

Page	Possible	Score
1	12	
2	8	
Total:	20	

UE12P26 – PE

Python numérique - Quiz du cours #2

29 septembre 2026

Cochez une seule réponse par question.

Ce quiz porte sur la séance 2 (NumPy : vectoriser, agréger, diffuser, masquer). Les questions 1 à 12 sont accessibles, les questions 13 à 20 sont plus techniques : un brouillon peut aider. Vous disposez de 10 minutes pour répondre aux 20 questions.

1. Soit `A` une matrice (tableau 2D). Quelle expression calcule la somme de **chaque ligne** ?

- ☐ `A.sum(axis=0)`
☐ `A.sum()`
☐ `A.sum(axis=1)`
☐ `np.sum(A, rows=True)`

2. Soit `tab` de forme (2, 3, 5). Quelle est la forme de `tab.sum(axis=0)` ?

- ☐ (2, 3)
☐ (3, 5)
☐ (2, 5)
☐ (2, 3, 5)

3. Quel masque sélectionne les éléments de `t` strictement négatifs **ou** strictement supérieurs à 100 ?

- ☐ `(t < 0) | (t > 100)`
☐ `(t < 0) or (t > 100)`
☐ `t < 0 | t > 100`
☐ `(t < 0) || (t > 100)`

4. Que renvoie `np.where(t > 0, t, 0)` ?

- ☐ Les indices des éléments strictement positifs de `t`
☐ Un booléen : vrai si tous les éléments sont positifs
☐ Rien : `t` est modifié en place
☐ Un nouveau tableau où les éléments positifs sont conservés et les autres remplacés par 0

5. Pour un tableau 2D `t`, que renvoie `t.argmax()` (sans `axis`) ?

- ☐ L'indice du plus grand élément dans le tableau **aplati**
☐ Un couple (`i`, `j`) de coordonnées
☐ La valeur du plus grand élément
☐ Un tableau d'indices, un par ligne

6. Quelle expression compte le nombre d'éléments de `t` strictement supérieurs à 5 ?

- ☐ `len(t > 5)`
☐ `np.sum(t) > 5`
☐ `(t > 5).sum()`
☐ `t[t > 5].count()`

7. À quoi sert le décorateur `@np.vectorize` appliqué à une fonction `f(x)` écrite pour un scalaire (avec un `if`) ?

- ☐ À compiler `f` en C pour la rendre aussi rapide qu'une `ufunc`
☐ À pouvoir appeler `f` sur un tableau, élément par élément, sans gain de vitesse notable
☐ À convertir automatiquement les listes Python en `ndarray`
☐ À rendre `f` utilisable avec `np.sin` et `np.cos`

8. Laquelle de ces opérations est **valide** grâce au broadcasting ?

- ☐ `np.ones((4, 3)) + np.arange(4)`
☐ `np.ones((2, 3)) + np.ones((3, 2))`
☐ `np.ones(5) + np.ones(4)`
☐ `np.ones((4, 3)) + np.arange(3)`

9. Quelle expression teste si **au moins un** élément de `t` est nul ?

- ☐ `np.all(t == 0)`
☐ `np.any(t == 0)`
☐ `t == 0`
☐ `0 in t.all()`

10. Soit `im` une image RGB de forme (H, W, 3). Quelle expression calcule l'image en niveaux de gris par **moyenne des 3 canaux** (forme (H, W)) ?

- ☐ `im.mean(axis=2)`
☐ `im.mean(axis=0)`
☐ `im.mean()`
☐ `im[:, :, 0] + im[:, :, 1] + im[:, :, 2]`

11. Soit `tab = np.arange(24).reshape(2, 3, 4)`. Que vaut `tab.sum(axis=(0, 1)).shape` ?

- ☐ (2, 3)
☐ (3, 4)
☐ (4,)
☐ () : un seul nombre

12. Dans `np.multiply(t, 2, out=t)`, quel est l'effet du paramètre `out=t` ?

- ☐ Il renvoie une copie de `t` multipliée par 2
☐ Il précise le `dtype` du résultat
☐ Il provoque une erreur car `t` est à la fois entrée et sortie
☐ Le résultat est écrit directement dans `t`, sans créer de tableau temporaire

13. Qu'affiche ce code ?

```
im = np.array([250, 10], dtype=np.uint8)
print(im + 10)
```

- ☐ [260 20]
- ☐ [4 20]
- ☐ [255 20]
- ☐ Une erreur `OverflowError`

14. Soit `im` une image `uint8`. Quelle écriture **éclaircit** l'image de 50 sans dépassement de capacité (les valeurs sont plafonnées à 255) ?

- ☐ `big = im.astype(int) + 50`
`np.clip(big, 0, 255).astype(np.uint8)`
- ☐ `im + 50`
- ☐ `np.clip(im + 50, 0, 255)`
- ☐ `(im + 50) % 255`

15. Qu'affiche ce code ?

```
t = np.arange(6).reshape(2, 3)
t[t % 2 == 1] = -1
print(t)
```

- ☐ `[[0 1 2] [3 4 5]]` (t n'est pas modifié)
- ☐ `[-1 -1 -1]`
- ☐ Une erreur : la forme du masque ne correspond pas
- ☐ `[[ 0 -1 2] [-1 4 -1]]`

16. Qu'affiche ce code ?

```
t = np.array([1, 5, 10])
s = t[t > 3]
s[0] = 0
print(t)
```

- ☐ [1 0 10]
- ☐ [0 5 10]
- ☐ [1 5 10]
- ☐ Une erreur : `s` n'est pas modifiable

17. Que se passe-t-il à l'exécution de ce code ?

```
a = np.ones((3, 4))
b = np.arange(3)
c = a + b
```

- ☐ `ValueError` : les formes `(3, 4)` et `(3,)` ne sont pas compatibles
- ☐ `c` a la forme `(3, 4)`, `b` est ajouté à chaque colonne
- ☐ `c` a la forme `(3, 4)`, `b` est ajouté à chaque ligne
- ☐ `c` a la forme `(3, 3)`

18. Qu'affiche ce code ?

```
t = np.array([[3, 9, 1],
              [7, 2, 8]])
print(t.min(axis=0), t.argmin(axis=1))
```

- ☐ [1 2] [2 1]
- ☐ [3 2 1] [2 1]
- ☐ [3 2 1] [1 2]
- ☐ [1 2] [0 1]

19. Que produit ce code ?

```
t = np.arange(10)
m = t > 2 & t < 8
```

- ☐ Un masque des éléments compris entre 3 et 7
- ☐ Un masque entièrement à `True`
- ☐ Un masque entièrement à `False`
- ☐ Une erreur `ValueError` (« truth value of an array is ambiguous »)

20. Soit `x = np.array([1, 2, 3])`. Que vaut `x.reshape(-1, 1) * x` ?

- ☐ [1 4 9]
- ☐ 14 (le produit scalaire)
- ☐ `[[1 2 3] [2 4 6] [3 6 9]]`
- ☐ Une erreur : formes `(3, 1)` et `(3,)` incompatibles

15. [1] []

16. [1] []

17. [1] []

18. [1] []

19. [1] []

20. [1] []