

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RI33PYT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Introduction	23
2. Contenu de l'ouvrage	23
3. Progressivité de l'ouvrage	24
4. À destination des enseignants et élèves	25
5. À destination des chercheurs ou doctorants	27
6. À destination de ceux qui viennent d'un autre langage	27

Partie 1 : Les atouts de Python

Chapitre 1.1

Clés théoriques

1. Petite histoire des langages informatiques	29
1.1 Informatique théorique	29
1.2 Chronologie de l'informatique	30
1.2.1 Évolutions des problématiques liées à l'informatique	30
1.2.2 Chronologie des langages informatiques	31
2. Typologie des langages de programmation	35
2.1 Paradigmes	35
2.1.1 Définition	35
2.1.2 Paradigme impératif et dérivés	36
2.1.3 Paradigme objet et dérivés	37
2.1.4 Programmation orientée aspect	37
2.1.5 Paradigme fonctionnel	37
2.1.6 Paradigme logique	38
2.1.7 Programmation concurrente	38
2.1.8 Synthèse	38
2.2 Interopérabilité	39

2.3	Niveau de programmation	41
2.3.1	Machine	41
2.3.2	Bas niveau	41
2.3.3	Haut niveau	42
2.4	Typage	43
2.4.1	Faible vs fort	43
2.4.2	Statique vs dynamique	43
2.5	Grammaire	43
2.5.1	Langages formels	43
2.5.2	Syntaxe	44
3.	Python et le reste du monde	45
3.1	Positionnement stratégique du langage Python	45
3.1.1	Segments de marchés	45
3.1.2	Niveau de complexité	45
3.1.3	Forces du langage	45
3.1.4	Points faibles	46
3.2	Intégration avec d'autres langages	46
3.2.1	Extensions C	46
3.2.2	Intégration de programmes écrits en C	47
3.2.3	Intégration de programmes Python dans du C	47
3.2.4	Intégration de programmes écrits en Java	47
3.2.5	Intégration de programmes Python dans Java	47
3.2.6	Autres intégrations	47

Chapitre 1.2

Présentation de Python

1.	Philosophie	49
1.1	Python en quelques lignes	49
1.1.1	D'où vient le nom « Python » ?	49
1.1.2	Présentation technique	49
1.1.3	Présentation conceptuelle	50
1.2	Comparaison avec d'autres langages	50
1.2.1	Shell	50
1.2.2	Perl	51
1.2.3	C, C++	51
1.2.4	Java	52
1.2.5	PHP	54

1.3	Grands principes	55
1.3.1	Le zen de Python	55
1.3.2	Le développeur n'est pas stupide	55
1.3.3	Documentation	56
1.3.4	Python est livré piles incluses	56
1.3.5	Duck Typing	56
1.3.6	Notion de code pythonique	57
2.	Histoire de Python.	57
2.1	La genèse	57
2.2	Extension du périmètre fonctionnel	58
2.3	Évolution de la licence	62
2.4	Avenir	62
3.	Gouvernance	63
3.1	Développement	63
3.1.1	Branches	63
3.1.2	Communauté.	64
3.2	Mode de gouvernance.	65
3.2.1	Créateur du langage.	65
3.2.2	PEP	65
3.2.3	Prise de décisions	65
3.2.4	Contribuer à Python	66
4.	Que contient Python ?	66
4.1	Une grammaire et une syntaxe	66
4.2	Plusieurs implémentations.	66
4.3	Une bibliothèque standard.	67
4.4	Des bibliothèques tierces	67
4.5	Des frameworks	67
5.	Phases d'exécution d'un programme Python.	67
5.1	Chargement de la machine virtuelle	67
5.2	Compilation	68
5.3	Interprétation	69

Chapitre 1.3

Pourquoi choisir Python

1. Qualités du langage	71
1.1 Ticket d'entrée	71
1.2 Qualités intrinsèques	73
1.3 Couverture fonctionnelle	73
1.4 Domaines d'excellence	74
1.5 Garanties	75
2. Diffusion	76
2.1 Entreprises	76
2.2 Le monde de la recherche	77
2.3 Le monde de l'éducation	78
2.4 Communauté	79
3. Références	80
3.1 Poids lourds de l'industrie informatique	80
3.1.1 Google	80
3.1.2 Mozilla	81
3.1.3 Microsoft	81
3.1.4 Canonical	81
3.1.5 Cisco	82
3.2 Entreprises innovantes	82
3.2.1 Services de stockage en ligne	82
3.2.2 Informatique dématérialisée	82
3.2.3 Forge	82
3.2.4 Réseaux sociaux	83
3.3 Éditeurs de contenus	83
3.3.1 Disney Animation Studio	83
3.3.2 YouTube	83
3.3.3 Box ADSL	83
3.3.4 Spotify	83
3.4 Éditeurs de logiciels	83
4. Retours d'expérience	84
4.1 Internet des objets	84
4.2 Système et développement web	85
4.3 Enseignement	85
4.4 Embarqué	86

4.5 Développement web..... 87
4.6 ERP..... 87

Chapitre 1.4
Installer son environnement de travail

1. Introduction..... 89
2. Installer Python..... 89
 2.1 Pour Windows..... 89
 2.2 Pour Mac..... 92
 2.3 Pour GNU/Linux et BSD..... 92
 2.4 Par la compilation..... 93
 2.5 Pour un smartphone..... 94
3. Installer une bibliothèque tierce..... 94
 3.1 À partir de Python 3.4..... 94
 3.2 Pour une version inférieure à Python 3.4..... 96
 3.3 Pour Linux..... 96
4. Créer un environnement virtuel..... 97
 4.1 À quoi sert un environnement virtuel ?..... 97
 4.2 Pour Python 3.3 ou version supérieure..... 97
 4.3 Pour toute version de Python..... 98
 4.4 Pour Linux..... 99
5. Installer Anaconda..... 100
 5.1 Pour Windows..... 100
 5.2 Pour Linux..... 103
 5.3 Pour Mac..... 103
 5.4 Mettre à jour Anaconda..... 104
 5.5 Installer une bibliothèque externe..... 104
 5.6 Environnements virtuels..... 104
6. Docker..... 104
7. La console Python..... 105
 7.1 Démarrer la console Python..... 105
 7.2 BPython..... 105
 7.3 IPython..... 106
 7.4 IPython Notebook..... 106

