DS - Paquet Cadeau

Consignes

Les consignes suivantes **doivent** être respectées sous peine d'être pénalisé par des **points négatifs**.

Le code doit être clair.
- Évitez les lignes à rallonge. S'il faut introduire une variable intermédiaire, faîte le.
- Nommez vos variables avec du sens. Évitez toutefois des noms trop longs. Auquel cas décrivez le rôle de la variable en commentaire.
- Ne surchargez pas de commentaires. Les commentaires utiles sont :
  - Spécification de la fonction
  - Description du rôle d'une variable
  - Description du rôle d'une boucle complexe.

Soignez la présentation de votre code.
- Marquez clairement les indentations.
- Éviter les ratures trop nombreuses dans un code. Une ou deux maximum.
- Écrivez votre instruction sur plusieurs lignes si elle est longue en ajoutant simplement une indentation après le retour à la ligne.

Respectez la syntaxe Python. - *0 à l'exercice si syntaxe inventée.*
- Vérifiez que ce que vous écrivez respecte la syntaxe.
- Les petites erreurs n'implique pas 0 (oublie de :, = au lieu de ==,...)
- Si vous avez une idée, mais que vous n'arrivez pas à l'écrire, n'inventez pas. Écrivez votre idée en commentaire.

Pour nommer une fonction utilisez le nom donné dans l'énoncé - *0 à l'exercice sinon*

Utilisation de `print` ou de `input` interdite pour le moment - *0 à l'exercice sinon*

Rappels de Python : Si `L` est une liste, alors `L[i]` désigne le `i`-ème élément de cette liste et `len(L)` la longueur de la liste. La commande `L[i] = x` affecte la valeur de l'expression `x` au `i`-ème élément de la liste `L`. L'expression `[]` construit une liste vide. L'expression `n * [x]` construit une liste de longueur `n` contenant `n` occurrences de `x`. La commande `L.append(x)` modifie la liste associée à `L` en lui rajoutant un nouvel élément final contenant `x`. La commande `L.pop()` modifie la liste `L` en supprimant son dernier élément et en renvoyant sa valeur. Si `L1` est un autre liste, `L + L1` renvoie la concaténation de `L` et `L1`.

Si `T` est un n-uplet, alors `T[i]` désigne le `i`-ème élément de ce n-uplet. `(1,2)` permet de créer le couple $(1, 2)$.

Important : Seules les opérations sur les listes apparaissant dans le paragraphe précédent sont autorisées dans les réponses. Si une fonction Python standard est nécessaire, elle devra être réécrite.

---

1 Exercices

1.1 Cours

Question 1.1.

Quels sont les indices valides d'une séquence de taille n ?

Question 1.2.

Que renvoient les expressions `ord(chr(12))` et `chr(ord('A')+12)` ?

Question 1.3.

Donnez l'instruction permettant d'ouvrir le fichier `nadine.py` situé dans le dossier `tajine` du disque `C:\` et d'affecter l'objet fichier obtenu à une variable nommée `fichier_nadine`.

Question 1.4.

Voici un ensemble d'instructions exécutées l'une après l'autre :

```
l1 = [12] * 3
l2 = l1
l1.append(0)
l2[0] = l1[-1]
l2.pop()
```

Donnez la liste associée à `l1` après l'exécution de ces instructions.

1.2 Algorithmes Classiques

Question 1.5.

La suite de Syracuse est définie par récurrence :

$$S_{n+1} = \begin{cases} \dfrac{S_n}{2} & \text{si } S_n \text{ est pair,} \\ 3S_n + 1 & \text{si } S_n \text{ est impair.} \end{cases}$$

Ecrire une fonction `syracuse` avec la spécification suivante :
Entrées : un entier $s \geq 1$
Sortie : Le premier entier $n \geq 0$ tel que $S_n = 1$

Question 1.6.

Écrire une fonction `recherche_lineaire` avec la spécification suivante :
Entrées : une liste l et un élément x
Sortie : L'indice de la première occurrence de x dans l, ou -1 si x n'est pas dans l

Votre fonction devra utiliser une boucle `while` pour parcourir les éléments de la liste

Question 1.7.

Écrire une fonction `comptage` avec la spécification suivante :
Entrées : une liste l et un élément x
Sortie : Le nombre de fois que x apparaît dans l

2 Problème - Algorithme du Papier Cadeau

Étant donné un ensemble de point du plan, l'enveloppe convexe de cet ensemble est un polyèdre dont les sommets sont des points de l'ensemble et qui contient tous les points de l'ensemble.

