1/ Calcul matriciel

```
import numpy as np
import numpy.linalg as alg
```

Création, manipulation

`np.array(list)`	création à partir d'une liste (de listes si besoin) Python (ou d'un itérable)
`np.linspace(a,b,n)`	subdivision de $[a,b]$ en n points, extrémités comprises - on peut ajouter `endpoint=False` si on ne veut pas du dernier point
`np.zeros(dim)`	tableau de 0 avec les dimensions données (dim est un tuple)
`np.ones(dim)`	idem avec des 1
`np.eye(n)`	matrice carrée identité de taille n
`np.eye(p,q,n)`	matrice de taille $p \times q$ avec une diagonale de 1 décalée de n crans
`np.diag(list)`	matrice carrée de diagonale $list$
`np.fromfunction(f,dim)`	crée le tableau à partir de la fonction f, avec le format dim
`A.shape`, `A.reshape(dim)`	format de la matrice, reformatage de la matrice
`np.concatenate((A,B),axis=0)`	concaténation vecticale (horizontale avec `axis=1`)
+,*,`np.dot(A,B)` ou `A.dot(B)`	somme, produit (scalaire ou case à case), produit matriciel
`A.T` ou `np.transpose(A)`	transposition
`alg.matrix_power(A,n)`	puissances de `A`
`alg.det(A)`, `alg.trace(A)`	détermiant, trace de `A`
`alg.matrix_rank(A)`	rang de la matrice
`alg.inv(A)`, `alg.solve(A,b)`	inverse de `A`, résolution de `Ax=b`
`alg.eigvals(A)`, `alg.eig(A)`	valeurs propres, éléments propres (val. et vect.)
`np.vdot(u,b)`, `np.cross(u,v)`	produit scalaire, vectoriel

2/ Polynômes

```
from numpy.polynomial import Polynomial
```

Création, opérations

`p = Polynomial([-3, 2, 0, 1])`	création de $-3+2X+X^3$
`p(val)`, `p.degree()`, `p.coeff`	évaluation, degré, tableau des coefficients
`p.deriv(n)`, `p.roots()`	polynôme dérivé (`n` nombre de fois), racines du polynôme
`p.integ(1,p0)`	primitive avec constante
`p.integ(n,tab)`	primitives successives avec tableau des constantes
+,*,**, // , %	opérations usuelles

3/ Graphiques

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
```

Tracé 2D

`plt.plot(X,Y,options)`	tracé classique
`plt.axis('equal')`	pour un repère orthonormé
`plt.grid()`	affichage d'une grille

Pour les tracés en 3D, voir la documentation.

4/ Probabilités

```
import numpy.random as rd
```

Création, manipulation

`rd.randint(a,b,n)`	tirage uniforme d'un ou `n` (facultatif) entiers dans $[a,b[$
`rd.randint(a,b,(p,q))`	matrice aléatoire d'entiers
`rd.random()`, `rd.random(dim)`	loi uniforme sur $[0,1[$.
`rd.binomial(n,p,nombre)`	loi binomiale $B(n,p)$
`rd.geometric(p,nombre)`	loi géométrique de paramètre p
`rd.poisson(l,nombre)`	loi de Poisson

5/ Analyse numérique

```
import numpy as np
import scipy.optimize as resol
import scipy.integrate as integr
import matplotlib.pyplot as plt
```

Résolutions

`resol.fsolve(f,x0)`	résolution numérique au voisinage de `x0`
`resol.root(f,[vals])`	idem avec `f` à plusieurs variables et conditions
`integ.quad(f,a,b)`	intégration numérique (`np.inf` pour l'infini)
`integ.odeint(f,x0,T)`	résolution de $y' = f(y,t)$ avec condition initiale `x0` sur le tableau d'abscisses `T`