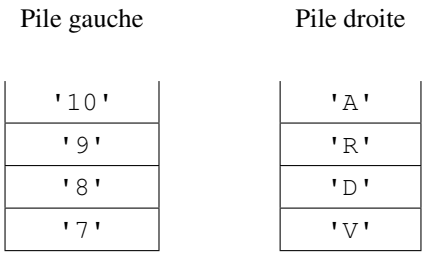
Centres étrangers - juin 2021 - sujet 2 (corrigé)

Exercice 1 (Piles)

1. On obtient tout d'abord les deux piles suivantes :

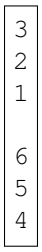


D'où la nouvelle liste : `['10','A','9','R','8','D','7','V']`

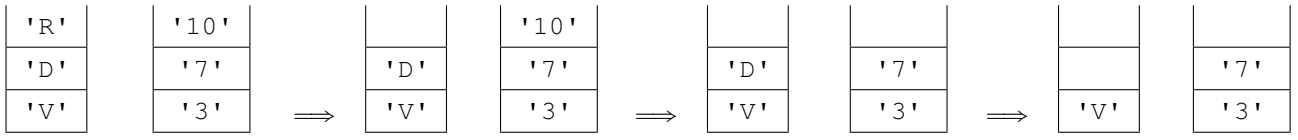
2. La fonction suivante convient :

```
def liste_vers_pile(L):
    """prend en paramètre une liste et renvoie une pile"""
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        p_temp.empiler(L[i])
    return p_temp
```

3. On obtient l'affichage suivant :



4. (a) On a le schéma suivant :

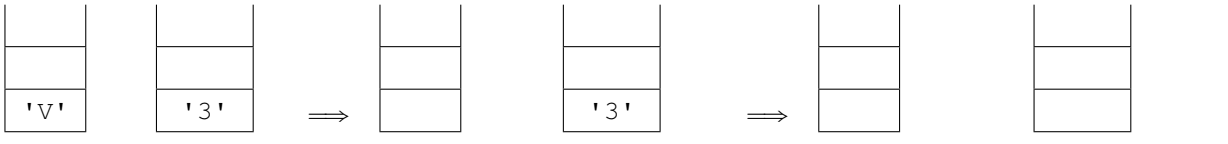


Etape 0 : liste initiale vide `[]`

Etape 1 : on dépile `'R'` `['R']`

Etape 2 : on dépile `'10'` `['R','10']`

Etape 3 : on dépile `'D'` `['R','10','D']`



Etape 4 : on dépile `'7'` `['R','10','D','7']`

Etape 5 : on dépile `'V'` `['R','10','D','7','V']`

Etape 6 : on dépile `'3'` `['R','10','D','7','V','3']`

(b) La fonction suivante convient :

```
def fusion(p1, p2):
    L = []
    while not p1.est_vide():
        L.append(p1.depiler())
        L.append(p2.depiler())
    return L
```

5. La fonction suivante convient :

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('_____')
    else:
        elt = p_temp.depiler()
        print('| ', elt, ' |')
        affichage_pile(p_temp)
```

Exercice 2 (Tableaux)

1. La fonction suivante convient :

```
def mur(l,i,j):
    return l[i][j]==1
```

2. (a) La variable `d` représente le carré de la distance entre entre les cases `case1` et `case2`. Deux cases adjacentes ont une distance égale à 1. Donc `d==1` (et donc la fonction `voisine`) renvoie `True` si `case1` et `case2` sont adjacentes et `False` dans le cas contraire.

   (b) La fonction suivante convient :

```
def adjacentes(l):
    adj = True
    n = len(l)
    for i in range(n-1):
        if not voisine(l[i], l[i+1]):
            adj = False
    return adj
```

3. Avant le début de la boucle `while` la variable `i` prend la valeur `0`. A chaque itération de la boucle la variable `i` est incrémentée d'une unité (`i = i + 1`) . La boucle va donc « s'arrêter » quand `i = len(cases)`. Comme le tableau `cases` n'est pas infini, la boucle va donc obligatoirement se terminer.

4. La fonction suivante convient :

