1 Résumé rapide des bibliothèques utiles

- import numpy as np
  ↪ pour faire du calcul numérique, permet en particulier de manipuler des tableaux ainsi que des matrices
- import numpy.linalg as al
  ↪ pour faire de l'algèbre linéaire sur les matrices
- import numpy.random as rd
  ↪ pour générer des nombres aléatoires et simuler des v.a.r. aléatoires usuelles
- import matplotlib.pyplot as plt
  ↪ pour faire des représentations graphiques (suites, fonctions, données statistiques,...)
- import pandas as pd
  ↪ pour manipuler et traiter des quantités importantes de données à la manière d'un tableur

2 Structures de code élémentaires

- La boucle `for` :

```
for k in range(n):
    [Bloc d'instructions à exécuter pour chaque valeur de k]
```

  ↪ Dans ce premier exemple, k varie de 0 à $n-1$ inclus.

```
for k in range(a,b):
    [Bloc d'instructions à exécuter pour chaque valeur de k]
```

  ↪ Dans ce deuxième exemple, k varie de a à $b-1$ inclus.

- La boucle `while` :

```
while [Condition booléenne]:
    [Bloc d'instructions à répéter tant que la condition booléenne est vérifiée]
```

- La structure conditionnelle `if` :

```
if [Condition booléenne]:
    [Bloc d'instructions à exécuter si la condition booléenne est vérifiée]
else:
    [Bloc d'instructions à exécuter si la condition booléenne n'est pas vérifiée]
```

  et si il y a plus de deux cas :

```
if [Condition booléenne 1]:
    [Bloc d'instructions à exécuter si la condition booléenne 1 est vérifiée]
elif [Condition booléenne 2]:
    [Bloc d'instructions à exécuter si la condition booléenne 2 est vérifiée]
else:
    [Bloc d'instructions à exécuter si aucune des conditions booléennes précédentes
    n'est vérifiée]
```

  ↪ On peut mettre autant de `elif` que l'on souhaite.

3 Catalogue des commandes

3.1 Fonctions et scripts

Il ne faut pas confondre la syntaxe d'une fonction avec celle d'un script. La fonction, une fois définie, peut être réutilisée ailleurs autant de fois que l'on veut. Un script effectue sa tâche une et une seule fois, à chaque fois que l'on compile le code (raccourci : `F5`).

- `def nom_fonction(p_1,p_2,...,p_n):`
  ↪ En-tête d'une fonction qui prend en entrée n paramètres `p_1,p_2,...,p_n`. Les paramètres d'entrée sont fixés, on ne peut pas les modifier dans le corps de la fonction.
- `return x`
  ↪ commande qui marque la sortie d'une fonction et indique que la fonction doit dans ce cas renvoyer ce qui est contenu dans la variable `x`. Il est possible d'écrire plusieurs lignes contenant la commande `return`, par exemple dans une structure conditionnelle.
- `print(x)`
  ↪ commande qui affiche dans la console le contenue de la variable `x`. On n'utilise cette commande que dans un script, jamais dans une fonction.

3.2 Listes

- `L = []`
  ↪ crée une liste vide
- `L = [k for k in range(a,b,p)]`
  ↪ crée une liste contenant les nombres $a, a+p, a+2p, \dots, a+kp$ où k est un entier tel que $a+kp < b$ et $a+(k+1)p \geqslant b$. Les paramètres `a`, `b` et `p` doivent être des entiers.
- `L.append(a)`
  ↪ rajoute l'élément `a` à la fin de la liste `L`
- `del L[k]`
  ↪ supprime l'élément à la position `k` de la liste `L`. En particulier, `del L[0]` supprime le premier élément de la liste `L`.
- `L[k]`
  ↪ renvoie l'élément de la liste `L` situé à la position `k` ($\texttt{k} \in \mathbb{N}$) en partant du début. Attention, les éléments des listes sont numérotés à partir de 0. En particulier, `L[0]` renvoie le premier élément de la liste `L`.
- `L[-k]`
  ↪ renvoie l'élément de la liste `L` situé à la position `k` ($\texttt{k} \in \mathbb{N}^*$) en partant de la fin. En particulier, `L[-1]` renvoie le dernier élément de la liste `L`.
- `len(L)`
  ↪ renvoie la taille (ou la longueur) de la liste `L`
- `sum(L)`
  ↪ renvoie la somme des éléments contenus dans la liste `L`
- `min(L)`
  ↪ renvoie le plus petit des éléments contenus dans la liste `L`
- `max(L)`
  ↪ renvoie le plus grand des éléments contenus dans la liste `L`

