

Editions ENI

Algorithmique

Techniques fondamentales de programmation

**Exemples en Python (nombreux exercices corrigés)
BTS, DUT informatique**

(2^e édition)

Collection
Ressources Informatiques

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **RI2PYALG** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1 Introduction à l'algorithmique

1. Les fondements de l'informatique	13
1.1 Architecture de Von Neumann	13
1.2 La machine de Turing	17
1.3 Représentation interne des instructions et des données	19
1.3.1 Le binaire	19
1.3.2 Les octets et les mots	22
1.3.3 L'hexadécimal	23
2. L'algorithmique	24
2.1 Programmer, c'est un art	24
2.2 Définition : L'algorithme est une recette	26
2.3 Pourquoi utiliser un algorithme ?	27
2.4 Le formalisme	28
2.4.1 Les algorigrammes	29
2.4.2 L'algorithme sous forme de texte	30
2.5 La complexité	32
2.6 Les structures algorithmiques	35
3. Les langages d'implémentation	36
3.1 Quel langage ?	36
3.2 Classifications des langages	39
3.2.1 Haut niveau, bas niveau	39
3.2.2 Diverses classifications	40
3.2.3 Compilé ou interprété	41

2 Algorithmique

Techniques fondamentales de programmation (exemples en Python)

3.3	La machine virtuelle	42
3.4	Python	44
3.4.1	Les avantages	44
3.4.2	Un premier programme Python	47
4.	Exercices	48

Chapitre 2 **Les variables et opérateurs**

1.	La variable	51
1.1	Principe	51
1.2	Déclaration	54
1.3	Les types	54
1.3.1	Les nombres	56
1.3.2	Autres types numériques	58
1.3.3	Les caractères	59
1.3.4	Le type booléen	61
1.4	Affectation	63
1.4.1	Affectation de valeurs	63
1.4.2	Affectation de variables	66
1.5	Saisie et affichage	67
1.5.1	La fonction print()	68
1.5.2	La fonction input()	69
1.6	Les constantes	70
2.	Opérateurs et calculs	71
2.1	Les affectations	71
2.2	Les opérateurs arithmétiques	71
2.3	Les opérateurs booléens	76
2.4	Les opérateurs de comparaison	79
2.4.1	L'égalité	80
2.4.2	La différence	81
2.4.3	Inférieur, supérieur	82

- 2.5 Le cas des chaînes de caractères 83
- 2.6 La précedence des opérateurs 84
- 3. Pour aller plus loin 85
 - 3.1 Les nombres négatifs 85
 - 3.2 La représentation des nombres réels 87
 - 3.3 Les dates 92
 - 3.4 Les caractères 93
- 4. Types et langages 95
 - 4.1 Langages typés ou non 95
 - 4.2 La gestion de la mémoire 96
- 5. Les types spécifiques à Python 97
 - 5.1 Les listes 97
 - 5.2 Les tuples 99
 - 5.3 Le type set 101
 - 5.4 Les dictionnaires 102
- 6. Exercices 103

Chapitre 3
Tests et logique booléenne

- 1. Les tests et conditions 107
 - 1.1 Principe 107
 - 1.2 Que tester ? 109
 - 1.3 Tests SI 111
 - 1.3.1 Forme simple 111
 - 1.3.2 Forme complexe 113
 - 1.4 Tests imbriqués 116
 - 1.5 Choix multiples 120
 - 1.6 Des exemples complets 122
 - 1.6.1 Le lendemain d’une date 122
 - 1.6.2 La validité d’une date 126
 - 1.6.3 L’heure dans n secondes 127

4 Algorithmique

Techniques fondamentales de programmation (exemples en Python)

2.	L'algèbre booléen	131
2.1	L'origine des tests	131
2.2	Petites erreurs, grosses conséquences	132
2.2.1	Ariane 5	133
2.2.2	Mars Climate Orbiter	133
2.3	George Boole	134
2.4	L'algèbre	135
2.4.1	Établir une communication	135
2.4.2	La vérité	137
2.4.3	La loi ET	137
2.4.4	La loi OU	138
2.4.5	Le contraire	139
2.4.6	Les propriétés	139
2.4.7	Quelques fonctions logiques	143
2.4.8	Avec plus de deux variables	146
2.5	Une dernière précision	149
3.	Exercices	150

Chapitre 4 Les boucles

1.	Les structures itératives	153
1.1	Définition	153
1.2	Quelques usages simples	154
2.	Tant Que	155
2.1	Structure générale	155
2.2	Boucles infinies et "break"	156
2.3	Des exemples	158
2.3.1	Une table de multiplication	158
2.3.2	Une factorielle	160
2.3.3	x à la puissance y	161
2.3.4	Toutes les tables de multiplication	163

2.3.5	Saisie de notes et calcul de moyennes	165
2.3.6	Rendez la monnaie	172
2.3.7	Trois boucles	176
3.	Répéter ... Jusqu'à	178
3.1	Différences fondamentales.	178
3.2	Quelques exemples adaptés	180
3.2.1	La factorielle	180
3.2.2	Les trois boucles	180
4.	Pour ... Fin Pour	181
4.1	Une structure pour compter...	181
4.2	... mais pas indispensable	182
4.3	Quelle structure choisir ?	182
4.4	Un piège à éviter	183
4.5	Quelques exemples	184
4.5.1	De nouveau trois boucles	184
4.5.2	La factorielle	186
4.5.3	Racine carrée avec précision.	187
4.5.4	Calcul du nombre PI.	189
5.	Exercices	192

Chapitre 5
Les tableaux et structures

1.	Présentation	195
1.1	Principe et définition	195
1.1.1	Simplifier les variables	195
1.1.2	Les dimensions	197
1.1.3	Les types	198
1.1.4	Déclaration	199
1.1.5	Utilisation	200
1.1.6	Les tableaux dynamiques.	200
1.2	Python et les tableaux	202

6 **Algorithmique**

Techniques fondamentales de programmation (exemples en Python)

1.3	Représentation en mémoire	207
1.3.1	Représentation linéaire	207
1.3.2	Représentation par référence	209
2.	Manipulations simples	211
2.1	Recherche d'un élément	211
2.2	Le plus grand/petit, la moyenne	214
2.3	Le morpion	216
3.	Algorithmes avancés	221
3.1	Les algorithmes des tris	221
3.1.1	Le principe	221
3.1.2	Le tri par création	222
3.1.3	Le tri par sélection	222
3.1.4	Le tri à bulles	225
3.1.5	Le tri par insertion	229
3.1.6	Le tri Shell	232
3.2	Recherche par dichotomie	235
4.	Structures et enregistrements	238
4.1	Principe	238
4.2	Déclaration	239
4.2.1	Type structuré	239
4.2.2	Enregistrement	240
4.3	Utiliser les enregistrements	242
4.3.1	Utiliser les champs	242
4.3.2	Un enregistrement dans une structure	244
4.3.3	Un tableau dans une structure	245
4.4	Les tableaux d'enregistrements	247
4.4.1	Les tables	247
4.4.2	Une table comme champ	248
4.5	Et Python ?	249
5.	Exercices	250