8.	Installer un IDE	107
8.1	Liste d'IDE	107
8.2	Présentation de PyCharm	107
8.3	Configuration de PyCharm	108

Partie 2 : Guide Python

Chapitre 2.1

Les premiers pas

1.	Avant de commencer	113
1.1	Quelques notions importantes	113
1.1.1	Comment fonctionne un ordinateur ?	113
1.1.2	Qu'est-ce qu'un programme informatique ?	114
1.1.3	Qu'est-ce qu'un code source ?	114
1.2	Quelques conventions utilisées dans ce livre	114
1.2.1	Code Python	114
1.2.2	Terminal	115
1.2.3	Mise en forme	115
1.3	Quelle est la meilleure méthode pour apprendre ?	116
2.	Premier programme	116
2.1	Hello world !	116
2.2	Affectation	118
2.3	Valeur booléenne	119
2.4	Type	120
2.5	Exceptions	121
2.6	Bloc conditionnel	124
2.7	Conditions avancées	125
2.8	Bloc itératif	126
3.	Premier jeu : Devine le nombre	128
3.1	Description du jeu	128
3.2	Aides	128
3.2.1	Gestion du hasard	128
3.2.2	Étapes de développement	128
3.3	Pour aller plus loin	129

Chapitre 2.2
Fonctions et modules

- 1. Les fonctions 131
 - 1.1 Pourquoi utiliser des fonctions ? 131
 - 1.2 Introduction aux fonctions 133
 - 1.2.1 Comment déclarer une fonction. 133
 - 1.2.2 Gestion d'un paramètre. 134
 - 1.2.3 Comment rendre une fonction plus générique 136
 - 1.2.4 Paramètres par défaut 138
 - 1.3 Problématiques de couplage et duplication de code 139
 - 1.3.1 Niveau de ses fonctions. 139
 - 1.3.2 Notion de complexité 141
 - 1.3.3 Bonnes pratiques 143
- 2. Les modules 144
 - 2.1 Introduction 144
 - 2.1.1 Qu'est-ce qu'un module ? 144
 - 2.1.2 Comment crée-t-on un module Python ? 145
 - 2.1.3 Organiser son code 145
 - 2.2 Gérer le code de ses modules 145
 - 2.2.1 Exécuter un module, importer un module. 145
 - 2.2.2 Gérer une arborescence de modules 146
- 3. Terminer le jeu. 147
 - 3.1 Créer des niveaux 148
 - 3.2 Déterminer un nombre de coups maximal 148
 - 3.3 Enregistrer les meilleurs scores. 148
 - 3.4 Intelligence artificielle 148

Chapitre 2.3
Les principaux types

- 1. Chaînes de caractères 149
 - 1.1 Syntaxe 149
 - 1.2 Formatage d'une chaîne 150
 - 1.3 Notion de casse. 151
 - 1.4 Notion de longueur. 152
 - 1.5 Appartenance 152
 - 1.6 Notion d'occurrence. 153

1.7	Remplacement	154
1.8	Notion de caractère	154
1.9	Typologie des caractères	155
1.10	Séquencer une chaîne de caractères	156
2.	Listes	156
2.1	Syntaxe	156
2.2	Indices	157
2.3	Valeurs	158
2.4	Hasard	159
2.5	Techniques d'itération	160
2.6	Tri	162
3.	Dictionnaires	164
3.1	Présentation des dictionnaires	164
3.2	Parcourir un dictionnaire	165
3.3	Exemple	165

Chapitre 2.4

Les classes

1.	Syntaxe	167
2.	Notion d'instance courante	168
3.	Opérateurs	170
4.	Héritage	172
4.1	Spécialisation	172
4.2	Programmation par composants	173

Partie 3 : Les fondamentaux du langage

Chapitre 3.1

Algorithmique de base

1.	Délimiteurs	175
1.1	Instruction	175
1.2	Une ligne de code = une instruction	175
1.3	Commentaire	176
1.4	Une instruction sur plusieurs lignes	176
1.5	Mots-clés	176

1.6	Mots réservés	177
1.7	Indentation	178
1.8	Symboles	179
1.9	Opérateurs	182
1.9.1	Opérateur expression d'affectation :=	185
1.10	Utilisation du caractère souligné	186
1.11	PEP-8	187
1.12	PEP-7	187
1.13	PEP-257	187
2.	Instructions	187
2.1	Définitions	187
2.1.1	Variable	187
2.1.2	Fonction	189
2.1.3	Fonctions lambda	190
2.1.4	Classe	191
2.1.5	Instruction vide	192
2.1.6	Suppression	192
2.1.7	Renvoyer le résultat de la fonction	193
2.2	Instructions conditionnelles	194
2.2.1	Définition	194
2.2.2	Condition	194
2.2.3	Instruction if	194
2.2.4	Instruction elif	195
2.2.5	Instruction else	195
2.3	Utilisation d'une expression d'affectation	197
2.3.1	Instruction switch	197
2.3.2	Interruptions	197
2.3.3	Approfondissement des conditions	198
2.3.4	Performances	199
2.4	Itérations	200
2.4.1	Instruction for	200
2.4.2	Instruction while	200
2.4.3	Quelle différence entre for et while ?	201
2.4.4	Instruction break	201
2.4.5	Instruction return	203
2.4.6	Instruction continue	203
2.4.7	Instruction else	203

