Université Sultan Moulay Slimane Faculté Polydisciplinaire Béni Mellal
Département INFORMATIQUE (MIP)

Filière : Science de données et sécurité des systèmes d'information
A.U : 2023-2024

Sujet

Compte Rendu de TP03 – Python pour la science des données

Présenté Par :
MAFTOUH Omar

Encadré Par :
Z. BOUSALEM
Y. MADANI

Introduction :

Ce travail pratique se concentre sur la manipulation des tableaux et des matrices en utilisant la bibliothèque NumPy en Python. Nous explorerons diverses opérations telles que la création de tableaux, l'ajout d'éléments, la multiplication, la recherche d'indices, la manipulation de formes, ainsi que des opérations de base sur les matrices comme la multiplication, la transposée et l'inversion.

Exercice01 :

Dans cet exercice, nous allons explorer diverses opérations sur les tableaux NumPy, telles que l'ajout, la multiplication, la recherche d'indices, la création de tableaux booléens, l'utilisation de masques et le redimensionnement.

```
# Importation du bibliothèque NumPy en utilisant l'alias np
import numpy as np

# Creation un tableau 1D appelé tab1 avec les entiers de 0 à 5
tab1 = np.arange(6)

# Creation d'un tableau 2D de taille 3,3 , remplit avec des entiers
aléatoires(entre 1 et 10)
tab2 = np.random.randint(1 , 10 , size=(3,3))

# Ajouter 2 à chaque élément du tab1
np.add(tab1 , 2)

# Multiplier chaque élément de tab2 par 2
np.multiply(tab2 , 2)

# Afficher la forme et la dimension des deux tableaux
print(f"Forme du tab1 : {tab1.shape} , dimension du tab1 : {tab1.ndim}") # tab
01
print(f"Forme du tab2 : {tab2.shape} , dimension du tab2 : {tab2.ndim}") # tab
02

# Afficher le nombre des lignes et des colonnes de tab2
print(f"Nombre du ligne du tab2 : {tab2.shape[0]} , Nombre du colone du tab2 :
{tab2.shape[1]}")

# Afficher la deuxième ligne de tab2
print(f"la deuxième ligne de tab2 : {tab2[1]}")

# Afficher un sous-tableau 2x2 de tab2 en excluant la première ligne et la
première colonne
print(f"tab2 sans ligne 1 & colone 1 :\n {tab2[ 1 : , 1 : ]}")

# les indices des valeurs maximales & minimales le long de l'axe 0 dans le
tableau tab2.
print(f"Les indices des valeurs maximales de le long axe 0 : {np.argmax(tab2 ,
axis=0)}")
```

```
print(f"Les indices des valeurs minimales de le long axe 0 : {np.argmin(tab2 , axis=0)}")

# les indices des valeurs maximales & minimales le long de l'axe1 dans le tableau tab2.
print(f"Les indices des valeurs maximales de le long axe 1 : {np.argmax(tab2 , axis=1)}")
print(f"Les indices des valeurs minimales de le long axe 1 : {np.argmin(tab2 , axis=1)}")

"""
    Creation du tableau booléen sup_a_3 de même taille que tab1, où chaque élément est égal à
    True si l’élément correspond dans tab1 est supérieur à 3, sinon la valeur est False
"""
sup_a_3 = np.array(tab1 > 3 , ndmin=(tab1.ndim))
print(f"Superieur a 3 : {sup_a_3}")

# les éléments de tab1 qui sont supérieurs à 3 , en utilisant le masque
print(f"Les elements sup que 3 dans tab1 : {tab1[sup_a_3]}")

# Redimensionner tab1 en un tableau2D de taille 2x3.
tab1_reshaped = tab1.reshape(2 , 3)
print(f"Tab1 en format 2D et de taille 2*3 :\n {tab1_reshaped}")

# La creation du tableau tab3 de la forme suivante : [4, -5, 2]
tab3 = np.array([4 , -5 , 2])

# Ajouter le tableau tab3 comme nouvelle ligne au tableau tab1
tab4 = np.vstack((tab1_reshaped , tab3))
print(f"Tab4 :\n {tab4}")

# Calculer la somme des éléments de tab4
print(f"La somme d\'element du tab4 : {np.sum(tab4)}")

# Calculez la moyenne de chaque colonne de tab2
print(f"Calculer la moyenne des element du tab2 de chaque colonne : {np.round(np.mean(tab2 , axis=0) , decimals=2)}")
```

Résultat du programme :

```
Forme du tab1 : (6,) , dimension du tab1 : 1
Forme du tab2 : (3, 3) , dimension du tab2 : 2
Nombre du ligne du tab2 : 3 , Nombre du colone du tab2 : 3
la deuxième ligne de tab2 : [5 7 6]
tab2 sans ligne 1 & colone 1 :
 [[7 6]
 [7 3]]
Les indices des valeurs maximales de le long axe 0 : [0 1 0]
Les indices des valeurs minimales de le long axe 0 : [2 0 2]
Les indices des valeurs maximales de le long axe 1 : [0 1 1]
Les indices des valeurs minimales de le long axe 1 : [1 0 2]
Superieur a 3 : [False False False False  True  True]
Les elements sup que 3 dans tab1 : [4 5]
Tab1 en format 2D et de taille 2*3 :
 [[0 1 2]
 [3 4 5]]
Tab4 :
 [[ 0  1  2]
 [ 3  4  5]
 [ 4 -5  2]]
La somme d'element du tab4 : 16
Calculer la moyenne des element du tab2 de chaque colonne : [5.33 6.33 5.33]
```