Chapitre 6
Les sous-programmes

- 1. Présentation 253
 - 1.1 Principe 253
 - 1.2 Déclaration et définition 255
 - 1.2.1 Dans un algorithme 255
 - 1.2.2 En Python 257
 - 1.3 Appel 258
 - 1.4 Fonctions et procédures 260
 - 1.4.1 Les procédures 260
 - 1.4.2 Les fonctions 261
 - 1.5 Variables locales et globales 263
 - 1.5.1 Locales 263
 - 1.5.2 Globales 264
 - 1.5.3 Variables globales et Python 266
 - 1.6 Les paramètres 266
 - 1.6.1 Les procédures 267
 - 1.6.2 Les fonctions 269
 - 1.6.3 Paramètres et Python 271
 - 1.6.4 Petite application fonctionnelle 274
 - 1.7 Sous-programmes prédéfinis 276
 - 1.7.1 Un choix important 276
 - 1.7.2 Quelques exemples 277
 - 1.8 Dernier cas : les tableaux 284
- 2. Les sous-programmes récursifs 287
 - 2.1 Principe 287
 - 2.2 Un premier exemple : la factorielle 288
 - 2.3 Un exemple pratique : les tours de Hanoï 291
- 3. Exercices 293

Chapitre 7 **Les fichiers**

1. Les différents fichiers	295
1.1 Préambule	295
1.2 Problématique	296
1.3 Définition	297
1.4 Les formats	297
1.4.1 Types de contenus	297
1.4.2 Le fichier binaire	299
1.4.3 Le fichier texte	300
1.4.4 Quel format utiliser ?	302
1.5 Les accès aux fichiers	303
1.5.1 Séquentiel	303
1.5.2 Accès direct	304
1.5.3 Indexé	304
1.5.4 Autre ?	304
2. Les enregistrements	305
2.1 Les délimiteurs	305
2.2 Largeur fixe	308
2.3 Principes d'accès	309
2.3.1 Étapes de base	309
2.3.2 Identificateurs de fichiers et canaux	310
2.3.3 Les modes d'ouverture	312
3. Fichier texte séquentiel	313
3.1 Ouvrir et fermer un fichier	313
3.2 Lire et écrire des enregistrements	314
3.2.1 Lecture	314
3.2.2 Écriture	316
3.3 Les enregistrements structurés	320
3.4 Exemple en Python	322

- 4. Les fichiers binaires 325
 - 4.1 Nouvelles instructions 325
 - 4.2 Exemple 325
- 5. Exercices 327

Chapitre 8
Notions avancées

- 1. Les pointeurs et références 329
 - 1.1 Rappels sur la mémoire et les données 329
 - 1.1.1 Structure de la mémoire 329
 - 1.1.2 Python : des limites qui n'en sont pas 331
 - 1.1.3 Brefs exemples en C 332
 - 1.2 Le pointeur 332
 - 1.2.1 Principe et définition 332
 - 1.2.2 Le C roi des pointeurs 334
 - 1.2.3 Applications 335
 - 1.3 Notation algorithmique 338
 - 1.3.1 Déclarer et utiliser les pointeurs 338
 - 1.3.2 Allocation dynamique 341
 - 1.4 Python et les références 343
 - 1.4.1 Différences entre le C et Python 343
 - 1.4.2 Références sur les objets 343
 - 1.4.3 Les types primitifs 346
 - 1.4.4 Références sur structures 346
 - 1.4.5 Le piège en Python 348
- 2. Les listes chaînées 349
 - 2.1 Listes chaînées simples 349
 - 2.1.1 Principe 349
 - 2.1.2 Création 353
 - 2.1.3 Parcours de la liste 355
 - 2.1.4 Recherche 355
 - 2.1.5 Ajout d'un élément 357

10 **Algorithmique**

Techniques fondamentales de programmation (exemples en Python)

2.1.6	Suppression d'un élément	360
2.1.7	Supprimer toute la liste	363
2.1.8	Parcours récursif	364
2.2	L'implémentation en Python	364
2.3	Autres exemples de listes	368
2.3.1	Listes circulaires	368
2.3.2	Listes d'éléments triés	368
2.3.3	Listes doublement chaînées	368
2.3.4	Files et piles	369
3.	Les arbres	370
3.1	Principe	370
3.2	Définitions	372
3.2.1	Base	372
3.2.2	Terminologie	372
3.2.3	Description horizontale	373
3.2.4	Description verticale	373
3.2.5	L'arbre binaire	373
3.3	Parcours d'un arbre	374
3.4	Arbre binaire ordonné	377
3.4.1	Principe	377
3.4.2	Recherche d'un élément	377
3.4.3	Ajout d'un élément	379
3.4.4	Suppression d'un nœud	380
4.	Exercices	381

Chapitre 9

Une approche de l'objet

1.	Principe de l'objet, une notion évidente	383
1.1	Avant de continuer	383
1.2	Rappels sur la programmation procédurale	384
1.2.1	Les données	384
1.2.2	Les traitements	385

1.3	L'objet	385
1.3.1	Dans la vie courante	385
1.3.2	En informatique	387
1.4	Classe, objets	391
1.5	Déclaration et accès	392
1.6	Les méthodes	394
1.7	Portée des membres	395
1.8	Encapsulation des données	397
1.9	L'héritage	399
1.9.1	Principe	399
1.9.2	Commerce	401
1.9.3	Hiérarchie	402
1.9.4	Simple ou multiple	403
1.10	Le polymorphisme	404
1.10.1	Principe	404
1.10.2	Le polymorphisme ad hoc	404
1.10.3	Le polymorphisme d'héritage	405
1.10.4	Le polymorphisme paramétrique	407
2.	Manipuler les objets	408
2.1	Les constructeurs	408
2.1.1	Déclaration	408
2.1.2	Appel implicite	409
2.1.3	L'héritage	411
2.2	Les destructeurs	413
2.3	Les membres statiques	414
2.4	Classes et méthodes abstraites	416
2.5	Interfaces	419
3.	L'objet en Python	421
3.1	Les langages objet	421
3.2	Déclaration des classes et objets	422
3.3	Héritage	425
3.4	Interfaces	426

12 _____ Algorithmique

Techniques fondamentales de programmation (exemples en Python)

4. Exercices 429

Chapitre 10 **Corrigés des exercices**

1. Introduction à l'algorithmique. 431
2. Les variables et opérateurs 435
3. Tests et logique booléenne. 442
4. Les boucles 450
5. Les tableaux et structures 464
6. Les sous-programmes 471
7. Les fichiers. 477
8. Notions avancées 483
9. Une approche de l'objet 486

