Atelier 2 - Tous les sujets - UE S2 Projet

1 Objectif

Programmer la représentation python et la représentation à l'écran du jeu. Réaliser la saisie de coordonnées. Vous rendrez un seul fichier .py sans mentionner votre nom et ne contenant pas la suite du programme. Le programme doit être écrit en python 3 et compréhensible par vos pairs.

2 Détails

Votre programme devra afficher 3 "grilles", tester la fonction `est_dans_grille` (voir plus loin) et permettre la saisie de coordonnées.

2.1 Représentation

2.1.1 Structure de données

Vous devez préciser la structure de données, c'est-à-dire les variables et leur type, dont vous avez besoin pour représenter les éléments du jeu.

Pour illustrer votre choix de structures de données, vous pourrez coder une ou plusieurs fonctions qui permettent de donner des valeurs pour la ou les variables en début, en milieu et en fin de partie (quelques coups avant). Ces configurations (début, milieu et fin de partie) seront utilisées tout au long du projet pour permettre aux relecteurs de tester votre programme sans jouer une partie entière.

2.1.2 Affichage de l'état de la partie

Vous devez définir une ou plusieurs fonctions qui affichent la grille et les informations pour jouer (joueur ou "couleur" dont c'est le tour, lignes, colonnes...). C'est à vous de choisir un affichage explicite pour l'utilisateur tout en n'étant pas trop chargé. Voici deux exemples convenables :

Quelques indications :

- au moins la grille doit être en paramètre
- il est possible de décomposer : afficher un grille de taille $n \times n$, c'est afficher n lignes. Afficher une ligne, c'est afficher n cases...
- vous NE devez PAS utiliser de bibliothèque graphique particulière (tkinter ou pygame par exemple).
- pour éviter les erreurs d'encodage (lorsqu'on travail sur des systèmes d'exploitations différents ou qu'on utilise des caractères spéciaux) : ajoutez `# -*- coding: utf-8 -*-` en début de programme. Même commentée cette ligne est "lue" (interprétée) par python qui tolère ainsi les accents et autres caractères encodés en utf-8.

2.2 Saisie

2.2.1 Jeux de tests

Vous devez définir une fonction `est_dans_grille(ligne, colonne, grille)` et éventuellement ses fonctions auxiliaires. Cette fonction devra vérifier que les coordonnées (ligne,colonne) appartiennent

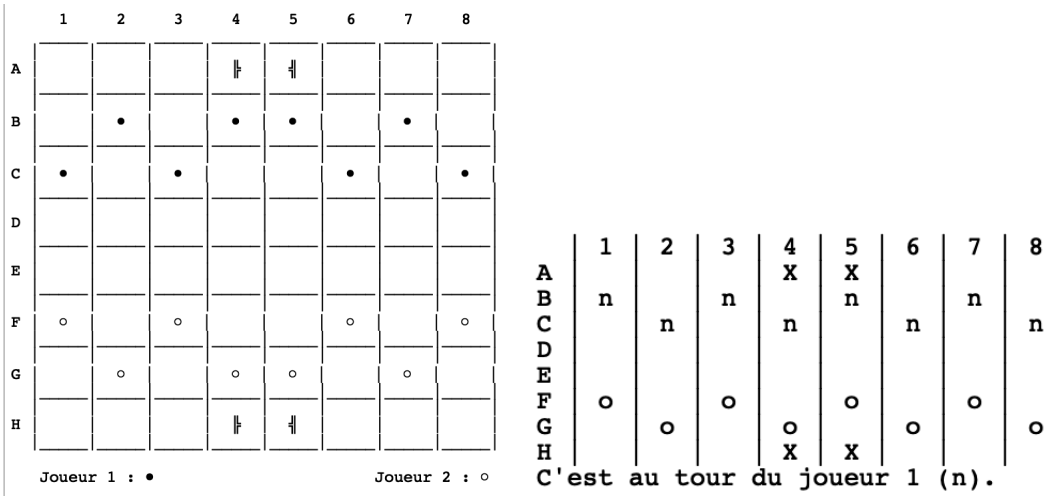

FIGURE 1 – Exemples d'affichage

à la grille.

Cette fonction servira lors de la saisie, pour vérifier que le joueur n'a pas saisi "n'importe quoi". Par exemple "X22" n'est pas valide dans une grille 4×4.

Selon vos choix d'implémentation, ligne et colonne peuvent être des caractères ou des entiers, il est probablement plus simple qu'il s'agisse d'entiers.