3.3 Bibliothèque numpy as np

3.3.1 Constantes et fonctions

- `np.e`
  ↪ renvoie une valeur approchée de la constante e = exp(1).
- `np.pi`
  ↪ renvoie une valeur approchée de la constante π.
- `np.exp`
  ↪ fonction exponentielle.
- `np.log`
  ↪ fonction logarithme.
- `np.sqrt`
  ↪ fonction racine carrée.
- `np.abs`
  ↪ fonction valeur absolue.
- `np.floor`
  ↪ fonction partie entière inférieure.
- `np.ceil`
  ↪ fonction partie entière supérieure.
- `np.math.factorial(n)`
  ↪ renvoie la valeur de $n!$.

3.3.2 Création de tableaux et de matrices

- `np.arange(a,b,p)`
  ↪ même syntaxe que la fonction range de **Python** mais le résultat est un tableau. Le début du tableau est `a` (inclus), la fin du tableau est `b` (exclue) et le pas entre deux éléments du tableau est `p`. Les arguments `a` et `p` sont facultatifs (s'ils sont omis, `a=0` et `p=1`).
  Exemple : `np.arange(5)` renvoie le tableau `(0,1,2,3,4)`.
- `np.linspace(a,b,n)`
  ↪ renvoie un tableau contenant `n` éléments régulièrement espacés, le premier étant `a` (inclus) et le dernier étant `b` (inclus).
  Exemple : `np.linspace(0,10,11)` renvoie le tableau `(0,1,2,3,4,5,6,7,8,9,10)`.
- `np.array(L)`
  ↪ prend en paramètre une liste de listes `L` et renvoie la matrice M dont les lignes sont les listes contenues dans `L`. Plus précisément, la ligne numéro `k` de la matrice M coïncide avec la liste `L[k]`.
  Exemple : `np.array([[0, 0.7, 0.3], [0.3, 0.4, 0.3], [0.5, 0.5, 0 ]])` crée la matrice
  $$\begin{pmatrix} 0 & 0.7 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0 \end{pmatrix}$$
- `np.zeros([n,m])`
  ↪ renvoie la matrice de taille `n` × `m` remplie de 0.
- `np.ones([n,m])`
  ↪ renvoie la matrice de taille `n` × `m` remplie de 1.
- `np.eye(n)`
  ↪ renvoie la matrice identité d'ordre `n`.
- `np.shape(M)`
  ↪ renvoie un couple (n,p) où n est le nombre de lignes de M et p est le nombre de colonnes de M.
- `np.transpose(M)`
  ↪ renvoie la transposée de la matrice M.
- `np.dot(M,N)`
  ↪ renvoie la matrice `MN` (produit matriciel mathématique). Attention, ne pas confondre avec `M * N` qui renvoie une matrice obtenue comme le produit termes à termes des coefficients de `M` et `N`.