Index 495

Editions ENI

Python 3

Les fondamentaux du langage

(2^e édition)

Collection
Ressources Informatiques

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RI23PYT** dans la zone de recherche
et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Contenu de l'ouvrage	31
2. Progressivité de l'ouvrage	32
3. À destination des enseignants et élèves	33
4. À destination des chercheurs ou doctorants	34
5. À destination de ceux qui viennent d'un autre langage	35

Partie 1 : Les atouts de Python

Chapitre 1.1

Python dans le paysage informatique

1. Petite histoire des langages informatiques	37
1.1 Informatique théorique	37
1.2 Chronologie de l'informatique	37
1.2.1 Évolutions des problématiques liées à l'informatique	37
1.2.2 Chronologie des langages informatiques	39
1.3 Histoire de Python	43
1.3.1 La genèse	43
1.3.2 Extension du périmètre fonctionnel	43
1.3.3 Évolution de la licence	44
1.3.4 Avenir	44
2. Typologie des langages de programmation	45
2.1 Paradigmes	45
2.1.1 Définition	45
2.1.2 Paradigme impératif et dérivés	46
2.1.3 Paradigme objet et dérivés	46
2.1.4 Programmation orientée aspect	47
2.1.5 Paradigme fonctionnel	47
2.1.6 Paradigme logique	47

2.1.7	Programmation concurrente	48
2.1.8	Synthèse	48
2.2	Interopérabilité	49
2.3	Niveau de programmation	50
2.3.1	Machine	50
2.3.2	Bas niveau	51
2.3.3	Haut niveau	51
2.4	Typage	52
2.4.1	Faible vs fort	52
2.4.2	Statique vs dynamique	52
2.5	Grammaire	53
2.5.1	Langages formels	53
2.5.2	Syntaxe	54
3.	Python et le reste du monde	54
3.1	Positionnement stratégique du langage Python	54
3.1.1	Segments de marchés	54
3.1.2	Niveau de complexité	54
3.1.3	Forces du langage	55
3.1.4	Points faibles	55
3.2	Intégration avec d'autres langages	56
3.2.1	Extensions C	56
3.2.2	Intégration de programmes écrits en C	56
3.2.3	Intégration de programmes Python dans du C	56
3.2.4	Intégration de programmes écrits en Java	56
3.2.5	Intégration de programmes Python dans Java	56
3.2.6	Autres intégrations	57

Chapitre 1.2

Présentation de Python

1.	Philosophie	59
1.1	Python en quelques lignes	59
1.1.1	D'où vient le nom « Python » ?	59
1.1.2	Présentation technique	59
1.1.3	Présentation conceptuelle	60
1.2	Comparaison avec d'autres langages	60
1.2.1	Shell	60
1.2.2	Perl	61

- 1.2.3 C, C++ 61
- 1.2.4 Java 62
- 1.2.5 PHP 64
- 1.3 Grands principes 64
 - 1.3.1 Le zen de Python 64
 - 1.3.2 Le développeur n'est pas stupide 65
 - 1.3.3 Documentation 66
 - 1.3.4 Python est livré piles incluses 66
 - 1.3.5 Duck Typing 66
 - 1.3.6 Notion de code pythonique 66
- 2. Gouvernance 67
 - 2.1 Développement 67
 - 2.1.1 Branches 67
 - 2.1.2 Communauté. 67
 - 2.2 Mode de gouvernance. 68
 - 2.2.1 Créateur du langage. 68
 - 2.2.2 PEP 68
 - 2.2.3 Prise de décisions 68
- 3. Que contient Python ? 69
 - 3.1 Une grammaire et une syntaxe 69
 - 3.2 Plusieurs implémentations. 69
 - 3.3 Une bibliothèque standard. 70
 - 3.4 Des bibliothèques tierces 70
 - 3.5 Des frameworks 70
- 4. Phases d'exécution d'un programme Python. 70
 - 4.1 Chargement de la machine virtuelle 70
 - 4.2 Compilation 70
 - 4.3 Interprétation 71

Chapitre 1.3
Pourquoi choisir Python

- 1. Qualités du langage 73
 - 1.1 Ticket d'entrée 73
 - 1.2 Qualités intrinsèques 74
 - 1.3 Couverture fonctionnelle 75
 - 1.4 Domaines d'excellence 76

1.5	Garanties	77
2.	Diffusion.	78
2.1	Entreprises	78
2.2	Le monde de la recherche	79
2.3	Le monde de l'éducation	80
2.4	Communauté	81
3.	Références.	82
3.1	Poids lourds de l'industrie informatique	82
3.1.1	Google	82
3.1.2	Mozilla	82
3.1.3	Microsoft	83
3.1.4	Canonical	83
3.1.5	Cisco	83
3.2	Entreprises innovantes	84
3.2.1	Services de stockage en ligne	84
3.2.2	Informatique dématérialisée	84
3.2.3	Forge	84
3.2.4	Réseaux sociaux	84
3.3	Éditeurs de contenus.	85
3.3.1	Disney Animation Studio	85
3.3.2	YouTube	85
3.3.3	Box ADSL	85
3.3.4	Spotify	85
3.4	Éditeurs de logiciels.	85
4.	Retours d'expérience	86
4.1	Internet des objets	86
4.2	Système et développement web.	87
4.3	Enseignement	87
4.4	Embarqué.	88
4.5	Développement web.	89
4.6	ERP.	89

Chapitre 1.4
Installer son environnement de travail

- 1. Introduction 91
- 2. Installer Python 91
 - 2.1 Pour Windows 91
 - 2.2 Pour Mac 94
 - 2.3 Pour GNU/Linux et BSD 94
 - 2.4 Par la compilation 95
 - 2.5 Pour un smartphone 96
- 3. Installer une bibliothèque tierce 96
 - 3.1 À partir de Python 3.4 96
 - 3.2 Pour une version inférieure à Python 3.4 98
 - 3.3 Pour Linux 98
- 4. Créer un environnement virtuel 99
 - 4.1 À quoi sert un environnement virtuel ? 99
 - 4.2 Pour Python 3.3 ou version supérieure 99
 - 4.3 Pour toute version de Python 100
 - 4.4 Pour Linux 101
- 5. Installer Anaconda 102
 - 5.1 Pour Windows 102
 - 5.2 Pour Linux 105
 - 5.3 Pour Mac 105
 - 5.4 Mettre à jour Anaconda 106
 - 5.5 Installer une bibliothèque externe 106
 - 5.6 Environnements virtuels 106
- 6. La console Python 106
 - 6.1 Démarrer la console Python 106
 - 6.2 BPython 107
 - 6.3 IPython 107
 - 6.4 IPython Notebook 108
- 7. Installer un IDE 108
 - 7.1 Liste d'IDE 108
 - 7.2 Présentation de PyCharm 109
 - 7.3 Configuration de PyCharm 109

