

Copies, Changement de dimensions et Concaténation

Copie
Changement de dimensions
Concaténation
Conclusion

Programmation Python Partie 2 : Copies, Changement de dimensions et Concaténation

Alexandre Gramfort

Copy

NumPy ne copie pas automatiquement les tableaux.

```
In [121]: A = np.array([[0, 2], [3, 4]])  
print(A)
```

```
[[0 2]  
 [3 4]]
```

```
In [122]: B = A # assignation sans copie
```

```
In [123]: B[0, 0] = 10 # changer B affecte A  
print(B)
```

```
[[10 2]  
 [ 3 4]]
```

```
In [124]: print(A) # A est modifié
```

```
[[10 2]  
 [ 3 4]]
```

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions

Concaténation

Conclusion

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions

Concaténation

Conclusion

Pour éviter ce comportement, on peut demander que B soit une copie de A

```
In [125]: A = np.array([[0, 2], [3, 4]])
```

```
In [126]: B = A.copy() # assignation avec copie
```

```
In [127]: B[0,0] = -5 # B est modifié  
print(B)
```

```
[[ -5  2]  
 [ 3  4]]
```

```
In [128]: print(A) # A n'est pas modifié
```

```
[[0 2]  
 [3 4]]
```

Changement de taille

```
In [129]: A = np.array([[1, 2], [3, 4]])
```

```
In [130]: n, m = A.shape  
print(n, m)
```

(2, 2)

```
In [131]: B = A.reshape((n * m,))  
print(B.shape)  
print(B)
```

(4,)
[1 2 3 4]

```
In [132]: B = A.reshape((n * m, 1))  
print(B.shape)  
print(B)
```

(4, 1)
[[1]
 [2]
 [3]
 [4]]

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions

Concaténation

Conclusion

Concaténer, répéter des *arrays*

En utilisant les fonctions `tile`, `vstack`, `hstack`, et `concatenate`, on peut créer des vecteurs/matrices plus grandes à partir de vecteurs/matrices plus petites.

`tile`

```
In [133]: A = np.array([[1, 2], [3, 4]])  
          print(A)
```

```
[[1 2]  
 [3 4]]
```

```
In [134]: print(np.tile(A, (1, 2))) # répéter la matrice 2 fois
```

```
[[1 2 1 2]  
 [3 4 3 4]]
```

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions

Concaténation

Conclusion

concatenate

```
In [135]: print(A)
```

```
[[1 2]
 [3 4]]
```

```
In [136]: B = np.array([[5, 6]])
```

```
In [137]: np.concatenate((A, B), axis=0)
```

```
Out[137]: array([[1, 2],
                 [3, 4],
                 [5, 6]])
```

```
In [138]: np.concatenate((A, B.T), axis=1)
```

```
Out[138]: array([[1, 2, 5],
                 [3, 4, 6]])
```

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions

Concaténation

Conclusion

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions
Concaténation

Conclusion

hstack et vstack

```
In [139]: np.vstack((A, B))
```

```
Out[139]: array([[1, 2],  
                [3, 4],  
                [5, 6]])
```

```
In [140]: np.hstack((A, B.T))
```

```
Out[140]: array([[1, 2, 5],  
                [3, 4, 6]])
```

Copies, Changement de dimensions et Concaténation

Copie

Changement de dimensions
Concaténation

Conclusion

FIN

```
In [141]: import scipy
```

Rendez-vous sur <http://docs.scipy.org/doc/scipy/reference/> pour découvrir cette librairie d'algorithmes (optimisation, stats, signal, intégration numérique etc.)

```
In [142]: import antigravity
```