## Recommandations générales

**Édition :** Utiliser un IDE (Integrated Development Environment) pour interpréter du Python, comme Pyzo ou Spyder. Pour nommer les fichiers, variables ou fonctions ne pas utiliser de caractères spéciaux : accents, espaces, etc. Utiliser des noms de variables et de fonctions qui facilitent la programmation, sans toutefois l'alourdir.
Commenter efficacement ses programmes à l'aide du symbole # (pour décrire l'utilité d'une fonction ou les hypothèses non-évidentes auxquelles sont soumis ses arguments d'entrée).

**Exécution :** Il faut exécuter le fichier .py afin de pouvoir utiliser les fonctions qui apparaissent dedans souvent. Toute modification d'une fonction ne sera effective qu'après nouvelle exécution du fichier la contenant.

**Déboguage :** Pour la recherche d'une **erreur de programmation ou de code**, commencer par l'identifier...
La commande `print(x)` glissée dans une fonction ou un programme permet de renvoyer la valeur de la variable x et ainsi de mieux comprendre l'erreur du programme.
Dans la recherche d'une erreur affectant le résultat d'un algorithme, une **stratégie de deboguage** efficace consiste à décliner une série de tests élémentaires et progressifs qui permettent de valider les différentes étapes de l'algorithme indépendamment les unes des autres. Cette validation peut/doit être effectuée au fil de la programmation qui est alors elle-même progressive.

**Attention !** L'indentation en Python est très importante, une mauvaise indentation renverra des erreurs. Durant les TP, il est important d'aérer le code afin de faciliter la lecture (pour tout le monde), par exemple en mettant des espaces autour des opérations +, =, etc.

## Environnements à utiliser

| | |
|---|---|
| `from math import *` | importe toutes les fonctions mathématiques du module math |
| `import numpy as np` | importe le module qui manipule les vecteurs et matrices |
| `import matplotlib.pyplot as plt` | importe le module qui trace des courbes |

## Quelques fonctions

`sqrt, log, exp, cos, sin, tan, cosh, sinh, tanh, abs, floor, round`
Ajouter `np.` devant ces fonctions pour les appliquer à des vecteurs composante par composante
`x.real` et `x.imag` renvoient les parties réelle et imaginaire d'un élément `x`
`pi, e, 1j` (pour le complexe $i$)

## Structures de contrôle

| | |
|---|---|
| `def f(x1, x2):`<br>`  ...`<br>`  return y1, y2` | Fonction `f` d'arguments `x1. x2`, renvoie `y1, y2` |
| `for i in range(a,b,p) :`<br>`  ...` | boucle pour `i` allant de `a` à `b-1` avec un pas de `p` (`p=1` par défaut) |
| `while condition :`<br>`  ...` | boucle `tant que` ; `condition` est un booléen |
| `if cond_1 :`<br>`  ...`<br>`elif cond_2 :`<br>`  ...`<br>`else :`<br>`  ...` | On peut utiliser plusieurs `elif` successifs. |

### Booléens et opérateurs logiques

| | |
|---|---|
| `True, False` | Booléens VRAI, FAUX |
| `==, >, <, >=, <=, !=` | Tests $=, >, <, \geq, \leq, \neq$ respectivement |
| `and, or, not` | Opérateurs logiques ET, OU, NON |

### Définition de vecteurs et matrices

| | |
|---|---|
| `np.array([2,5,7])` | Vecteur (ou tableau) ${}^t(2,5,7)$ |
| `np.array([[1,2],[4,5]])` | Matrice $\begin{pmatrix}1 & 2\\ 4 & 5\end{pmatrix}$ |
| `np.arange(a,b,p)` | Vecteur des valeurs entières de `a` à `a+kp` avec le pas `p` (`p=1` par défaut), où `a+kp` est le plus proche de `b` par valeurs strictement inférieures. |
| `np.linspace(a,b,N)` | Vecteur de `N` valeurs équiréparties entre `a` et `b` inclus |
| `np.zeros(n)` | Vecteur nul de taille `n` |
| `np.zeros((m,n))` | Matrice nulle de taille `m`×`n` |
| `np.ones((m,n))` | Matrice de `1` de taille `m`×`n` |
| `np.eye(m,n)` | Matrice de type identité de taille `m`×`n` |
| `np.diag(X)` | Construction de la matrice carrée dont la (0-ème) diagonale est le vecteur `X` |
| `np.diag(X,i)` | Construction de la matrice carrée dont la `i`-ième diagonale est le vecteur `X` |

### Manipulation de vecteurs et matrices

| | |
|---|---|
| `len(X)` | Longueur d'un vecteur (ou nombre de lignes d'une matrice `X`) |
| `np.shape(A)` ou `A.shape` | Taille `m`×`n` de la matrice `A` |
| `np.shape(A)[0]` | Nombre de lignes de la matrice `A` |
| `np.shape(A)[1]` | Nombre de colonnes de la matrice `A` |
| `np.size(A)` ou `A.size` | Nombre d'éléments de la matrice `A` |
| `A(a:b,c:d)` | Extraction de la sous-matrice composée des lignes `a` à `b-1` et des colonnes `c` à `d-1` |
| `A[:,0]` | Extraction de la première colonne de `A` |
| `A[1:-1,0)` | Extraction de la première colonne de `A` excepté les premier et dernier coefficients |
| `np.diag(A)` | Extraction du vecteur formant la diagonale de `A` |
| `np.diag(A,i)` | Extraction du vecteur formant la `i`-ième diagonale de `A` |
| `A+x` | Ajoute la valeur `x` à tous les éléments de `A` |
| `A#B` où `#` $\in \{*,/,\hat{}\}$, | Opération composante par composante entre les `np.array` `A` et `B` |
| `A.dot(B)` ou `A@B` | Produit matriciel de `A` avec `B` (où `A` et `B` sont des `np.array`) |
| `A.T` ou `np.transpose(A)` | Transposée de la matrice `A` |

### Opérations d'algèbre linéaire

| | |
|---|---|
| `np.linalg.norm(X,p)` | $\|X\|_p$ (`p`$\in [1,+\infty[$ ou `p=inf`) |
| `np.linalg.norm(A,p)` | $\|A\|_p$ (norme subordonnée lorsque `p`$\in \{$`1,2,inf`$\}$,de Frobenius si `p='fro'`) |
| `np.linalg.det(A)` | Déterminant de la matrice `A` |
| `np.linalg.matrix_rank(A)` | Rang de la matrice `A` |
| `np.trace(A)` | Trace de la matrice `A` |
| `np.linalg.inv(A)` | Inverse de la matrice `A` |
| `np.linalg.solve(A,b)` | Solution du système linaire `Ax=b` |
| `np.linalg.eigvals(A)` | Valeurs propres de `A` avec multiplicité |
| `np.linalg.eig(A)` | Couple contenant les valeurs propres et les vecteurs propres de la matrice `A` (vecteurs propres rangés en colonne) |
| `np.vdot(X,Y)` | Produit scalaire entre les vecteurs `X` et `Y` |