Mémo Python

1 Structures de bases

1.1 Fonctions

Fonction explicite

```
def nom_fonction(variables):
    (éventuelles instructions ici)
    return ...
```

Suite définie par récurrence simple

```
def nom_suite(n):
    u = # première valeur de la suite
    for i in range(n) : # adapter si besoin
        u = ... # nouvelle valeur du terme en fonction de l'ancien
    return u
```

1.2 Conditions et tests

```
if (condition) :
    instructions
elif (condition) :
    instructions
else :
    instructions
```

1.3 Boucles

Boucle bornée (for)

Permet de répéter une instruction un nombre de fois connu à l'avance.

```
for element in sequence :
    instructions
# désindenter pour terminer la boucle
```

Boucle non bornée (while)

Permet de répéter une instruction tant qu'une certaine condition est vraie (attention aux boucles infinies)

```
while (condition):
    instructions
```

Application classique : algorithme de seuil

```
def seuil(A):
    U = # première valeur de la suite
    N = 0 # ou première valeur de n
    while U < A : # ou U > A, selon les cas
        U = ... # nouvelle valeur de la suite
        N = N + 1
    return N
```

2 Listes

2.1 Accéder à des éléments

`L[pos]`	renvoie l'élément de la liste `L` située à la position `pos`. Les positions commencent à 0
`L[debut:fin]`	renvoie les éléments de la liste `L` de la position de `debut` inclue à la position `fin` exclue.
`L[:fin]`	renvoie les éléments de la liste `L` depuis son premier élément inclu jusqu'à la position `fin` exclue.
`L[debut:]`	renvoie les éléments de la liste `L` depuis la position `debut` inclue jusqu'à la fin.
`L[-1]`	renvoie le dernier élément de la liste

2.2 Range

`range(a)`	crée une "liste" contenant tous les entiers de 0 inclus à `a` exclu.
`range(a,b)`	crée une "liste" contenant tous les entiers de `a` inclus à `b` exclu.
`range(a,b,pas)`	crée une "liste" contenant tous les entiers de `a` inclus à `b` exclu en progressant de `pas` entre chaque entier.

2.3 Méthodes et fonctions sur les listes

`len(L)`	renvoie le nombre d'éléments de la liste `L`
`x in L`	renvoie `True` si l'élément `x` est dans `L` et `False` sinon
`L.append(x)`	ajoute l'élément `x` à la fin de la liste `L`
`L.pop()`	retire et renvoie le dernier élément de la liste `L`
`del L[pos]`	retire l'élément situé à la position `pos` de la liste `L`
`L[i] = x`	remplace l'élément à la position `i` de la liste `L` par la valeur `x`

3 Module `numpy`

3.1 Module classique

```
1 import numpy as np
```

Constantes mathématiques et fonctions usuelles

`np.e`	Nombre d'Euler e (soit exp(1))
`np.pi`	Constante π

Les fonctions numpy peuvent être appliquées à un nombre ou à un tableau entier. Dans ce cas, la fonction est appliquée à tous les éléments du tableau.

`np.exp`	fonction exponentielle en base e
`np.log`	fonction logarithme népérien
`np.sqrt`	fonction racine carrée
`np.floor`	fonction Partie entière
`np.abs`	fonction valeur absolue

Générations de tableaux

`np.linspace(a,b,n)`	crée un tableau de `n` valeurs allant de `a` à `b` tous deux inclus
`np.arange(a,b,p)`	crée le tableau des valeurs de `a` inclus à `b` exclus par pas de `p`

Calcul matriciel

`np.array([[...], ... [...]])`	renvoie une matrice générée ligne par ligne
`np.shape(A)`	taille du tableau `A`. Renvoie un couple `n,p` (lignes, colonnes)
`A[i,j]`	renvoie le coefficient `(i,j)` de `A`. La numérotation commence à 0.
`A[i,:], A[:,j]`	renvoie la ligne `i` de `A` (respectivement la colonne `j` de `A`)
`np.zeros((n,p))`	renvoie une matrice de taille `n` $\times$ `p` composée de zéros
`np.ones((n,p))`	renvoie une matrice de taille `n` $\times$ `p` composée de 1
`np.eye(n)`	renvoie la matrice identité de taille `n`
`np.transpose(A)`	renvoie la transposée de `A`
`np.dot(A,B)`	renvoie le produit matriciel `AB`

Statistiques

`np.sum`	renvoie la somme des éléments
`np.cumsum`	renvoie la somme cumulée des éléments
`np.min`	renvoie le plus petit élément
`np.max`	renvoie le plus grand élément
`np.mean`	renvoie la moyenne des valeurs
`np.var`	renvoie la variance des valeurs
`np.std`	renvoie l'écart-type des valeurs
`np.median`	renvoie la médiane des valeurs

Les fonctions précédentes peuvent s'appliquer à une liste seule ou à une matrice. Dans ce cas, elles sont appliquées sur chaque colonne ou chaque ligne. Par exemple, si `A` est une matrice.

