

SUPINFO Academic Dept.

**Algorithmique en Python**

Graded exercise

Durée : 4 heures

Version 1.0

Last update: 27/11/2014

Use: Students/Staff

Author: Laurent GODEFROY

SOMMAIRE

0 Préambule 3

1 Problème : LE jeu de Ping 3

1.1 Le jeu de Ping en lui même 3

1.1.1 Les règles du jeu 3

1.1.2 Des algorithmes pour jouer à ce jeu 9

1.2 Un jeu qui n’a a priori rien à voir 12

1.2.1 D’autres règles de jeu 12

1.2.2 Des algorithmes pour jouer à cet autre jeu 12

1.3 Un lien ? 15

# Préambule

Votre rendu se fera sous la forme d’un document « .py » que vous nommerez en respectant la convention **ID-1ADS-votreCampus-TP** (exemple : 66280-1ADS-Tours-TP). Vous le joindrez à un mail dont l’objet sera ce même nom.

Vous enverrez ce mail à **votre formateur**, et vous mettrez **en copie** la boîte [1ADS@supinfo.com](mailto:1ADS@supinfo.com). Si vous ne mettez pas cette dernière adresse en copie aucune réclamation ultérieure ne sera recevable.

Le non respect de ces consignes entrainera automatiquement une non correction de votre examen et par conséquent une note de 0 (cf. règlement des études).

Pour rappel, seuls les quatre premiers supports de cours au format ppt sont autorisés. En aucun cas la correction des exercices des séances Labs n'est consultable. Pas de consultations de sites web.

# Problème : LE jeu de Ping

**Remarque préliminaire : aucunes connaissances en mathématiques ne sont nécessaires pour résoudre ce problème.**

L’auteur de ce sujet a découvert les règles de ce jeu dans un article de Jean-Paul Delahaye « Un jeu à épisodes pour l’été » de la revue « Pour La Science ».

## Le jeu de Ping en lui même

### Les règles du jeu

Ce jeu solitaire se déroule sur un plateau rectangulaire constitué de cases, et avec des pions bicolores, noirs d’un côté et blancs de l’autre. Il y a toujours exactement un pion par case. Au départ tous les pions montrent leur face blanche. A chaque tour de jeu, on choisit une case, et tous les pions sur les cases avoisinantes changent de couleur. Le but est de retourner tous les pions afin qu’ils montrent tous leur face noire.

Le nombre de « cases avoisinantes » dépend de la position de la case choisie sur le plateau : il y en a trois si l’on est sur un coin, cinq si l’on est sur un bord sans être sur un coin, et huit si l’on est « à l’intérieur » du plateau.

* Les trois cases avoisinantes (en bleu) d’un coin (en rouge) :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

* Les cinq cases avoisinantes (en bleu) d’une case sur le bord mais pas dans un coin (en rouge) :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

* Les huit cases avoisinantes (en bleu) d’une case à l’intérieur du plateau (en rouge) :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Exemple de déroulement d’une partie où l’on joue sur un tableau de quatre cases :

* Situation de départ, tous les pions montrent leur face blanche :

|  |  |
| --- | --- |
|  |  |
|  |  |

* On choisit la case en haut à gauche, on obtient :

|  |  |
| --- | --- |
|  |  |
|  |  |

* On choisit la case en haut à droite, on obtient :

|  |  |
| --- | --- |
|  |  |
|  |  |

* On choisit la case en bas à droite, on obtient :

|  |  |
| --- | --- |
|  |  |
|  |  |

* On choisit la case en bas à gauche, on obtient

|  |  |
| --- | --- |
|  |  |
|  |  |

Et c’est gagné !

Un autre exemple, moins trivial avec un plateau de quatre cases sur quatre cases :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la première ligne, première colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la première ligne, quatrième colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la deuxième ligne, deuxième colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la deuxième ligne, troisième colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la troisième ligne, première colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Choix de la case située sur la troisième ligne, quatrième colonne :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Et c’est gagné !

### Des algorithmes pour jouer à ce jeu

**Vous êtes invités à lire cette partie en entier avant de commencer à écrire vos codes.**

Ici le plateau sera une liste à deux dimensions constituée de booléens, « True » représentant la couleur blanche et « False » la couleur noire.

1. Ecrire une fonction ne prenant pas de paramètre et qui retourne les valeurs de deux entiers saisis par l’utilisateur. On fera en sorte que ces deux entiers soient nécessairement strictement positifs.
2. Ecrire une fonction prenant en paramètre deux entiers que l’on sait strictement positifs, **n** et **m** par exemple, et qui retourne une liste à deux dimensions comportant **n** lignes et **m** colonnes dont tous les éléments sont égaux à « True ».
3. Ecrire une procédure réalisant l’affichage une liste à deux dimensions constituée de booléens, liste passée en paramètre, avec la convention suivante : un « True » est représenté par un ‘O’ et un « False » par un ‘X’.   
   Par exemple le deuxième plateau de la page 7 sera affiché comme suit

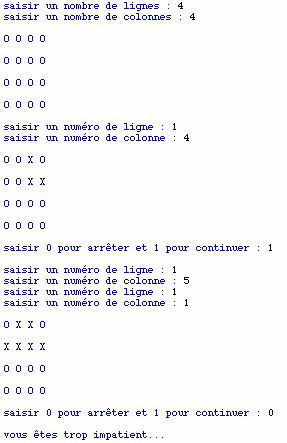
express:Users:laurentgodefroy:Desktop:Image 10.png

1. Ecrire une fonction prenant en paramètre deux entiers strictement positifs, **n** et **m** par exemple, qui fait saisir à l’utilisateur deux entiers **i** et **j** tels que et qui retourne la valeur de ces deux entiers.

