Abderrahmane Fadil

Mathématiques et statistiques appliquées avec Python

Cours et exercices corrigés



ellipses

Mathématiques et statistiques appliquées avec Python

Cours et exercices corrigés

Références sciences

Mathématiques et statistiques appliquées avec Python

Cours et exercices corrigés

Abderrahmane Fadil

ellipses

Collection Références sciences

dirigée par Paul de Laboulaye
paul.delaboulaye@editions-ellipses.fr

Retrouvez tous les livres de la collection et des extraits sur www.editions-ellipses.fr

ISBN 9782340-104280
Dépôt légal : mai 2025
©Ellipses Édition Marketing S.A.
8/10 rue la Quintinie 75015 Paris

DANGER LE PHOTOCOPILLAGE TUE LE LIVRE

Le Code de la propriété intellectuelle et artistique n'autorisant, aux termes des alinéas 2 et 3 de l'article L. 122-5, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Avant-propos

Les mathématiques constituent les bases fondamentales des avancées scientifiques et technologiques de tous les temps. Elles ont permis le développement de modèles formels et de méthodes rigoureuses afin d'appréhender les situations réelles et la résolution des problèmes dans divers domaines : scientifiques, économiques, biologiques, logistiques et bien d'autres.

Le besoin croissant de mettre en œuvre ces méthodes à travers des disciplines, également basées sur les mathématiques, a permis ainsi le développement de l'algorithmique, des environnements de développement, de la gestion et l'analyse de données, etc. Dans cette dynamique, on aboutit à ce jour à des avancées considérables en data sciences, en Deep Learning et en intelligence artificielle.

Ce livre s'adresse à des étudiants et élèves-ingénieurs en :

- licence dans des filières scientifiques ;
- écoles de commerce ;
- cycle préparatoire des écoles d'ingénieurs ;
- formation en data sciences ;
- auto-formation.

Ce livre a pour objectif d'aborder des chapitres de mathématiques et de statistiques descriptives :

- rappels des bases et des concepts ;
- méthodes de résolution ;
- pratiques avec développement de scripts Python correspondants.

Le choix de Python est motivé par les facilités d'apprentissage et de mise en œuvre. Il contient des librairies riches et étoffées. Il s'est également et naturellement imposé dans les développements en data sciences et intelligence artificielle.

Table des matières

Chapitre 1
Notions de base du développement Python **13**
1 Introduction 13
2 Script Python 13
2.1 Représentation systémique 14
2.2 Script Python 15
2.3 Résultat 16
3 Les instructions fondamentales 17
3.1 L'affectation 17
3.2 Les entrées / sorties 17
3.3 Les structures de contrôle conditionnelles 18
3.3.1 Syntaxes 18
3.3.2 Exemple 19
3.4 Les structures de contrôle répétitives 21
3.4.1 La boucle for 21
3.4.2 Exemple de la suite de Catalan 22
3.4.3 La boucle while 23
3.4.4 Exemple de script de test statistique avec while 23
3.4.5 Test statistique 24
4 Les structures de données 25
4.1 Les listes 26
4.1.1 Définition et caractéristiques 26
4.1.2 Création et initialisation de listes 26
4.1.3 Opérations sur liste 27
4.2 Les tableaux 28
4.2.1 Définition et caractéristiques 28
4.2.2 Création et initialisation de tableaux 29
4.2.3 Opérations sur les tableaux 31
4.3 Les tuples 32
4.3.1 Définition et caractéristiques 32
4.3.2 Création et initialisation d'un tuple 33
4.3.3 Opérations sur les tuples 33
4.4 Les ensembles 35
4.4.1 Définition et caractéristiques 35
4.4.2 Création et initialisation d'un ensemble 36
4.4.3 Opérations sur les ensembles 36

4.5 Les dictionnaires 39
4.5.1 Définition et caractéristiques 39
4.5.2 Création et initialisation d'un ensemble 39
4.5.3 Opérations sur les dictionnaires 40

Chapitre 2
Calcul scientifique 43
1 Définition et généralités 43
1.1 Définition 43
1.2 Importation 44
1.3 Utilisation 44
1.4 Exemple 45
2 Rappel sur les ensembles numériques 46
3 Librairie MATH 47
4 Librairie cMATH 49
5 Librairie FRACTIONS 50
6 Calcul scientifique ave la librairie SCIPY 52
6.1 Les sous-librairies scipy 52
6.2 Les fonctions 53
6.3 Résolution d'un programme linéaire par la méthode du simplex 54
6.3.1 Programme linéaire : script de résolution 55
6.3.2 Résultat 56
6.4 Résolution d'un programme d'affectation linéaire 56
6.4.1 Méthode hongroise : script de résolution 57
6.4.2 Résultat 58
6.5 Résolution d'équations différentielles 58
6.5.1 Équation différentielle 59
6.5.2 Implémentation 59
6.5.3 Test 60

