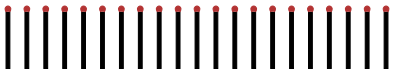
Projet n°1
Le jeu de Nim



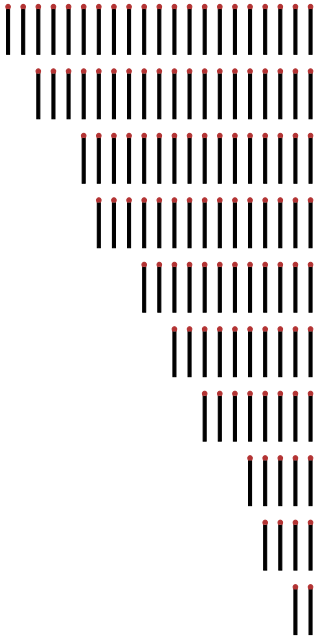
I Présentation du jeu de Nim

Le jeu de Nim [1] est un jeu de stratégie qui oppose deux joueurs placés face à un certain nombre d'allumettes [2]. Chacun leur tour, les joueurs prennent une, deux ou trois allumettes. Suivant les variantes, le joueur qui prend la dernière allumette est soit le gagnant, soit le perdant.

Stratégie gagnante. Si le gagnant est celui qui prend la dernière allumette, on se rend compte qu'il se passe quelque-chose de particulier s'il reste quatre allumettes. En effet, celui qui se retrouve à jouer lorsqu'il reste quatre allumettes a perdu : s'il en prend une, l'autre en prend trois, s'il en prend deux, l'autre en prend deux, s'il en prend trois, l'autre en prend une ; dans tous les cas, c'est l'autre joueur qui gagne.

Une stratégie gagnante lorsque le gagnant est celui qui prend la dernière allumette consiste donc à laisser quatre allumettes à son adversaire. Ceci est possible en laissant, à chaque étape, un multiple de quatre allumettes sur le jeu.

Exemple de partie. Voici un exemple de partie avec 21 allumettes dans laquelle le joueur n°2 applique la stratégie gagnante.



Le joueur 1 en prend 2
Le joueur 2 en prend 3
Le joueur 1 en prend 1
Le joueur 2 en prend 3
Le joueur 1 en prend 2
Le joueur 2 en prend 2
Le joueur 1 en prend 3
Le joueur 2 en prend 1
Le joueur 1 en prend 2
Le joueur 2 en prend 2 et gagne la partie.

II Présentation du projet

Ce projet consiste à implémenter en Python un jeu de Nim interactif. L'utilisateur pourra choisir de jouer contre un autre joueur ou seul contre la machine. Il choisira aussi le nombre d'allumettes au début de la partie ainsi que la variante jouée : le joueur qui prend la dernière allumette est soit le gagnant, soit le perdant du jeu. Vous devez donc implémenter les deux variantes et les deux stratégies gagnantes correspondantes.

Pour vous aider, la structure du code (le squelette) a déjà été écrite dans le fichier `nim.py` disponible dans le Capytale *186d-247491*. Il existe deux manière de lancer le jeu :

1. Il existe en fait plusieurs jeux de Nim. Voir `https://fr.wikipedia.org/wiki/Jeux_de_Nim`.
2. On utilise ici des allumettes mais des billes, des pièces, etc. font très bien l'affaire.

- depuis `Jupyter` (ou tout autre environnement Python) avec

```
from nim import jouer

jouer()
```

- depuis la ligne de commande Linux, avec

```
$ ./nim.py
```

en ayant pris soin, au préalable, de rendre le fichier `nim.py` exécutable avec la commande

```
$ chmod u+x nim.py
```

Un affichage possible du programme attendu est donné ci-dessous.

```
Jeu de Nim
Jouer contre la machine ? [o/n]  o
Le gagnant est celui qui prend la dernière allumette ? [o/n]  o
Combien d'allumettes ? [1-100]  21
|||||||||||||||||||||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  1
||||||||||||||||||||
[Joueur 2] : l'ordinateur prend 1 allumette(s).
|||||||||||||||||||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  3
|||||||||||||||
[Joueur 2] : l'ordinateur prend 1 allumette(s).
||||||||||||||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  3
||||||||||||
[Joueur 2] : l'ordinateur prend 1 allumette(s).
|||||||||||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  3
||||||||
[Joueur 2] : l'ordinateur prend 1 allumette(s).
|||||||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  3
||||
[Joueur 2] : l'ordinateur prend 1 allumette(s).
|||
[Joueur 1] : retirer combien d'allumettes ? [1-3]  3
Le joueur 1 a gagné !
Rejouer ? [o/n]  n
Au revoir
```

