Commencé le	Saturday 23 March 2024, 15:11
État	Terminé
Terminé le	Saturday 23 March 2024, 16:41
Temps mis	1 heure 30 min
Note	**31,00** sur 40,00 (**77,5**%)

Question **1**

Correct

Note de 1,00 sur 1,00

Retourner les éléments de la diagonale allant d'en bas à gauche jusqu'en haut à droite pour une matrice carrée.

Return the diagonal elements from bottom left to top right for a square matrix.

En 2 lignes

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def antidiag(A):
2     return np.flipud(A).diagonal()
```

	Test	Résultat attendu	Résultat obtenu	
✔	`A = np.random.randint(0,20,size=(5,5))` `print(np.all(antidiag(A) == np.diag(A[::-1, :])))`	`True`	`True`	✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Python3):

```
1 def antidiag(A):
2     return np.diag(A[::-1, :])
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **2**

Incorrect

Note de 0,00 sur 1,00

Retourner la valeur la plus au milieu d'un tableau en 2 dimensions. Si une dimension du tableau est paire, on prend l'indice inférieur (exemple : dim == 4, alors l'indice à prendre est 1).

Return the middlemost value of a 2-dimensional array. If a dimension of the array is even, we take the lower index (example: dim == 4, then the index to take is 1).

En 2 lignes

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def middle(A):
    return A[(len(A) - 1) // 2][(len(A) - 1) // 2]
```

Solution de l'auteur de la question (Python3):

```
def middle(A):
    return A[int((A.shape[0] - 1)/2), int((A.shape[1] - 1)/2) ]
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question **3**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui retourne le nombre d'éléments d'une matrice dont la valeur est multiple de 3.

```
array([[ 3,  7,  8],
       [ 1, -3,  4],
       [ 3,  6, -6]])
```

donne 5

A faire en 2 lignes de 30 caractères max.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def count_tri(A):
    return (A % 3 == 0).sum()
```

Solution de l'auteur de la question (Python3):

```
def count_tri(A):
    return (A%3 == 0).sum()
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **4**

Correct

Note de 1,00 sur 1,00

Écrire la fonction correlation(u,v) qui prend 2 vecteurs et retourne leur corrélation.

La corrélation est la covariance divisée par le produit des écarts types de u et de v. Elle varie entre -1 et 1.

Write the function correlation (u, v) which takes 2 vectors and returns their correlation.

The correlation is the covariance divided by the product of the standard deviations of u and v. It varies between -1 and 1.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 cov = lambda x,y : np.dot((x - x.mean()), (y - y.mean())) / len(x)
2 correlation = lambda x,y: cov(x,y) / (np.std(x) * np.std(y))
3
```

Solution de l'auteur de la question (Python3):

```
1 cov = lambda x,y : np.dot((x - x.mean()), (y - y.mean())) / len(x)
2 correlation = lambda x,y: cov(x,y) / (np.std(x) * np.std(y))
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **5**

Correct

Note de 1,00 sur 1,00

Retourner une liste avec toutes les valeurs du bord d'un tableau 2D. L'ordre des valeurs n'est pas important.

Return a list with all the edge values of a 2D array. The order of the values is not important.

Exemple : si A = np.array([[1,2,3],[4,5,6],[7,7,7]]) alors on retourne [1,2,3,4,6,7,7,7]

En 3 lignes max

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def couronne(A):
    return list(A[[0, -1], :].flatten()) + list(A[1 : -1, [0, -1]].flatten())
```

Solution de l'auteur de la question (Python3):

```
def couronne(A):
    return list(A[[0,-1],:].flatten()) +   list(A[1:-1, [0,-1]].flatten())
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **6**

Correct

Note de 1,00 sur 1,00

Soit un tenseur qui représente une image de taille (n,m) en couleur (RGB). Le tenseur est un tableau de taille (n,m,3). On désire ajouter une couche qui est le niveau de gris donc la définition est

gris = 0.2126 R + 0.7152 G + 0.0722 B

Le résultat est une matrice de dimension (n,m)..