2.4.8	Générateurs	204
2.5	Constructions fonctionnelles	207
2.5.1	Construction conditionnelle	207
2.5.2	Générateurs	207
2.5.3	Compréhensions de listes	207
2.5.4	Compréhensions d'ensembles	208
2.5.5	Compréhensions de dictionnaires	208
2.6	Compréhensions et expressions d'affectation	208
2.7	Gestion des exceptions	208
2.7.1	Présentation rapide des exceptions	208
2.7.2	Lever une exception	209
2.7.3	Pourquoi lever une exception ?	209
2.7.4	Assertions	210
2.7.5	Capturer une exception	211
2.7.6	Effectuer un traitement de l'exception	212
2.7.7	Gérer la sortie du bloc de capture	214
2.7.8	Gérer le non-déclenchement d'exceptions	214
2.7.9	Prise et libération de ressources	216
2.7.10	Programmation asynchrone	217
2.8	Divers	218
2.8.1	Gérer des imports	218
2.8.2	Traverser les espaces de nommage	219
2.8.3	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
2.3	Obliger un paramètre à être positionnel (Positional-only)	245
2.3.1	Annotations	245
2.3.2	Types hint	249
3.	Classe	251
3.1	Déclaration	251
3.1.1	Signature	251
3.1.2	Attribut	251
3.1.3	Méthode	252
3.1.4	Bloc local	252
3.2	Instanciation	253
3.2.1	Syntaxe	253
3.2.2	Relation entre l'instance et la classe	253
4.	Module	254
4.1	À quoi sert un module ?	254
4.2	Déclaration	254
4.3	Instructions spécifiques	254
4.4	Comment appréhender le contenu d'un module ?	255
4.5	Compilation des modules	256

Chapitre 3.3

Modèle objet

1.	Tout est objet	259
1.1	Principes	259
1.1.1	Quel sens donner à « objet » ?	259
1.1.2	Adaptation de la théorie objet dans Python	260
1.1.3	Généralités	261
1.2	Classes	261
1.2.1	Introduction	261
1.2.2	Déclaration impérative d'une classe	262
1.2.3	Instance	262
1.2.4	Objet courant	264
1.2.5	Déclaration par prototype d'une classe	264
1.2.6	Tuples nommés	267
1.3	Méthodes	267
1.3.1	Déclaration	267
1.3.2	Appel de méthode	269
1.3.3	Méthodes et attributs spéciaux	271
1.3.4	Constructeur et initialisateur	275
1.3.5	Gestion automatisée des attributs	276
1.3.6	Intérêt du paradigme objet	276
1.3.7	Relation entre objets	277
1.4	Héritage	277
1.4.1	Polymorphisme par sous-typage	277
1.4.2	Surcharge de méthode	278
1.4.3	Surcharge des opérateurs	280
1.4.4	Polymorphisme paramétrique	281
1.4.5	Héritage multiple	283
2.	Autres outils de la programmation objet	285
2.1	Principes	285
2.2	Interfaces	285
2.3	Attributs	288
2.4	Propriétés	290
2.5	Emplacements	292
2.6	Métaclasses	294
2.7	Classes abstraites	296

- 2.8 La Zope Component Architecture. 299
 - 2.8.1 Présentation 299
 - 2.8.2 Installation 299
 - 2.8.3 Définir une interface et un composant 300
 - 2.8.4 Autres fonctionnalités. 301
 - 2.8.5 Avantages de la ZCA 301
- 3. Fonctions spéciales et primitives associées 301
 - 3.1 Personnalisation 301
 - 3.1.1 Classes 301
 - 3.1.2 Instances. 303
 - 3.1.3 Comparaison 304
 - 3.1.4 Évaluation booléenne 304
 - 3.1.5 Relations d’héritage ou de classe à instance. 305
 - 3.2 Classes particulières 305
 - 3.2.1 Itérateurs 305
 - 3.2.2 Conteneurs. 308
 - 3.2.3 Instances assimilables à des fonctions 308
 - 3.2.4 Ressources à protéger 309
 - 3.2.5 Types 310
 - 3.2.6 Classes de données. 310

Chapitre 3.4

Types de données et algorithmes appliqués

- 1. Nombres. 311
 - 1.1 Types 311
 - 1.1.1 Entiers. 311
 - 1.1.2 Réels 312
 - 1.1.3 Socle commun aux nombres entiers et réels 313
 - 1.1.4 Méthodes dédiées aux nombres entiers 314
 - 1.1.5 Méthodes dédiées aux nombres réels 315
 - 1.1.6 Complexes 315
 - 1.2 La console Python, la calculatrice par excellence 316
 - 1.2.1 Opérateurs mathématiques binaires 316
 - 1.2.2 Opérateurs binaires particuliers 317
 - 1.2.3 Opérateurs mathématiques unaires 318
 - 1.2.4 Arrondis 319
 - 1.2.5 Opérateurs de comparaison 321

