Bien que le programme se limite aux tableaux de taille fixe, les exercices marqués (app) nécessitent d'ajouter des éléments progressivement, ce qu'on fera en utilisant `r.append(valeur)` pour ajouter un élément à la fin d'un tableau `r`, plutôt que `r = r+[valeur]`.

Quelques erreurs à éviter en manipulant un tableau :

- se tromper dans les indices. En particulier, il ne faut pas oublier que les indices commencent à 0 et terminent à `len(t)-1` ;
- penser qu'on manipule des « copies » d'éléments alors qu'on manipule l'élément original à travers un alias. Cette erreur se rencontre avec tous les éléments muables, pas seulement les tableaux.

1 Parcours de tableau par indice

Soit `t = ['a','b','d','z']`.

Que retournent t[1], t[4], t[-1], t[-4] et t[0] ?

2 Parcours de matrice par indice

Soit `m = [['a','b'], ['d','z']]`.

a. Comment accède-t-on à la case en bas à gauche (celle qui vaut ici 'd') dans cette matrice 2 × 2 ?

b. Compléter le code suivant pour qu'il affiche les 4 éléments de `m` : 'd a b z', dans cet ordre : `print(… , … , … , … )`.

c. Compléter le code suivant pour que la fonction affiche les dimensions de `m`. Ces dimensions doivent être calculées à partir de `m`.

```
def affiche_dimensions(mat):
    """ affiche les dimensions de mat """
    print(f'{...} lignes et {...} colonnes')
# affiche : 2 lignes et 2 colonnes pour m
# 1 lignes et 3 colonnes pour [[2,3,5]]
```

d. Compléter le code `t…` pour qu'il renvoie la valeur en bas à gauche d'une matrice `t`.

3 Parcours de matrice par ligne

Écrire une fonction `m_l` qui prend en paramètres une matrice `m` et un entier `i`, et retourne la *i*-ième ligne de `m` si elle existe, `None` sinon.

Par exemple : `m_l([['a','b'], ['d','z']],1)` retourne `['d','z']`.

4 Une fonction mystérieuse

a. Que calcule la fonction suivante ?

```
def inconnu(x, m):
    tot = 0
    for i in len(m):
        tot = tot + m[i][i]
    return tot
```

b. Dans le code ci-dessous quelle instruction manque pour obtenir `[[0,1,1,1],[0,0,1,1],[0,0,0,1],[0,0,0,0]]` à partir d'une matrice m de dimensions 4 × 4 « remplie de » 0 ?

```
for i in range(4):
    for j in range(4):
        if ... :
            m[i][j] = 1
```

a `i < j`	b `i > j`	c `i+j == 4`
d `i-j == 4`	e `i-j == 3`	f `i+j == 3`

c. Même question pour obtenir `[[0,0,0,1], [0,0,1,0], [0,1,0,0], [1,0,0,0]]`.

d. Reécrire les fonctions b. et c. pour ne pas parcourir d'éléments inutilement, en éliminant ou modifiant l'énumeration `for j …`.

5

Écrire (sans utiliser la fonction Python `sum`) une fonction qui prend en paramètre un tableau et retourne le total de ses éléments.

L'adapter pour retourner une moyenne.

6

Écrire (sans utiliser la fonction Python `sum`) une fonction qui prend en paramètre une matrice et retourne la somme des éléments de la matrice.

7

Écrire (sans utiliser `min`) une fonction qui prend en paramètre un tableau et retourne son plus petit élément.

8 Recherche d'occurrence (app)

a. Écrire une fonction `trouve(v,t)` qui prend en paramètres un tableau `t` et une valeur `v`, et retourne la liste des indices des éléments de `t` égaux à `v`.

b. Écrire une fonction `compte(v,t)` qui prend en paramètres un tableau `t` et une valeur `v`, et retourne le nombre d'éléments égaux à `v`.

c. Même question que a. avec une matrice, les indices sont donc des paires d'entiers.

9 Compréhension et fonction range

a. Que vaut `[x*x for x in range(6,-1,-2)]` ? Le résultat et calcul sont-ils les mêmes que `[x*x for x in range(6,-1,-1) if x%2==0]` ?

b. Que vaut `[x*x for x in range(6,-1,2)]` ?