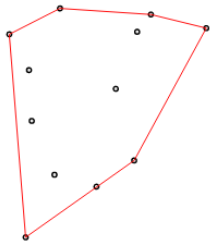

FIGURE 1 – enveloppe convexe d'un ensemble de points

Le but du problème est de résoudre le problème suivant : étant donné un ensemble de points du plan, qu'elle est l'enveloppe convexe de cet ensemble de point. Il existe plusieurs algorithmes permettant de résoudre ce problème. On implémentera une version de l'algorithme dit *du papier cadeau* ou *algorithme de Jarvis*. Puis on proposera une implémentation de l'algorithme d'Andrew, autre algorithme permettant de résoudre le problème.

2.1 Points

Les points du plan seront représentés par des couples de flottant `(x,y)` tel que `x` est l'abscisse du point et `y` dont ordonnée. On rappelle qu'étant donné un couple `p` de flottant, les instructions `x = p[0]` et `y = p[1]` permette respectivement de récupérer l'abscisse et l'ordonnée du point représenté par `p`.

Étant donnés deux points A et B on rappelle la définition de la distance euclidienne entre A et B, notée AB :

$$AB = ((x_A - x_B)^2 + (y_A - y_B)^2)^{\frac{1}{2}}$$

Question 2.1. *Distance*

Implémenter une fonction `distance` qui prend en entrée deux points sous la forme de deux couples de flottant et qui renvoie la distance euclidienne entre ces deux points.

Question 2.2. *Orientation*

On considère trois points du plan A, B et C. Intuitivement, on dit que le triplet (A, B, C) est dans le sens trigonométrique si C est à gauche de la droite (AB). Formellement pour $A(x_A, y_A)$, $B(x_B, y_B)$

et $C(x_C, y_C)$, on pose $z = (x_A - x_B)(y_C - yB) - (y_A - y_B)(x_c - x_b)$ et on donne la définition suivante :

$$(A, B, C) \text{ est dans le sens trigonométrique } \iff \begin{cases} A = B \\ z < 0 \\ z = 0 \text{ et } A \neq C \text{ AC} < \text{AB} \end{cases}$$

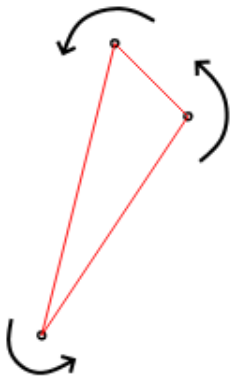

FIGURE 2 – sens trigonométrique de trois points

Implémenter une fonction `trigo` qui prend en entrée trois point A, B et C, dans cet ordre et sous la forme de deux couples de flottant, et qui renvoi un booléen vrai si et seulement si (A, B, C) sont dans le sens trigonométrique.

Question 2.3. *Ordre*

On considère deux points $A(x_A, y_A)$ et $B(x_B, y_B)$. On dit que A est *strictement plus petit* que B si $x_A < x_B$ ou si $x_A = x_B$ et $y_A < y_B$.

Implémentez une fonction `inférieur` qui prend en entrée deux point A et B, dans cet ordre et sous la forme de deux couples de flottant, et qui renvoie un booléen vrai si et seulement si A est strictement plus petit que B.

2.2 Minimums

Question 2.4. *Point minimum*

Implémentez une fonction `point_min` qui prend en entrée un ensemble de points disjoints du plan, sous la forme d'une liste de couples de flottants, et qui renvoie l'unique point de l'ensemble qui est strictement plus petit que tous les autres au sens de la question *1.3 Ordre*.

Question 2.5. *Premier entier*

Implémentez une fonction `min_int_diff` qui prend en entrée une liste d'entiers positifs, l et qui renvoie le premier entier $n \geq 0$ qui n'est pas dans l.

Question 2.6. *Minimum d'une liste extraite*

Implémentez une fonction `min_diff` qui prend en entrée une liste l de n points et une liste d'entiers t entre 0 et $n-1$ et qui renvoie l'indice du minimum de l parmi les éléments d'indices $i \notin t$. On pourra utiliser la fonction `min_int_diff` même si on n'a pas répondu à la question.

2.3 Algorithme de Jarvis

L'idée de l'algorithme du papier cadeau et de partir du plus petit point de l'ensemble et de chercher le point *le plus à gauche* pour l'ajouter dans l'enveloppe, comme si on enroulait du papier cadeau autour des points.