Partie 2 : Guide Python

Chapitre 2.1

Les premiers pas

1. Avant de commencer	115
1.1 Quelques notions importantes	115
1.1.1 Comment fonctionne un ordinateur ?	115
1.1.2 Qu'est-ce qu'un programme informatique ?	116
1.1.3 Qu'est-ce qu'un code source ?	116
1.2 Quelques conventions utilisées dans ce livre	116
1.2.1 Code Python	116
1.2.2 Terminal	117
1.2.3 Mise en forme	117
1.3 Quelle est la meilleure méthode pour apprendre ?	118
2. Premier programme	118
2.1 Hello world !	118
2.2 Affectation	120
2.3 Valeur booléenne	121
2.4 Type	122
2.5 Exceptions	123
2.6 Bloc conditionnel	126
2.7 Conditions avancées	127
2.8 Bloc itératif	128
3. Premier jeu : Devine le nombre	130
3.1 Description du jeu	130
3.2 Aides	130
3.2.1 Gestion du hasard	130
3.2.2 Étapes de développement	130
3.3 Pour aller plus loin	131

Chapitre 2.2
Fonctions et modules

- 1. Les fonctions 133
 - 1.1 Pourquoi utiliser des fonctions ? 133
 - 1.2 Introduction aux fonctions 135
 - 1.2.1 Comment déclarer une fonction. 135
 - 1.2.2 Gestion d'un paramètre. 136
 - 1.2.3 Comment rendre une fonction plus générique 138
 - 1.2.4 Paramètres par défaut 140
 - 1.3 Problématiques de couplage et duplication de code 141
 - 1.3.1 Niveau de ses fonctions. 141
 - 1.3.2 Notion de complexité 143
 - 1.3.3 Bonnes pratiques 145
- 2. Les modules 146
 - 2.1 Introduction 146
 - 2.1.1 Qu'est-ce qu'un module ? 146
 - 2.1.2 Comment crée-t-on un module Python ? 147
 - 2.1.3 Organiser son code 147
 - 2.2 Gérer le code de ses modules 147
 - 2.2.1 Exécuter un module, importer un module. 147
 - 2.2.2 Gérer une arborescence de modules 148
- 3. Terminer le jeu. 149
 - 3.1 Créer des niveaux 149
 - 3.2 Déterminer un nombre de coups maximal 150
 - 3.3 Enregistrer les meilleurs scores. 150
 - 3.4 Intelligence artificielle 150

Chapitre 2.3
Les principaux types

- 1. Chaînes de caractères 151
 - 1.1 Syntaxe 151
 - 1.2 Formatage d'une chaîne 152
 - 1.3 Notion de casse. 153
 - 1.4 Notion de longueur. 153
 - 1.5 Appartenance 154
 - 1.6 Notion d'occurrence. 154

1.7	Remplacement	155
1.8	Notion de caractère	155
1.9	Typologie des caractères	156
1.10	Séquencer une chaîne de caractères	157
2.	Listes	158
2.1	Syntaxe	158
2.2	Indices	158
2.3	Valeurs	159
2.4	Hasard	160
2.5	Techniques d'itération	161
2.6	Tri	163
3.	Dictionnaires	165
3.1	Présentation des dictionnaires	165
3.2	Parcourir un dictionnaire	166
3.3	Exemple	166

Chapitre 2.4

Les classes

1.	Syntaxe	169
2.	Notion d'instance courante	170
3.	Opérateurs	172
4.	Héritage	174
4.1	Spécialisation	174
4.2	Programmation par composants	175

Partie 3 : Les fondamentaux du langage

Chapitre 3.1

Algorithmique de base

1.	Délimiteurs	177
1.1	Instruction	177
1.2	Une ligne de code = une instruction	177
1.3	Commentaire	178
1.4	Une instruction sur plusieurs lignes	178
1.5	Mots-clés	178

1.6	Mots réservés	179
1.7	Indentation	180
1.8	Symboles	181
1.9	Opérateurs	184
1.10	Utilisation du caractère souligné	187
1.11	PEP-8	188
1.12	PEP-7	188
1.13	PEP-257	188
2.	Instructions	188
2.1	Définitions	188
2.1.1	Variable	188
2.1.2	Fonction	190
2.1.3	Fonctions lambda	191
2.1.4	Classe	192
2.1.5	Instruction vide	193
2.1.6	Suppression	193
2.1.7	Renvoyer le résultat de la fonction	194
2.2	Instructions conditionnelles	195
2.2.1	Définition	195
2.2.2	Condition	195
2.2.3	Instruction if	195
2.2.4	Instruction elif	196
2.2.5	Instruction else	196
2.2.6	Instruction switch	198
2.2.7	Interruptions	198
2.2.8	Approfondissement des conditions	198
2.2.9	Performances	199
2.3	Itérations	200
2.3.1	Instruction for	200
2.3.2	Instruction while	201
2.3.3	Quelle différence entre for et while ?	201
2.3.4	Instruction break	201
2.3.5	Instruction return	203
2.3.6	Instruction continue	203
2.3.7	Instruction else	204
2.3.8	Générateurs	205

2.4	Constructions fonctionnelles	207
2.4.1	Construction conditionnelle	207
2.4.2	Générateurs	208
2.4.3	Compréhensions de listes	208
2.4.4	Compréhensions d'ensembles	208
2.4.5	Compréhensions de dictionnaires	208
2.5	Gestion des exceptions	208
2.5.1	Présentation rapide des exceptions	208
2.5.2	Lever une exception	209
2.5.3	Pourquoi lever une exception ?	210
2.5.4	Assertions	210
2.5.5	Capturer une exception	211
2.5.6	Effectuer un traitement de l'exception	212
2.5.7	Gérer la sortie du bloc de capture	214
2.5.8	Gérer le non-déclenchement d'exceptions	214
2.5.9	Prise et libération de ressources	216
2.5.10	Programmation asynchrone	217
2.6	Divers	218
2.6.1	Gérer des imports	218
2.6.2	Traverser les espaces de nommage	219
2.6.3	Fonctions print, help, eval et exec	221

Chapitre 3.2

Déclarations

1.	Variable	223
1.1	Qu'est-ce qu'une variable ?	223
1.1.1	Contenu	223
1.1.2	Contenant	223
1.1.3	Modes de modification d'une variable	225
1.2	Typage dynamique	228
1.2.1	Affectation : rappels	228
1.2.2	Primitive type et nature du type	228
1.2.3	Caractéristiques du typage Python	229
1.3	Visibilité	231
1.3.1	Espace global	231
1.3.2	Notion de bloc	232

