

Les éléments à télécharger sont disponibles à l'adresse suivante :

<http://www.editions-eni.fr>

Saisissez la référence ENI de l'ouvrage **HSRI43PYT** 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	26
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	38
2.1.6 Paradigme logique	38
2.1.7 Programmation concurrente	38
2.1.8 Synthèse	39
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	44
2.5.1	Langages formels	44
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	47
3.2.1	Extensions C	47
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	50
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	56
1.3.3	Documentation	56
1.3.4	Python est livré piles incluses	56
1.3.5	Duck Typing	57

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	67
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	68
5.1	Chargement de la machine virtuelle	68
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	74
1.4	Domaines d'excellence	74
1.5	Garanties	75
2.	Diffusion	76
2.1	Entreprises	76
2.2	Le monde de la recherche	78
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	83
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	86
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	93
2.4	Par la compilation	94
2.5	Pour un smartphone	94
3.	Installer une bibliothèque tierce	95
3.1	À partir de Python 3.4	95
3.2	Pour une version inférieure à Python 3.4	97

3.3 Pour Linux	97
4. Créer un environnement virtuel	97
4.1 À quoi sert un environnement virtuel ?	97
4.2 Pour Python 3.3 ou version supérieure	98
4.3 Pour toute version de Python	98
4.4 Pour Linux	100
5. Gestion des dépendances	101
6. Installer Anaconda	102
6.1 Pour Windows	102
6.2 Pour Linux	103
6.3 Pour Mac	103
6.4 Mettre à jour Anaconda	103
6.5 Installer une bibliothèque externe	103
6.6 Environnements virtuels	104
7. Docker	104
8. La console Python	105
8.1 Démarrer la console Python	105
8.2 BPython	105
8.3 IPython	106
8.4 IPython Notebook	106
9. Installer un IDE	107
9.1 Liste d'IDE	107
9.2 Présentation de PyCharm	108
9.3 Configuration de PyCharm	108
10. VSCode	112

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	123
2.7	Conditions avancées	125
2.8	Bloc itératif	125
3.	Premier jeu : Devine le nombre	127
3.1	Description du jeu	127
3.2	Aides	127
3.2.1	Gestion du hasard	127
3.2.2	Étapes de développement	128
3.3	Pour aller plus loin	128

Chapitre 2-2

Fonctions et modules

1.	Les fonctions	129
1.1	Pourquoi utiliser des fonctions ?	129
1.2	Introduction aux fonctions	131
1.2.1	Comment déclarer une fonction	131
1.2.2	Gestion d'un paramètre	132
1.2.3	Comment rendre une fonction plus générique	134
1.2.4	Paramètres par défaut	136
1.3	Problématiques de couplage et duplication de code	137
1.3.1	Niveau de ses fonctions	137
1.3.2	Notion de complexité	139
1.3.3	Bonnes pratiques	141
2.	Les modules	142
2.1	Introduction	142
2.1.1	Qu'est-ce qu'un module ?	142
2.1.2	Comment crée-t-on un module Python ?	143
2.1.3	Organiser son code	143
2.2	Gérer le code de ses modules	143
2.2.1	Exécuter un module, importer un module	143
2.2.2	Gérer une arborescence de modules	144

3. Terminer le jeu	145
3.1 Créer des niveaux	146
3.2 Déterminer un nombre de coups maximal	146
3.3 Enregistrer les meilleurs scores	146
3.4 Intelligence artificielle	146

Chapitre 2-3

Les principaux types

1. Chaînes de caractères	147
1.1 Syntaxe	147
1.2 Formatage d'une chaîne	148
1.3 Notion de casse	148
1.4 Notion de longueur	149
1.5 Appartenance	150
1.6 Notion d'occurrence	150
1.7 Remplacement	151
1.8 Notion de caractère	151
1.9 Typologie des caractères	152
1.10 Séquencer une chaîne de caractères	153
2. Listes	154
2.1 Syntaxe	154
2.2 Indices	154
2.3 Valeurs	155
2.4 Hasard	157
2.5 Techniques d'itération	157
2.6 Tri	160
3. Dictionnaires	162
3.1 Présentation des dictionnaires	162
3.2 Parcourir un dictionnaire	162
3.3 Exemple	163