Chapitre 3
Visualisation graphique des données 63
1 Visualisation des données 63
2 Les fonctions de PYPLOT 63
2.1 Syntaxe 64
2.2 Catégories de fonctions 64

3 Les graphiques 2D 64
3.1 Création 64
3.1.1 Exemple 1 de fonction définie par intégrale 65
3.1.2 Visualisation de l'exemple 1 65
3.2 Les options de la fonction plot() 66
3.2.1 Exemple 2 de fonction définie par intégrale 66
3.2.2 Visualisation de l'exemple 2 66
3.3 Autres fonctions 67
3.3.1 Exemple précédent 67
3.3.2 Résultat 68
3.4 Plusieurs courbes 69
3.4.1 Exemple 69
3.4.2 Résultat 70
4 Les graphiques 3D 72
4.1 Courbe filaire en 3D 72
4.1.1 Exemple 72
4.1.2 Visualisation d'une courbe filaire 73
4.2 Surface en 3D 73
4.2.1 Exemple 74
4.2.2 Visualisation d'une surface en 3D 75
5 Autres types de graphiques 75
5.1 Nuage de points 76
5.1.1 Exemple 76
5.1.2 Résultat 77
5.2 Histogramme 77
5.2.1 Exemple 78
5.2.2 Résultat 79
5.3 Lignes de niveau 81
5.3.1 Exemple : les cercles concentriques 81
5.3.2 Résultat 82
5.4 Graphique circulaire 85
5.4.1 Exemple 1 85
5.4.2 Résultat de l'exemple 1 86
5.4.3 Exemple 2 86
5.4.4 Résultat de l'exemple 2 87
5.4.5 Exemple 3 87
5.4.6 Résultat de l'exemple 3 88

5.5 Graphique en beignet 89
5.5.1 Exemple 89
5.5.2 Résultat 90
5.5.3 Image d'une matrice 90
5.5.4 Exemple 90
5.5.5 Résultat 91
5.6 Graphique en radar 92
5.6.1 Exemple 92
5.6.2 Résultat 94
5.7 Diagramme en boîte 94
5.7.1 Exemple 1 94
5.7.2 Résultat de l'exemple 1 95
5.7.3 Exemple 2 95
5.7.4 Résultat de l'exemple 2 96

Chapitre 4
Nombres complexes et mathématiques symboliques 97
1 Structure algébrique 97
1.1 Produit cartésien 97
1.2 Loi de composition interne 97
1.3 Groupe 97
1.4 Corps commutatif 98
2 Corps des nombres réels 98
2.1 Partie entière 98
2.1.1 Définition 98
2.1.2 Propriétés 99
2.2 Valeur absolue 99
2.2.1 Définition 99
2.2.2 Propriétés 99
3 Corps des nombres complexes 99
3.1 Définition 99
3.2 Propriété 100
3.3 Partie réelle, partie imaginaire 100
3.3.1 Définitions 100
3.3.2 Propriétés 101
3.4 Module et argument 101
3.4.1 Définition : module 101
3.4.2 Propriétés : module 102
3.4.3 Définition : argument 102
3.4.4 Propriétés : argument 103

3.5 Interprétation géométrique 104
3.6 Racine n^{ième} d'un nombre complexe 104
3.6.1 Propriété 105
3.6.2 Résolution de l'équation 105
3.7 Racines cubiques de 1 105
3.7.1 Définition 105
3.7.2 Remarques 105
3.8 Fonctions complexes 106
3.8.1 Définition d'une fonction complexe 106
3.8.2 Visualisation d'une fonction complexe 106
4 Mathématiques symboliques et calcul polynomial 109
4.1 Importation de sympy 109
4.2 Création d'une variable 109
4.3 Écriture d'une expression 110
4.4 Évaluer une expression 111
4.5 Développement d'une expression 112
4.6 Factorisation d'une expression 113
4.7 Simplification d'une expression 114
4.8 Séparation de variables 115
4.9 Division polynomiale 116
4.10 Définition de fonction 118
4.10.1 Transformation d'une expression en fonction 118
4.10.2 Définition de fonction symbolique 118
4.11 Dérivée d'une fonction ou d'une expression 119
4.12 Primitive et intégrale d'une fonction 121
4.12.1 Définition de primitive 122
4.12.2 Propriété 122
4.12.3 Définition intégrale 122
4.12.4 Méthode integrate() 123
4.13 Limite d'une fonction 124
4.14 Résolution d'équations 125
4.14.1 Résolution d'équations algébriques 126
4.14.2 Résolution d'équations différentielles 128
4.15 Résolution d'équations linéaires en écriture matricielle 132