3.3.3 Fonctions sur les tableaux

- `np.mean(T)`
  ↪ renvoie la moyenne (empirique) des éléments du tableau `T`. Très utile pour calculer la moyenne empirique d'une réalisation d'un n-échantillon pour trouver une approximation de l'espérance d'une v.a.r.
- `np.var(x)`
  ↪ renvoie la variance empirique d'une série statistique `x`
- `np.std(x)`
  ↪ renvoie l'écart-type empirique d'une série statistique `x`

3.4 Bibliothèque numpy.linalg as al

- `al.inv(M)`
  ↪ renvoie l'inverse de la matrice `M` (ou un message d'erreur si `M` n'est pas inversible).
- `al.matrix_rank(M)`
  ↪ renvoie le rang de la matrice `M`
- `al.matrix_power(M,n)`
  ↪ renvoie la matrice `M` élevée à la puissance `n`
- `al.solve(M,V)`
  ↪ renvoie la solution de l'équation $MX = V$ d'inconnue X, où M est inversible.
- `al.eig(M,V)`
  ↪ renvoie les valeurs propres de la matrice `M` ainsi qu'un vecteur propre associé à chacune des valeurs propres.

3.5 Bibliothèque numpy.random as rd

3.5.1 Simulations de v.a.r. discrètes

- `rd.binomial(n,p,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{B}(n,p)$ (le paramètre d est optionnel)
- `rd.randint(a,b,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{U}(\llbracket a, b-1 \rrbracket)$ (le paramètre d est optionnel)
- `rd.choice(E,d)`
  ↪ simule un d-tirage équiprobable avec remise sur l'ensemble E où E est un tableau ou une liste (le paramètre d est optionnel)
- `rd.choice(E,d,replace=False)`
  ↪ simule un d-tirage équiprobable sans remise sur l'ensemble E où E est un tableau ou une liste (le paramètre d est optionnel)
- `rd.geometric(p,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{G}(p)$ (le paramètre d est optionnel)
- `rd.poisson(a,p)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{P}(a)$ (le paramètre d est optionnel)

3.5.2 Simulations de v.a.r. à densité

- `rd.random()`
  ↪ génère un nombre aléatoire dans $[0,1]$, *i.e.* simule une v.a.r. $X \hookrightarrow \mathcal{U}([0,1])$
- `rd.random(d)`
  ↪ génère d nombres aléatoires de manière indépendante dans $[0,1]$
- `rd.random([n,p])`
  ↪ génère un tableau à n lignes et p colonnes dont chaque case est un nombre aléatoire dans $[0,1]$. Ces nombres aléatoires sont choisis de manière indépendante.

- `rd.uniform(a,b,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{U}([a,b])$ (le paramètre d est optionnel)
- `rd.exponential(a,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{E}\left(\frac{1}{a}\right)$ (le paramètre d est optionnel)
- `rd.normal(m,s,d)`
  ↪ simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{N}(m,s)$ (le paramètre d est optionnel)

3.6 Bibliothèque matplotlib.pyplot as plt

- `plt.plot(X,Y)`
  ↪ trace les données contenues dans `Y` en ordonnées au dessus des abscisses contenues dans `X`. Peut être interprété comme le tracé du nuage de points associé à la série statistique double `(X,Y)`
- `plt.hist(X)`
  ↪ trace l'histogramme de la série statistique X
  On a alors plusieurs options possibles :
  - `plt.hist(X,cumulative = True)`
  - `plt.hist(X,density = True)`
  - `plt.hist([X,Y])`
- `plt.boxplot(X)`
  ↪ trace la boîte à moustache associée aux données contenues dans X

3.7 Bibliothèque pandas as pd

- `table.mean()`
  ↪ renvoie la liste des moyennes pour chaque colonne numérique
- `table['nom_de_la_colonne'].mean()`
  ↪ renvoie la moyenne des valeurs d'une colonne précise
- `table.std()`
  ↪ renvoie la liste des écarts-types pour chaque colonne numérique
- `table['nom_de_la_colonne'].std()`
  ↪ renvoie l'écart-type des valeurs d'une colonne précise
- `table.median()`
  ↪ renvoie la liste des médianes pour chaque colonne numérique
- `table['nom_de_la_colonne'].median()`
  ↪ renvoie la médiane des valeurs d'une colonne précise
- `table.min()`
  ↪ renvoie la liste des valeurs minimales pour chaque colonne numérique
- `table['nom_de_la_colonne'].min()`
  ↪ renvoie la valeur minimale d'une colonne précise
- idem avec `table.max()`