Nombre de lignes max = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def rgb2gray(I):
    return 0.2126 * I[:,:,0] + 0.7152 * I[:,:,1] + 0.0722 * I[:,:,2]
```

Solution de l'auteur de la question (Python3):

```
def rgb2gray(I):
    return 0.2126 * I[:,:,0] + 0.7152 * I[:,:,1] + 0.0722 * I[:,:,2]
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **7**

Correct

Note de 1,00 sur 1,00

Trouver la plus grande valeur paire d'une matrice. S'il n'y a pas de valeur paire, retourner None.

Nombre de lignes maximum = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def grand_paire(A):
    return A[A % 2 == 0].max() if np.any(A % 2 == 0) else None
```

Solution de l'auteur de la question (Python3):

```
def grand_paire(A):
    return A[A % 2 ==0].max() if np.any(A % 2 ==0) else None
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **8**

Incorrect

Note de 0,00 sur 1,00

Ecrire une fonction qui remplit une matrice (N,M) ligne par ligne avec les entiers entre 0 et N*M - 1.

```
python
fill_with_int(3,2)
array([[0, 1],
       [2, 3],
       [4, 5]])
```

A faire en 2 lignes de moins de 40 caractères.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def fill_with_int(n,m):
    return np.arange(n*m).reshape(n,m)
```

Solution de l'auteur de la question (Python3):

```
def fill_with_int(n,m):
    return np.arange(n*m).reshape(n,m)
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question **9**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui prend une matrice d'entier et la retourne modifiée pour n'avoir que des valeurs paires, les valeurs impaires ayant été augmenté d'un.

Nombre de lignes max = 3

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def matpaire(A):
    A[A % 2 == 1] += 1
    return A
```

Solution de l'auteur de la question (Python3):

```
def matpaire(A):
    A[A % 2 == 1] += 1
    return A
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **10**

Correct

Note de 1,00 sur 1,00

Soit 2 vecteurs a et b, créer la matrice C telle que C[i,j] = 1 / (a[i] - b[j])

Nombre de ligne max = 6

Attention, pas de for

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def matspe(a, b):
    def c(i,j):
        return 1/(a[i]-b[j])
    return np.fromfunction(c, shape=(len(a), len(b)), dtype=int)
```

Solution de l'auteur de la question (Python3):

```
def matspe(a, b):
    def c(i,j):
        return 1/(a[i]-b[j])

    return np.fromfunction(c, shape=(len(a), len(b)), dtype=int)

```

Correct

Note pour cet envoi : 1,00/1,00.

Question **11**

Correct

Note de 1,00 sur 1,00

Soit C = A B le produit matriciel des matrices A et B stockées comme deux tableaux Numpy. Écrire la fonction optimale qui retourne la valeur C[i,j]

En 2 lignes de 30 caractères max

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def C(i, j, A ,B):
    return A[i, :] @ B[:, j]
```

Solution de l'auteur de la question (Python3):

```
def C(i, j, A, B):
    return A[i,:] @ B[:,j]
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **12**

Incorrect

Note de 0,00 sur 1,00

Retourner la somme des éléments de la dernière colonne d'une matrice.

Return the sum of the elements of the last column of a matrix.

Nb de lignes max = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def sum_last_column(A):
    return A[:,-1].sum()
```

Solution de l'auteur de la question (Python3):

```
def sum_last_column(A):
    return A[:,-1].sum()
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question **13**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui triple la valeur des éléments d'indices impairs d'un vecteur (tableau 1D Numpy) en moins de 20 caractères par ligne sur 3 lignes. Le premier indice du vecteur est 0 donc pair.

Exemple : v = [1,2,3,4] triple_odds(v) donne [1,6,3,12]

En 3 lignes

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def triple_odds(v):
    v[1::2] *= 3
    return v
```