Chapitre 5
Éléments de statistiques descriptives **141**

1 Définitions 141
1.1 Population 141
1.2 Échantillon 141
1.3 Variable statistique 141
1.4 Variable quantitative 142
1.4.1 Variable quantitative continue 142
1.4.2 Variable quantitative discrète 142
1.5 Variable qualitative 142
1.5.1 Variable qualitative ordinale 143
1.5.2 Variable qualitative nominale 143
1.6 Statistiques descriptives 143
2 Mesures 143
3 Implémentation 144
3.1 Moyenne, variance et écart-type 144
3.1.1 Script 144
3.1.2 Résultat 146
3.2 Résumé statistique et matrice de corrélation 146
3.2.1 Script 146
3.2.2 Résultat 148

Chapitre 6
Éléments d'algèbre linéaire **153**

1 Introduction 153
2 Matrice des liaisons en trafic urbain 153
3 Calcul matriciel 154
3.1 Définition matrice 154
3.2 Exemple 155
3.3 Taille d'une matrice 155
3.4 Vecteur 155
3.5 Somme de deux matrices 156
3.6 Produit matriciel 156
3.7 Transposée d'une matrice 157
3.8 Produit scalaire 157
3.9 Mineur 157
3.10 Cofacteurs et comatrice 158

3.11 Déterminant d'une matrice 159
3.12 Matrice inverse 159
3.13 Matrice élémentaire 160
3.14 Matrice triangulaire 162
3.15 Matrice diagonale 163
3.16 Autres notions 163
4 Les tableaux et matrices avec la librairie NUMPY 164
4.1 Syntaxe 164
4.2 Création de tableaux 164
4.2.1 Création d'un tableau avec des données saisies dans le code 165
4.2.2 Création d'un tableau vide 166
4.2.3 Transformation d'une liste en un tableau 166
4.2.4 Création d'un tableau avec ones() ou zeros() 168
4.2.5 Création d'un tableau par copie d'un autre tableau 169
4.2.6 Importation d'un tableau sauvegardé dans un fichier 171
4.2.7 Création d'un tableau à partir d'un range 172
4.2.8 Création d'un tableau par linspace 173
4.3 Opérations sur les tableaux 173
4.3.1 Taille d'un tableau 173
4.3.2 Changement de taille d'un vecteur ou d'une matrice 175
4.3.3 Transformation d'une matrice en un tableau 1D 176
4.3.4 Le slicing d'une matrice 177
4.3.5 Image miroir d'une matrice 180
4.3.6 Scinder un tableau 181
5 Algèbre linéaire et calcul matriciel avec NUMPY 184
5.1 Opérations et calcul matriciel avec numpy 184
5.2 Résolution de A.X = B par pivot de Gauss 196
5.2.1 Méthode de résolution 196
5.2.2 Script de résolution 198
5.2.3 Exemple 200
5.3 Calcul du déterminant et matrice inverse 202
5.3.1 Méthode de calcul 202
5.3.2 Script de calcul 203
5.3.3 Exemples 206
5.4 Suites de matrices et suites numériques 208
5.4.1 Énoncé 208
5.4.2 Script de résolution 209
5.4.3 Exemples 211

Bibliographie 215
Index 217

Chapitre 1

Notions de base du développement Python

1 Introduction

Avant d'aborder les outils et méthodes de visualisation de données (graphiques, courbes, …), de mise en application de fondements mathématiques et statistiques (algèbre linéaire, statistiques descriptives, …), d'importation/exportation de structures de données complexes (tableaux, matrices, *tensors*, …) ou encore l'introduction au *Machine Learning* (analyse de données, …) , il est utile de faire quelques rappels des notions de bases du développement Python.

