Solution exercice

- **Question 1** – Créer un vecteur de valeurs nulles de taille 10.

```
a = np.zeros(10)
```

- **Question 2** – Créer un vecteur de valeurs nulles de taille (10, 10).

```
a = np.zeros((10, 10))
```

- **Question 3** – Créer un vecteur de valeurs nulles de taille 10 et remplacez la 5ème valeur par un 1.

```
a = np.zeros(10)
a[4] = 1
```

- **Question 4** – Créer un tableau de taille 50 et l'inverser (le premier élément deviendra le dernier).

```
a = np.random.randint(10, size=50)
a[::-1]
```

- **Question 5** – Créer une matrice 3×3 avec des valeurs allant de 0 à 8 (Indices : reshape, shape ...).

```
a = np.arange(9).reshape((3,3))
```

- **Question 6** – Créer une matrice identité de taille 3×3 (Indices : [https://fr.wikipedia.org/wiki/Matrice_identit%C3%A9] et documentation de *np.eye*).

```
a = np.eye(3)
```

- **Question 7** – Créer une matrice de dimension $3 \times 3 \times 3$ avec des valeurs aléatoires.

```
a = np.random.randint(10, size=(3, 3, 3))
```

- **Question 8** – Créer une matrice de dimension 10×10 avec des valeurs aléatoires et extraire le min, le max, la moyenne (*mean*), l'écart-type (*std* pour standard deviation) et le shape.

```
a = np.random.randint(20, size=(10, 10))
mean = np.mean(a)
std = np.std(a)
shape = a.shape
```

- **Question 9** – Créer une matrice 2D avec des 1 sur les bords et des 0 à l'intérieur. Exemple :

$$m = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

```
a = np.ones((10,10), dtype=int)
a[1:-1,1:-1] = 0
```

- **Question 10** – Créer une matrice aléatoire m de taille 10×10 et normalisez là avec la formule suivante :

$$n = \frac{m - \bar{m}}{\sigma(m)}$$

où $\bar{m}$ est la moyenne de m et $\sigma(m)$ l'écart-type de m.

```
m = np.random.randint(20, size=(10, 10))
n = (m - np.mean(m))/np.std(m)
```

- **Question 11** – Faire la multiplication d'une matrice 5×3 par une matrice 3×2. Quelle est la taille de la matrice résultante ?

```
a = np.random.randint(20, size=(5, 3))
b = np.random.randint(20, size=(3, 2))
np.dot(a,b).shape
```

- **Question 12** – Générer une matrice 1D avec des valeurs allant de 0 à 15 et rendre négatifs ($\times -1$) tous les nombres entre 5 et 12.

```
a = np.arange(16)
a[5:12] *= -1
```

- **Question 13** – Générer une matrice 2D avec des valeurs allant de 0 à 30 de taille 5×6 et rendre négatifs ($\times -1$) tous les nombres de l'avant dernière ligne (et uniquement de cette ligne là !).

```
a = np.arange(30).reshape((5,6))
a[-1,:] *= -1
```

- **Question 14** – Générer deux matrices 2D de taille 1000×1000 et faire la somme en parcourant chaque élément des deux matrices avec une boucle (boucle sur les lignes et colonnes) et en utilisant Numpy. Comparer les temps d'exécution des deux méthodes.

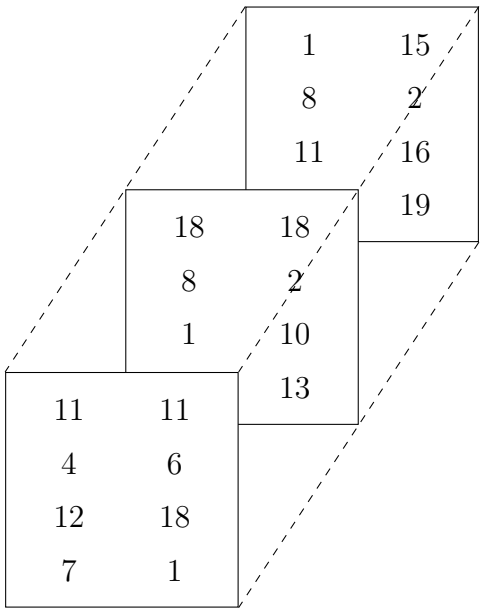
```
import time
start_time = time.time()
...
end_time = time.time()
```

```
a = np.random.randint(20, size=(1000, 1000))
b = np.random.randint(20, size=(1000, 1000))

#Boucle
assert a.shape == b.shape
c = np.zeros(a.shape, dtype=int)
start_time_loop = time.time()
for i in range(0, a.shape[0]):
    for j in range(0, a.shape[1]):
        c[i, j] = a[i, j] + b[i, j]
end_time_loop = time.time()
print('Time loop : ' + str(end_time_loop - start_time_loop) + '
   secondes')

#Matrix
assert a.shape == b.shape
start_time_matrix = time.time()
d = a + b
end_time_matrix = time.time()
print('Time matrix : ' + str(end_time_matrix - start_time_matrix) + '
   secondes')
```

- **Question 15** – Générer un tableau d'entiers de 845×893 avec des valeurs allant de 0 à 20 et compter le nombre d'occurrence de chaque élément du tableau. (Indice : regarder du coté de np.unique ...)

```
a = np.random.randint(20, size=(845, 893))
unique, counts = numpy.unique(a, return_counts=True)
dic_occurence = dict(zip(unique, count))
print(dic_occurence)
```

- **Question 16** – Générer trois tableau d'entiers de 5×5 avec des valeurs entières allant de 0 à 20. Faire un stack de ces tableaux dans un seul est unique tableau de dimension 3. Ci-dessous un exemple de tableau de dimension 3 :




```
a = np.random.randint(20, size=(5, 5))
b = np.random.randint(20, size=(5, 5))
c = np.random.randint(20, size=(5, 5))

d = np.zeros((a.shape[0], a.shape[1], 3), dtype=int)

d[:,:,0] = a
d[:,:,1] = b
d[:,:,2] = c
```

- **Question 17** – Générer une matrice de 1000×1000 avec des entiers aléatoires entre 0 et 50, la normaliser (cf. Question 10) et la binariser (tous les éléments supérieurs à 0.5 prennent la valeur 1, sinon ils prennent la valeur 0). Ne pas utiliser de boucles ! (Indices pour la normalisation : np.where).

```
a = np.random.randint(50, size=(1000, 1000))
a = (a - np.mean(a))/np.std(a)
a = np.where(a > 0.5, 1, 0)
```