Chapitre 2-4

Les classes

1. Syntaxe	165
2. Notion d'instance courante	166
3. Opérateurs	169
4. Héritage	170
4.1 Spécialisation	171
4.2 Programmation par composants	172

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
1.10 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	Instruction de correspondance	197
2.4	Utilisation d'une expression d'affectation	200
2.4.1	Instruction switch	200
2.4.2	Interruptions	201
2.4.3	Approfondissement des conditions	201
2.4.4	Performances	202
2.5	Itérations	203
2.5.1	Instruction for	203
2.5.2	Instruction while	203
2.5.3	Quelle différence entre for et while ?	204
2.5.4	Instruction break	204
2.5.5	Instruction return	206
2.5.6	Instruction continue	206
2.5.7	Instruction else	206
2.5.8	Générateurs	207
2.6	Constructions fonctionnelles	210
2.6.1	Construction conditionnelle	210
2.6.2	Générateurs	210
2.6.3	Compréhensions de listes	210
2.6.4	Compréhensions d'ensembles	211
2.6.5	Compréhensions de dictionnaires	211
2.7	Compréhensions et expressions d'affectation	211
2.8	Gestion des exceptions	211
2.8.1	Présentation rapide des exceptions	211
2.8.2	Lever une exception	212
2.8.3	Pourquoi lever une exception ?	212
2.8.4	Assertions	213
2.8.5	Capturer une exception	214
2.8.6	Effectuer un traitement de l'exception	215
2.8.7	Gérer la sortie du bloc de capture	217
2.8.8	Gérer le non-déclenchement d'exceptions	217
2.8.9	Gestionnaire de contexte	219
2.8.10	Programmation asynchrone	220

2.9	Divers	221
2.9.1	Gérer des imports	221
2.9.2	Traverser les espaces de nommage	222
2.9.3	Fonctions print, help, eval et exec	224

Chapitre 3-2

Déclarations

1.	Variable	227
1.1	Qu'est-ce qu'une variable ?	227
1.1.1	Contenu	227
1.1.2	Contenant	227
1.1.3	Modes de modification d'une variable	229
1.2	Typage dynamique	232
1.2.1	Affectation : rappels	232
1.2.2	Primitive type et nature du type	232
1.2.3	Caractéristiques du typage Python	233
1.3	Visibilité	236
1.3.1	Espace global	236
1.3.2	Notion de bloc	236
2.	Fonction	240
2.1	Déclaration	240
2.2	Paramètres	241
2.2.1	Signature d'une fonction	241
2.2.2	Notion d'argument ou de paramètre	242
2.2.3	Valeur par défaut	243
2.2.4	Valeur par défaut mutable	244
2.2.5	Paramètres nommés	245
2.2.6	Déclaration de paramètres extensibles	246
2.2.7	Passage de paramètres étoilés	247
2.2.8	Signature universelle	248
2.2.9	Obliger un paramètre à être nommé (keyword-only)	249
2.3	Obliger un paramètre à être positionnel (positional-only)	251
2.3.1	Annotations/type hint/typage statique	251
3.	Classe	255
3.1	Déclaration	255
3.1.1	Signature	255
3.1.2	Attribut	255
3.1.3	Méthode	256
3.1.4	Bloc local	256

3.2	Instanciation	257
3.2.1	Syntaxe	257
3.2.2	Relation entre l'instance et la classe	257
4.	Module	258
4.1	À quoi sert un module ?	258
4.2	Déclaration	258
4.3	Instructions spécifiques	259
4.4	Comment appréhender le contenu d'un module ?	260
4.5	Compilation des modules	260