Solution de l'auteur de la question (Python3):

```
def triple_odds(v):
    v[1::2] *= 3
    return v
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **14**

Correct

Note de 1,00 sur 1,00

Créer un vecteur (Numpy array) de n entiers tirés au sort entre -10 et 10 et retourner le résultat triés par ordre croissant.

Nombre de ligne max : 3

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def vecaltri(n):
    res = np.random.randint(-10, 10, size = n)
    return np.sort(res)
```

Solution de l'auteur de la question (Python3):

```
def vecaltri(n):
    res = np.random.randint(-10, 10, size=n)
    return np.sort(res)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **15**

Correct

Note de 1,00 sur 1,00

Mettre toutes les valeurs maximales d'une matrice à 0 et retourner la matrice modifée.

En 3 lignes et 25 caractères par ligne maximum.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def max2zero(A):
    A[A == A.max()] = 0
    return A
```

Solution de l'auteur de la question (Python3):

```
def max2zero(A):
    A[A == A.max()] = 0
    return A
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **16**

Correct

Note de 1,00 sur 1,00

Ecrire la fonction qui retourne l'indice de la colonne qui a la plus grande valeur.

Si 2 colonnes ont la plus grande valeur, retourner l'indice de colonne le plus petit.

En maximum 2 lignes de 35 caractères.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def max_col(A):
    return A.max(axis=0).argmax()
```

Solution de l'auteur de la question (Python3):

```
def max_col(A):
    return A.max(axis=0).argmax()
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **17**

Correct

Note de 1,00 sur 1,00

Écrire la fonction direction_principale(nuage) qui prend un nuage de points en 3 dimensions et retourne sa direction principale normalisée. Le nuage est un np.array de dimension (3, N).

Write the function main_direction(cloud) which takes a point cloud in 3 dimensions and returns its normalized main direction. The cloud is a np.array of dimension (3, N).

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def direction_principale(nuage):
    cov = np.cov(nuage)
    val, vec = lin.eig(cov)
    m = np.argmax(val)
    return vec.T[m]
```

Solution de l'auteur de la question (Python3):

```
def direction_principale(nuage):
    cov = np.cov(nuage)
    val, vec = lin.eig(cov)
    m = np.argmax(val)
    return vec.T[m]
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **18**

Correct

Note de 1,00 sur 1,00

On suppose que notre matrice représente la température sur une grille qui a la forme de la matrice (un rectangle) avec un écart de 1 entre deux points (valeurs de la matrice). Calculer la dérivé suivant y en tout point. On utilise la dérivée centrée sauf aux bords.

Let assume that our matrix represents the temperature on a grid that has the shape of the matrix (a rectangle) with a distance of 1 between two points (values of the matrix). Calculate the derivative along y at any point. The centered derivative is used except at the edges.

En 2 lignes

Exemple : soit A

```
array([[17. , 17.2, 17.8, 17.5],
       [17. , 17.9, 18. , 17.6],
       [17. , 18. , 18.3, 17.5],
       [17. , 17.5, 17.3, 17. ]])
```

alors $\frac{\partial(A)}{\partial y}} = $

```
array([[ 0.  ,  0.7 ,  0.2 ,  0.1 ],
       [ 0.  ,  0.4 ,  0.25,  0.  ],
       [ 0.  , -0.2 , -0.35, -0.3 ],
       [ 0.  , -0.5 , -1.  , -0.5 ]])
```

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def d_dy(A):
2     return np.gradient(A, axis=0)
```

Solution de l'auteur de la question (Python3):

```
1 def d_dy(A):
2     return np.gradient(A, axis=0)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **19**

Correct

Note de 1,00 sur 1,00

Ecrire une fonction qui échange les colonnes i et j de la matrice A. Pas de copie.

Nombre de lignes maximum = 3

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def xchange_col(A, i, j):
2     A[:, [i, j]] = A[:, [j, i]]
3     return A
```

Solution de l'auteur de la question (Python3):

```
1 def xchange_col(A, i, j):
2     A[:,[i,j]] = A[:,[j,i]]
3     return A
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **20**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui indique quelle est la valeur d'un vecteur la plus proche d'une valeur `a` donnée en argument.

Write a function that returns the closest value of a vector from a given value `a`.

