Numpy Cheat Sheet

Créer un tableau:

```
import numpy as np

a = np.array([1,2,3], [4,5,6])
```

Créer un tableau remplis d'un nombre:

```
a = np.full((2,2), yourNumber)
```

Générer des matrices de nombres aléatoires:

```
a = np.random.randint(10, size=(3,4))
```

La size définit la taille de la matrice, 10 définit le nombre maximal
Une lois peut être choisie

```
loc = 3
scale = 1.5
np.random.normal(loc, scale, size=(2,3))  # Gauss distribution
```

On peut reformater un tableau:

```
a = np.arange(3*4) #Genère une liste de 12 élèments
a.reshape(3,4) #Reformate la liste en matrice de 3 par 4

print(a.flat()) #Display la matrice sous forme de liste sans la modifier
a.flatten() # Remet le tableau sous forme de liste
```

On peut mélanger un tableau:

```
np.random.permutation(a) #Si c'est utilisé sur une matrice, seule la première
dimension
```

Opérations de base

On peut additionner un nombre ou une matrice à une autre matrice:

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

A += 1
A += A
```

On peut mutiplier une matrice avec une autre matrice:

```
A *= A #On multiplie seulements les élèments entre eux
A @ A #Multiplication de matrice classique
```

On peut transposer une matrice:

```
A.T
```

Pour additionner une liste:

```
A.sum()
np.sum(A)
```

Pour avoir une moyenne:

```
A.mean()
np.mean(A)

A.average() #Moyenne pondérée
np.average(A)
```

Pour multiplier tous les elements entre eux:

```
A.prod()
np.prod(A)
```

Pour trouver le min et le max:

```
A.min()
np.min(A)

A.max()
np.max(A)

# L'index du max et du min
A.argmin()
np.argmin(A)
A.argmax()
np.argmax(A)
```

Pour faire des additions et multiplications cumulatives:

```
A.cumsum()
np.cumsum(A)

A.cumprod(A)
np.cumprod(A)
```

Pour voir l'écart avec l'élément suivant:

```
np.diff(A)
```

Manipulation de tableau

Filtre via les indices

Les indices fonctionnent sous la forme:
[debut:fin:pas]

Si des parametres ne sont pas spécifiés, ils sont remplacés par des valeurs par defauts:
[0:len:1]

La dernière column est désigné par -1

```
a = np.arange(12).reshape(3,4)
a[1,:] #Toute la deuxième ligne
a[0:2,-1] #Toutes les dernières columns des deux premières lignes
a[:,::2] #Tous les nombres d'indices pairs
```

Filtre logique

On peut utiliser les opérateurs de comparaisons pour verifier une condition sur toutes les valeurs de la matrice. Ce filtrage renvoie des tableau de boolean

```
fil = a > 5
data = a[a>5] #Pour recuperer toutes les valeurs
```

On peut donc faire des opérations pour verifier le resultat du filtrage du

```
fill.sum() #Permet de savoir combien de True
fill.prod() #Permet de verifier si tous les cas sont vrais
```

Filte avec Where

On peut filtrer une matrice avec where, cela permet de generer une matrice de même taille remplie de d'une valeur definie dans toutes les cases qui ne valident pas la conditions, les autres valeurs sont conservées.

```
np.where(a > 5, a, 0) #la valeur de remplissage ici est 0
```