Chapitre 3-3

Modèle objet

1.	Tout est objet	263
1.1	Principes	263
1.1.1	Quel sens donner à « objet » ?	263
1.1.2	Adaptation de la théorie objet dans Python	264
1.1.3	Généralités	265
1.2	Classes	266
1.2.1	Introduction	266
1.2.2	Déclaration impérative d'une classe	266
1.2.3	Instance	267
1.2.4	Objet courant	268
1.2.5	Déclaration par prototype d'une classe	269
1.2.6	Tuples nommés	271
1.3	Méthodes	272
1.3.1	Déclaration	272
1.3.2	Appel de méthode	273
1.3.3	Méthodes et attributs spéciaux	276
1.3.4	Constructeur et initialisateur	279
1.3.5	Gestion automatisée des attributs	280
1.3.6	Intérêt du paradigme objet	281
1.3.7	Relation entre objets	281
1.4	Héritage	282
1.4.1	Polymorphisme par sous-typage	282
1.4.2	Surcharge de méthode	283
1.4.3	Surcharge des opérateurs	285
1.4.4	Polymorphisme paramétrique	285
1.4.5	Héritage multiple	287

2. Autres outils de la programmation objet	289
2.1 Principes	289
2.2 Interfaces	290
2.3 Attributs	292
2.4 Propriétés	294
2.5 Emplacements	297
2.6 Métaclasses	298
2.7 Classes abstraites	300
2.8 La Zope Component Architecture	303
2.8.1 Présentation	303
2.8.2 Installation	303
2.8.3 Définir une interface et un composant	304
2.8.4 Autres fonctionnalités	305
2.8.5 Avantages de la ZCA	305
3. Fonctions spéciales et primitives associées	305
3.1 Personnalisation	305
3.1.1 Classes	305
3.1.2 Instances	307
3.1.3 Comparaison	308
3.1.4 Évaluation booléenne	308
3.1.5 Relations d'héritage ou de classe à instance	309
3.2 Classes particulières	309
3.2.1 Itérateurs	309
3.2.2 Conteneurs	312
3.2.3 Instances assimilables à des fonctions	312
3.2.4 Ressources à protéger	313
3.2.5 Types	314
3.2.6 Classes de données	314

Chapitre 3-4

Nombres, booléens et algorithmes appliqués

1. Nombres	315
1.1 Types	315
1.1.1 Entiers	315
1.1.2 Réels	316
1.1.3 Socle commun aux nombres entiers et réels	317
1.1.4 Méthodes dédiées aux nombres entiers	318
1.1.5 Méthodes dédiées aux nombres réels	319
1.1.6 Complexes	319

1.2	La console Python, la calculatrice par excellence	320
1.2.1	Opérateurs mathématiques binaires	320
1.2.2	Opérateurs binaires particuliers	321
1.2.3	Opérateurs mathématiques unaires	322
1.2.4	Arrondis	323
1.2.5	Opérateurs de comparaison	326
1.2.6	Opérations mathématiques n-aires	327
1.2.7	Fonctions mathématiques usuelles.	328
1.3	Représentations d'un nombre	333
1.3.1	Représentation décimale	333
1.3.2	Représentation par un exposant.	333
1.3.3	Représentation par une fraction.	333
1.3.4	Représentation hexadécimale	334
1.3.5	Représentation octale	335
1.3.6	Représentation binaire	336
1.3.7	Opérations binaires	336
1.3.8	Longueur de la représentation mémoire d'un entier	338
1.4	Conversions	340
1.4.1	Conversion entre entiers et réels	340
1.4.2	Conversion entre réels et complexes	340
1.4.3	Conversion vers un booléen	341
1.5	Travailler avec des variables.	342
1.5.1	Un nombre est non mutable.	342
1.5.2	Modifier la valeur d'une variable	342
1.5.3	Opérateurs d'incrémentation	343
1.6	Statistiques	344
2.	Booléens	345
2.1	Le type booléen	345
2.1.1	Classe bool	345
2.1.2	Les deux objets True et False	346
2.1.3	Différence entre l'opérateur d'égalité et d'identité	346
2.2	Évaluation booléenne	346
2.2.1	Méthode générique	346
2.2.2	Objets classiques	346