1.2.6	Opérations mathématiques n-aires	322
1.2.7	Fonctions mathématiques usuelles	323
1.3	Représentations d'un nombre	329
1.3.1	Représentation décimale	329
1.3.2	Représentation par un exposant	329
1.3.3	Représentation par une fraction	329
1.3.4	Représentation hexadécimale	330
1.3.5	Représentation octale	331
1.3.6	Représentation binaire	332
1.3.7	Opérations binaires	332
1.3.8	Longueur de la représentation mémoire d'un entier	334
1.4	Conversions	335
1.4.1	Conversion entre entiers et réels	335
1.4.2	Conversion entre réels et complexes	336
1.4.3	Conversion vers un booléen	336
1.5	Travailler avec des variables	337
1.5.1	Un nombre est non mutable	337
1.5.2	Modifier la valeur d'une variable	338
1.5.3	Opérateurs d'incrément	338
1.6	Statistiques	339
2.	Séquences	340
2.1	Présentation des différents types de séquences	340
2.1.1	Généralités	340
2.1.2	Les listes	341
2.1.3	Les n-uplets	342
2.1.4	Conversion entre listes et n-uplets	344
2.1.5	Socle commun entre liste et n-uplet	344
2.1.6	Notion d'itérateur	345
2.2	Utilisation des indices et des tranches	347
2.2.1	Définition de l'indice d'un objet et des occurrences	347
2.2.2	Utiliser l'indice pour adresser la séquence	349
2.2.3	Retrouver les occurrences d'un objet et leurs indices	350
2.2.4	Taille d'une liste, comptage d'occurrences	351
2.2.5	Utiliser l'indice pour modifier ou supprimer	352
2.2.6	Itération simple	354
2.2.7	Présentation de la notion de tranches (slices)	357
2.2.8	Cas particulier de la branche 2.x de Python	366

2.2.9	Utilisation basique des tranches	367
2.2.10	Utilisation avancée des tranches	368
2.3	Utilisation des opérateurs	370
2.3.1	Opérateur +	370
2.3.2	Opérateur *	371
2.3.3	Opérateur +=	373
2.3.4	Opérateur *=	375
2.3.5	Opérateur in	376
2.3.6	Opérateurs de comparaison	377
2.4	Méthodes de modifications	378
2.4.1	Ajouter des éléments dans une liste et un n-uplet	378
2.4.2	Supprimer un objet d'une liste et d'un n-uplet	380
2.4.3	Solutions de contournement pour la modification de n-uplets	384
2.4.4	Renverser une liste ou un tuple	385
2.4.5	Trier une liste	386
2.5	Utilisation avancée des listes	388
2.5.1	Opérations d'ensemble	388
2.5.2	Pivoter une séquence	389
2.5.3	Itérer correctement	390
2.5.4	Programmation fonctionnelle	391
2.5.5	Compréhensions de listes	394
2.5.6	Itérations avancées	395
2.5.7	Combinatoire	400
2.6	Adapter les listes à des besoins spécifiques	402
2.6.1	Liste d'entiers	402
2.6.2	Présentation du type array	403
2.6.3	Utiliser une liste comme pile	405
2.6.4	Utiliser une liste comme file d'attente	406
2.6.5	Conteneur plus performant	406
2.6.6	Utiliser des listes pour représenter des matrices	407
2.6.7	Liste sans doublons	408
2.7	Autres types de données	411
3.	Ensembles	413
3.1	Présentation	413
3.1.1	Définition d'un ensemble	413
3.1.2	Différences entre set et frozenset	414

3.1.3	Utilisation pour dédoubler des listes	415
3.1.4	Rajouter une relation d'ordre	415
3.2	Opérations ensemblistes	416
3.2.1	Opérateurs pour un ensemble à partir de deux autres	416
3.2.2	Opérateurs pour modifier un ensemble à partir d'un autre	417
3.2.3	Méthodes équivalentes à la création ou modification ensembliste	418
3.2.4	Méthodes de comparaison des ensembles	418
3.2.5	Exemples non classiques d'utilisation	419
3.3	Méthodes de modification d'un ensemble	423
3.3.1	Ajouter un élément	423
3.3.2	Supprimer un élément	423
3.3.3	Vider un ensemble	424
3.3.4	Dupliquer un élément	424
3.3.5	Sortir une valeur d'un ensemble	425
3.3.6	Utiliser un ensemble comme un recycleur d'objets	426
3.3.7	Algorithmique avancée : résolution du problème des n-dames	428
4.	Chaînes de caractères	430
4.1	Présentation	430
4.1.1	Définition	430
4.1.2	Vocabulaire	431
4.1.3	Spécificités de la branche 2.x	432
4.1.4	Changements apportés par la branche 3.x	433
4.1.5	Chaîne de caractères en tant que séquence de caractères	435
4.1.6	Caractères	437
4.1.7	Opérateurs de comparaison	438
4.2	Formatage de chaînes de caractères	441
4.2.1	Opérateur modulo	441
4.2.2	Méthodes de formatage sur l'ensemble de la chaîne	446
4.2.3	Nouvelle méthode de formatage des variables dans une chaîne	448
4.2.4	Littéraux formatés	451
4.3	Opérations d'ensemble	452
4.3.1	Séquençage de chaînes	452
4.3.2	Opérations sur la casse	454
4.3.3	Recherche sur une chaîne de caractères	455

4.3.4	Informations sur les caractères	456
4.4	Problématiques relatives à l'encodage	457
4.4.1	Encodage par défaut	457
4.4.2	Encodage du système.	458
4.4.3	L'unicode, référence absolue	458
4.4.4	Autres encodages	459
4.4.5	Ponts entre l'unicode et le reste du monde	460
4.4.6	Revenir vers l'Unicode.	461
4.5	Manipulations de bas niveau avancées	461
4.5.1	Opérations de comptage	461
4.5.2	Une chaîne de caractères vue comme une liste	462
4.5.3	Une chaîne de caractères vue comme un ensemble de caractères	463
4.6	Représentation mémoire	463
4.6.1	Présentation du type bytes	463
4.6.2	Lien avec les chaînes de caractères	464
4.6.3	Présentation du type bytearray	466
4.6.4	Gestion d'un jeu de caractères	468
5.	Dictionnaires	473
5.1	Présentation	473
5.1.1	Définition.	473
5.1.2	Évolutions et différences entre les branches 2.x et 3.x	474
5.1.3	Vues de dictionnaires.	475
5.1.4	Instanciation	477
5.1.5	Compréhension de dictionnaire	477
5.2	Manipuler un dictionnaire	478
5.2.1	Récupérer une valeur d'un dictionnaire	478
5.2.2	Modifier les valeurs d'un dictionnaire	479
5.2.3	Supprimer une entrée d'un dictionnaire	480
5.2.4	Dupliquer un dictionnaire.	480
5.2.5	Utiliser le dictionnaire comme agrégateur de données	481
5.2.6	Méthodes d'itération.	482
5.3	Utilisation avancée des dictionnaires	482
5.3.1	Rajouter une relation d'ordre	482
5.3.2	Algorithmiques classiques.	486
5.3.3	Adapter les dictionnaires à des besoins spécifiques	488
5.3.4	Représentation universelle de données	490

