TP N° 05 et 06 - Module R107

Le but de ce TP est de commencer à programmer un jeu de bataille navale en mode texte en python.

1. Introduction :

Soit la variable suivante : `g=[[1,2,3],['a','b','c']]`.

1. Quel est le contenu de `g[0][1]` ? De `g[1][2]` ?
2. Comment afficher `g` sous la forme de 2 lignes de 3 valeurs séparées par des espaces ?

```
1 2 3
a b c
```

3. Comment afficher `g` sous la forme de 3 lignes et 2 colonnes ?

```
1 a
2 b
3 c
```

2. Bataille navale :

Le but du jeu est de « couler » tous les bateaux du joueur adverse. Chaque joueur possède les bateaux suivants :

- 1 PORTE_AVION (noté P) de taille 4
- 2 CROISEURS (noté C) de taille 3
- 3 TORPILLEURS (noté T) de taille 2
- 4 SOUS_MARIN (noté S) de taille 1

Ils sont placés secrètement, sans se toucher, sur une grille carrée de 8 lignes par 10 colonnes (dans notre cas).
Chaque joueur, à son tour, « fait feu » sur une case donnée de l'adversaire. S'il s'agit du morceau d'un bateau ennemi, l'adversaire indique « touché » et le joueur attaquant peut « tirer » une seconde fois. Quand tous les morceaux d'un bateau sont touchés, on doit dire « touché-coulé ». Si un joueur manque sa cible, on dit alors « dans l'eau » et le tour du joueur est terminé. Le joueur ayant coulé le premier, tous les bateaux de son adversaire a gagné la partie.

1. Tester le jeu en ligne par exemple sur http://fr.battleship-game.org/.
2. On décide de représenter les morceaux de chaque bateau par une lettre et les cases vides par un tiret. Créer manuellement une liste de liste contenant les 8 lignes et les 10 colonnes de la grille suivante :

```
- C C C - - - T T -
- - - - - - - - - -
C - - - - - - - - -
C - P P P P - - - T
C - - - - - - - - T
- - S - T - - - - -
- - - - T - - S - S
- S - - - - - - - -
```

Fig. 1 : Version simple

```
  A B C D E F G H I J
1 - C C C - - - T T -
2 - - - - - - - - - -
3 C - - - - - - - - -
4 C - P P P P - - - T
5 C - - - - - - - - T
6 - - S - T - - - - -
7 - - - - T - - S - S
8 - S - - - - - - - -
```

Fig 2 : Version avec numérotation

3. Créer ensuite une fonction qui affiche uniquement le contenu d'une grille (Voir fig. 1 : Version simple).
   EN OPTION : Afficher la numérotation des lignes et des colonnes (Voir fig. 2). Pour les colonnes, on peut, soit utiliser une liste contenant les lettres `ABCD...`, soit afficher la lettre à partir de son code ASCII : la fonction `ord(c)` donne le code ASCII de la lettre `c` et la fonction `chr(x)` donne le caractère correspondant au code ASCII `x`.
4. Comment obtenir le contenu de la case « C4 » ? Faire une fonction qui transforme la chaîne passée en paramètre en coordonnées de la case dans la grille (`i` pour les lignes et `j` pour les colonnes). Pour la valeur `C4`, on doit obtenir `i=3` et `j=2`. Tester ensuite votre fonction pour les cases « H7 » et « F6 ».

```
cStr="C4"
i,j=getCoordonnees(cStr)
print("La case",cStr,"i =",i,"j =",j,"contient :",g[i][j])
```

5. Comment noter les cases indiquées par l'adversaire dans la grille, `@` quand un bateau est touché et `*` quand c'est dans l'eau.
6. Faire une boucle dans laquelle on entre une case au clavier, on modifie la grille et on la ré-affiche.