```
[In]  : closest(np.arange(5), 2.1)
[Out] : 2
```

2 lignes max

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def closest(v, a):
2     return v[np.argmin(np.abs(v - a))]
```

Solution de l'auteur de la question (Python3):

```
1 def closest(v, a):
2     return v[np.argmin(np.abs(v - a))]
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **21**

Incorrect

Note de 0,00 sur 1,00

Créer une matrice aléatoire n x m telle que la somme des éléments est égale à n*m et que toutes les valeurs de la matrice soient différentes (sauf vraiement vraiment pas de chance !).

Nombre de lignes maximum = 3

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def rand_euh(n, m):
2     res = np.random.random(size=(n, m))
3     return (n*m / res.sum()) * res
4
```

Solution de l'auteur de la question (Python3):

```
1 def rand_euh(n, m):
2     res = np.random.random(size=(n, m))
3     return (n*m / res.sum()) * res
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question **22**

Correct

Note de 1,00 sur 1,00

Écrire la fonction qui indique si une matrice est triangulaire supérieur ou non (donc qui retourne True ou False).

Nombre de ligne = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def est_triangulaire_sup(A):
    return np.all(A == np.triu(A))
```

Solution de l'auteur de la question (Python3):

```
def est_triangulaire_sup(A):
    return np.all(A == np.triu(A))
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **23**

Correct

Note de 1,00 sur 1,00

Notre tableau 2D A représente dans les colonnes impaires les poids des colonnes paires pour calculer la moyenne pondérée de chaque colonne paire. Donnez ces moyennes pondérées.

Our 2D array A represents in the odd columns the weights of the even columns to calculate the weighted average of each even column. Give these weighted averages.

En 2 lignes

Exemple : soit A

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

les premiers poids sont 0,4,8 pour les valeurs 1,5,9. La moyenne pondérée est 7.666... La 3e colonne a les poids de la 4e colonne. Le résultat final est

```
array([7.66666667, 8.77777778])
```

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def moyenne_pondérée(A):
    return np.average(A[:, 1::2], weights = A[:, ::2], axis = 0)
```

Solution de l'auteur de la question (Python3):

```
def moyenne_pondérée(A):
    return np.average(A[:,1::2], weights=A[:,::2], axis=0)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **24**

Correct

Note de 1,00 sur 1,00

Transformer une matrice pour que sa moyenne soit 0 et son écart type égal à 1.

Réponse en 2 lignes de moins de 35 caractères

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def normalize(A):
    return (A - A.mean()) / a.std()
```

Solution de l'auteur de la question (Python3):

```
def normalize(A):
    return (A - A.mean()) / A.std()
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **25**

Correct

Note de 1,00 sur 1,00

Retourner une matrice dont les élements sont translatés afin que l'éĺément i,j devient l'élément 0,0, que ceux avant i passent à droite et ceux avant j passent en bas.

En 2 lignes

Exemple : def

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

donne le résultat suivant avec i,j == 1,2

```
array([[ 6,  7,  4,  5],
       [10, 11,  8,  9],
       [ 2,  3,  0,  1]])
```

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def translate(A, i, j):
    return np.roll(np.roll(A,-i, axis=0), -j, axis=1)
```

Solution de l'auteur de la question (Python3):

```
def translate(A, i, j):
    return np.roll(np.roll(A,-i,axis=0), -j, axis=1)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **26**

Correct

Note de 1,00 sur 1,00

Ranger par ordre croissant les valeurs d'une matrice sans changer la structure de la matrice (ordre : ligne, colonne c.a.d. A[i,j] <= A[i+1,k] et A[i,j] <= A[i,j+1] pour tout i,j,k).

Nombre de lignes maximum = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def range_mat(A):
    return np.array(sorted(A.flatten())).reshape(A.shape)
```

