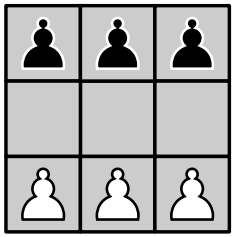
Hexapawn

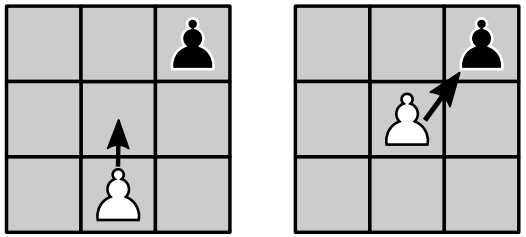
Règles du jeu

Hexapawn a été inventé par Martin Gardner, un écrivain américain de vulgarisation scientifique. C'est une sorte de jeu d'échecs miniature sur un damier de 9 cases, avec 3 pions par joueur (d'où son, qui signifie 6 pions).



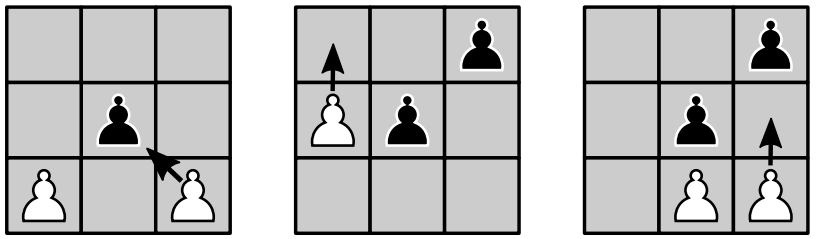
Chaque joueur a trois pions positionnés sur le bord du plateau, à l'opposée de ceux de son adversaire. Chaque point dispose des deux possibilités de déplacement habituelles aux échecs :

- **Déplacement simple :** Avancer tout droit sur une case libre
- **Attaque :** Prendre un pion adverse en avançant en diagonale.



Il y a trois façons de gagner la partie :

- **Destruction :** Prendre tous les pions adverses
- **Conquête :** Atteindre une case de la zone de départ de l'adversaire.
- **Blocage :** Empêcher l'adversaire de pouvoir jouer un coup supplémentaire.



Dans ce TP, les joueurs blancs joueront en premier. L'objectif du TP est de simuler des parties entre deux joueurs :

- Dans un premier temps, les deux joueurs jouent à chaque coup un coup aléatoire parmi tous ceux qui leur sont possibles.
- Dans un second temps, on mettra en place une intelligence artificielle pour le second joueur afin de lui permettre d'apprendre de ses erreurs au fur et à mesure de ses défaites.

Dans les deux cas, une visualisation graphique du cumul des scores des deux joueurs au fur et à mesure des parties permettra d'apprécier les différences de résultat.

1 Représentation Python

On représente un plateau de jeu en Python par une matrice (c'est-à-dire une liste de listes) d'entiers. Si `p` représente un tel plateau de jeu, alors `p[i][j]` contient un 1 si la case contient un pion blanc, un 2 si elle contient un point noir, et enfin un 0 si elle est vide. L'indice i désigne l'indice de la ligne de la grille (numérotée du bas vers le haut), tandis que l'indice j est celui de colonne (de la gauche vers la droite). Ainsi, le plateau initial est représenté par

```
p = [[1,1,1],[0,0,0],[2,2,2]]
```

1. Ecrire une foncton `init` sans argument qui renvoie la matrice représentant le jeu initial.
2. Ecrire une fonction `affiche` qui prend en argument une matrice et réalise à l'aide de `print` un affichage sympathique du plateau de jeu correspondant.

Dans toute la suite, on pourra utiliser la fonction suivante pour créer une copie d'un plateau sans risquer des effets de bords.

```
def copier(p):
    return [[j for j in i] for i in p]
```

2 Calcul des coups jouables et du gagnant

3. Ecrire une foncton `pieces` qui prend en argument l'indice `j` d'un joueur (1 ou 2 en pratique) et un plateau `p` et qui renvoie la liste des couples de coordonnées de ses pièces.
4. Ecrire une fonction `coups_jouables_pion` qui prend en argument l'indice `j` d'un joueur, un couple de coordonnées `pos` d'un de ses pions, et un plateau `p`, et qui renvoie la liste de tous les plateaux que l'on peut obtenir en jouant depuis ce pion.