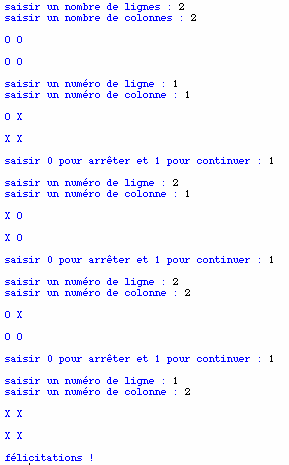


1. Ecrire une procédure prenant en paramètre une liste à deux dimensions constituée de booléens, et deux entiers naturels positifs, le premier inférieur ou égal au nombre de lignes de la liste et le second inférieur ou égal au nombre de colonnes. Ces deux derniers entiers représentent donc les coordonnées d’une case. Cette procédure inversera les valeurs des cases de la liste, qui sont avoisinantes à la case dont les coordonnées sont passées en paramètre. Cette notion de « cases avoisinantes » est bien sûr celle donnée dans la sous-partie précédente « les règles du jeu ».
2. Ecrire une fonction prenant en paramètre une liste à deux dimensions constituée de booléens et qui retourne « True » si toutes les cases de la liste valent « False » et « False » sinon.
3. Ecrire une fonction ne prenant pas de paramètre et qui retourne un booléen indiquant si l’utilisateur à envie de continuer à jouer ou pas.
4. Ecrire une procédure principale utilisant toutes les fonctions et procédures précédentes afin de faire jouer un utilisateur au jeu de Ping jusqu’à qu’il ait gagné ou qu’il ait envie d’arrêter. On indiquera si le joueur a gagné ou pas.

Voici un exemple de partie terminée prématurément :



Et voici un exemple de partie gagnée :



## Un jeu qui n’a a priori rien à voir

### D’autres règles de jeu

Cet autre jeu solitaire se déroule lui aussi sur un plateau rectangulaire constitué de cases. Sur chaque case le joueur peut poser ou non une étoile. Il sera déclaré vainqueur si chaque case est entourée (au même sens que précédemment) par un nombre impair d’étoile.

Exemple de jeu gagnant sur un plateau de quatre cases sur quatre cases :

|  |  |  |  |
| --- | --- | --- | --- |
| ★ | ★ | ★ |  |
|  |  | ★ | ★ |
|  | ★ |  | ★ |
|  |  |  | ★ |

### Des algorithmes pour jouer à cet autre jeu

**Vous êtes invités à lire cette partie en entier avant de commencer à écrire vos codes.**

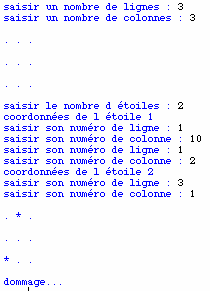
Ici le plateau sera une liste à deux dimensions constituée de booléens, « True » représentant la présence d’une étoile et « False » son absence.

1. Ecrire une fonction prenant en paramètre deux entiers que l’on sait strictement positifs, **n** et **m** par exemple, et qui retourne une liste à deux dimensions comportant **n** lignes et **m** colonnes dont tous les éléments sont égaux à « False ».
2. Ecrire une procédure prenant en paramètre une liste à deux dimensions constituée de booléens. Cette procédure commence par demander un entier strictement positif à l’utilisateur. Elle fait ensuite saisir les coordonnées d’un nombre d’étoiles égal à cet entier, en faisant en sorte que l’on ne puisse saisir deux fois les mêmes coordonnées. On pensera également à ne pas « sortir » du plateau. Pour chacune des ces coordonnées valides, elle affectera la case correspondante du plateau avec la valeur « True ».
3. Ecrire une procédure réalisant l’affichage une liste à deux dimensions constituée de booléens passée en paramètre, avec la convention suivante : un « True » est représenté par un ‘\*’ et un « False » par un ‘.’   
   Par exemple le plateau de la page 12 sera affiché comme suit :

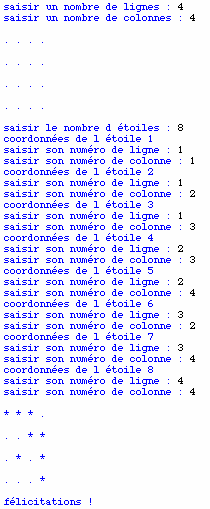
express:Users:laurentgodefroy:Desktop:Image 7.png

1. Ecrire une fonction prenant en paramètre une liste à deux dimensions constituée de booléens, et qui retourne « True » si toutes les cases du plateau sont entourées d’un nombre impair de cases possédant la valeur « True ». Elle retournera « False » sinon.
2. Ecrire une procédure principale utilisant toutes les fonctions et procédures précédentes afin de faire jouer un utilisateur à ce jeu. On indiquera si le joueur a gagné ou pas.

Voici un exemple de partie perdue :



Et un exemple de partie gagnée :



## Un lien ?

Quel lien pouvez-vous faire entre les jeux des deux parties précédentes ?