Chapitre 3-5
Séquences et algorithmes appliqués

1.	Présentation des différents types de séquences	349
1.1	Généralités	349
1.2	Les listes	350
1.3	Les n-uplets	351
1.4	Conversion entre listes et n-uplets	353
1.5	Socle commun entre liste et n-uplet	353
2.	Notion d'itérateur	354
3.	Utilisation des indices et des tranches	356
3.1	Définition de l'indice d'un objet et des occurrences	356
3.2	Utiliser l'indice pour adresser la séquence	358
3.3	Retrouver les occurrences d'un objet et leurs indices	359
3.4	Taille d'une liste, comptage d'occurrences	360
3.5	Utiliser l'indice pour modifier ou supprimer	361
3.6	Itération simple	363
3.7	Présentation de la notion de tranches (slices)	366
3.8	Cas particulier de la branche 2.x de Python	375
3.9	Utilisation basique des tranches	376
3.10	Utilisation avancée des tranches	377
4.	Utilisation des opérateurs	379
4.1	Opérateur +	379
4.2	Opérateur *	380
4.3	Opérateur +=	383
4.4	Opérateur *=	384
4.5	Opérateur in	385
4.6	Opérateurs de comparaison	386
5.	Méthodes de modifications	387
5.1	Ajouter des éléments dans une liste et un n-uplet	387
5.2	Supprimer un objet d'une liste et d'un n-uplet	389
5.3	Solutions de contournement pour la modification de n-uplets	393
5.4	Renverser une liste ou un tuple	394
5.5	Trier une liste	395
6.	Utilisation avancée des listes	398
6.1	Opérations d'ensemble	398
6.2	Pivoter une séquence	399
6.3	Itérer correctement	400
6.4	Programmation fonctionnelle	401

6.5	Compréhensions de listes	403
6.6	Itérations avancées	405
6.7	Combinatoire	409
7.	Adapter les listes à des besoins spécifiques	412
7.1	Liste d'entiers	412
7.2	Présentation du type array	414
7.3	Utiliser une liste comme pile	415
7.4	Utiliser une liste comme file d'attente	416
7.5	Conteneur plus performant	416
7.6	Utiliser des listes pour représenter des matrices	417
7.7	Liste sans doublons	419
8.	Autres types de données	421

Chapitre 3-6

Ensembles et algorithmes appliqués

1.	Présentation	425
1.1	Définition d'un ensemble	425
1.2	Différences entre set et frozenset	426
1.3	Utilisation pour dédoublonner des listes	427
1.4	Rajouter une relation d'ordre	427
2.	Opérations ensemblistes	428
2.1	Opérateurs pour un ensemble à partir de deux autres	428
2.2	Opérateurs pour modifier un ensemble à partir d'un autre	429
2.3	Méthodes équivalentes à la création ou modification ensembliste	430
2.4	Méthodes de comparaison des ensembles	430
2.5	Exemples non classiques d'utilisation	431
3.	Méthodes de modification d'un ensemble	435
3.1	Ajouter un élément	435
3.2	Supprimer un élément	435
3.3	Vider un ensemble	436
3.4	Dupliquer un élément	436
3.5	Sortir une valeur d'un ensemble	437
3.6	Utiliser un ensemble comme un recycleur d'objets	438
3.7	Algorithmique avancée : résolution du problème des n-dames	441

Chapitre 3-7
Chaînes de caractères et algorithmes appliqués

1.	Présentation	443
1.1	Définition	443
1.2	Vocabulaire	444
1.3	Spécificités de la branche 2.x	445
1.4	Changements apportés par la branche 3.x	446
1.5	Chaîne de caractères en tant que séquence de caractères	448
1.6	Caractères	450
1.7	Opérateurs de comparaison	451
2.	Formatage de chaînes de caractères	454
2.1	Opérateur modulo	454
2.2	Méthodes de formatage sur l'ensemble de la chaîne	459
2.3	Nouvelle méthode de formatage des variables dans une chaîne	462
2.4	Littéraux formatés	465
3.	Opérations d'ensemble	466
3.1	Séquençage de chaînes	466
3.2	Opérations sur la casse	468
3.3	Recherche sur une chaîne de caractères	470
3.4	Informations sur les caractères	470
4.	Problématiques relatives à l'encodage	472
4.1	Encodage par défaut	472
4.2	Encodage du système	472
4.3	L'unicode, référence absolue	472
4.4	Autres encodages	473
4.5	Ponts entre l'unicode et le reste du monde	474
4.6	Revenir vers l'Unicode	475
5.	Manipulations de bas niveau avancées	476
5.1	Opérations de comptage	476
5.2	Une chaîne de caractères vue comme une liste	477
5.3	Une chaîne de caractères vue comme un ensemble de caractères	478
6.	Représentation mémoire	478
6.1	Présentation du type bytes	478
6.2	Lien avec les chaînes de caractères	479
6.3	Présentation du type bytearray	480
6.4	Gestion d'un jeu de caractères	482