- 2. Fonction 235
 - 2.1 Déclaration 235
 - 2.2 Paramètres 236
 - 2.2.1 Signature d’une fonction 236
 - 2.2.2 Notion d’argument ou de paramètre 237
 - 2.2.3 Valeur par défaut 237
 - 2.2.4 Valeur par défaut mutable 239
 - 2.2.5 Paramètres nommés 240
 - 2.2.6 Déclaration de paramètres extensibles 240
 - 2.2.7 Passage de paramètres étoilés 242
 - 2.2.8 Signature universelle 242
 - 2.2.9 Obliger un paramètre à être nommé (keyword-only) 243
 - 2.2.10 Annotations 245
 - 2.2.11 Types hints 248
- 3. Classe 249
 - 3.1 Déclaration 249
 - 3.1.1 Signature 249
 - 3.1.2 Attribut 250
 - 3.1.3 Méthode 250
 - 3.1.4 Bloc local 251
 - 3.2 Instanciation 251
 - 3.2.1 Syntaxe 251
 - 3.2.2 Relation entre l’instance et la classe 252
- 4. Module 252
 - 4.1 À quoi sert un module ? 252
 - 4.2 Déclaration 253
 - 4.3 Instructions spécifiques 253
 - 4.4 Comment appréhender le contenu d’un module ? 254
 - 4.5 Compilation des modules 255

Chapitre 3.3
Modèle objet

- 1. Tout est objet 257
 - 1.1 Principes 257
 - 1.1.1 Quel sens donner à « objet » ? 257
 - 1.1.2 Adaptation de la théorie objet dans Python 258

1.1.3	Généralités	259
1.2	Classes	259
1.2.1	Introduction	259
1.2.2	Déclaration impérative d'une classe	260
1.2.3	Instance	260
1.2.4	Objet courant	262
1.2.5	Déclaration par prototype d'une classe	262
1.2.6	Tuples nommés	265
1.3	Méthodes	265
1.3.1	Déclaration	265
1.3.2	Appel de méthode	267
1.3.3	Méthodes et attributs spéciaux	269
1.3.4	Constructeur et initialiseur	273
1.3.5	Gestion automatisée des attributs	274
1.3.6	Intérêt du paradigme objet	274
1.3.7	Relation entre objets	275
1.4	Héritage	275
1.4.1	Polymorphisme par sous-typage	275
1.4.2	Surcharge de méthode	276
1.4.3	Surcharge des opérateurs	278
1.4.4	Polymorphisme paramétrique	279
1.4.5	Héritage multiple	281
2.	Autres outils de la programmation objet	283
2.1	Principes	283
2.2	Interfaces	283
2.3	Attributs	286
2.4	Propriétés	288
2.5	Emplacements	290
2.6	Métaclasses	292
2.7	Classes abstraites	294
2.8	La Zope Component Architecture	297
2.8.1	Présentation	297
2.8.2	Installation	297
2.8.3	Définir une interface et un composant	298
2.8.4	Autres fonctionnalités	299
2.8.5	Avantages de la ZCA	299

3. Fonctions spéciales et primitives associées	299
3.1 Personnalisation	299
3.1.1 Classes	299
3.1.2 Instances	301
3.1.3 Comparaison	302
3.1.4 Évaluation booléenne	302
3.1.5 Relations d'héritage ou de classe à instance	303
3.2 Classes particulières	303
3.2.1 Itérateurs	303
3.2.2 Conteneurs	306
3.2.3 Instances assimilables à des fonctions	306
3.2.4 Ressources à protéger	307
3.2.5 Types	308

Chapitre 3.4

Types de données et algorithmes appliqués

1. Nombres	309
1.1 Types	309
1.1.1 Entiers	309
1.1.2 Réels	310
1.1.3 Socle commun aux nombres entiers et réels	311
1.1.4 Méthodes dédiées aux nombres entiers	312
1.1.5 Méthodes dédiées aux nombres réels	313
1.1.6 Complexes	313
1.2 La console Python, la calculatrice par excellence	314
1.2.1 Opérateurs mathématiques binaires	314
1.2.2 Opérateurs binaires particuliers	315
1.2.3 Opérateurs mathématiques unaires	316
1.2.4 Arrondis	317
1.2.5 Opérateurs de comparaison	319
1.2.6 Opérations mathématiques n-aires	320
1.2.7 Fonctions mathématiques usuelles	321
1.3 Représentations d'un nombre	327
1.3.1 Représentation décimale	327
1.3.2 Représentation par un exposant	327
1.3.3 Représentation par une fraction	327
1.3.4 Représentation hexadécimale	328

1.3.5	Représentation octale	329
1.3.6	Représentation binaire.....	330
1.3.7	Opérations binaires	330
1.3.8	Longueur de la représentation mémoire d'un entier	332
1.4	Conversions	333
1.4.1	Conversion entre entiers et réels	333
1.4.2	Conversion entre réels et complexes	333
1.4.3	Conversion vers un booléen	334
1.5	Travailler avec des variables	335
1.5.1	Un nombre est non mutable	335
1.5.2	Modifier la valeur d'une variable	335
1.5.3	Opérateurs d'incrémement	336
1.6	Statistiques	337
2.	Séquences	338
2.1	Présentation des différents types de séquences.....	338
2.1.1	Généralités	338
2.1.2	Les listes	339
2.1.3	Les n-uplets.....	340
2.1.4	Conversion entre listes et n-uplets	342
2.1.5	Socle commun entre liste et n-uplet	342
2.1.6	Notion d'itérateur	343
2.2	Utilisation des indices et des tranches.....	345
2.2.1	Définition de l'indice d'un objet et des occurrences	345
2.2.2	Utiliser l'indice pour adresser la séquence	346
2.2.3	Retrouver les occurrences d'un objet et leurs indices	347
2.2.4	Taille d'une liste, comptage d'occurrences.....	349
2.2.5	Utiliser l'indice pour modifier ou supprimer	350
2.2.6	Itération simple	352
2.2.7	Présentation de la notion de tranches (slices)	355
2.2.8	Cas particulier de la branche 2.x de Python.....	363
2.2.9	Utilisation basique des tranches.....	364
2.2.10	Utilisation avancée des tranches.....	365
2.3	Utilisation des opérateurs.....	368
2.3.1	Opérateur +	368
2.3.2	Opérateur *	369
2.3.3	Opérateur +=	371
2.3.4	Opérateur *=	372