Exercice 02 :

Cet exercice se concentre sur des opérations avancées sur les matrices, incluant la génération de matrices aléatoires, la multiplication matricielle, la transposée, l'exclusion de colonnes, l'inversion, et la vérification de l'identité matricielle.

```
import numpy as np

# Définir la moyenne et l'écart-type
moyenne = 3
ecart_type = 2

# Générer le tableau de valeurs aléatoires suivant une distribution normale
tab_random = np.random.normal(moyenne, ecart_type, 12)
print(f"tab_random : \n{tab_random}")

# Redimensionner tab_random en une matrice(mat1) de taille 3,4
mat1 = tab_random.reshape(3 , 4)
print(f"mat1 : \n{mat1}")

# Remplacez toutes les valeurs de la matrice mat1 par leur valeur entière
inférieure
print(f"valeur entière inférieure du matrice mat1 : \n{np.floor(mat1)}")

# Convertir les éléments de mat1 en int
mat1 = np.int64(mat1)
# Ou mat1 = mat1.astype(int)
print(f"les elements du mat1 format int : \n{mat1}")

# Creation du matrice mat2 de taille 4,3 avec des valeur entre 2 et 14
mat2 = np.arange(2 , 14)
mat2_reshaped = mat2.reshape(4 , 3)
print(f"mat2 : {mat2_reshaped}")

# Calculer la multiplication matricielle de mat1 et mat2
print(f"La multiplication du mat1 & mat2 : \n{np.dot(mat2_reshaped , mat1)}")

# Calculer la transposée de la première matrice mat1
print(f"Transpose du mat1 : \n{mat1.T}")

# Creation d'une nouvelle matrice carrée 3x3 en excluant la dernière colonne
de la matrice mat1
matCarree = np.array([mat1[ : , :-1]])
print(f"Matrice Carree 3*3 : \n{matCarree}")
inverseMat1 = None
# Trouver l'inverse de la matrice mat1
if np.linalg.det(matCarree) != 0 :
    inverseMat1 = np.linalg.inv(matCarree)
    inverseMat1 = np.round(inverseMat1 , 2)
    print(f"Inverse du mat1 : \n{inverseMat1}")
```

```
else :
    print("Determinant egale 0")

# Multipliez la matrice résultante de l'inversion par la matrice originale
if inverseMat1 is not None :
    idnt = np.dot(inverseMat1 , matCarree)
    idnt = np.round( idnt , 2 )
    print(f"Matrice identité : \n{idnt}")
else :
    print("Invalide inversse")
```

Résultat du programme :

```
tab_random :
[ 3.16588519  2.24963567  3.16501875  3.60161352  3.71715377  4.44506685
  2.14243207 -0.53063688 -0.21247177  1.14318965  1.88426659  5.49084526]
mat1 :
[[ 3.16588519  2.24963567  3.16501875  3.60161352]
 [ 3.71715377  4.44506685  2.14243207 -0.53063688]
 [-0.21247177  1.14318965  1.88426659  5.49084526]]
valeur entière inférieure du matrice mat1 :
[[ 3.  2.  3.  3.]
 [ 3.  4.  2. -1.]
 [-1.  1.  1.  5.]]
les elements du mat1 format int :
[[3 2 3 3]
 [3 4 2 0]
 [0 1 1 5]]
mat2 : [[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]
 [11 12 13]]
La multiplication du mat1 & mat2 :
[[15 20 16 26]
 [33 41 34 50]
 [51 62 52 74]
 [69 83 70 98]]
Transpose du mat1 :
[[3 3 0]
 [2 4 1]
 [3 2 1]
 [3 0 5]]
Matrice Carree 3*3 :
[[[3 2 3]
  [3 4 2]
  [0 1 1]]]
Inverse du mat1 :
[[[ 0.22  0.11 -0.89]
  [-0.33  0.33  0.33]
  [ 0.33 -0.33  0.67]]]
Matrice identité :
[[[[ 0.99 -0.01 -0.01]]
```

```
 [[ 0.    0.99  0.  ]]
 [[ 0.    0.01  1.  ]]]]
```

Conclusion :

Ce compte rendu met en lumière les différentes opérations que nous avons effectuées sur les tableaux et les matrices en utilisant NumPy. Nous avons exploré diverses manipulations telles que l'ajout, la multiplication, la recherche d'indices, la création de masques, la transposée, l'inversion, et la vérification de l'identité matricielle. Ces manipulations illustrent la puissance et la flexibilité offertes par la bibliothèque NumPy pour le traitement efficace des données numériques en Python.