Examen de TP, fin du premier semestre

Ouvrez dans pyzo le fichier `TP_note_sem1.py` situé dans le répertoire `Devoirs/fleck/TP_note_sem1/`. Après l'avoir ouvert, et avant toute autre chose, changez la valeur de la variable `numero_du_TP` et mettez-la à la valeur précisée ci-après:

```
1 numero_du_TP = 2
```

N'oubliez pas que l'exécution doit se faire via `Ctrl-SHIFT-E` pour que tout marche bien.

Partie I

Tir à la corde

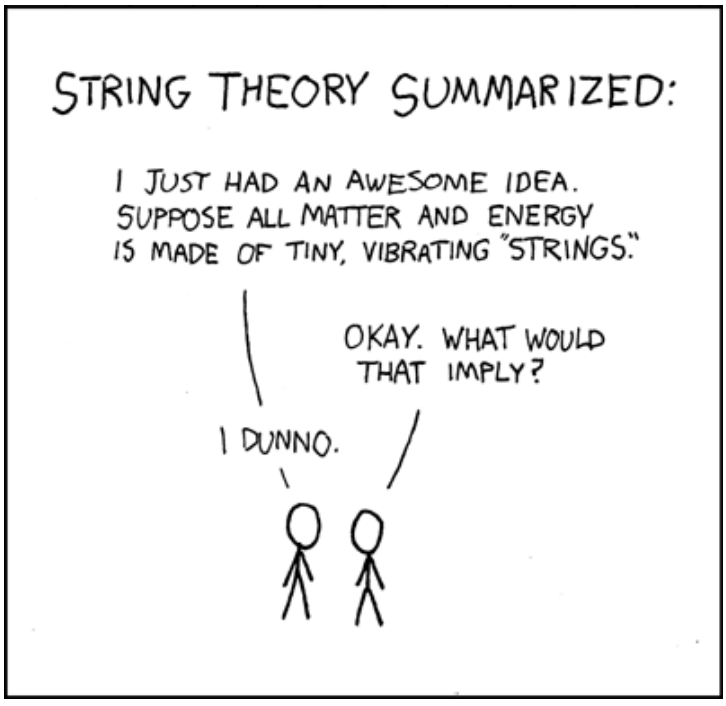
Un championnat de « tir à la corde » est organisé par l'auberge dans laquelle vous vous trouvez actuellement. Vous êtes chargé de l'enregistrement des deux équipes et les membres de chacune d'elles viennent vous voir, en alternance, pour la pesée. En calculant le poids total dans chaque équipe, vous essayez de faire un premier pronostic.

Votre programme accepte une unique liste en argument qui contient les poids de chacune des personnes, en alternance : les premier, troisième, cinquième... sont dans l'équipe 1 alors que les second, quatrième, sixième....sont dans l'équipe 2.

Ce que doit faire votre programme :

1. Vérifier s'il y a bien le même nombre de joueurs dans chaque équipe. Si ce n'est pas le cas, le programme doit renvoyer `None`.
2. Faire la somme des poids des joueurs de chaque équipe et renvoyer dans l'ordre le poids total de l'équipe 1, le poids total de l'équipe 2 et une chaîne[1] de caractères dont le contenu sera `"L'equipe X est avantagee."` où bien sûr le `"X"` sera remplacé par le numéro de l'équipe dont le poids total est le plus important.

On vous garantit que les poids des deux équipes ne seront jamais égaux.



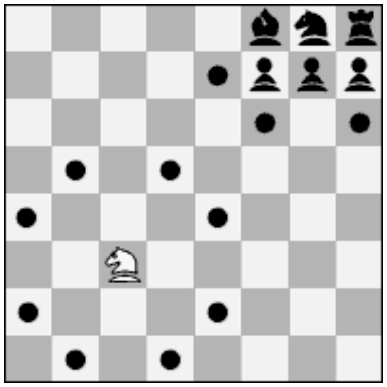
`This works on pretty much every level.`

[1] Sans accents, avec une majuscule au début et un point à la fin.

Partie II

Tempête sur l'échiquier

1 Introduction



Vous devez écrire un programme qui détermine, dans une partie d'échecs, si un cavalier peut prendre une pièce de l'adversaire.

Rappelons que les échecs se jouent sur un plateau carré de 8 cases par 8. Un cavalier se déplace en « L », c'est-à-dire en avançant de deux cases horizontalement ou verticalement, puis en avançant d'une case à la perpendiculaire de la direction précédente. Le cavalier peut se déplacer même si les cases intermédiaires de son déplacement contiennent une pièce (amie ou ennemie). La figure ci-contre montre toutes les possibilités de déplacements de deux cavaliers.