   Indications : On rappelle qu'il y au plus trois possibilités : avancer ou prendre en diagonale (vers le gauche ou vers la droite). On suggère l'organisation suivante :

   - Initialiser une liste vide à laquelle on ajoutera au plus 3 plateaux (les 3 éventuels coups jouables)
   - Tester l'une après l'autre la possibilité d'avancer, attaquer à gauche et attaquer à droite.
   - Les blancs montent, les noirs descendent. Pour éviter d'avoir à distinguer deux cas en fonction du joueur, on pourra remarquer que :
     - L'adversaire du joueur `j` est `3-j`
     - La quantité `d=3-2*j` vaut 1 si $j = 1$ et -1 si $j = 2$. Ainsi, quel que soit le joueur, un pion depuis la position `pos` atterrit dans tous les cas sur la ligne `pos[0]+d` (et il faut penser à vérifier que cette valeur est bien comprise entre 0 et 2).
5. Ecrire une fonction `coups_jouables` qui prend en argument l'indice `j` d'un joueur et un plateau `p`, et qui renvoie la liste de tous les plateaux que l'on peut obtenir en jouant un coup.
6. Ecrire une fonction `gagnant` qui prend en argument l'indice `j` d'un joueur dont c'est le tour, et un plateau `p` et qui renvoie le numéro du gagnant s'il y en a un. Dans le cas contraire, la fonction renverra 0. On rappelle que si un joueur n'a plus aucun coup à jouer, il perd la partie.

3 Parties au hasard

7. Ecrire une fonction `partie_hasard_affichee()` sans arguments qui fait jouer une partie jusqu'à la victoire d'un des deux joueurs, chaque joueur jouant au hasard. On fera afficher le plateau après chaque coup grâce à la fonction de la question 2, ainsi que le nom du gagnant.

   On pourra importer le module `random` et utiliser la fonction `random.choice` qui prend en argument une liste `L` et renvoie un élément aléatoire de cette liste (avec équiprobabilité).
8. Modifier la fonction précédente pour qu'elle renvoie uniquement le numéro du gagnant, sans faire d'affichage.
9. Tester vos fonctions sur environ 1000 parties en traçant un graphe ayant en abscisse le nombre de parties jouées et en ordonnée le nombre de victoires de chaque joueur. Quelle semble être la proportion de victoire du joueur 1 ?

4 Parties avec apprentissage

Dans toute cette partie, on suppose disposer d'une variable globale `perdants` qui est une liste initialement vide. Cette liste sera utilisée pour stocker le plateau à l'issue de son **dernier** coup joué à chaque défaite du joueur 2.

10. Ecrire une fonction `elimine_coups_perdants` qui prend en argument une liste de plateaux `L` et qui renvoie une nouvelle liste contenant les éléments de `L` qui n'apparaissent pas dans la liste globale `perdants`.
11. Créer une nouvelle fonction `partie_hasard_affichee_apprenante` à nouveau sans arguments qui fait jouer une partie jusqu'à la victoire d'un des deux joueurs (en affichant à nouveau le nom du vainqueur). Les 2 joueurs jouent à nouveau des coups au hasard, si ce n'est que le joueur 2 élimine systématiquement tous les plateaux perdants de ces coups possibles. S'il se retrouve à ne pouvoir joueur aucun coup non perdant, on arrêtera la partie en déclarant le joueur 1 vainqueur, en affichant le message `position perdante pour le joueur` 2 et on stockera à nouveau le dernier plateau dans la liste `perdants`.
12. Modifier la fonction précédente en supprimant tout l'affichage pour qu'elle renvoie le nom du vainqueur. Effectuer ensuite à nouveau un graphique dressant le nombre de victoires de chaque joueur en fonction du nombre de parties jouées sur une cinquantaine de parties. Que remarquez-vous ?