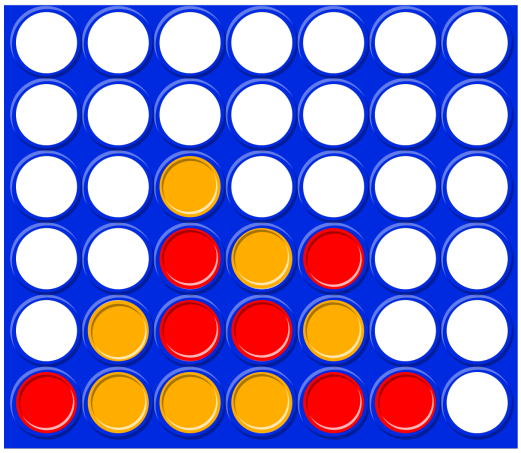
INFO-F101 – Programmation
Projet 2
"Puissance 4"

Année académique 2018–2019



Le "puissance 4" est un jeu de société pour deux joueurs édité par l'entreprise MB. Le jeu est composé de 7 colonnes adjacentes d'une hauteur permettant d'y insérer 6 jetons. Un joueur possède 21 jetons jaunes, l'autre 21 rouges. Le cadre en plastique est généralement bleu. Tour à tour, les joueurs déposent un jeton dans la colonne de leur choix, celui-ci tombe dans la position libre la plus basse. Une fois qu'une colonne est pleine, il n'est plus permis d'y glisser un jeton. Le premier joueur parvenant à aligner 4 jetons de sa couleur, horizontalement, verticalement ou en diagonale gagne la partie : "4 pions alignés, c'est gagné !" Si les 42 jetons ont été placés sans alignement de 4 jetons d'une même couleur, la partie est nulle.

Consignes

Pour ce second projet, nous vous demandons d'implémenter un "puissance 4" ainsi qu'une intelligence artificielle permettant à un joueur d'affronter l'ordinateur. Celle-ci est relativement basique et développée par niveaux. Dans le premier, l'intelligence artificielle (IA) se contente de placer son jeton au hasard parmi les colonnes possibles. Dans le second niveau, l'IA joue au hasard sauf si l'adversaire possède un alignement de 3 jetons dans une fenêtre de 4 et qu'un choix permet de briser cet alignement. Dans le troisième niveau, en plus de ce qui est décrit ci-avant, si un choix permet à l'IA d'obtenir un alignement de 3 jetons dans une fenêtre de 4, elle opère ce choix. Il vous est également demandé de développer un quatrième niveau de votre choix dont vous détaillerez le fonctionnement dans un *docstring* au sein de votre programme.

Modalités

Vous êtes libres au niveau de l'implémentation de votre programme. Néanmoins, afin de pouvoir tester celui-ci, nous vous demandons d'utiliser la fonction `randint` du module `random` d'une manière strictement définie. L'interface utilisateur est également fixée, vous n'avez aucune liberté à ce niveau. Votre interface doit correspondre en tout point à l'exemple donné dans ce document. Un module UpyLab sera disponible, il vous permettra de tester votre solution et la conformité de son interface. Des tests automatiques seront exécutés sur votre programme. Si vous déviez de ce qui est demandé, ces tests échoueront et ceci sera pénalisé.

Au début de la partie, votre programme commence par demander à l'utilisateur le niveau de l'IA ainsi qu'une *seed*. Cette graine permet de "fixer" le hasard. Après un appel à `seed(29)`, le premier appel à `randint(1,7)` donnera toujours 5 et le suivant 1 (ceci peut dépendre de votre version de *python*) ! Il doit y avoir un appel unique à `seed` au début de votre programme avec la *seed* donnée par l'utilisateur. Un appel à `randint(a,b)` permet de demander un nombre aléatoire entre `a` et `b`, bornes incluses. Il est important que vous respectiez les consignes suivantes scrupuleusement. Un seul appel à `randint` doit être effectué par coup de l'IA. Par défaut, celui-ci est réalisé avec `(1,7)` comme paramètre représentant les colonnes 1 à 7. Si deux colonnes ne représentent pas un choix valide —parce qu'elles sont pleines par exemple— l'appel est effectué avec `(1,5)` comme paramètre. Il s'agit des choix possibles considérés de manière croissante. Ainsi, si les colonnes 2 et 5 sont pleines, le mappage est le suivant (avec entier : colonne) : 1 : 1, 2 : 3, 3 : 4, 4 : 6, 5 : 7. Dans le cadre du second niveau de l'IA, si plusieurs choix sont possibles afin de briser l'alignement de l'adversaire, il est fait appel à `randint` de la même manière. Si deux choix permettent de briser l'alignement, un appel à `randint(1,2)` est effectué. L'exemple ci-après vous indique comment importer et utiliser le module.

```
>>> from random import randint, seed
>>> seed(29)
>>> randint(1,7)
5
```

