Echecs

L'exercice consiste a trouver les positions possibles dans une grille, on attend de l'eleve a retranscrire les regles de deplacement en indices sur une matrice qui fait office d'abstraction pour un plateau d'echecs. Personellement pour les pieces fou, tour et reine j'ai utilise une boilerplate que je modifie selon mes besoins. Exemple pour le fou:

```
moves = []
    directions = [(1,  1), (1, -1), (-1,  1), (-1, -1)]

    for dx, dy in directions:
        i, j = x + dx, y + dy
        while  0 <= i <  8 and  0 <= j <  8:
            moves.append((i, j))
            i += dx
            j += dy

    return moves
```

j'ai une liste contenant les soustractions/addition necessaire pour obtenir la case. Puisque ces pieces n'ont pas de limitations (on peut se deplacer en E4 comme en F3), on itere jusqu'a ce qu'on atteint les bords du plateau, c'est a dire quand on est a l'indice 0 ou 8 sur au moins un des deux axes

Pareille pour tour et reine:

```
def cases_tour(x, y):
    moves = []
    directions = [(0,  1), (1, 0), (-1, 0), (0, -1)]

    for dx, dy in directions:
        i, j = x + dx, y + dy
        while  0 <= i <  8 and  0 <= j <  8:
            moves.append((i, j))
            i += dx
            j += dy

    return moves


def cases_reine(x, y):

    moves = []
    directions = [(0,  1), (1, 0), (-1, 0), (0, -1), (1,1), (-1,-1), (1,-1), (-1, 1)]

    for dx, dy in directions:
        i, j = x + dx, y + dy
        while  0 <= i <  8 and  0 <= j <  8:
            moves.append((i, j))
            i += dx
            j += dy
```

```
    return moves
```

Pour les pieces comme le roi et le cavalier, j'ai juste enleve la boucle while, pour que les deplacements se font qu'une fois:

```
def cases_roi(col,lig):
      moves = []
   directions = [(0,  1), (1, 0), (-1, 0), (0, -1), (1,1), (-1,-1), (1,-1), (-1, 1)]
   for x, y in directions:
       if 0<x+col< 8 and 0<lig+y<8:
           moves.append((x+lig, y+col))

   return moves

def cases_cavalier(lig, col):

    moves = []
    directions = [(1,2), (1, -2), (2,1), (2, -1), (-2, 1), (-2, -1), (-1, 2), (-1,
-2)]
    for x, y in directions:
        if 0<=x+col< 8 and 0<=lig+y<8:
            moves.append((y+lig, x+col))
    return moves
```

pour le pion, j'ai simplement verifie les coordonnes, et soustrait -2 sur y si il est encore place correctement, sinon soustrait seulement par -1

```
def cases_pion(col,lig):
        if lig == 6:
        return [(col, lig-1), (col, lig-2)]
    return [(col, lig-1)]
```

Parcours

les 2 premiers parcours sont realisable facilement, jvais juste mettre le code pour les effectuer `def parcours_ligne(n):`

```
        i = 0
    result = []
    while i<n:
        j=0
        while j<n:
            result.append((j,i))
            j+=1
        i+=1
    print(result, n)
    return result
```

```
def parcours_colonne(n):
        i = 0
    result = []
    while i<n:
        j=0
        while j<n:
            result.append((i, j))
            j+=1
        i+=1
    return result
```

on itere juste par colonne ou par ligne

Pour sinusoide et zigzag, tant que la ligne/colonne a pas ete parcouru, on itere dans la colonne/ligne, a la fin de la boucle interieur, on soustrait pour avoir une boucle qui parcours a l'envers. Les 2 fonctions sont similaire, il suffit seulement de changer les variables

```
def parcours_sinusoidal(n):
    result = []
    i = 0
    j = 0
    direct = 1
    while j<=n:
        while 0<=i<n and 0<=j<n:
            result.append((j,i))
            i+=direct
        direct = -direct
        i+=direct
        j+=1
    print(result)
    return result


def parcours_zigzag(n):
    result = []
    i = 0
    j = 0
    direct = 1
    while j<=n:
        while 0<=i<n and 0<=j<n:
            result.append((i,j))
            i+=direct
            print(result)
        direct = -direct
        i+=direct
        print()
        j+=1
    print(result)
    return result
```

```
def parcours_serpentin(n):
        result = []
    i = 0
    j = 0
    inc = 0
    mini = 0
    while inc<n**2:
     while i<n-1:
         print(i,j)
         result.append((i,j))
         inc+=1
         i+=1
     print('premiere')
     while j<n-1:
         result.append((i,j))
         print(i,j)
         inc+=1
         j+=1
     print('2ieme')
     while i>mini:
         result.append((i,j))
         print(i,j)
         inc+=1
         i-=1
     print('3em')
     mini+=1
     while j>mini:
         result.append((i,j))
         print(i,j)
         j-=1
         inc+=1

     print('fin')
     n-=1
    return result
```

Puissance 4