L'échiquier vous sera à chaque fois fourni sous la forme d'une liste contenant 8 listes, elles-même contenant chacune 8 éléments représentatifs des pièces posées à cet endroit. Les lettres majuscules représentent les pièces blanches, les minuscules les pièces noires, et les `'.'` représentent les cases vides. Les cavaliers sont représentés par la lettre `'c'` (ou `'C'`), et les autres pièces par d'autres lettres de l'alphabet.

Pour faciliter vos explorations, vous disposez d'un echiquier d'exemple nommé `exemple_echiquier` que vous pouvez afficher à l'aide de la fonction `affiche_echiquier`, soit sous forme de liste de liste, soit sous forme plus compacte en ne gardant que les caractères adéquats.

```
>>> affiche_echiquier(exemple_echiquier)
[['t', 'c', '.', 'd', 'r', 'f', '.', 't'],
 ['p', 'p', 'p', '.', 'p', 'p', 'p', 'p'],
 ['.', '.', '.', 'p', '.', '.', '.', 'c'],
 ['.', '.', '.', '.', '.', 'f', '.', '.'],
 ['.', '.', 'C', '.', '.', 'P', '.', '.'],
 ['.', '.', 'P', '.', 'D', '.', 'P', '.'],
 ['P', 'P', '.', '.', '.', '.', '.', 'P'],
 ['T', '.', 'F', '.', 'R', 'F', 'C', 'T']]
>>> affiche_echiquier(exemple_echiquier,raw=True)
tc.drf.t
ppp.pppp
...p...c
.....f..
..C..P..
..P.D.P.
PP.....P
T.F.RFCT
```

2 Détection des cavaliers

Écrire une première fonction `detection_cavaliers` qui détecte la positions de tous les cavaliers noirs présents sur le plateau. En particulier,

- s'il n'y en a pas, elle doit renvoyer `None` ;
- s'il y en a un ou plus[2], elle doit renvoyer une liste des doublets des positions[3] (sous forme `(ligne,colonne)`) de tous les cavaliers noirs présents sur l'échiquier.

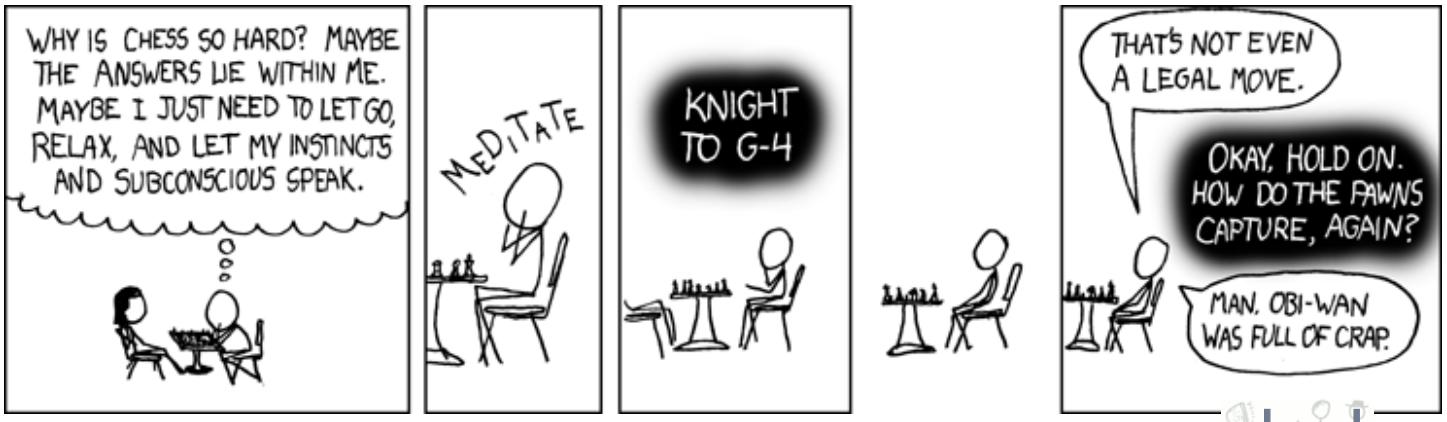
[2]NB: il peut y en avoir plus de deux du fait du phénomène de « promotion de pion »: quand un pion arrive au bout de l'échiquier, il peut se transformer en n'importe quelle pièce de sa couleur, notamment un troisième ou quatrième cavalier (il peut au maximum y avoir 12 cavaliers sur l'échiquier).

[3]L'ordre dans lequel vous renvoyez les positions n'aura pas d'incidence dans les tests de ce TP.