2 Script Python

Un script informatique est une séquence d'instructions, écrites dans un langage dit de scripts et qui fait appel à un outil dit interpréteur afin d'exécuter les instructions. L'interpréteur est souvent associé à l'éditeur de code, qui permet d'exécuter la séquence d'instructions en interprétant les instructions l'une après l'autre de façon séquentielle. L'interpréteur ne crée pas un exécutable, contrairement à un compilateur et c'est là la différence majeure avec un langage compilé comme le C++ par exemple, et donc pour toute nouvelle exécution du script l'interpréteur est appelé et refait le même travail, à savoir lecture et interprétation des instructions l'une après l'autre de façon séquentielle.

Un programme écrit sous Python est un script et a besoin donc d'un programme auxiliaire, dit interpréteur, pour lire et exécuter au fur et à mesure les instructions du script.

Un script Python peut utiliser des fonctions et procédures prédéfinies, ce qui facilite la tâche du développeur. Ces fonctions sont classées par catégories et regroupées sous des librairies importables. Pour les éléments de langage, ces librairies sont dites aussi des modules, des bibliothèques ou encore des packages.

Pour pouvoir importer ces librairies et utiliser leurs fonctions prédéfinies il faut les avoir auparavant installées sous l'éditeur Python utilisé pour le développement.

Un script doit tenir compte également des contraintes d'environnement telles que la source de données lors d'importation de données ou encore le modèle de données lors d'une étape d'apprentissage par exemple en *Deep Learning*.

2.1 Représentation systémique

La *figure 1* ci-après est une représentation systémique d'un script. En effet, une séquence d'instructions est définie afin de réaliser un traitement de données.

Ce traitement agit sur des données en entrée e_1, e_2 , …, e_n où n est le nombre de ces données en entrée (c'est-à-dire les données qui alimentent le script).

Le traitement utilise ces données pour en produire d'autres. À la fin du traitement, ces données produites sont les données en sortie du script s_1, s_2, …, s_m où m est le nombre de ces données en sortie qui correspondent aux résultats escomptés du traitement (cf. *figure 1* ci-dessous).

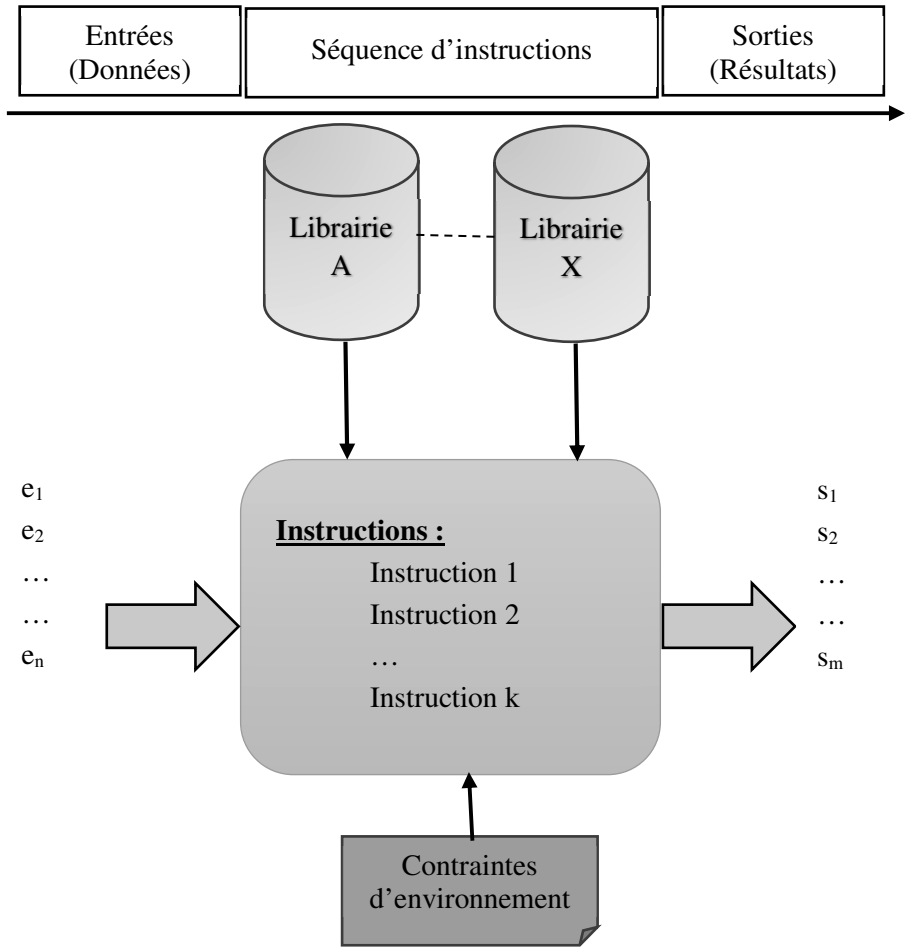


Figure 1. *Script modèle*

Voici, ci-dessous un exemple où on retrouve tous les éléments du script modèle, à savoir :

- importation de librairies ;
- contraintes d'environnement comme l'importation de données à partir d'un fichier texte ;
- les entrées / sorties ;
- les instructions de traitement.