6.	Booléens	491
6.1	Le type booléen	491
6.1.1	Classe bool	491
6.1.2	Les deux objets True et False	492
6.1.3	Différence entre l'opérateur d'égalité et d'identité	492
6.2	Évaluation booléenne	492
6.2.1	Méthode générique	492
6.2.2	Objets classiques	492
7.	Données temporelles	493
7.1	Gérer une date calendaire	493
7.1.1	Notion de date calendaire	493
7.1.2	Travailler sur une date	494
7.1.3	Considérations astronomiques	495
7.1.4	Considérations historiques	495
7.1.5	Considérations techniques	495
7.1.6	Représentation textuelle	496
7.2	Gérer un horaire ou un moment d'une journée	498
7.2.1	Notion d'instant	498
7.2.2	Notion de fuseau horaire	499
7.2.3	Représentation textuelle	499
7.3	Gérer un instant absolu	500
7.3.1	Notion d'instant absolu	500
7.3.2	Rapport avec les notions précédentes	501
7.3.3	Représentation textuelle	502
7.3.4	Gestion des fuseaux horaires	503
7.3.5	Créer une date à partir d'une représentation textuelle	503
7.4	Gérer une différence entre deux dates ou instants	503
7.4.1	Notion de différence et de résolution	503
7.4.2	Considérations techniques	505
7.4.3	Utilisation avec des dates calendaires	506
7.4.4	Utilisation avec des horaires	506
7.4.5	Utilisation avec des dates absolues	506
7.4.6	La seconde comme unité de base	506
7.4.7	Précision à la nanoseconde	507
7.5	Spécificités des fuseaux horaires	507

- 7.6 Problématiques de bas niveau 508
 - 7.6.1 Timestamp et struct_time 508
 - 7.6.2 Mesures de performances 509
- 7.7 Utilisation du calendrier. 511
 - 7.7.1 Présentation du module calendar 511
 - 7.7.2 Fonctions essentielles du calendrier 516

Partie 4 : Les fonctionnalités

Chapitre 4.1
Manipulation de données

- 1. Manipuler des fichiers 519
 - 1.1 Ouvrir un fichier 519
 - 1.2 Lire un fichier 520
 - 1.3 Écrire un fichier. 521
 - 1.4 Comparer deux fichiers 522
- 2. Utilitaire de sauvegarde. 524
- 3. Lire un fichier de configuration 524
- 4. Format d'export/Import 525
 - 4.1 CSV 525
 - 4.1.1 Exploiter un fichier CSV 526
 - 4.1.2 Génération d'un fichier CSV 529
 - 4.2 JSON 531
 - 4.3 Base64 534
 - 4.4 Pickle 534
- 5. Compresser et décompresser un fichier 537
 - 5.1 Tarfile 537
 - 5.2 Gzip 539
 - 5.3 Bz2 539
 - 5.4 Zipfile 540
 - 5.5 Interface de haut niveau. 542
- 6. Outils de manipulation de données 543
 - 6.1 Générer des nombres aléatoires 543
 - 6.2 Expressions régulières. 544

7. Cryptographie légère	548
7.1 Nombre aléatoire sécurisé	548
7.2 Fonctions de chiffrement	548
7.3 Code d'authentification de message	550
7.4 Empreinte de fichier	551
7.5 Stéganographie	552
7.6 Communication inter-applicative sécurisée	555

Chapitre 4.2

Bases de données

1. Introduction	559
2. Accès à une base de données relationnelle	559
2.1 Point d'entrée	559
2.2 MySQL	560
2.3 PostgreSQL	565
2.4 SQLite	567
2.5 Oracle	567
3. Utilisation d'un ORM	568
3.1 Qu'est-ce qu'un ORM ?	568
3.2 ORM proposés par Python	568
3.3 SQLAlchemy	569
4. Autres bases de données	575
4.1 CSV	575
4.2 NoSQL	581
4.3 Base de données orientée objet : ZODB	581
4.4 Base de données orientée graphe : Neo4j	586
4.5 Base de données de type clé-valeur : Redis	587
4.6 Bases de données orientées documents : CouchDB et MongoDB	589
4.7 Bases de données natives XML : BaseX, eXist	590
4.8 Cassandra	591
4.9 Bases de données orientées colonnes : HBase	591
4.10 Big Data : l'écosystème Hadoop	593
5. LDAP	595
5.1 Protocole	595
5.2 Serveurs	595
5.3 Terminologie	596

- 5.4 Installation 596
- 5.5 Ouvrir une connexion à un serveur 596
- 5.6 Effectuer une recherche 598
- 5.7 Synchrone vs asynchrone 599
- 5.8 Connexions sécurisées 599

Partie 5 : Mise en pratique

Chapitre 5.1

Créer une application web en 30 minutes

- 1. Description de l'application à construire 601
- 2. Mise en place 602
 - 2.1 Isolation de l'environnement 602
 - 2.2 Création du projet 603
 - 2.3 Paramétrage 603
 - 2.4 Premiers essais 604
- 3. Réalisation de l'application 605
 - 3.1 Modèles 605
 - 3.2 Vues 608
 - 3.3 Contrôleurs 609
- 4. Pour aller plus loin 613