Exemple:

```
>>> detection_cavaliers(exemple_echiquier)
[(0, 1), (2, 7)]
```

3 Positions possibles



`You know that 'sweep the pieces off the board and see it in your mind' thing? Doesn't work.`

Une fois la détection des cavaliers effectuées, écrire une fonction `deplacements_possibles` qui détermine, à partir d'une position donnée sous forme d'un doublet `(ligne,colonne)` la liste des positions accessibles[4] si le cavalier était à la position de départ indiquée. Par exemple

```
>>> position_initiale = (5, 2)
>>> deplacements_possibles(position_initiale)
[(3, 1), (3, 3), (4, 0), (4, 4), (6, 0), (6, 4), (7, 1), (7, 3)]
>>> position_initiale2 = (0, 6)
>>> deplacements_possibles(position_initiale2)
[(1, 4), (2, 5), (2, 7)]
```

Il n'y a pas besoin de donner l'échiquier en argument car on se contente de vérifier que le cavalier peut se déplacer sur le plateau à partir de la position donnée, indépendamment de la présence d'autres pièces au points d'arrivée ou même de la présence effective d'un cavalier au point de départ[5].

4 Plat de résistance

Vous avez à présent toutes les armes pour répondre à la question posée. En particulier, votre dernière fonction `prise_possible` doit renvoyer:

- `True` s'il existe un cavalier noir en position de prendre une pièce de l'équipe adverse[6] ;
- `False` si aucun cavalier noir ne peut prendre de pièce adverse ;
- `None` s'il n'y a plus de cavalier noir sur l'échiquier.

Exemple:

```
>>> prise_possible(exemple_echiquier)
False
```

[4] L'ordre dans lequel vous donnez ces positions n'aura pas d'importance pour les tests de ce TP.

[5] En d'autres termes: *Si* il y avait un cavalier à cet endroit, quels seraient toutes les cases *potentiellement* accessibles en l'absence de toute autre pièce.

[6] Y compris le roi, même si aux échecs une telle possibilité n'existe pas.

Partie III

Cours à la carte

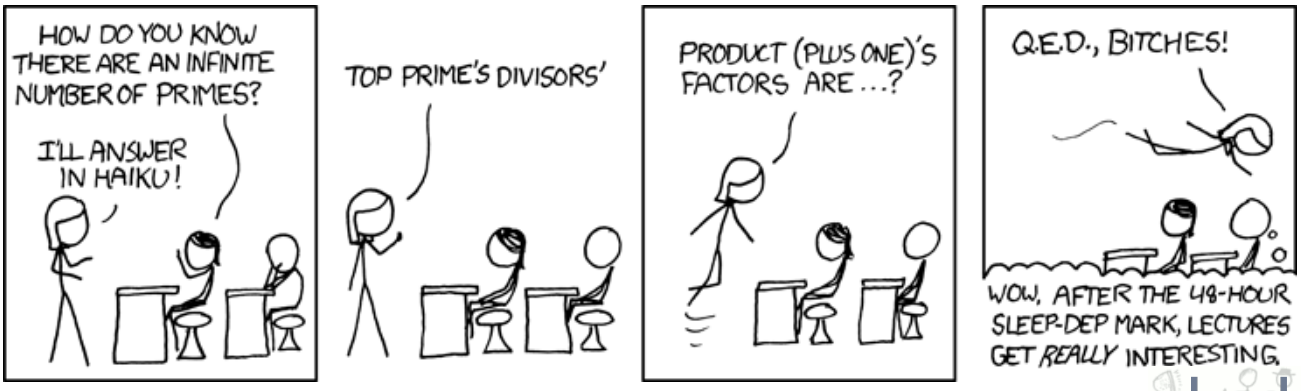
L'école où vous étudiez est très libre : vous pouvez choisir les cours auxquels vous vous rendez. Dans une journée, N cours se succèdent. La seule contrainte est que vous devez obligatoirement assister à au moins M cours de cette journée.

Chacun de ces cours vous rapporte un certain nombre de points P pour le diplôme délivré par votre école. Étant quelqu'un de très fainéant, vous désirez vous rendre à un minimum de cours (donc à M cours), et vous souhaitez aussi que ces cours se suivent afin de ne pas perdre de temps à attendre au milieu de votre journée. Mais comme vous êtes en plus quelqu'un de malin, vous voulez également que ces cours vous rapportent un maximum de points pour le diplôme.

Étant donné la liste des points rapportés par chaque cours et le nombre minimal de cours à suivre, déterminez le nombre maximal de points que vous pourrez obtenir pour votre diplôme en vous fatiguant un minimum et en ayant tous vos cours à la suite.

Exemple:

```
>>> cours = [1, 5, 3, 2, 6, 3, 1]
>>> M = 4
>>> flemmard(M,cours) # choisit les cours à 5,3,2 et 6 points
16
```



After somewhere around 40 hours,
there's no academic reason to go to the class.
Only go for the hallucinations.

Crap, I have levitation class at 25:131. Better set the alarm to 'cinnamon'.