Solution de l'auteur de la question (Python3):

```
def range_mat(A):
    return np.array(sorted(A.flatten())).reshape(A.shape)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **27**

Correct

Note de 1,00 sur 1,00

Écrire la fonction qui retourne le ratio entre la plus grande valeur propre en valeur absolue et la plus petite en valeur absolue.

max_i (|λ_i|) / min_i (|λ_i|)

`import numpy.linalg as lin`.

est déjà fait

A faire en 4 lignes de 30 caractères max.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def ratio_eig(A):
    e, b = lin.eig(A)
    e = np.abs(e)
    return e.max() / e.min()
```

Solution de l'auteur de la question (Python3):

```
def ratio_eig(A):
    e,_ = lin.eig(A)
    e = np.abs(e)
    return e.max() / e.min()
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **28**

Incorrect

Note de 0,00 sur 1,00

Soit une matrice A, retourner la solution x de A x = diag(A).

On a déjà fait import numpy.linalg as lin

Nombre max de ligne = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def axdiag(A):
    return lin.solve(A, np.diag(A)
```

Solution de l'auteur de la question (Python3):

```
def axdiag(A):
    return lin.solve(A, np.diag(A))
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question **29**

Correct

Note de 1,00 sur 1,00

Créer une matrice NxN dont les colonnes valent 0, 1, 2, ... N-1.

```
python
col_range(3)
array([[0., 0., 0.],
       [1., 1., 1.],
       [2., 2., 2.]])
```

Max 4 lignes de 25 caractères.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def col_range(n):
    res = np.empty((n,n))
    res[:] = np.arange(n)
    return res.T
```

Solution de l'auteur de la question (Python3):

```
def col_range(n):
    res = np.empty((n,n))
    res[:] = np.arange(n)
    return res.T
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **30**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui découpe un tensor de dimension d (une matrice est de dimension 2) en k tenseurs de dimension d-1 avec k la taille du tenseur dans sa dernière dimension.

Write a function that plits a d dimensions tensor (a matrix is a 2D tensor) in k tensors of d-1 dimensions with k the size of our tensor in its last dimension (axis).

```
In [28]: A
Out[28]:
array([[7, 2, 7],
       [6, 0, 9],
       [5, 2, 1]])

In [29]: cut(A)
Out[29]:
[array([[7],
        [6],
        [5]]),
 array([[2],
        [0],
        [2]]),
 array([[7],
        [9],
        [1]])]
```

2 lines

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 def cut(T):
2     return np.split(T, T.shape[-1], -1)
```

Solution de l'auteur de la question (Python3):

```
1 def cut(T):
2     return np.split(T, T.shape[-1], -1)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **31**

Correct

Note de 1,00 sur 1,00

Soit un vecteur v, on désire créer la matrice A qui a

- v en 1ere colonne,
- v décalé d'un cran en 2e colonne (A[0,1] = v[1] et A[-1,1] = v[0]),
- v décalé de 2 crans en 3e colonne (A[0,2] = v[2] et A[-1,2] = v[1]),
- etc

`for` autorisé.

Nombre max de lignes = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def matspe(v):
    return np.stack([np.roll(v, -i) for i in range(len(v))], axis=1)
```

Solution de l'auteur de la question (Python3):

```
def matspe(v):
    return np.stack([np.roll(v,-i) for i in range(len(v))], axis=1)
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **32**

Correct

Note de 1,00 sur 1,00

Écrire une fonction qui échange la valeur maximale d'une matrice avec sa valeur minimale.

Nombre de lignes max = 3

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def maxmin(A):
    A.flat[[A.argmax(), A.argmin()]] = A.min(), A.max()
    return A
```

Solution de l'auteur de la question (Python3):

```
def maxmin(A):
    A.flat[[A.argmax(),A.argmin()]] = A.min(), A.max()
    return A
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **33**

Correct

Note de 1,00 sur 1,00

Remplacer toutes les valeurs négatives par leur opposé si cet opposé n'existe pas déjà dans la matrice.