Chapitre 5.2

Créer une application console en 10 minutes

- 1. Objectif 615
- 2. Enregistrer le script 616
- 3. Création des données 616
- 4. Parseur d'arguments 617

Chapitre 5.3

Créer une application graphique en 20 minutes

- 1. Objectif 619
 - 1.1 Fonctionnel 619
 - 1.2 Technique 619

2. Présentation rapide de Gtk et d'astuces	620
2.1 Présentation	620
2.2 Astuces	620
3. Démarrer le programme	621
4. Interface graphique avec Glade	623
5. Créer le composant graphique	626
6. Contrôleur	628
7. Autres bibliothèques graphiques	629
7.1 TkInter	629
7.2 wxPython	629
7.3 PyQt	629
7.4 PySide	630
7.5 Autres	630

Chapitre 5.4

Créer un jeu en 30 minutes avec PyGame

1. Présentation de PyGame	631
2. Réalisation d'un jeu Tetris	632
2.1 Présentation du jeu	632
2.2 Présentation des problématiques	633
2.3 Création des constantes	633

Annexes

1. Objets mutables et non mutables	645
2. Table Unicode	647
2.1 Script	647
3. Bytes	647
3.1 Script	647
3.2 Résultat	647
4. Guide de portage vers Python 3	650

Index	653
-----------------	-----

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EI3PYT** dans la zone de recherche
et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Partie 1

Traitement des données

Chapitre 1.1

Motifs de conception

1.	Définition.	15
1.1	Positionnement par rapport à la notion d'objet	15
1.2	Organisation du chapitre	16
1.3	Positionnement par rapport à d'autres concepts	16
2.	Motifs de création	17
2.1	Singleton	17
2.2	Fabrique.	17
2.3	Fabrique abstraite	20
2.4	Monteur	21
2.5	Prototype.	23
3.	Motifs de structuration.	26
3.1	Adaptateur	26
3.2	Pont	29
3.3	Composite.	32
3.4	Décorateur.	34
3.5	Façade	36
3.6	Poids-mouche	38
3.7	Proxy	40
4.	Motifs de comportement	42
4.1	Chaîne de responsabilité	42
4.2	Commande	43
4.3	Itérateur.	45

4.4	Memento	47
4.5	Visiteur	48
4.6	Observateur	49
4.7	Stratégie	51
4.8	Fonction de rappel	52
5.	ZCA	53
5.1	Rappels	53
5.2	Adaptateur	53
5.3	Utilitaire	55
5.4	Fabrique	56
5.5	Pour aller plus loin	57

Chapitre 1.2

XML

1.	XML et les technologies qui gravitent autour	59
1.1	Définition de XML, terminologie associée	59
1.2	Notion de schéma	60
1.3	Avantages et inconvénients de XML	61
1.4	Différentes manières de parcourir un fichier XML	62
1.5	Modules Python dédiés au XML	62
2.	Valider un document XML	63
2.1	Document XML	63
2.2	Schéma DTD	64
2.3	Schéma XSD	64
2.4	Schéma RNG (RelaxNG)	65
2.5	Schematron	65
3.	DOM	66
3.1	Lecture	66
3.2	Écriture	67
4.	SAX	68
4.1	Support de SAX dans lxml	68
4.2	API SAX Allégée	69
5.	XPath	70
6.	XSLT	72

- 7. Cas spécifique des fichiers HTML 74
 - 7.1 Problématique 74
 - 7.2 Parser un fichier HTML à la façon DOM 74
 - 7.3 Parser un fichier HTML à la façon SAX 76

Chapitre 1.3
Génération de contenu

- 1. PDF 79
 - 1.1 Présentation 79
 - 1.1.1 Format PDF 79
 - 1.1.2 Avantages 79
 - 1.1.3 Inconvénients 79
 - 1.1.4 Présentation de la bibliothèque libre 80
 - 1.2 Bas niveau 80
 - 1.2.1 Bibliothèque de données 80
 - 1.2.2 Canvas 82
 - 1.3 Haut niveau 83
 - 1.3.1 Styles 83
 - 1.3.2 Flux de données 85
 - 1.3.3 Création d'un visuel 87
 - 1.3.4 Template de page 87
 - 1.3.5 Page contenant plusieurs zones 88
- 2. OpenDocument 90
 - 2.1 Présentation 90
 - 2.2 ezodf2 91
 - 2.2.1 Installation 91
 - 2.2.2 OpenDocument Texte 91
 - 2.2.3 OpenDocument Tableur 91
 - 2.2.4 Aller plus loin 92
 - 2.3 Alternatives 92
 - 2.3.1 lpod 92
 - 2.3.2 Génération à partir de templates 92
- 3. Travailler avec des images 93
 - 3.1 Représentation informatique d'une image 93
 - 3.2 Présentation de Pillow 94

3.3	Formats d'images matricielles	95
3.4	Récupérer des informations d'une image	97
3.5	Opérations d'ensemble sur une image	98
3.6	Travailler avec les calques ou les pixels	100

Chapitre 1.4

Qualité

1.	Programmation dirigée par les tests	103
1.1	Tests unitaires	103
1.1.1	Principes	103
1.1.2	Interprétation	104
1.1.3	Couverture	105
1.1.4	Outils	105
1.1.5	Autres outils	107
1.2	Tests de non-régression	107
1.2.1	Actions de développement	107
1.2.2	Gestion de la découverte d'une anomalie par une MOA	108
1.3	Tests fonctionnels	109
1.4	Tests de performance	110
1.5	Tests syntaxiques	112
1.6	Intégration continue	113
2.	Programmation dirigée par la documentation	114
2.1	Documentation interne	114
2.1.1	À destination des développeurs	114
2.1.2	À destination des utilisateurs	115
2.2	Documentation externe	115
2.2.1	Présentation	115
2.2.2	Démarrage rapide	115
2.2.3	Résultat	118
3.	Optimisation	118
3.1	Qualimétrie	118
3.2	Outils de débogage	120
3.3	Outils de profilage	120