Le programme doit être exécutable via la commande suivante aux salles machines du bâtiment NO :

```
~>python3 projet2.py
```

Voici un exemple d'exécution du programme attendu :

```
Bienvenue au Puissance 4
Veuillez choisir le niveau de l'IA (1-4) : 1
Veuillez entrer votre graine : 29
|       |
|       |
|       |
|       |
|       |
|       |
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 4
|       |
|       |
```

```
|       |
|       |
|       |
|   XO  |
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 4
|       |
|       |
|       |
|       |
|   X   |
|0  XO  |
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 5
|       |
|       |
|       |
|       |
|   XX  |
|0 0XO  |
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 3
|       |
|       |
|       |
|    0  |
|  XXX  |
|0 0XO  |
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 4
|       |
|       |
|       |
|   XO  |
|  XXX  |
|0 0XO 0|
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 6
|       |
|       |
|    0  |
|   XO  |
|  XXX  |
|0 0XOX0|
-1234567-
Dans quelle colonne souhaitez-vous jouer ? 6
|       |
|       |
|    0  |
```

```
|   X0  |
|  XXXX |
|0 0X0X0|
-1234567-
Félicitations, vous avez gagné la partie !
```

Quelques jours après la publication de cet énoncé, un module UpyLab (`http://upylab.ulb.ac.be`) vous permettra de tester votre solution. Veuillez à reproduire à l'identique l'output de l'exemple de fonctionnement ci-dessus afin de permettre l'exécution de tests automatiques sur votre solution. Un fichier contenant les chaînes de caractères utilisées vous est fourni sur l'Université Virtuelle : `projet2.py`, il est encodé en `UTF8`, il vous est demandé de l'utiliser. Vous ne devriez pas avoir besoin d'autres chaînes de caractères. Afin d'afficher une chaîne de caractères sans passer ensuite à la ligne, vous transmettrez un second paramètre à la fonction `print()` de la manière suivante :

```
print(maChaine,end="")
```

Pour afficher à l'écran plusieurs chaînes de caractères sur une même ligne, vous pouvez également concaténer celles-ci grâce à l'opérateur + de la manière suivante :

```
print(maChaine+uneAutre+" "+str(88)+" fois "+str(val))
```

Notez l'utilisation de `str()` avec un nombre en paramètre afin de transformer celui-ci en chaîne de caractère pouvant être concaténée.

Veillez à commenter votre programme avec pertinence. La lisibilité de votre code sera prise en considération, n'hésitez pas à structurer votre code à l'aide de fonctions.

Consignes pour la remise du projet

Le projet devra être remis via deux canaux : une version imprimée au secrétariat étudiant (Maryka Peetroons - P.2N8.104) et une version électronique sur l'Université virtuelle. Pour le document imprimé, n'utilisez pas de farde et/ou chemise en plastique. Si il comporte plusieurs pages, agrafez les. Veillez à soigner la présentation de ce document. Même imprimé en noir et blanc, utilisez la coloration syntaxique afin d'en améliorer la lisibilité et évitez les retours à la ligne non indentés pour vos commentaires. Les consignes pour la remise du projet sont disponibles en ligne sur la page du cours sur l'Université Virtuelle. Les consignes sont à respecter *scrupuleusement* ; relisez-les attentivement avant la remise !

Votre projet sera testé à l'aide de la commande suivante : `python3 projet2.py`.

Pour toute question concernant l'énoncé, nous vous invitons à vous adresser à Cédric Ternon (cternon@ulb.ac.be)

Date limite de remise. Le lundi 5 novembre 2018 à 13h.