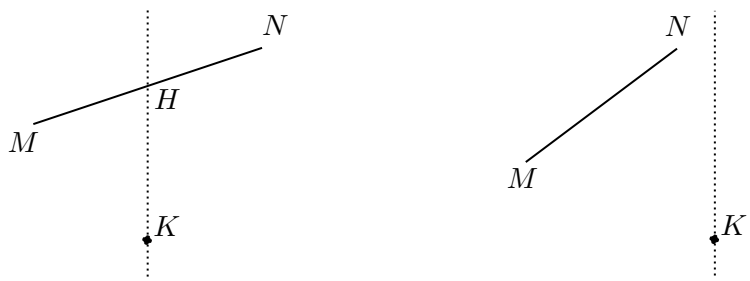

Le but de l'exercice est de déterminer informatiquement si deux chemins sont homotopes dans le plan privé d'un ensemble fini de points. Un chemin est vu comme une succession de points reliés par des segments et que l'on stocke dans une liste, chaque point étant lui-même une liste de deux coordonnées cartésiennes.

```
K_1 = [5.3, 6] # exemple de point interdit ou << trou >>
Chemin = [[0, 0], [1.2, 2.3], [3.4, 5.6], ...., [10, 10]] # ex de chemin
```

Je mets à votre disposition le fichier Python `homotopie_pour_eleves.py` contenant des exemples de chemins d'extrémités ($A = (0, 0)$, $B = (10, 10)$), des exemples de points interdits ainsi que des fonctions permettant de les représenter graphiquement. Vous utiliserez ces outils pour tester vos fonctions et illustrer les résultats. Ce fichier est accessible sur cahier-de-prepa https://cahier-de-prepa.fr/pc*-poincare/docs?informatique. L'énoncé que vous tenez en main est lui aussi disponible sous forme de fichier pdf et permet d'activer directement ce lien, ainsi que celui du dossier partagé où vous devrez déposer votre travail (non pas les codes Python, mais les figures obtenues dans la question 12).

**1.** À l'aide des fonctions fournies, représenter graphiquement les chemins $\mathcal{C}_0$, $\mathcal{C}_1$ et $\mathcal{C}_6$ et l'ensemble de points interdits $[K_3, K_7, K_9]$. Parmi les paires $(\mathcal{C}_0, \mathcal{C}_1)$, $(\mathcal{C}_0, \mathcal{C}_6)$ et $(\mathcal{C}_1, C6)$, lesquelles sont homotopes et lesquelles ne le sont pas ?

**2.** Coder la fonction `traverse(M, N, K)` prenant en argument trois points $M$, $N$ et $K$ et renvoyant un booléen indiquant si le segment $[MN]$ coupe la droite verticale passant par $K$. C'est le cas si et seulement si $(x_M - x_K)$ et $(x_N - x_K)$ sont de signes opposés.

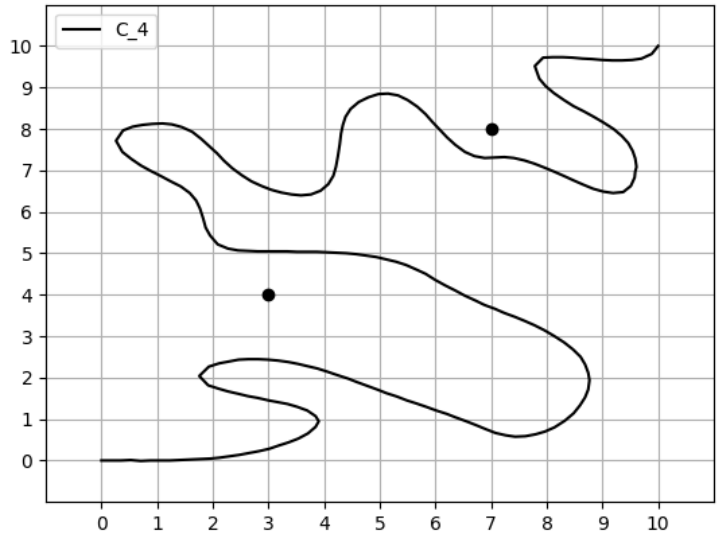


**3.** Dans le cas où la fonction précédente renvoie `True`, on montre que

$$\overline{KH} = \frac{(y_M - y_K)(x_N - x_M) + (y_N - y_M)(x_K - x_M)}{x_N - x_M} \quad .$$

Écrire la fonction `dessus_ou_dessous(M, N, K)` qui renvoie la chaîne d'un seule caractère `'+'` si $H$ est au dessus de $K$ et la chaîne `'-'` sinon. Le cas où $K$ et $H$ sont confondus n'est pas envisagé.