- 3.4 Règles d'optimisation 122
 - 3.4.1 Pourquoi optimiser ? 122
 - 3.4.2 Règles générales 123
 - 3.4.3 Profiler un algorithme 124
 - 3.4.4 Optimiser l'utilisation de la mémoire 133

Partie 2

Programmation, système, réseau et scientifique

Chapitre 2.1

Programmation système

- 1. Présentation 135
 - 1.1 Définition 135
 - 1.2 Objectifs du chapitre 136
- 2. Appréhender son système d'exploitation. 136
 - 2.1 Avertissement 136
 - 2.2 Système d'exploitation. 136
 - 2.3 Processus courant 137
 - 2.4 Utilisateurs et groupes 138
 - 2.5 Mots de passe et authentification 140
 - 2.6 Constantes pour le système de fichiers. 141
 - 2.7 Gérer les chemins 142
 - 2.8 Utilitaires. 142
 - 2.8.1 Comparer deux fichiers 142
 - 2.8.2 Utilitaire de sauvegarde. 144
 - 2.8.3 Lire un fichier de configuration 145
 - 2.8.4 Pickle. 146
 - 2.9 Compresser et décompresser un fichier. 148
 - 2.9.1 Tarfile 148
 - 2.9.2 Gzip 150
 - 2.9.3 Bz2 151
 - 2.9.4 Zipfile 151
 - 2.9.5 Interface de haut niveau 153
- 3. Gestion d'un fichier 154
 - 3.1 Changer les droits d'un fichier. 154
 - 3.2 Changer de propriétaire ou de groupe 156

3.3	Récupérer des informations sur un fichier	157
3.4	Supprimer un fichier	157
4.	Alternatives simples à des commandes bash usuelles	158
4.1	Répertoires	158
4.2	Fichiers	160
4.3	Module de haut niveau	161
4.4	Recherche d'un fichier	163
5.	Exécuter des commandes externes	164
5.1	Exécuter et afficher le résultat	164
5.2	Exécuter et récupérer le résultat	165
5.3	Pour Python 3.5 et supérieur	166
6.	IHM système	166
6.1	Présentation	166
6.2	Travailler avec des arguments	167
6.3	Fermeture du programme	171
6.4	Entrée standard	172
6.5	Sortie standard	173
6.6	Sortie d'erreur	174
6.7	Taille du terminal	175
7.	Curses	176
7.1	Présentation	176
7.2	Gérer le démarrage et l'arrêt	176
7.3	Affichage de texte	177
7.4	Gestion des attributs	177
7.5	Gestion des couleurs	178
7.6	Gestion des événements	178
8.	Accès aux périphériques	178
8.1	Introduction	179
8.2	Voir les partitions	179
8.3	Voir les clés USB	180
8.4	Liste des informations disponibles	181
8.5	Détecter le branchement d'une clé USB	181
8.6	Communiquer via un port série	182
8.7	Communiquer via un port USB	183

Chapitre 2.2
Programmation réseau

- 1. Écrire un serveur et un client 185
 - 1.1 Utilisation d'une socket TCP..... 185
 - 1.2 Utilisation d'une socket UDP 189
 - 1.3 Création d'un serveur TCP 191
 - 1.4 Création d'un serveur UDP 193
 - 1.5 Un peu plus loin sur le sujet 194
- 2. Utiliser un protocole standard 195
 - 2.1 Client HTTP 195
 - 2.2 Serveur HTTP 196
 - 2.3 Proxy 199
 - 2.4 Cookies 200
 - 2.5 FTP et SFTP..... 200
 - 2.6 SSH 202
 - 2.7 POP et POPS 204
 - 2.8 IMAP et IMAPS 205
 - 2.9 SMTP et SMPTS..... 206
 - 2.10 NNTP 211
 - 2.11 IRC..... 212
- 3. Wake-on-LAN 215
 - 3.1 Prérequis 215
 - 3.2 Mise en œuvre..... 215

Chapitre 2.3
Programmation web

- 1. Services web..... 217
 - 1.1 REST 217
 - 1.2 SOAP 218
 - 1.3 Pyro 220
- 2. Client web 221
- 3. Web scrapping 225

Chapitre 2.4**Programmation scientifique**

1.	Calcul scientifique	227
1.1	Présentation	227
1.2	Python, une alternative libre et crédible	227
1.3	Vue d'ensemble de quelques bibliothèques	228
2.	Tableaux multidimensionnels	229
2.1	Création	229
2.2	Déterminer la composition d'un tableau	230
2.3	Générateurs de tableaux	231
2.4	Opérations basiques	232
2.5	Opérateur crochet	234
3.	Matrices	236
4.	Génération de graphiques	237
4.1	Syntaxe MATLAB	238
4.2	Syntaxe objet	239
4.3	Mise en forme	239
4.4	Graphiques 3D	244
5.	Introduction à Pandas	247
5.1	Présentation	247
5.2	Séries	247
5.3	Dataframes	251

Partie 3**Programmation concurrente****Chapitre 3.1****Programmation concurrente**

1.	Notion de performance	259
1.1	Programmation bloquante	259
1.2	Utilisation de ressources CPU	260
1.3	Organisation de l'application	261

2.	Terminologie	262
2.1	Processus	262
2.2	Fil d'exécution	263
2.3	Programmation non bloquante	264
2.4	Notion de GIL	264
3.	Présentation des paradigmes	265
3.1	Programmation asynchrone	265
3.2	Programmation parallèle	265
3.3	Programmation distribuée	266
3.4	Programmation concurrente	267