Chapitre 3-8
Dictionnaires et algorithmes appliqués

1.	Présentation	489
1.1	Définition	489
1.2	Évolutions et différences entre les branches 2.x et 3.x	490
1.3	Vues de dictionnaires	491
1.4	Instanciation	493
1.5	Compréhension de dictionnaire	494
2.	Manipuler un dictionnaire	494
2.1	Récupérer une valeur d'un dictionnaire	494
2.2	Modifier les valeurs d'un dictionnaire	495
2.3	Supprimer une entrée d'un dictionnaire	497
2.4	Dupliquer un dictionnaire	497
2.5	Utiliser le dictionnaire comme agrégateur de données	498
2.6	Méthodes d'itération	499
3.	Utilisation avancée des dictionnaires	499
3.1	Rajouter une relation d'ordre	499
3.2	Algorithmiques classiques	500
3.3	Adapter les dictionnaires à des besoins spécifiques	503
3.4	Représentation universelle de données	505

Chapitre 3-9
Données temporelles et algorithmes appliqués

1.	Gérer une date calendaire	507
1.1	Notion de date calendaire	507
1.2	Travailler sur une date	508
1.3	Considérations astronomiques	509
1.4	Considérations historiques	509
1.5	Considérations techniques	509
1.6	Représentation textuelle	510
2.	Gérer un horaire ou un moment d'une journée	512
2.1	Notion d'instant	512
2.2	Notion de fuseau horaire	513
2.3	Représentation textuelle	513
3.	Gérer un instant absolu	514
3.1	Notion d'instant absolu	514
3.2	Rapport avec les notions précédentes	515
3.3	Représentation textuelle	517

3.4	Gestion des fuseaux horaires	517
3.5	Créer une date à partir d'une représentation textuelle	517
4.	Gérer une différence entre deux dates ou instants	518
4.1	Notion de différence et de résolution	518
4.2	Considérations techniques	519
4.3	Utilisation avec des dates calendaires	520
4.4	Utilisation avec des horaires	520
4.5	Utilisation avec des dates absolues	520
4.6	La seconde comme unité de base	521
4.7	Précision à la nanoseconde	521
5.	Spécificités des fuseaux horaires	521
6.	Problématiques de bas niveau	522
6.1	Timestamp et struct_time	522
6.2	Mesures de performances	523
7.	Utilisation du calendrier	526
7.1	Présentation du module calendar	526
7.2	Fonctions essentielles du calendrier	530

Partie 4 : Les fonctionnalités

Chapitre 4-1

Manipulation de données

1.	Manipuler des fichiers	533
1.1	Ouvrir un fichier	533
1.2	Lire un fichier	534
1.3	Écrire un fichier	535
1.4	Comparer deux fichiers	536
2.	Utilitaire de sauvegarde	538
3.	Lire un fichier de configuration	538
4.	Format d'export/import	539
4.1	CSV	539
4.1.1	Exploiter un fichier CSV	540
4.1.2	Génération d'un fichier CSV	543
4.2	JSON	546
4.3	Base64	548
4.4	Pickle	549

5.	Compresser et décompresser un fichier	551
5.1	Tarfile	551
5.2	Gzip	553
5.3	BZ2	554
5.4	Zipfile	554
5.5	Interface de haut niveau	556
6.	Outils de manipulation de données	557
6.1	Générer des nombres aléatoires	557
6.2	Expressions régulières	558
7.	Cryptographie légère	562
7.1	Nombre aléatoire sécurisé	562
7.2	Fonctions de chiffrement	563
7.3	Code d'authentification de message	565
7.4	Empreinte de fichier	566
7.5	Stéganographie	566
7.6	Communication inter-applicative sécurisée	570