10 Construire un tableau par compréhension

a. Écrire un tableau par compréhension qui comprend tous les nombres pairs inférieurs à 100.

b. Écrire un tableau par compréhension qui comprend les 100 premiers nombres pairs.

11

Écrire un tableau par compréhension qui comprend les 52 cartes d'un jeu.

Chaque carte est un tuple `(valeur, couleur)` où la valeur est un entier entre 2 et 13 (13 étant la valeur de l'as) et la couleur est `"trèfle"`, `"carreau"`, `"cœur"`, `"pique"`.

12 Compréhension avec condition

Quelle syntaxe renvoie les éléments de `t` supérieurs à 5 ?

a. `[x if x>5 for x in t]`

b. `[x for x in t if x>5]`

13 Filtrer un tableau en itérant

Compléter le code suivant pour que `res` enregistre les indices des nombres de `t` compris entre 10 et 100 (inclus).

```
t = [2,100,54.2,10]
res = []
for x ... :
    if x >= 10 and ...:
        res.append(x)
```

14 Filtrer un tableau par compréhension

Soit `t = [1,2,4,6]`.

a. Construire élégamment un tableau `t_pair` contenant les éléments pairs de `t`.

b. Construire un tableau `t_pair_idx` contenant l'indice de chaque élément pair de `t`.

15 Copier un tableau `t0` d'entiers

Créer une copie (indépendante) d'un tableau d'entiers `t0` : modifier `t0` ne doit pas affecter `t`.

16 Min de fonction sur un tableau

Écrire une fonction `minimum(t,f)` qui prend en paramètres un tableau `t` et une fonction `f`, et retourne l'indice de l'élément de `t` qui minimise `f` (le premier s'il y en a plusieurs).

17 Max, avec les indices

a. Écrire une fonction `maxi(t)` qui renvoie l'indice du maximum du tableau `t`. Le premier s'il y en a plusieurs.

b. Même question, mais avec le dernier indice.

c. Même question, mais en renvoyant tous les indices auxquels se trouve la valeur max. (app)

18 Copier une matrice

Écrire une fonction qui crée une copie (indépendante) d'une matrice d'entiers `t0`.

19 Autre fonction mystérieuse

Que calcule la fonction suivante ?

```
def inconnu(x, m):
    l = []
    for (i,a) in enumerate(m):
        for (j,b) in enumerate(a):
            if x == b:
                l.append((i,j))
    return l
```

20 Échange d'éléments

Construire une fonction qui prend en paramètres un tableau `t`, deux indices `i` et `j`, et échange `t[i]` avec `t[j]` (la fonction affichera «impossible» si les indices `i` ou `j` n'existent pas – on n'acceptera pas les indices négatifs).

21 Parcours de matrice : par ligne et colonne

a. Écrire une fonction `m_c` qui prend en paramètres une matrice `m` et un entier `i`, et qui retourne la `i`-ième colonne de `m` si elle existe, `None` sinon.

Par exemple, `m_c([['a','b'], ['d','z']],0)` retournera `['a','d']`.

b. Écrire une fonction `cherche_m` qui prend une matrice de chaines `m` et une chaîne de caractères `s`, et affiche, pour chaque case égale à `s` les indices de cette case, la ligne qui la contient, et la colonne qui la contient.

Par exemple, `cherche_m([['aa','b'], ['d','z'], ['aab','aa']],'aa')` affichera (sur 2 lignes) :

```
0 0 ['aa', 'd', 'aab']
2 1 ['b', 'z', 'aa']
```

22 Remplir une matrice

Écrire un programme qui construise la matrice illustrée ci-dessous :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

23 Puissance 4

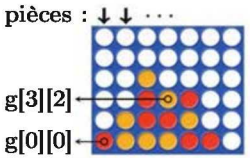
Des indications sont données dans le fichier `puissance4.py` pour b. et c. On peut représenter le jeu *Puissance 4* par une matrice g où $g[i][j]$ représente le contenu de la colonne j dans la rangée i : *None* pour vide, 0 pour jaune et 1 pour rouge. Chaque joueur fait tomber une pièce de sa couleur à tour de rôle dans la colonne de son choix. Le premier joueur à aligner (à l'horizontale,

verticale ou en diagonale) 4 pièces de sa couleur gagne. Sur la figure, $g[0][0] = 1$ et $g[3][2] = 1$, et si *jaune* pose une pièce dans la 2e colonne alors *rouge* peut gagner en posant une pièce en $g[1][2]$.