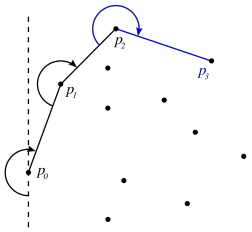

FIGURE 3 – jarvis

Question 2.7. *Point le plus à gauche*

On fixe un point P et un ensemble de points $\mathcal{E}$. Le point le plus à gauche de P dans E est défini comme le point $Q \in E$ tel que $\forall Q' \in E$, (P, Q', Q) est dans le sens trigonométrique.

Implémentez une fonction `plus_a_gauche` qui prend en entrée un point P, représenté par un couple de flottant, et un ensemble de points E non vide, représenté par une liste de couples de flottant, et qui renvoie le point de plus à gauche de E dans P.

Question 2.8. *Papier Cadeau*

L'enveloppe convexe d'un ensemble de points sera représentée par une liste de points $[e_0, e_1, ..., e_n]$ tels que $e_0 = e_n$ et pour tout $i < n$, tous les points p de l'ensemble, (e_i, e_{i+1}, p) n'est pas dans le sens trigo.

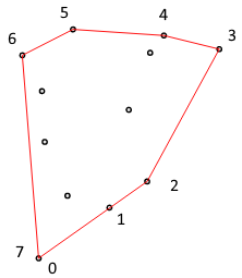

FIGURE 4 – liste des 8 points de l'enveloppe convexe dans l'ordre.

L'algorithme du papier cadeau prend en entrée un ensemble $\mathcal{E}$ de deux points ou plus, sous la forme d'une liste de couples de flottants, renvoie l'enveloppe convexe de l'ensemble sous la forme d'une liste comme décrit ci-dessus, et fonctionne comme ceci :

1. Initialise une liste `env` vide.
2. Calcule e_0 le plus petit point de l'ensemble et l'ajoute à la fin de `env`.
3. Calcule e comme le point le plus à gauche du dernier élément de `env` dans $\mathcal{E}$ et l'ajoute à la fin de `env`.
4. Tant que le dernier élément de `env` n'est pas égal au premier élément de `env`, répéter l'étape 3.

Écrire une fonction `jarvis` qui implémente l'algorithme du papier cadeau.

2.4 Algorithme d'Andrew

L'algorithme d'Andrew tri d'abord l'ensemble des points en utilisant la relation définie dans la question *1.3 Ordre* pour diminuer le nombre de comparaisons à réaliser.

Question 2.9. *Tri des points*

Pour trier les points, on décide d'utiliser l'algorithme de tri nommé *tri par sélection*. Supposons qu'on cherche à trier l. Le tri par sélection réalise les étapes de calcul suivantes :

1. Initialiser une liste r vide.
2. Calculer le minimum, m_0, de l et l'ajouter à la fin de r.
3. Calculer le minimum, m_1, de l privée de m_0 et l'ajouter à la fin de r.
4. Calculer le minimum, m_2, de l privée de m_0 et m_1 et l'ajouter à la fin de r.
5. etc. jusqu'à ce que r contienne tous les éléments de r.

En utilisant `point_min` et `min_diff`, même si vous n'avez pas répondu à ces questions, implémenter une fonction `tri_points` qui prend en entrée un ensemble de point, sous la forme d'une liste de couples de flottant, et renvoi une copie de la liste triée par la méthode du tri par sélection.

Attention, la liste prise en entrée ne doit pas être modifiée par votre fonction.

Question 2.10. *Enveloppe convexe*

L'algorithme d'Andrew est décrit par les étapes suivantes sur une liste l de points :

1. Définir une copie `t` triée de `l`.
2. Initialiser deux listes vides `haut` et `bas`.
3. Pour tout point P de l
   (a) On note H et G le dernier et l'avant-dernier points de `haut` s'ils existent.
   (b) Tant que H et G existent et que (P, H, G) n'est pas dans le sens trigonométrique, supprimer H de `haut`
   (c) Ajouter P à la fin de `haut`
   (d) On note B et C le dernier et l'avant-dernier points de `bas` s'ils existent.
   (e) Tant que B et C existent et que (C, B, P) n'est pas dans le sens trigonométrique, supprimer B de `bas`
   (f) Ajouter P à la fin de `bas`
4. On concatène `haut` privé de son dernier élément et `bas`, privée de son premier élément, puis inversé.

Écrire une fonction `andrew` qui implémente l'algorithme d'Andrew.