2.3.5	Opérateur in	373
2.3.6	Opérateurs de comparaison	374
2.4	Méthodes de modifications	375
2.4.1	Ajouter des éléments dans une liste et un n-uplet	375
2.4.2	Supprimer un objet d'une liste et d'un n-uplet	378
2.4.3	Solutions de contournement pour la modification de n-uplets	381
2.4.4	Renverser une liste ou un tuple	382
2.4.5	Trier une liste	383
2.5	Utilisation avancée des listes	386
2.5.1	Opérations d'ensemble	386
2.5.2	Pivoter une séquence	387
2.5.3	Itérer correctement	388
2.5.4	Programmation fonctionnelle	389
2.5.5	Compréhensions de listes	391
2.5.6	Itérations avancées	392
2.5.7	Combinatoire	397
2.6	Adapter les listes à des besoins spécifiques	399
2.6.1	Liste d'entiers	399
2.6.2	Présentation du type array	401
2.6.3	Utiliser une liste comme pile	403
2.6.4	Utiliser une liste comme file d'attente	403
2.6.5	Conteneur plus performant	404
2.6.6	Utiliser des listes pour représenter des matrices	405
2.6.7	Liste sans doublons	406
2.7	Autres types de données	408
3.	Ensembles	410
3.1	Présentation	410
3.1.1	Définition d'un ensemble	410
3.1.2	Différences entre set et frozenset	412
3.1.3	Utilisation pour dédoubler des listes	412
3.1.4	Rajouter une relation d'ordre	413
3.2	Opérations ensemblistes	413
3.2.1	Opérateurs pour un ensemble à partir de deux autres	413
3.2.2	Opérateurs pour modifier un ensemble à partir d'un autre	414
3.2.3	Méthodes équivalentes à la création ou modification ensembliste	415

3.2.4	Méthodes de comparaison des ensembles	415
3.2.5	Exemples non classiques d'utilisation.	417
3.3	Méthodes de modification d'un ensemble	420
3.3.1	Ajouter un élément	420
3.3.2	Supprimer un élément	421
3.3.3	Vider un ensemble	421
3.3.4	Dupliquer un élément	421
3.3.5	Sortir une valeur d'un ensemble	422
3.3.6	Utiliser un ensemble comme un recycleur d'objets.	423
3.3.7	Algorithmique avancée : résolution du problème des n-dames	426
4.	Chaînes de caractères.	428
4.1	Présentation.	428
4.1.1	Définition	428
4.1.2	Vocabulaire.	429
4.1.3	Spécificités de la branche 2.x.	430
4.1.4	Changements apportés par la branche 3.x.	431
4.1.5	Chaîne de caractères en tant que séquence de caractères	433
4.1.6	Caractères	435
4.1.7	Opérateurs de comparaison.	436
4.2	Formatage de chaînes de caractères	438
4.2.1	Opérateur modulo	438
4.2.2	Méthodes de formatage sur l'ensemble de la chaîne	443
4.2.3	Nouvelle méthode de formatage des variables dans une chaîne	446
4.3	Opérations d'ensemble	449
4.3.1	Séquençage de chaînes.	449
4.3.2	Opérations sur la casse	451
4.3.3	Recherche sur une chaîne de caractères	452
4.3.4	Informations sur les caractères	453
4.4	Problématiques relatives à l'encodage	455
4.4.1	Encodage par défaut.	455
4.4.2	Encodage du système.	455
4.4.3	L'unicode, référence absolue	455
4.4.4	Autres encodages	456
4.4.5	Ponts entre l'unicode et le reste du monde.	457
4.4.6	Revenir vers l'Unicode	458

4.5	Manipulations de bas niveau avancées	459
4.5.1	Opérations de comptage	459
4.5.2	Une chaîne de caractères vue comme une liste	460
4.5.3	Une chaîne de caractères vue comme un ensemble de caractères	460
4.6	Représentation mémoire	461
4.6.1	Présentation du type bytes	461
4.6.2	Lien avec les chaînes de caractères	462
4.6.3	Présentation du type bytearray	463
4.6.4	Gestion d'un jeu de caractères	465
5.	Dictionnaires	470
5.1	Présentation	470
5.1.1	Définition	470
5.1.2	Évolutions et différences entre les branches 2.x et 3.x	471
5.1.3	Vues de dictionnaires	472
5.1.4	Instanciation	474
5.1.5	Compréhension de dictionnaire	474
5.2	Manipuler un dictionnaire	475
5.2.1	Récupérer une valeur d'un dictionnaire	475
5.2.2	Modifier les valeurs d'un dictionnaire	476
5.2.3	Supprimer une entrée d'un dictionnaire	477
5.2.4	Dupliquer un dictionnaire	477
5.2.5	Utiliser le dictionnaire comme agrégateur de données	478
5.2.6	Méthodes d'itération	479
5.3	Utilisation avancée des dictionnaires	479
5.3.1	Rajouter une relation d'ordre	479
5.3.2	Algorithmiques classiques	483
5.3.3	Adapter les dictionnaires à des besoins spécifiques	485
5.3.4	Représentation universelle de données	487
6.	Booléens	488
6.1	Le type booléen	488
6.1.1	Classe bool	488
6.1.2	Les deux objets True et False	489
6.1.3	Différence entre l'opérateur d'égalité et d'identité	489
6.2	Évaluation booléenne	489
6.2.1	Méthode générique	489
6.2.2	Objets classiques	489

7.	Données temporelles	490
7.1	Gérer une date calendaire	490
7.1.1	Notion de date calendaire	490
7.1.2	Travailler sur une date	491
7.1.3	Considérations astronomiques	492
7.1.4	Considérations historiques	492
7.1.5	Considérations techniques	492
7.1.6	Représentation textuelle	493
7.2	Gérer un horaire ou un moment d'une journée	495
7.2.1	Notion d'instant	495
7.2.2	Notion de fuseau horaire	496
7.2.3	Représentation textuelle	496
7.3	Gérer un instant absolu	497
7.3.1	Notion d'instant absolu	497
7.3.2	Rapport avec les notions précédentes	498
7.3.3	Représentation textuelle	499
7.3.4	Gestion des fuseaux horaires	500
7.3.5	Créer une date à partir d'une représentation textuelle	500
7.4	Gérer une différence entre deux dates ou instants	500
7.4.1	Notion de différence et de résolution	500
7.4.2	Considérations techniques	502
7.4.3	Utilisation avec des dates calendaires	503
7.4.4	Utilisation avec des horaires	503
7.4.5	Utilisation avec des dates absolues	503
7.4.6	La seconde comme unité de base	503
7.5	Spécificités des fuseaux horaires	504
7.6	Problématiques de bas niveau	505
7.6.1	Timestamp et struct_time	505
7.6.2	Mesures de performances	506
7.7	Utilisation du calendrier	508
7.7.1	Présentation du module calendar	508
7.7.2	Fonctions essentielles du calendrier	512