a. Écrire une fonction qui prend en entrée une matrice g et 2 entiers i, j, et vérifie si g contient un alignement horizontal partant de $g[i][j]$ vers la droite (4 valeurs non vides identiques).

b. En déduire une fonction qui prend en entrée une matrice g et détermine si elle contient au moins un alignement.

c. (Titan) On suppose que c'est à *jaune* de jouer. Écrire une fonction qui renvoie la colonne dans laquelle il peut poser une pièce pour gagner immédiatement s'il y en a une, et –1 sinon.



24 Liste de caractères

Écrire (sans utiliser `join`) une fonction qui convertit un tableau de caractères en une chaîne de caractères.

25 Choix aléatoire

La fonction Python `random.randint(a,b)` renvoie un entier aléatoire parmi `a`, `a+1`,..., `b`. Compléter le code ci-dessous pour que `f` renvoie un élément aléatoire du tableau `t`.

```
from random import randint
def f(t):
    i = randint(...,...)
    return ...
```

26 Inversion de tableau

Écrire (sans utiliser la fonction Python `reverse`) une fonction qui prend en entrée un tableau `t` de nombres et retourne un autre tableau contenant les éléments de `t` dans l'ordre inverse.

27 Les vaches de Narayana

Une vache donne naissance à une autre tous les ans en début d'année, qui elle-même donne naissance à une autre chaque année à partir de sa quatrième année.

- Partant d'une vache qui vient de donner naissance, créer un tableau `t` de taille 20 tel que `t[i]` soit le nombre de vaches à la fin de l'année `i` (aucune ne mourant entre temps).

● NOTE

Ce problème est du au mathématicien indien Pandit Narayana (1340-1400). Un problème similaire a été proposé par l'italien Leonardo Fibonacci dans son Liber abaci en 1202 concernant la reproduction des lapins. La suite de Fibonacci se retrouve beaucoup plus fréquemment dans la nature et les œuvres d'arts. Les mathématiciens indiens étudiaient déjà ces équations récurrentes quelques siècles av J.-C. pour des problèmes de combinatoire grammaticale (compter le nombre de manières d'alterner syllabes longues et courtes dans une phrase de longueur donnée).

28 Suite de Syracuse

La suite de Syracuse `u(n)` d'un entier `x` est définie comme suit : `u(0) = x`, et pour tout `n>=0`, `u(n+1) = u(n)//2` si `u(n)` est pair, et `3*u(n)+1` sinon.

Écrire une fonction qui prend en paramètre un entier `u0` et retourne dans un tableau les 100 000 premières valeurs de la suite de Syracuse de `u0`.

29 Somme de matrices

Écrire une fonction qui prend en paramètres deux matrices rectangulaires de mêmes dimensions et retourne la somme des deux.

Pour les entrées `[[1,3,7],[2,4,5]]` et `[[0,0,3],[6,4,20]]`, elle renvoie `[[1,3,10],[8,8,25]]`.

30 Tableau de fonctions

On donne les instructions suivantes.

```
from math import sin, cos, tan, exp
tab = [sin,cos,tan,exp]
```

a. Que retourne `tab[0](0)` ?

b. Même question pour `[f(0) for f in tab]`.

31 Counting sort

Écrire une fonction qui prend un tableau `t` d'entiers entre 0 et 100, et retourne pour chaque entier combien de fois il apparait. Il est demandé un programme efficace qui ne parcourt le tableau qu'une fois. On appelle cet algorithme linéraire le *tri comptage* (*counting sort* en anglais).

32 Crible d'Eratosthène

Implémenter le crible d'Eratosthène en suivant les indications du fichier `eratoshene.py`.

33 Calcul de dénivelé en randonnée

Un randonneur est équipé d'un capteur enregistrant toutes les minutes l'altitude (en nombre – pas forcément entier – de mètres) à laquelle il se trouve.

- Compléter le code du fichier `denivele.py` pour calculer le dénivelé total, la plus longue montée et son dénivelé.