Pour cette fonction `est_dans_grille(...)` vous écrirez une fonction `test_est_dans_grille()` qui contiendra les jeux de tests (assert) selon le principe du Test Driven Development (voir "Bowling" vu en TP d'Algo). Voici un exemple de fonction de test pour la fonction qui ferait la sommes des éléments positifs d'une liste :

```
1 def test_somme_positifs():
2     assert somme_positifs([])==0, "liste vide, somme nulle"
3     assert somme_positifs([-1,-2,-3])==0, "liste uniquement negatifs, somme nulle"
4     assert somme_positifs([1,2,-3,1,12])==16, "liste mixte, 1+2+1+12"
```

Code 1 – "Exemple fonction de test"

Pour cet atelier, cette fonction sera intégrée dans votre fichier principal (un seul fichier à rendre).

2.3 Saisie de coordonnées

Vous devez définir une ou plusieurs fonctions qui permettent de saisir des coordonnées jusqu'à ce qu'elles correspondent bien à une position **dans** la grille (en utilisant la fonction précédente), sans vous préoccuper de savoir si cette position peut être jouée en respect des règles du jeu (atelier suivant). Cette fonction doit renvoyer les coordonnées vérifiées vis à vis de l'appartenance dans la grille.

Quelques conseils :

— Convertir un entier en lettre et une lettre en entier
Les lettres ont un code "ASCII". Par exemple, la lettre "A" a le code 65, "B" 66 et "a" 97...
Pour passer de l'un à l'autre et permettre des calculs on peut utiliser les fonctions python

suivantes : `ord("a")` vaut 97 et `chr(97)` vaut "a".

— Une chaîne de caractères est une liste
On peut facilement manipuler les caractères d'une chaîne de caractère. Par exemple pour `chaine = "A1"` on aura `chaine[0]` qui vaut `"A"` et `chaine[1]` qui vaut `"1"`

3 Evaluation

Voici une liste non exhaustive des aspects sur lesquels vous serez évalués :

— Respect des consignes (env 1/20) :
  — Anonymat,
  — Un seul fichier, .py
  — Uniquement la partie demandée
  — Compilation : le code ne doit pas comporter d'erreurs (en cas d'erreur l'évaluateur pourra légitimement mettre 0)

— Fonctionnalités :
L'évaluation portera sur l'existence, la correction et la pertinence du code demandé
  — Structure de données (env 5/20)
  — Affichage (env 3/20)
  — `est_dans_grille` et sa fonction de tests (env 5/20)
  — Saisie (env. 3/20)

— Clean code (env 3/20) :
  — le nom des variables est explicite et respecte les conventions
  — le programme est composé de fonctions courtes et explicites
  — les fonctions s'appliquent à des paramètres (et pas de variables globales)
  — le programme est commenté pour améliorer la lisibilité et compréhension
  — le programme inclut des tests significatifs pour chaque fonction
  — le programme est essentiellement construit à l'aide de fonctions, il y a un code principal (de moins de 15 lignes)

4 Squelette proposé

A titre indicatif, voici un squelette minimaliste de ce à quoi pourrait ressembler votre programme. Il est souvent plus facile de faire à votre envie mais pour faciliter l'évaluation, le correcteur ne doit pas avoir à corriger votre code pour l'exécuter.

Prévoyez donc un programme principal qui réalise :

— l'appel de l'affichage pour les 3 configurations,
— l'appel de la fonction de test pour `est_dans_grille`,
— et l'appel d'une saisie.

```
1 # -*- coding: utf-8 -*- ## Pour s'assurer de la compatiblite entre correcteurs
2
3 #### REPRESENTATION DES DONNEES
4 ###initialisation des grilles et autres variables de jeu
5 ...
6
7 #### REPRESENTATION GRAPHIQUE
```

```python
def afficher_grille(grille...) :
    ...

#### SAISIE
###fonctions de verification
#jeux de tests
def test_est_dans_grille():
    assert ...

#verification dans grille
def est_dans_grille(...) :
...

###fonctions de saisie
def saisir_coordonnees(...) :
    ...

#### CODE PRINCIPAL
# execution affichage sur les 3 grilles et autres variables de jeux
afficher_grille(grille...)
afficher_grille(grille...)
afficher_grille(grille...)

#execution test est_dans_grille
test_est_dans_grille() #uniquement pour la mise au point, a conserver pour le correcteur

#affichage des coordonnees saisies
print(saisir_coordonnees( ... ))
```

5 Astuces / conseils complémentaires Python

Ces astuces sont proposées mais ne sont pas forcément nécessaires.

— Afficher un message lors de la saisie : `reponse = input(message)`
Cette ligne est équivalente aux deux suivantes :
`print(message)`
`reponse = input()`

— Afficher un message sans retour à la ligne : `print(message,end="")`
Le paramètre end est un mot clé en python qui précise quel caractère de fin la fonction `print` doit utiliser en fin du message. Par défaut, en l'absence de précision sur ce paramètre, python ajoute un saut de ligne.

Rappels :

— pas d'objet
— pas de récursivité
— pas de try/except
— pas de bibliothèque non standard