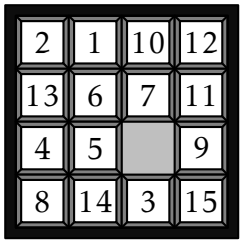
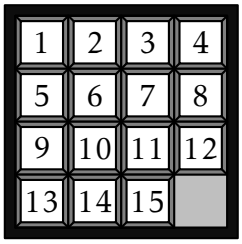
Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI 1re Année **2022-23**

Projet Python – Le taquin

Le taquin

Ce jeu est formé de plusieurs carreaux glissant les uns contre les autres. Chaque carreau contient un nombre ou un bout d'image et l'objectif est de reformer l'image ou de mettre les nombres dans l'ordre.





Nous allons reproduire ce jeu en Python. Le joueur pourra contrôler au clavier le déplacement au clavier.

Préparatifs

Pour simplifier la programmation du jeu, nous allons utiliser les constantes suivantes :

```
import random

N = 3    # grille N*N
LONGUEUR_MAX = len(str(N**2-1))  # plus grand nombre
TOUCHE_HAUT = 'z'
TOUCHE_BAS = 's'
TOUCHE_GAUCHE = 'q'
TOUCHE_DROITE = 'd'
DIRECTIONS = [TOUCHE_HAUT, TOUCHE_BAS, TOUCHE_GAUCHE, TOUCHE_DROITE]
```

Le plateau sera représenté par un tableau de dimensions $N \times N$ et chaque case sera un entier entre 1 et $N^2 - 1$, ou 0 pour la case vide. La variable `N` pourrait être donnée en paramètre des fonctions, mais la mettre ainsi en constante globale permet d'alléger la syntaxe des fonctions. La valeur `LONGUEUR_MAX` correspond au nombre de caractères nécessaires pour afficher la plus grande valeur du plateau de jeu. Vous pouvez l'utiliser pour l'affichage de votre plateau.

Exercice 1 : Écrire une fonction `generer_plateau()` qui génère un tableau $N \times N$ avec les cases numérotées de gauche à droite et de haut en bas. Le 0 se trouve en bas à droite.

```
>>> generer_plateau()    # avec N = 3
[[1, 2, 3], [4, 5, 6], [7, 8, 0]]
>>> generer_plateau()    # avec N = 4
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
```

La fonction suivante permet de faire un affichage minimal:

```
def afficher_plateau(plateau):
    for l in plateau:
        print(" ".join([str(v).rjust(LONGUEUR_MAX) for v in l]))
```

```
>>> afficher_plateau(generer_plateau())
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  0
```

Exercice 2 : Améliorer la fonction `afficher_tableau(plateau)` pour avoir un affichage plus complexe. Voici un exemple possible :

```
>>> afficher_plateau(generer_plateau())
+----+----+----+----+
|  1 |  2 |  3 |  4 |
+----+----+----+----+
|  5 |  6 |  7 |  8 |
+----+----+----+----+
|  9 | 10 | 11 | 12 |
+----+----+----+----+
| 13 | 14 | 15 |    |
+----+----+----+----+
```

Déplacement des carreaux

Exercice 3 : Écrire une fonction `echanger(plateau, i1, j1, i2, j2)` qui échange les valeurs de `plateau[i1][j1]` et de `plateau[i2][j2]`.
Afin de pouvoir remettre les cases dans l'ordre, il faut déplacer la case vide. Les coordonnées de cette case sont donc très importantes. Pour la suite, les variables `i0` et `j0` correspondent à la ligne et à la colonne de la case vide, allant de 0 à $N-1$.

Exercice 4 : Écrire une fonction `bouger(plateau, direction, i0, j0)` qui modifie le plateau en déplaçant la case vide dans la direction indiquée, et renvoie la nouvelle position de la case vide. Si le déplacement est impossible, le plateau n'est pas modifié et la case ne bouge pas. Les directions sont `TOUCHE_HAUT`, `TOUCHE_BAS`...

```
>>> plateau = generer_plateau()
>>> bouger(plateau, TOUCHE_HAUT, N-1, N-1)
(2, 3)
>>> afficher_plateau(plateau)
+----+----+----+----+
|  1 |  2 |  3 |  4 |
+----+----+----+----+
|  5 |  6 |  7 |  8 |
+----+----+----+----+
|  9 | 10 | 11 |    |
+----+----+----+----+
| 13 | 14 | 15 | 12 |
+----+----+----+----+
```

Exercice 5 : Écrire une fonction `melanger(plateau, nb, i0, j0)` qui fait `nb` déplacements de la case vide dons les coordonnées sont données en paramètres. La fonction renvoie la position finale de la case vide. Votre fonction peut tirer un déplacement au hasard, même s'il n'est pas possible, ou alors s'assurer que le déplacement est possible et n'annule pas celui fait juste avant.
Attention, il ne faut pas échanger les cases au hasard, sinon il y a une chance sur deux que le plateau obtenu soit impossible à résoudre. Il faut impérativement déplacer la case vide.

Le jeu

Exercice 6 : Écrire une fonction `jouer()` qui génère un tableau, le mélange et laisse le joueur indiquer dans quelle direction il veut déplacer la case vide. Vous pouvez laisser le joueur continuer à l'infini ou vous pouvez tester s'il a réussi à remettre toutes les cases dans l'ordre.

```
>>> jouer()
+---+---+---+
|   | 7 | 1 |
+---+---+---+
| 8 | 3 | 2 |
+---+---+---+
| 6 | 4 | 5 |
+---+---+---+
d
+---+---+---+
| 7 |   | 1 |
+---+---+---+
| 8 | 3 | 2 |
+---+---+---+
| 6 | 4 | 5 |
+---+---+---+
d
+---+---+---+
| 7 | 1 |   |
+---+---+---+
| 8 | 3 | 2 |
+---+---+---+
| 6 | 4 | 5 |
+---+---+---+
s
+---+---+---+
| 7 | 1 | 2 |
+---+---+---+
| 8 | 3 |   |
+---+---+---+
| 6 | 4 | 5 |
+---+---+---+
...
```