III Tests

Il vous est demandé de tester intensivement votre projet. On s'attachera en particulier à tester la réponse de la fonction `prise_ia` pour tous les nombres d'allumettes possibles entre 1 et 100, pour les deux variantes du jeu.

Les tests sont à placer dans le fichier `test_nim.py` qui possède déjà plusieurs tests et qui permet d'automatiser leur exécution et d'obtenir une synthèse [3].

3. Cet environnement de tests a été conçu pour le cours de NSI. En pratique, on utilise plutôt des solutions plus avancées comme Unittest, Doctest ou Pytest par exemple.

Dès que vous ajoutez une fonction dont le nom commence par `test_` au fichier `test_nim.py`, celle-ci est automatiquement ajoutée au jeu de tests. Pour exécuter le jeu de tests, deux possibilités s'offrent à vous :

- depuis `Jupyter` (ou tout autre environnement Python) avec

```
from test_nim import tester

tester()
```

- depuis la ligne de commande Linux, avec

```
$ ./test_nim.py
```

en ayant pris soin, au préalable, de rendre le fichier `test_nim.py` exécutable avec la commande

```
$ chmod u+x test_nim.py
```

L'affichage produit est le suivant :

```
1/4         afficher_jeu                OK
2/4         prise_ia                    échec /!\ (AssertionError)
3/4         reponse_entier              OK
4/4         reponse_oui_non             OK
Synthèse : 3/4 tests passés (75.0%)
```


Info

Attention, pour exécuter le script nim.py ou le jeu de test test_nim.py depuis Capytale, il sera nécessaire de mettre à jours la liste des fichiers annexes avec la dernière version.

IV Bonus

Si votre programme est complet, des bonus pourront être attribués pour l'implémentation des fonctionnalités suivantes[4] :

- Pour être sûre de gagner, lorsqu'elle est utilisée, « l'intelligence artificielle » débute ou non la partie en fonction du nombre d'allumettes et de la variante du jeu. Exemple : si le gagnant est celui qui prend la dernière allumette et qu'il y en a 21 au début, il vaut mieux commencer. Si en revanche le perdant est celui qui prend la dernière allumette et qu'il y en a 21, il vaut mieux ne pas commencer !
- Le joueur qui débute la partie est tiré au sort.
- Le joueur n°2 n'applique plus la stratégie gagnante mais prend un nombre aléatoire d'allumettes.
- Implémenter la stratégie gagnante lorsque l'on peut prendre 1, 2 ou 4 allumettes mais pas 3.
- Réaliser l'affichage du jeu avec Turtle.

V Critères d'évaluation

Les projets sont à réaliser en groupe. Les membres du groupe doivent se montrer actifs pendant les séances, sinon, un malus sera appliqué.

L'indentation devra être impeccable et le code lisible. En particulier, toutes les fonctions, à l'exception des fonctions de test, devront comporter une dosctring claire.

Votre programme devra passer le jeu de tests des enseignants. Votre jeu de test devra être le plus complet possible et être lisible (présence de commentaires).

Un compte-rendu d'une page ou deux, au format PDF, devra reprendre les points importants de votre implémentation, de vos tests et des éventuels bonus. Il convient par exemple d'expliquer la stratégie gagnante retenue dans le cas où le perdant est celui qui prend la dernière allumette.

4. Si vous voulez proposer une autre fonctionnalité, parlez-en avec l'enseignant pour savoir si celle-ci pourra donner lieu à un bonus.

VI Rendu des projets

Les projets seront à rendre sur Pronote au plus tard le jeudi 21 Janvier 2022. Il est demandé de réaliser une archive ZIP contenant trois fichiers :

- le fichier du jeu, `nim.py` ;
- le fichier de tests, `test_nim.py`
- et le compte-rendu au format PDF.