34 Run-length encoding (app)

Une des techniques pour « compresser » un tableau est le codage par plages (en anglais RLE pour *Run-Length Encoding*). Il s'agit d'indiquer pour chaque suite de valeurs identiques le nombre de fois (au moins 1) que cette valeur est répétée. On se retrouve alors avec un tableau de paires (entier, valeurs) généralement plus court que le tableau initial. Écrire une fonction renvoyant le codage par plages d'un vecteur.

Par exemple, `t = ['aa','aa','b','c','c','c','aa']` sera représenté par `[(2,'aa'),(1,'b'),(3,'c'),(1,'aa')]`.

● NOTE

RLE est surtout utilisée pour comprimer des images, dans lesquelles il est fréquent que des pixels voisins aient des valeurs similaires. Le format JPEG utilise ainsi RLE en combinaison avec d'autres techniques.

35 Vecteurs creux

Une des techniques pour « compresser » des tableaux ou matrices *creux*, c'est-à-dire contenant une forte proportion de 0 (*sparse matrices* en anglais) est de n'indiquer que la liste des valeurs non nulles avec leur index.

a. Écrire une fonction qui prend un tableau et le représente comme un vecteur creux : `v = [0,0,4,0,0,0,5,6]` sera représenté par `[(2,4),(6,5),(7,6)]`.

b. Coder un encodage analogue pour les matrices, mais cette fois sous forme d'un dictionnaire : `m = [[0,4],[5,6]]` sera représenté par `{(0,1):4,(1,0):5,(1,1):6}`. Pour un code élégant, on notera qu'il est possible d'imbriquer des compréhensions : `{x for ligne in m if True for x in ligne if True}`.

● NOTE

Pour aller plus loin : On utilise implicitement cet encodage lorsque l'on représente les voisinages dans un graphe par des listes plutôt que des matrices. On rencontre aussi fréquemment des matrices creuses en intelligence artificielle, ou lorsque l'on applique la méthode des éléments finis.

36 Delta encoding

Une des techniques pour « compresser » un tableau est le codage par différence (en anglais *Delta encoding*). Il s'agit d'indiquer pour chaque valeur sa différence avec la valeur précédente (plutôt que la valeur elle-même). On se retrouve alors avec un tableau de valeurs assez petites donc nécessitant moins de place en mémoire.

- Écrire une fonction renvoyant l'encodage par différence d'un vecteur. Par exemple, `[1000,800,802,1000,1003]` sera représenté par `[1000,-200,2,198,3]`.

● NOTE

Lorsque les valeurs varient peu, les différences sont petites et peuvent donc être représentées en utilisant peu de bits. Des variations de cette idée sont mises en œuvre dans le protocole HTTP, JPEG, les logiciels de gestion de versions, etc.

37 Jeu d'échec (app)

On modélise par une matrice 8×8 une fin de partie d'échec sans roques. Les cases de la matrice contiennent un tuple (couleur, pièce), comme par exemple `('noir','cavalier')`, sauf les cases vides qui contiennent None.

- Compléter le code du fichier `jeu-echec.py` pour définir le mouvement des pièces, vérifier si un plateau correspond à une situation d'échec ou d'échec et mat. Le fichier se contente bien sûr de modéliser des cas « simples » et non pas une partie entière.

38 Collisions dans le plan

On donne dans un tableau la position, l'orientation (Nord, Sud, Est ou Ouest) et la vitesse de voitures sur un plan routier en forme de grille (comme Manhattan). Deux voitures se croisent à une intersection si elles y arrivent exactement au même moment. Les voitures, considérées comme de simples points, peuvent se doubler sans ralentir.

- Compléter le code du fichier `croisements.py` pour identifier les voitures qui se croisent.

39 Le chiffre de César

Compléter le fichier `cesar.py` pour coder un texte en utilisant le chiffrement de César.

40 Le chiffre de Vigenère

Compléter le fichier `vigenere.py` pour coder un texte en utilisant le chiffrement de Vigenère.

41 Dessiner une image

a. Construire une matrice pixel par pixel représentant le drapeau "Kroaz du" (sans les mouchetures d'hermine). Vous pourrez l'afficher en utilisant la fonction `affiche` du TP dithering.