

Nous aurons besoin de la bibliothèque **numpy**, abrégée en **np**, ainsi que de la bibliothèque **numpy.linalg** à importer ainsi :

```
import numpy.linalg as al
```

Syntaxe	Ce qu'elle renvoie
<code>np.zeros([n,p])</code>	matrice à n lignes et p colonnes dont tous les coefficients sont 0
<code>np.ones([n,p])</code>	matrice à n lignes et p colonnes dont tous les coefficients sont 1
<code>np.eye(n)</code>	matrice identité de taille n
<code>np.array([L1,L2,...,Ln])</code> (où L1,...,Ln sont des listes)	matrice dont la première ligne vaut L1, la seconde L2, ...
<code>np.diag(L)</code>	la matrice diagonale dont les coefficients diagonaux sont les éléments de la liste L
<code>np.shape(A)</code>	le couple (n,p) où n est le nombre de lignes et p de colonnes de A
<code>A[i,j]</code>	le coefficient (i,j) du tableau, donc le coefficient $(i-1,j-1)$ de la matrice A
<code>A+B</code> (si A et B sont deux matrices de même taille)	la matrice $A+B$
<code>A*B</code> (si A et B sont deux matrices de même taille)	la matrice dont le coefficient (i,j) est le produit des coefficients (i,j) de A et B : PAS LA MATRICE AB!
<code>np.dot(A,B)</code> (si compatibilité...)	la matrice AB
<code>A**2</code>	la matrice dont les coefficients de A sont élevés au carré : PAS LA MATRICE A^2 !
<code>al.matrix_power(A,k)</code> (si A est carrée)	la matrice A^k
<code>np.transpose(A)</code>	la matrice tA
<code>al.inv(A)</code> (si A est inversible)	la matrice A^{-1}
<code>A==B</code> (si A et B sont deux matrices de même taille)	la matrice composée de booléens : True si les coefficients (i,j) de A et B sont égaux, False sinon.
<code>(A==B).all()</code>	True si A et B sont égales, False sinon.
<code>al.matrix_rank(A)</code>	le rang de A
<code>al.solve(A,B)</code> (si A est inversible et si compatibilité)	l'unique solution de l'équation $AM = B$ d'inconnue M (où B peut être une matrice colonne ou non)
<code>al.eig(A)</code> (si A est carrée)	un couple (V,P) où V est un tableau ligne des valeurs propres de A et P un tableau de vecteurs propres associés

Exemple

Si on veut
 $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, on saisit :
`np.array([[1,2,3],[4,5,6]])`
 Une matrice est en fait une liste de listes transformée en tableau.

Attention !

Comme pour les listes, la numérotation des lignes et colonnes d'une matrice commence à 0.

Petite remarque

Plus généralement, si f est une fonction définie sur les listes et tableau (`np.exp`, `np.log`...), alors $f(A)$ renvoie la matrice dont le coefficient (i,j) est l'image par f du coefficient (i,j) de A

Précisions...

- Les vecteurs propres donnés sont *normés*...
- Si A est diagonalisable, alors P sera une matrice de passage de la base canonique vers une base de vecteurs propres.

On considère la matrice $A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 6 & -11 & 6 \end{pmatrix}$.

1. A l'aide de Python, calculer $A^3 - 6A^2 + 11A - 6I_3$.

```
1 import numpy as np
2 import numpy.linalg as al
```

```

3
4 A=np.array([[0,1,0],[0,0,1],[6,-11,6]])
5 A2=al.matrix_power(A,2)
6 A3=al.matrix_power(A,3)
7 I=np.eye(3)
8 print(A3-6*A2+11*A-6*I)

```

2. En déduire que A est inversible. A l'aide de **Python**, calculer et afficher A^{-1} de deux façons différentes.

```

1 import numpy as np
2 import numpy.linalg as al
3
4 A=np.array([[0,1,0],[0,0,1],[6,-11,6]])
5 A2=al.matrix_power(A,2)
6 I=np.eye(3)
7 invA=al.inv(A)
8 invAbis=1/6*(A2-6*A+11*I)
9 print(invA)
10 print(invAbis)

```

3. Sans utiliser la commande **al.eig**, déterminer les valeurs propres de A .

♣ Indication...
 Quelles sont les caractérisations des VP ?

```

1 import numpy as np
2 import numpy.linalg as al
3
4 A=np.array([[0,1,0],[0,0,1],[6,-11,6]])
5 I=np.eye(3)
6 #les racines du polynôme X^3-6X^2+11X-6 (annulateur de A) sont 1,2,3
7 #sont-elles bien valeurs propres ?
8
9 for x in [1,2,3]:
10     if al.matrix_rank(A-x*I)!=3:
11         print(x, "est valeur propre de A")
12     else:
13         print(x, "n'est pas valeur propre de A")

```

4. Sans exécuter les lignes suivantes, sachant que la commande **al.eig(A)[0]** renvoie **array([1,2,3])**, quelle devrait être la matrice affichée à l'issue de l'exécution ?

```

1 import numpy as np
2 import numpy.linalg as al
3
4 A=np.array([[0,1,0],[0,0,1],[6,-11,6]])
5 P=al.eig(A)[1]
6 invP=al.inv(P)
7 B=np.dot(invP,np.dot(A,P))
8 print(B)

```

On aura $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$.

En effet :

- **al.eig(A)[0]** renvoie un tableau composé des valeurs propres de A : il y en a 3 différentes, A est donc diagonalisable...
- et dans ce cas, **al.eig(A)[1]** renvoie un tableau composé si possible d'une base de vecteurs propres de A associés aux valeurs propres (dans le même ordre)... Puisqu'ici A est diagonalisable, ce tableau est la matrice de passage de la base canonique vers une base de vecteurs propres.

Par formule de changement de base, la matrice $P^{-1}AP$ est diagonale constituée des valeurs propres de A .