```
def echappe(cases, laby):
    nb_lig = len(laby)
    nb_col = len(laby[0])
    return cases[0]==(0,0) and cases[len(cases)-1]==(nb_lig-1,nb_col-1) and teste(cases,laby)
```

Exercice 3 (Booléens et caractères)

1. $89 = 2^6 + 2^4 + 2^3 + 2^0$, donc $89 = (0101\ 1001)_2$
2. On a l'opération suivante :

$$\begin{array}{rr} & 1100\ 1110 \\ \oplus & 0110\ 1011 \\ \hline = & 1010\ 0101 \end{array}$$

3. La fonction suivante convient :

```
def xor_crypt(message, cle):
    lst = []
    for i in range(len(message)):
        n = xor(ord(message[i]), ord(cle[i]))
        lst.append(n)
    return lst
```

4. La fonction suivante convient :

```
def generer_cle(mot,n):
    cle = ""
    long_mot = len(mot)
    long_n = len(n)
    div = long_mot // long_n
    reste = long_mot % long_n
    for i in range(div):
        cle = cle + n
    for i in range(reste):
        cle = cle + n[i]
    return cle
```

5. On a le tableau suivant :

A	B	A ⊕ B	(A ⊕ B) ⊕ B
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

On remarque que `A ⊕ B) ⊕ B` correspond à `A`.
Soit le message d'origine `m`, soit le message chiffré `m'` et soit la clé `k`.
Pour obtenir `m'` il faut faire un `m ⊕ k`.
Pour obtenir `m` à partir de `m'` et `k`, il suffit donc de faire un `m' ⊕ k` (`m'` correspond à `A ⊕ B`, `m` correspond à `A` et `k` correspond à `B`).

Exercice 4 (SQL)

1. (a) L'attribut `nom` de la table `licencies` ne peut pas servir de clé primaire, car il peut exister des homonymes et que la clé primaire doit être unique.
   (b) L'attribut `id_licencie` peut jouer le rôle de clé primaire.
2. (a) Cette requête renvoie le nom et le prénom des licenciés qui jouent dans l'équipe des `-12 ans`.
   (b) En utilisant `*` à la place de `prenom, nom`, on obtient l'ensemble des attributs.
   (c) La requête suivante convient :

```
SELECT date
FROM matchs
WHERE equipe = 'Vétérans' AND lieu = 'Domicile'
```

3. La requête suivante convient :

```
INSERT INTO licencies(id_licencie, prenom, nom, annee_naissance, equipe)
VALUES(287, 'Jean', 'Lavenu', 2001, 'Hommes 2')
```

4. La requête suivante convient :

```
UPDATE Licencies
SET equipe = 'Vétérans'
WHERE prenom = 'Joseph' AND nom = 'Cuviller'
```

5. La requête suivante convient :

```
SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE adversaire = 'LSC' AND date = 2021-06-19
```

Exercice 5 (POO)

1. (a) `Obj_bandeau.get_pixel_rgb(1)` renvoie le tuple `(0, 0, 255)` car LED1 est bleue.
   (b) `Adafruit_WS2801.RGB_to_color(0, 0, 255)` renvoie l'entier `16711680` (`num_color` du bleu).
   (c) Ces deux instructions permettent d'afficher le code de la couleur la led LED0, c'est-à-dire `255`.
2. (a) On a le tableau suivant :

bleu	bleu	bleu	bleu	bleu	blanc	blanc	blanc	blanc	blanc	rouge	rouge	rouge	rouge	rouge

   (b) On a le tableau suivant :

vert	jaune	jaune	vert	jaune	jaune	vert	jaune	jaune	vert	jaune	jaune	vert	jaune	jaune

3. (a) La chaîne de documentation suivante convient :

```
"""
Initialise une instance de la classe Bandeau.
Elle prend en paramètre le nombre de leds utilisées dans le bandeau.
"""
```

   (b) Le commentaire suivant convient :

```
# Permet de faire passer les leds 6 et 7 au bleu.
```