```
In : A
Out:
array([[-2,  7, -8],
       [-8,  1,  7],
       [ 4,  8,  4]])

In : presqu_abs(A)
Out:
array([[ 2,  7, -8],
       [-8,  1,  7],
       [ 4,  8,  4]])
```

Nombre de lignes maximum = 5

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```python
def presqu_abs(A):
    a = np.isin(A, -A[A > 0])
    A = np.abs(A)
    A[a] *= -1
    return A
```

Solution de l'auteur de la question (Python3):

```python
def presqu_abs(A):
    mask = np.isin(A, -A[A>0])
    A = np.abs(A)
    A[mask] *= -1
    return A
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **34**

Correct

Note de 1,00 sur 1,00

Ranger les lignes d'une matrice par ordre croissant de la moyenne des valeurs de la ligne.

[[1,5,3], [0,1,2],[4,6,8]] donne [[0,1,2], [1,5,3], [4,6,8]] car les moyennes des lignes initiales sont 3, 1 et 6.

Nombre de ligne max = 4

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def trilignes(A):
    mean = A.mean(axis=1)
    sort = np.argsort(means)
    return A[order]
```

Solution de l'auteur de la question (Python3):

```
def trilignes(A):
    means = A.mean(axis = 1)
    order = np.argsort(means)
    return A[order]
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **35**

Incorrect

Note de 0,00 sur 2,00

Soit 2 tenseurs A et B. A est en 4 dimension et B en 5 dimension (rappel : une matrice est en 2 dimensions). On veut calculer la contraction tensorielle de A par B qui donne C :

$$ C[a,b,c,d,e] = \sum_i \sum_j A[a,i,j,b] * B[c,d,i,e,j] $$

Nombre de lignes max = 2

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
ef contraction_tensorielle(A, B):
    return np.einsum('aijb,cdiej -> abcde', A, B)
```

Solution de l'auteur de la question (Python3):

```
def contraction_tensorielle(A, B):
    return np.einsum('aijb,cdiej -> abcde', A, B)
```

Incorrect

Note pour cet envoi : 0,00/2,00.

Question **36**

Correct

Note de 2,00 sur 2,00

Écrire une fonction qui donne la diagonale du produit matriciel de A par B (le résultat est un vecteur). On ne calculera pas le produit complet de A par B bien sûr.

En 2 lignes max de 40 caractères.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def diag_prod(A, B):
    return np.einsum("ij,ji->i", A, B)
```

Solution de l'auteur de la question (Python3):

```
def diag_prod(A, B):
    return np.einsum("ij,ji->i", A, B)
```

Correct

Note pour cet envoi : 2,00/2,00.

Question **37**

Incorrect

Note de 0,00 sur 2,00

Soit une liste de points en 2D qu'on peut imaginer comme les coordonnées (x,y) de villes. On veut un tableau D qui donne la distance à vol d'oiseau entre 2 villes. Donc a D[i,j] = D[j,i] = distance entre les villes (points) i et j.

Les n points sont donnés dans une matrice (n,2). Écrire la fonction qui prend ces points et retourne la matrice D des distances.

Nombre de ligne max = 4

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
def distpoints(P):
    x = np.array([P[:, 0]])
    y = np.array([P[:, 1]])
    return np.sqrt(np.square(x - x.T) + np.square(y - y.T))
```

Solution de l'auteur de la question (Python3):

```
def distpoints(P):
    x = np.array([P[:,0]])
    y = np.array([P[:,1]])
    return np.sqrt(np.square(x - x.T) + np.square(y - y.T))
```

Incorrect

Note pour cet envoi : 0,00/2,00.

◄ Annonces

Aller à…

Le cours ►