**4.** La recherche d'homotopie entre deux chemins passe par la détermination de leurs *séquences* respectives. Parcourons un chemin $\mathcal{C}$ de $A$ vers $B$. À chaque fois qu'on passe à la verticale de l'un des points $K_i$ interdits, on inscrit dans une liste le tuple `(i, '+')` ou le tuple `(i, '-')` selon que $H$ se trouve au dessus ou au dessous de $K_i$. Dans la suite, chacun de ces tuples sera appelé *marqueur* et c'est la liste qui les énumère dans l'ordre de leur rencontre qui constitue la séquence de ce chemin. Écrire la fonction `sequence(chemin, points_interdits)` renvoyant la séquence d'un chemin `chemin` associée aux points interdits de la liste `points_interdits`. Dans l'exemple de la figure ci-dessous où le point interdit gauche a pour indice 0 et le point interdit droit a pour indice 1, l'appel `sequence(C_4, [[3, 4], [7, 8]]))` doit renvoyer
`[(0, '-'), (0, '-'), (0, '-'), (1, '-'), (1, '-'), (0, '+'), (0, '+'), (1, '-')]`.




**5.** On introduit ici le concept de *séquence canonique* d'un chemin. L'idée est que passer deux fois de suite en dessus (ou deux fois de suite en dessous) d'un même point $K_i$ revient à ne rien faire. La séquence canonique se déduit de la séquence « brute » en la simplifiant de manière à y éliminer tous les doublets, c'est à dire les paires de marqueurs identiques successifs, tout en préservant l'ordre des autres. Il faut prendre garde au fait que de nouvelles répétitions peuvent surgir lorsqu'on en efface certaines et qu'il faut les éliminer elles aussi [1].

| séquence brute | séquence canonique |
|---|---|
| $A\underline{AA}BC\underline{AA}\,\underline{BB}E$ | $ABCE$ |
| $A\underline{B}\,\underline{CC}\,\underline{B}D$ | $AD$ |

Écrire la fonction `construit_canonique(Seq)` qui prend une séquence en paramètre et qui construit puis renvoie la séquence canonique associée. La séquence `Seq` ne doit pas être modifiée. Pour cela, il suffit de partir d'une liste `Seq_can` vide et de parcourir les éléments `Seq`. Chacun d'eux doit être inscrit dans `Seq_can` ou, au contraire, entraîner l'élimination du dernier élément de `Seq_can` s'il lui est identique.

**6.** Il est aussi possible de travailler en place. Écrire la fonction `rend_canonique(Seq)` *qui ne renvoie rien* mais qui transforme la séquence `Seq` de manière à la rendre canonique. Après l'exécution de cette fonction, `Seq` devient canonique ! On peut par exemple parcourir la séquence par indices dans une boucle `while` en supprimant deux éléments successifs avec la fonction `del` s'ils sont identiques.

1. Cette remarque appelle une écriture récursive mais nous procédons autrement dans la suite

**7.** Écrire une fonction `egalite(L1, L2)` qui indique par un booléen l'égalité de deux listes. On pourrait évidemment se contenter du test `L1 == L2` mais à titre d'exercice, je vous demande de coder naïvement cette fonction. On ne sais pas *a priori* si les deux listes présentent la même longueur.