Chapitre 4-2

Bases de données

1.	Introduction	573
2.	Accès à une base de données relationnelle	573
2.1	Point d'entrée	573
2.2	MySQL	574
2.3	PostgreSQL	579
2.4	SQLite	581
2.5	Oracle	582
3.	Utilisation d'un ORM	582
3.1	Qu'est-ce qu'un ORM ?	582
3.2	ORM proposés par Python	582
3.3	SQLAlchemy	583
3.3.1	Introspection sur une table existante	583
3.3.2	Manipuler des données sur une table existante	586
3.3.3	Décrire une base de données par le code	590
4.	Autres bases de données	591
4.1	CSV	591
4.2	NoSQL	597
4.3	Base de données orientée objet : ZODB	598
4.4	Base de données orientée graphe : Neo4j	602

4.5	Base de données de type clé-valeur : Redis	604
4.6	Bases de données orientées documents : CouchDB et MongoDB	605
4.7	Bases de données natives XML : BaseX, eXist	606
4.8	Cassandra	607
4.9	Bases de données orientées colonnes : HBase	607
4.10	Big Data : l'écosystème Hadoop	609
5.	LDAP	612
5.1	Protocole	612
5.2	Serveurs	612
5.3	Terminologie	612
5.4	Installation	613
5.5	Ouvrir une connexion à un serveur	613
5.6	Effectuer une recherche	614
5.7	Synchrone vs asynchrone	615
5.8	Connexions sécurisées	616

Partie 5 : Mise en pratique

Chapitre 5-1

Créer un environnement de travail en 10 minutes

1.	Description de l'application à construire	617
2.	Containers	618
2.1	Portainer	618
2.2	Base de données	619
3.	Créer son container Docker	621
4.	Installer ses bibliothèques Python	623

Chapitre 5-2

Créer une application web en 30 minutes

1.	Description de l'application à construire	625
2.	Mise en place	626
2.1	Isolation de l'environnement	626
2.2	Création du projet	627
2.3	Paramétrage	627
2.4	Premiers essais	629

3.	Réalisation de l'application	630
3.1	Modèles	630
3.2	Templates	632
3.3	Vues	634
4.	Pour aller plus loin	638

Chapitre 5-3

Créer une application console en 10 minutes

1.	Objectif	639
2.	Enregistrer le script	640
3.	Création des données	640
4.	Parseur d'arguments	641

Chapitre 5-4

Créer une application graphique en 20 minutes

1.	Objectif	643
1.1	Fonctionnel	643
1.2	Technique	643
2.	Présentation rapide de Gtk et d'astuces	644
2.1	Présentation	644
2.2	Astuces	644
3.	Démarrer le programme	645
4.	Interface graphique avec Glade	648
5.	Créer le composant graphique	650
6.	Contrôleur	652
7.	Autres bibliothèques graphiques	653
7.1	TkInter	653
7.2	wxPython	653
7.3	PyQt	653
7.4	PySide	654
7.5	Autres	654

Chapitre 5-5
Créer un jeu en 30 minutes avec PyGame

1. Présentation de PyGame	655
2. Réalisation d'un jeu Tetris	656
2.1 Présentation du jeu	656
2.2 Présentation des problématiques	657
2.3 Création des constantes	657

Annexes

1. Objets mutables et non mutables	669
2. Table Unicode	672
3. Bytes	672
4. Guide de portage vers Python 3	675
5. Comment déboguer	677
6. Comment tester la performance	678
 Index	681

Les éléments à télécharger sont disponibles à l'adresse suivante :