Chapitre 3.5

Motifs de conception

1. Définition	515
1.1 Positionnement par rapport à la notion d'objet	515
1.2 Organisation du chapitre	516
1.3 Positionnement par rapport à d'autres concepts	516
2. Motifs de création	517
2.1 Singleton	517
2.2 Fabrique	517
2.3 Fabrique abstraite	520
2.4 Monteur	521
2.5 Prototype	523
3. Motifs de structuration	526
3.1 Adaptateur	526
3.2 Pont	529
3.3 Composite	532
3.4 Décorateur	534
3.5 Façade	536
3.6 Poids-mouche	538
3.7 Proxy	540
4. Motifs de comportement	542
4.1 Chaîne de responsabilité	542
4.2 Commande	543
4.3 Itérateur	545
4.4 Memento	547
4.5 Visiteur	548
4.6 Observateur	549
4.7 Stratégie	551
4.8 Fonction de rappel	552
5. ZCA	553
5.1 Rappels	553
5.2 Adaptateur	553
5.3 Utilitaire	555
5.4 Fabrique	556
5.5 Pour aller plus loin	557

Partie 4 : Les fonctionnalités

Chapitre 4.1

Manipulation de données

1. Bases de données	559
1.1 Présentation	559
1.2 Accès à une base de données relationnelle	559
1.2.1 Point d'entrée	559
1.2.2 MySQL	560
1.2.3 PostgreSQL	564
1.2.4 SQLite	567
1.2.5 Oracle	567
1.3 Utilisation d'un ORM	568
1.3.1 Qu'est-ce qu'un ORM ?	568
1.3.2 ORM proposés par Python	568
1.3.3 SQLAlchemy	568
1.4 Autres bases de données	575
1.4.1 CSV	575
1.4.2 NoSQL	581
1.4.3 Base de données orientée objet : ZODB	581
1.4.4 Base de données de type clé-valeur : Redis	586
1.4.5 Bases de données orientées documents : CouchDB et MongoDB	587
1.4.6 Bases de données natives XML : BaseX, eXist	588
1.4.7 Cassandra	589
1.4.8 Bases de données orientées colonnes : HBase	590
1.4.9 Big Data : l'écosystème Hadoop	591
2. LDAP	594
2.1 Présentation	594
2.1.1 Protocole	594
2.1.2 Serveurs	594
2.1.3 Terminologie	594
2.2 Installation	595
2.3 Ouvrir une connexion à un serveur	595
2.4 Effectuer une recherche	596
2.5 Synchrone vs asynchrone	597

2.6	Connexions sécurisées	598
3.	XML	599
3.1	XML et les technologies qui gravitent autour	599
3.1.1	Définition de XML, terminologie associée	599
3.1.2	Notion de schéma	600
3.1.3	Avantages et inconvénients de XML	601
3.1.4	Différentes manières de parcourir un fichier XML	602
3.1.5	Modules Python dédiés au XML	602
3.2	Valider un document XML	603
3.2.1	Document XML	603
3.2.2	Schéma DTD	604
3.2.3	Schéma XSD	604
3.2.4	Schéma RNG (RelaxNG)	605
3.2.5	Schematron	605
3.3	DOM	606
3.3.1	Lecture	606
3.3.2	Écriture	607
3.4	SAX	608
3.4.1	Support de SAX dans lxml	608
3.4.2	API SAX Allégée	609
3.5	XPath	610
3.6	XSLT	612
3.7	Cas spécifique des fichiers HTML	613
3.7.1	Problématique	613
3.7.2	Parser un fichier HTML à la façon DOM	614
3.7.3	Parser un fichier HTML à la façon SAX	615
4.	Outils de manipulation de données	616
4.1	Encrypter une donnée	616
4.1.1	Fonctions de hachage	616
4.1.2	Code d'authentification de message	618
4.1.3	Stéganographie	620
4.2	Générer des nombres aléatoires	623
4.3	Expressions régulières	624
5.	Travailler avec des images	628
5.1	Représentation informatique d'une image	628
5.2	Présentation de Pillow	629

5.3	Formats d'images matricielles	630
5.4	Récupérer des informations d'une image	631
5.5	Opérations d'ensemble sur une image	633
5.6	Travailler avec les calques ou les pixels	635

Chapitre 4.2

Génération de contenu

1.	PDF	639
1.1	Présentation	639
1.1.1	Format PDF	639
1.1.2	Avantages	639
1.1.3	Inconvénients	639
1.1.4	Présentation de la bibliothèque libre	640
1.2	Bas niveau	640
1.2.1	Bibliothèque de données	640
1.2.2	Canvas	642
1.3	Haut niveau	643
1.3.1	Styles	643
1.3.2	Flux de données	645
1.3.3	Création d'un visuel	647
1.3.4	Template de page	647
1.3.5	Page contenant plusieurs zones	648
2.	OpenDocument	650
2.1	Présentation	650
2.2	ezodf2	651
2.2.1	Installation	651
2.2.2	OpenDocument Texte	651
2.2.3	OpenDocument Tableur	651
2.2.4	Aller plus loin	652
2.3	Alternatives	652
2.3.1	lpod	652
2.3.2	Génération à partir de templates	653

Chapitre 4.3
Programmation parallèle

- 1. Terminologie 655
 - 1.1 Processus 655
 - 1.2 Tâche 656
- 2. Utilisation d'une tâche 656
 - 2.1 Gestion d'une tâche 656
 - 2.1.1 Présentation 656
 - 2.1.2 Création 656
 - 2.2 Gestion de plusieurs tâches 659
 - 2.2.1 Lancement et contrôle 659
 - 2.2.2 Opportunité d'utiliser une tâche 661
 - 2.3 Résolution des problématiques liées 663
 - 2.3.1 Synchronisation 663
 - 2.3.2 Synchronisation conditionnelle 665
 - 2.3.3 Sémaphore 668
- 3. Utilisation de processus 669
 - 3.1 Gestion d'un processus 669
 - 3.1.1 Présentation 669
 - 3.1.2 Création 669
 - 3.2 Gestion de plusieurs processus 672
 - 3.2.1 Synchronisation 672
 - 3.2.2 Paralléliser un travail 672
 - 3.3 Résolution des problématiques liées 674
 - 3.3.1 Communication interprocessus 674
 - 3.3.2 Partage de données entre processus 676
 - 3.4 Opportunité d'utiliser les processus 677
 - 3.5 Démon 677
- 4. Exécution asynchrone 679
 - 4.1 Introduction 679
 - 4.2 Présentation 680