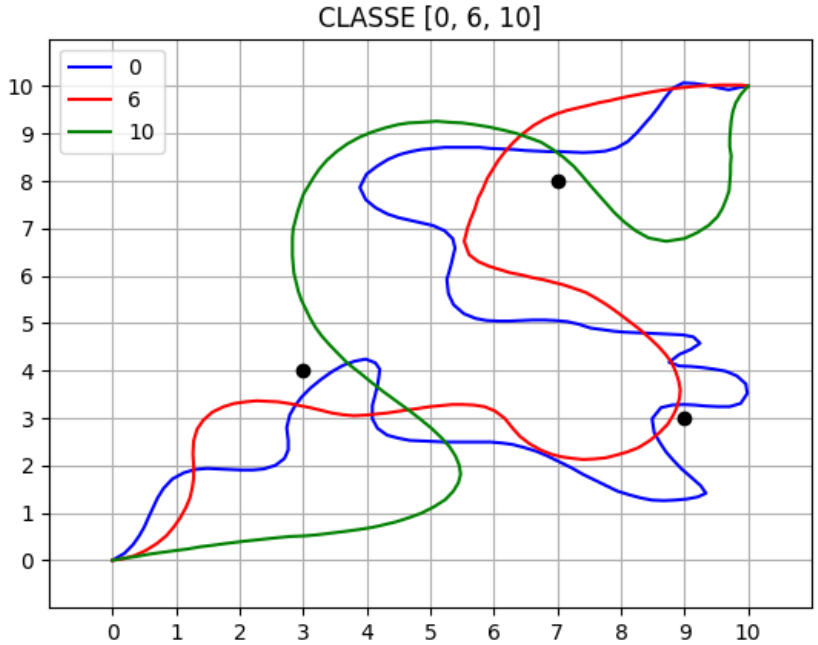
**8.** Deux chemins sont homotopes si et seulement si leurs séquences canoniques relatives aux mêmes points interdits sont égales. Écrire la fonction `sont_homotopes(C_A, C_B, liste_trous)` prenant en argument deux chemins `C_A` et `C_B` et une liste de points interdits, et renvoyant un booléen signalant leur éventuelle homotopie.

**9.** Considérons maintenant un ensemble de chemins $E$ représenté par une liste `E` (il s'agit donc d'une liste de listes de listes de deux coordonnées). Écrire la fonction `sont_homotopes_indices(E, i, j, liste_trous)` indiquant par un booléen si les chemins `E[i]` et `E[j]` sont homotopes.

**10.** On souhaite partitionner $E$ en sous-ensembles de telle manière que tous les chemins d'un même sous-ensemble soient homotopes et que deux chemins appartenant à des sous-ensembles distincts ne le soient pas. En termes mathématiques, on dit que l'homotopie est une *relation d'équivalence* et qu'on cherche les *classes d'équivalences* de $E$. Pour faciliter la lecture du résultat, on décide de travailler sur les indices des chemins dans la liste `E` plutôt que sur ces chemins eux-mêmes. On va donc construire une liste de listes, chacune énumérant les indices de chemins homotopes. Pour un ensemble de 10 chemins par exemple, la liste `[[0, 4, 5], [1, 8, 9], [2], [3, 7, 6]]` signifie qu'on peut le diviser en 4 classes d'homotopies et que les chemins d'indices 0, 4 et 5 sont homotopes alors que le chemin d'indice 2 n'est homotope à aucun autre.

Écrire la fonction `classes_equivalence(E, liste_trous)` qui renvoie cette liste des classes d'homotopie. Je vous suggère de partir d'une liste `Liste_classes` initialement vide et de parcourir par leurs indices les éléments de `E` et de regarder si chacun d'eux appartient à la classe d'un chemin déjà rencontré.

**11.** Pour conclure, écrire une fonction `analyse_et_represente(E, liste_trous)` qui détermine les classes d'homotopie d'un ensemble de chemins `E` et qui produit autant de figures qu'il y a de classes, chacune d'elles représentant par des couleurs (si possible distinctes) tous les chemins homotopes d'une même classe. On choisira les couleurs dans la liste `couleurs = ['black', 'blue', 'red', 'green', 'yellow', 'orange', 'grey']`. Avec `plt.title`, on intitulera chaque figure par la liste des chemins homotopes qu'elle représente. Voici l'exemple d'une seule figure représentant une classe de trois chemins homotopes, le plan ayant été privé de trois points signalés par des disques noirs.




**12.** Pour conclure, exécuter la fonction précédente avec l'ensemble de chemins fourni et la liste de points interdits $[K_3, K_7, K_9]$. Avec un logiciel de bureautique, regrouper les figures en un seul fichier au format pdf et le placer dans le dépôt prévu par le professeur : https://nuage04.apps.education.fr/index.php/s/zPygN8qCF6BT6Y4