7. Comment compter et afficher les coups ?
8. Comment déterminer quand tous les bateaux sont coulés ?
9. Comment afficher les coups de l'adversaire en couleur dans le terminal (rouge pour « touché » et bleu pour « dans l'eau ») voir par exemple le module `termcolor` (Si besoin : `pip3 install termcolor`).
10. Comment sauvegarder une grille dans un fichier ? Faire une fonction.
11. Comment lire un fichier de sauvegarde ? Faire une fonction.

3. Mise en place des bateaux :

Dans cette partie, on désire pouvoir placer les bateaux sur une grille vide.

1. Créer une fonction qui fabrique une grille vide.
2. Comment placer le PORTE_AVION (noté P) de taille 4, en position B2 orienté en vertical pour obtenir la grille de la figure 3 ?

```
   A B C D E F G H I J
1  - - - - - - - - - -
2  - P - - - - - - - -
3  - P - - - - - - - -
4  - P - - - - - - - -
5  - P - - - - - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
```

Figure 3

```
   A B C D E F G H I J
1  - - - - - - - - - -
2  - P P P P - - - - -
3  - - - - - - - - - -
4  - - - - - - - - - -
5  - - - - - - - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
```

Figure 4

3. Que faut-il changer pour obtenir la figure 4, avec le bateau en horizontal ?
4. Créer alors 2 fonctions (`placeBateauHorizontal()` et `placeBateauVertical()`) pour placer des bateaux sur une grille à partir de la position souhaitée, de la lettre représentant le bateau et de sa taille.
5. Si au lieu de placer le PORTE_AVION en B2, on choisit la case H6. Pourquoi ne peut-on pas placer le bateau en horizontal ou en vertical ? Quel test faut-il faire ? Créer alors les fonctions `verifHorizontal(grille,pos,taille)` et `verifVertical(grille,pos,taille)`.
6. Lorsque l'on doit placer les autres bateaux, il faut aussi vérifier que les bateaux ne se superposent pas et qu'ils ne se touchent pas également (Voir figures 1 et 2). Par exemple les grilles des figures 5 et 6 ci-dessous sont incorrectes :

```
   A B C D E F G H I J
1  - - - - - - - - - -
2  - P - - - - - - - -
3  - P - - - - - - - -
4  - C C C - - - - - -
5  - P - - - - - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
```

Figure 5 : Bateaux superposés

```
   A B C D E F G H I J
1  - - - - - - - - - -
2  - P P P P - - - - -
3  - - - - - C - - - -
4  - - - - - C - - - -
5  - - - - - C - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
```

Figure 6: Bateaux qui se touchent

Modifier les fonctions précédentes pour tenir compte de ces 2 contraintes.

7. Faire une boucle placer tous les bateaux sur la grille en demandant à chaque fois à l'utilisateur la position et l'orientation.
8. Comment générer une grille avec un placement aléatoire (Voir module `random` et la fonction `randint()`)?

4. Programmation réseau :

Dans cette partie, on désire pouvoir jouer à distance entre 2 machines. On vous fourni un exemple de client/serveur réalisant un échange de textes entre 2 programmes.

1. Tester les fichiers fournis : lancer en premier `bnServer-base.py` dans une console et `bnClient-base.py` dans une autre. Modifier les message envoyés entre le client et le serveur.
2. Adapter alors les 2 programmes pour réaliser une connexion distante entre 2 joueurs de bataille navale. On pourra partir avec une grille sauvegardée en fichier des 2 cotés ou bien utiliser le système aléatoire de la question 3.8. Pour simplifier un peu le système, on ne fera uniquement qu'un 1 seul coup par joueur.
   Le client jouera donc en premier en envoyant une position `B3` par exemple et le serveur devra répondre en fonction de sa grille (`TOUCHE` ou bien `DANS L'EAU`). Ensuite ce sera au tour du serveur d'envoyer une position, etc... A chaque coup on devra afficher la grille personnelle et la grille de tir.
3. Finaliser le jeu pour que quand un joueur touche un bateau, il puisse rejouer à nouveau.