2.2 Script Python

```
import numpy as np
import os

# Création de la matrice A
A=np.zeros(shape=(3,3))

# Chargement des données à partir d'une source externe
os.chdir("C:\\Exemples\Exemples Maths Data")
A=np.loadtxt("Matrice_ load.txt",delimiter="\t",dtype=float)
print("A = \n",A)

# taille et transpose d'une matrice
tA=np.zeros(shape=(A.shape[1],A.shape[0]))
tA=np.transpose(A)
print("tA = \n",tA)

# Déterminant d'une matrice carrée
d=np.linalg.det(A)
print("d = ",d)

# Valeurs propres et vecteurs propres
M=np.random.random(size=(2,2))*10
print("M = \n",M)
# vvp : vecteur des valeurs propres
# mvp : matrice des vecteurs propres (vecteurs colonnes)
vvp, mvp=np.linalg.eig(M)
```

```
print("vvp = ", vvp)
print("mvp = \n", mvp)

# M.v = l*v où v est un vecteur propre et l la valeur propre
correspondante
for i in range(M.shape[1]):
    vi=mvp[:,i]
    li=vvp[i]
    print("vérification n°", i+1, " : ",
          np.all(np.isclose(np.dot(M,vi),li*vi)))
```

2.3 Résultat

```
A =
 [[ 45.   62.  168. ]
  [-34.   15.5 187. ]
  [  1.   -1.2   0. ]]

tA =
 [[ 45.  -34.    1. ]
  [ 62.   15.5  -1.2]
  [168.  187.    0. ]]

d =  25942.4

M =
 [[2.45 1.87]
  [9.04 7.68]]

vvp =  [0.19 9.94]
mvp =
 [[-0.63 -0.24 ]
  [ 0.76 -0.97]]

vérification n° 1  :  True
vérification n° 2  :  True
```

3 Les instructions fondamentales

3.1 L'affectation

L'opération d'affectation permet d'affecter à une variable une valeur constante, ou le résultat d'une expression, ou le résultat d'une fonction, en utilisant l'opérateur = où le receveur est à gauche de l'opérateur et la valeur à affecter est à droite.

Exemples

- y = 20 ;
- y = (x**2 +1) ;
- y = numpy.sin((np.pi /3) * x) ;
- etc.

3.2 Les entrées / sorties

Les fonctions d'entrées / sorties qui permettent la lecture et l'écriture de données sont les suivantes :

Fonction	Syntaxe	Remarques et exemples
input()	v = input([*message*])	C'est l'instruction de saisie. v : est une variable qui reçoit la valeur saisie. [] : signifie facultatif. *message* : est un message à destination de l'utilisateur. Exemple : x = input("Saisir x : ")
print()	print(*arguments*)	C'est l'instruction d'affichage. *arguments* : la liste des arguments à afficher. Le séparateur des arguments est **,** (la virgule). Exemple : print("v = ", round(v, 2))

Exemple d'affectation et d'entrées / sorties

```
from math import sqrt

x=float(input("Saisir la valeur réelle de x : "))

# Instructions d'affectation
y=x**3
v=sqrt(y)

print("v = ",round(v,2))
```

Résultat

```
v =  111.18
```

3.3 Les structures de contrôle conditionnelles

Une structure de contrôle conditionnelle permet, selon le résultat de test d'une ou plusieurs conditions, à un script :

- de réaliser un bloc d'instructions lorsque une condition est vraie puis l'interpréteur passe à la suite du script ;
- de réaliser un bloc d'instructions lorsque une condition est vraie sinon un autre bloc d'instructions est exécuté puis l'interpréteur passe à la suite du script ;
    - de réaliser un bloc d'instructions parmi plusieurs blocs possibles (bifurcation).

Cela correspond à trois syntaxes du contrôle conditionnel.

3.3.1 Syntaxes

Syntaxe 1	**Remarque**
if Condition : *Bloc instructions*	Si *Condition* est vraie alors un bloc d'instructions est exécuté, sinon l'interpréteur passe à la suite du script.