Chapitre 4.4**Programmation système et réseau**

1.	Présentation	687
1.1	Définition	687
1.2	Objectifs du chapitre	688
2.	Écrire des scripts système	688
2.1	Appréhender son système d'exploitation	688
2.1.1	Avertissement	688
2.1.2	Système d'exploitation	688
2.1.3	Processus courant	689
2.1.4	Utilisateurs et groupes	690
2.1.5	Constantes pour le système de fichiers	692
2.1.6	Gérer les chemins	692
2.2	Gestion d'un fichier	693
2.2.1	Ouvrir un fichier	693
2.2.2	Lire un fichier	694
2.2.3	Écrire un fichier	695
2.2.4	Changer les droits d'un fichier	696
2.2.5	Changer de propriétaire ou de groupe	697
2.2.6	Récupérer des informations sur un fichier	698
2.2.7	Supprimer un fichier	699
2.3	Alternatives simples à des commandes bash usuelles	699
2.3.1	Répertoires	699
2.3.2	Fichiers	702
2.3.3	Module de haut niveau	703
2.3.4	Recherche d'un fichier	705
2.4	Exécuter des commandes externes	705
2.4.1	Exécuter et afficher le résultat	705
2.4.2	Exécuter et récupérer le résultat	706
2.4.3	Pour Python 3.5	707
2.5	Utilitaires	707
2.5.1	Comparer deux fichiers	707
2.5.2	Utilitaire de sauvegarde	709
2.5.3	Lire un fichier de configuration	710
2.5.4	Pickle	711

2.6	Compresser et décompresser un fichier	713
2.6.1	Tarfile	713
2.6.2	Gzip	715
2.6.3	Bz2	716
2.6.4	Zipfile	716
2.6.5	Interface de haut niveau	718
3.	Travailler avec des arguments	719
3.1	Présentation	719
3.2	Mise en œuvre	720
4.	Programmation réseau	723
4.1	Écrire un serveur et un client	723
4.1.1	Utilisation d'une socket TCP	723
4.1.2	Utilisation d'une socket UDP	726
4.1.3	Création d'un serveur TCP	729
4.1.4	Création d'un serveur UDP	731
4.1.5	Un peu plus loin sur le sujet	731
4.2	Utiliser un protocole standard	733
4.2.1	HTTP	733
4.2.2	Proxy	737
4.2.3	Cookies	738
4.2.4	FTP et SFTP	738
4.2.5	SSH	740
4.2.6	POP et POPS	742
4.2.7	IMAP et IMAPS	743
4.2.8	SMTP et SMPTS	744
4.2.9	NNTP	748
4.2.10	IRC	749
4.3	Services web	752
4.3.1	REST	752
4.3.2	SOAP	753
4.3.3	Pyro	755
5.	Utilisation du matériel	756
5.1	Wake-on-LAN	756
5.1.1	Prérequis	756
5.1.2	Mise en œuvre	756
5.2	Utilisation du port série	757

Chapitre 4.5**Programmation asynchrone**

1. Utilité de la programmation asynchrone 759
2. Exemple de travail 760
3. Exemple retravaillé en utilisant des générateurs 762
4. Exemple retravaillé en asynchrone pour Python 3.4 763
5. Exemple retravaillé en asynchrone pour Python 3.5 765
6. Pour aller plus loin 766

Chapitre 4.6**Programmation scientifique**

1. Calcul scientifique 769
 - 1.1 Présentation 769
 - 1.2 Python, une alternative libre et crédible 769
 - 1.3 Vue d'ensemble de quelques bibliothèques 770
2. Tableaux multidimensionnels 771
 - 2.1 Création 771
 - 2.2 Déterminer la composition d'un tableau 772
 - 2.3 Générateurs de tableaux 773
 - 2.4 Opérations basiques 774
 - 2.5 Opérateur crochet 776
3. Matrices 778
4. Génération de graphiques 779
 - 4.1 Syntaxe MATLAB 780
 - 4.2 Syntaxe objet 781
 - 4.3 Mise en forme 781
 - 4.4 Graphiques 3D 786

Chapitre 4.7
Bonnes pratiques

- 1. Programmation dirigée par les tests 791
 - 1.1 Tests unitaires 791
 - 1.1.1 Principes 791
 - 1.1.2 Interprétation 792
 - 1.1.3 Couverture 793
 - 1.1.4 Outils 793
 - 1.1.5 Autres outils 795
 - 1.2 Tests de non-régression 795
 - 1.2.1 Actions de développement 795
 - 1.2.2 Gestion de la découverte d'une anomalie par une MOA 796
 - 1.3 Tests fonctionnels 797
 - 1.4 Tests de performance 798
 - 1.5 Intégration continue 800
- 2. Programmation dirigée par la documentation 801
 - 2.1 Documentation interne 801
 - 2.1.1 À destination des développeurs 801
 - 2.1.2 À destination des utilisateurs 801
 - 2.2 Documentation externe 802
 - 2.2.1 Présentation 802
 - 2.2.2 Démarrage rapide 802
 - 2.2.3 Résultat 805
- 3. Optimisation 805
 - 3.1 Qualimétrie 805
 - 3.2 Outils de débogage 807
 - 3.3 Outils de profilage 807
 - 3.4 Règles d'optimisation 808
 - 3.4.1 Pourquoi optimiser ? 808
 - 3.4.2 Règles générales 809
 - 3.4.3 Profiler un algorithme 810
 - 3.4.4 Optimiser l'utilisation de la mémoire 820

Partie 5 : Mise en pratique

Chapitre 5.1

Créer une application web en 30 minutes

1. Description de l'application à construire	823
2. Mise en place	824
2.1 Isolation de l'environnement	824
2.2 Création du projet	825
2.3 Paramétrage	825
2.4 Premiers essais	826
3. Réalisation de l'application	827
3.1 Modèles	827
3.2 Vues	830
3.3 Contrôleurs	831
4. Pour aller plus loin	835

Chapitre 5.2

Créer une application console en 10 minutes

1. Objectif	837
2. Enregistrer le script	837
3. Création des données	838
4. Parseur d'arguments	839

Chapitre 5.3

Créer une application graphique en 20 minutes

1. Objectif	841
1.1 Fonctionnel	841
1.2 Technique	841
2. Présentation rapide de Gtk et d'astuces	842
2.1 Présentation	842
2.2 Astuces	842
3. Démarrer le programme	843
4. Interface graphique avec Glade	845
5. Créer le composant graphique	848

- 6. Contrôleur 850
- 7. Autres bibliothèques graphiques 851
 - 7.1 TkInter 851
 - 7.2 wxPython 851
 - 7.3 PyQt. 851
 - 7.4 PySide 852
 - 7.5 Autres 852

Chapitre 5.4
Créer un jeu en 30 minutes avec PyGame

- 1. Présentation de PyGame 853
- 2. Réalisation d'un jeu Tetris 854
 - 2.1 Présentation du jeu 854
 - 2.2 Présentation des problématiques 855
 - 2.3 Création des constantes 855

Annexes

- 1. Objets mutables et non mutables 867
- 2. Table Unicode 869
 - 2.1 Script 869
- 3. Bytes 869
 - 3.1 Script 869
 - 3.2 Résultat 869
- 4. Guide de portage vers Python 3 872

- Index 875