- `np.mean(A)` renvoie la moyenne de tous les élements de `A` ;
- `np.mean(A,0)` renvoie la moyenne colonne par colonne des éléments de `A` ;
- `np.mean(A,1)` renvoie la moyenne ligne par ligne des éléments de `A` ;

3.2 Sous-module numpy.linalg

`import numpy.linalg as al`	importe `numpy.linalg` avec l'alias `al`
`al.inv(A)`	renvoie la matrice inverse de `A`
`al.matrix_power(A,n)`	renvoie la matrice `A` à la puissance `n`
`al.rank(A)`	renvoie le rang de la matrice `A`
`al.solve(A,B)`	résout l'équation matricielle `AX=B` d'inconnue `X`

3.3 Sous-module numpy.random

`import numpy.random as rd`	importe `numpy.random` avec l'alias `rd`
`rd.random()`	génère un nombre (flottant) au hasard de l'intervalle $]0;1[$
`rd.randint(a,b)`	génère un entier aléatoire entre `a` inclus et `b` exclu.
`rd.binomial(n,p)`	génère un entier simulé selon une loi binomiale de paramètres `n` et `p`
`rd.poisson(lambda)`	génère un entier simulé selon une loi de Poisson de paramètre `lambda`
`rd.geometric(p)`	génère un entier simulé selon une loi géométrique de paramètre `p`

Pour chaque loi, on peut générer soit un unique entier, soit une liste, soit une matrice aléatoire. Par exemple,

- `rd.binomial(n,p)` génère un unique entier selon la loi binomiale de paramètres `n` et `p` ;
- `rd.binomial(n,p,s)` renvoie une liste de `s` entiers générés selon la loi binomiale de paramètres `n` et `p` ;
- `rd.binomial(n,p, [s,t])` renvoie une matrice de taille `s` $\times$ `t` d'entiers générés selon la loi binomiale de paramètres `n` et `p` ;

4 Module `matplotlib.pyplot`

```
1 import matplotlib.pyplot as plt
```

Méthodes utiles

`plt.plot(abs, ord)`	place et relie les points dont les abscisses sont données par la liste `abs` et les ordonnées sont données par la liste `ord`
`plt.hist(list, bins = ...)`	trace l'histogramme où les valeurs des listes sont regroupés en classes. Le paramètres `bins` peut être un entier qui désigne le nombre de classes ou la liste des valeurs extrêmes des classes.
`plt.bar(abs, ord)`	trace le diagramme en barre ou `abs` est la liste des modalités et `ord` les effectifs associés
`plt.boxplot(liste)`	trace le diagramme en boîte associée à la liste `liste`
`plt.show()`	affiche le graphique en cours
`plt.close()`	ferme la fenêtre du graphique

5 Module `pandas`

```
1 import pandas as pd
```

5.1 Import d'un tableau

`df=pd.read_csv(fichier)`	importe le fichier csv `fichier` dans un dataframe `df`
`df=pd.read_excel(fichier)`	importe le fichier excel `fichier` dans un dataframe `df`

5.2 Exploitation du tableau

`df`	aperçu du dataframe
`df.head(n)`, `df.tail(n)`	affiche les `n` premières (ou dernières) lignes du dataframe
`df.shape`	taille du tableau
`df.info()`	affiche les colonnes et le type de chacune
`df[colonne]`	affiche les données de la colonne `colonne`. Attention, si le nom de la colonne est une chaîne de caractères, on écrit son nom entre guillemets
`df[colonne1, colonne2...]`	affiche les données de chaque colonne en entrée
`df.sort_values(colonne)`	trie les données selon la colonne `colonne`

5.3 Filtres

`df[ df[colonne] == 3 ]`	sélectionne les lignes pour lesquelles la valeur de `colonne` est 3.
`df[ (filtre1) & (filtre2)]`	applique `filtre1` et `filtre2` au dataframe `df`
`df[ (filtre1) | (filtre2)]`	affiche les lignes respectant `filtre1` OU `filtre2` dans`df`

5.4 Statistiques

`df.describe()`	affiche un résumé statistique pour chaque colonne
`df.mean()`	affiche les moyennes colonne par colonne
`df.std()`	affiche les écarts-types, colonne par colonne
`df.median()`	affiche la médiane, colonne par colonne
`df.count()`	affiche le nombre d'entrées, colonne par colonne