Chapitre 3.2

Programmation asynchrone : initiation

1.	Utilité de la programmation asynchrone	269
2.	Introduction à l'asynchrone	270
2.1	Notion de coroutine	270
2.2	Exécution d'une coroutine	271
2.3	Précisions sur le mot-clé await	272
2.4	Exécuter plusieurs coroutines en séries	273
2.5	Tâches asynchrones	274
2.6	Exécuter les tâches asynchrones de manière concurrente	276
2.7	Notion d'awaitable	276
2.8	Précisions sur <code>asyncio.sleep</code>	276
3.	Éléments de grammaire	277
3.1	Gestionnaire de contexte asynchrone	277
3.2	Générateurs asynchrones	278
3.3	Compréhensions asynchrones	278
3.4	Itération asynchrone	278
4.	Notions avancées	278
4.1	Introspection	278
4.2	Gestion du résultat ou de l'exception	282
4.3	Annuler une tâche	283
4.4	Se prémunir d'une annulation	285
4.5	Timeouts	286
4.6	Gestion globale fine de l'attente	286

4.7	Module contextvars	289
5.	Boucle événementielle asynchrone	292
5.1	Gestion de la boucle	292
5.2	Débogage	293
5.3	Notion de future	295
5.4	Annulation	297
5.5	Gestion des exceptions	298
6.	Utiliser la boucle événementielle	302
6.1	Utilisation des fonctions de retour (callbacks)	302
6.2	Planifier des appels à des fonctions classiques	305
6.3	Utiliser des signaux	306
6.4	Synchronisation	309
6.5	Communication entre coroutines	313
6.6	Exemple : lecture de l'entrée standard	317
6.7	Programmation parallèle et asynchrone	318
6.8	Exécuter du code bloquant	318
7.	L'asynchrone suivant les versions de Python	319
7.1	Python 3.7	319
7.2	Python 3.6	320
7.3	Python 3.5	320
7.4	Python 3.4	321
7.5	Python 3.3 et inférieur	321
7.6	Python 2.7	322
8.	Cas concret	322
8.1	Exemple de travail	322
9.	Exemple retravaillé en utilisant des générateurs	325
9.1	Télécharger en asynchrone	326
9.2	Parser en asynchrone	327
9.3	Faire plusieurs traitements en asynchrone	328
9.4	Utiliser un exécuteur	329
9.5	Pour aller plus loin	329

Chapitre 3.3
Programmation asynchrone : avancée

- 1. Transports et protocoles 331
 - 1.1 Introduction 331
 - 1.2 Transports 331
 - 1.3 Protocoles 335
 - 1.4 Mise en œuvre 336
 - 1.5 Exemple pour le protocole TCP 338
 - 1.6 Exemple pour le protocole SSL 346
 - 1.7 Exemple pour le protocole UDP 349
 - 1.8 Traiter d’autres types de protocoles réseau 352
 - 1.9 Exemple pour Subprocess 353
- 2. Flux 359
 - 2.1 Introduction 359
 - 2.2 Exemple avec le protocole TCP 359

Chapitre 3.4
Programmation asynchrone : alternatives

- 1. Gevent 363
 - 1.1 Introduction 363
 - 1.2 DNS 363
 - 1.3 Appels système 364
 - 1.4 Programmation asynchrone 365
- 2. Twisted 366
 - 2.1 Introduction 366
 - 2.2 Client/Serveur TCP 366
 - 2.3 Serveur/Client UDP 368
 - 2.4 AMP 369
 - 2.5 Autres protocoles 372
- 3. Trollius 372
- 4. HTTP asynchrone avec aiohttp 374
 - 4.1 Côté serveur 374
 - 4.2 Côté client 376

Chapitre 3.5

Programmation parallèle

1.	Utilisation d'un fil d'exécution	377
1.1	Gestion d'un fil d'exécution	377
1.1.1	Présentation	377
1.1.2	Création	377
1.2	Gestion de plusieurs fils d'exécution	380
1.2.1	Lancement et contrôle	380
1.2.2	Opportunité d'utiliser un fil d'exécution	382
1.3	Résolution des problématiques liées	384
1.3.1	Synchronisation	384
1.3.2	Synchronisation conditionnelle	386
1.3.3	Sémaphore	388
2.	Utilisation de processus	390
2.1	Gestion d'un processus	390
2.1.1	Présentation	390
2.1.2	Création	390
2.2	Gestion de plusieurs processus	393
2.2.1	Synchronisation	393
2.2.2	Paralléliser un travail	393
2.3	Résolution des problématiques liées	395
2.3.1	Communication interprocessus	395
2.3.2	Partage de données entre processus	397
2.4	Opportunité d'utiliser les processus	398
2.5	Démon	398
3.	Exécution asynchrone	400
3.1	Introduction	400
3.2	Présentation	401

Chapitre 3.6

Programmation distribuée

1.	Définitions	407
2.	ØMQ	407
2.1	Présentation générale	407
2.2	Pair	409
2.3	Client/Serveur	410
2.4	Publish/Subscribe	411
2.5	Push/Pull	415
2.6	Patron pipeline	416
3.	AMQP avec RabbitMQ	418
3.1	Installation	418
3.2	Configuration	418
3.3	Introduction	419
3.4	Notions importantes	421
3.5	Publish/Subscribe	422
3.6	Routage et sujets	423
4.	Celery	425
4.1	Présentation	425
4.2	Dans quel cas utiliser Celery ?	425
4.3	Installation	425
4.4	Configuration	426
4.5	Utilisation	427
4.6	Monitorer	429
5.	Crossbar	429
5.1	Présentation	429
5.2	WebSocket	430
5.3	Publish/Subscribe	436
	Index	439