<http://www.editions-eni.fr>

Saisissez la référence de l'ouvrage **EI23PYT** 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	18
2.3	Fabrique abstraite	21
2.4	Monteur	23
2.5	Prototype	25
3.	Motifs de structuration	28
3.1	Adaptateur	28
3.2	Pont	31
3.3	Composite	34
3.4	Décorateur	36
3.5	Façade	38
3.6	Poids-mouche	40
3.7	Proxy	42
4.	Motifs de comportement	44
4.1	Chaîne de responsabilité	44
4.2	Commande	45

4.3	Itérateur	47
4.4	Memento	49
4.5	Visiteur	50
4.6	Observateur	51
4.7	Stratégie	53
4.8	Fonction de rappel	54
5.	ZCA	55
5.1	Rappels	55
5.2	Adaptateur	55
5.3	Utilitaire	57
5.4	Fabrique	58
5.5	Pour aller plus loin	59

Chapitre 1.2

XML

1.	XML et les technologies qui gravitent autour	61
1.1	Définition de XML, terminologie associée	61
1.2	Notion de schéma	62
1.3	Avantages et inconvénients de XML	63
1.4	Différentes manières de parcourir un fichier XML	64
1.5	Modules Python dédiés au XML	64
2.	Valider un document XML	65
2.1	Document XML	65
2.2	Schéma DTD	66
2.3	Schéma XSD	67
2.4	Schéma RNG (RelaxNG)	67
2.5	Schematron	68
3.	DOM	68
3.1	Lecture	68
3.2	Écriture	69
4.	SAX	70
4.1	Support de SAX dans lxml	70
4.2	API SAX Allégée	71
5.	XPath	72

6.	XSLT	74
7.	Cas spécifique des fichiers HTML	76
7.1	Problématique	76
7.2	Parser un fichier HTML à la façon DOM	76
7.3	Parser un fichier HTML à la façon SAX	78

Chapitre 1.3

Génération de contenu

1.	PDF	81
1.1	Présentation	81
1.1.1	Format PDF	81
1.1.2	Avantages	81
1.1.3	Inconvénients	81
1.1.4	Présentation de la bibliothèque libre	82
1.2	Bas niveau	82
1.2.1	Bibliothèque de données	82
1.2.2	Canvas	84
1.3	Haut niveau	85
1.3.1	Styles	85
1.3.2	Flux de données	87
1.3.3	Création d'un visuel	89
1.3.4	Template de page	89
1.3.5	Page contenant plusieurs zones	90
2.	Générer un document PDF depuis une page HTML	92
2.1	Présentation	92
2.2	Installation	92
2.3	Utilisation	93
3.	OpenDocument	93
3.1	Présentation	93
3.2	ezodf2	94
3.2.1	Installation	94
3.2.2	OpenDocument Texte	94
3.2.3	OpenDocument Tableur	94
3.2.4	Aller plus loin	94
3.3	Alternatives	95

4.	Travailler avec des images	95
4.1	Représentation informatique d'une image	95
4.2	Présentation de Pillow	96
4.3	Formats d'images matricielles	97
4.4	Récupérer des informations d'une image	99
4.5	Opérations d'ensemble sur une image	100
4.6	Travailler avec les calques ou les pixels	102
5.	Fichiers de configuration	104
5.1	Format YAML	104
5.2	Format TOML	105

Chapitre 1.4

Qualité

1.	Programmation dirigée par les tests	107
1.1	Tests unitaires	107
1.1.1	Principes	107
1.1.2	Interprétation	108
1.1.3	Couverture	109
1.1.4	Outils	109
1.1.5	Autres outils	111
1.2	Tests de non-régression	111
1.2.1	Actions de développement	111
1.2.2	Gestion de la découverte d'une anomalie par une MOA	112
1.3	Tests fonctionnels	113
1.4	Tests de performance	114
1.5	Tests syntaxiques	116
1.6	Intégration continue	117
2.	Programmation dirigée par la documentation	118
2.1	Documentation interne	118
2.1.1	À destination des développeurs	118
2.1.2	À destination des utilisateurs	119
2.2	Documentation externe	119
2.2.1	Présentation	119
2.2.2	Démarrage rapide	119
2.2.3	Résultat	122

3.	Optimisation	122
3.1	Qualimétrie	122
3.2	Outils de débogage	124
3.3	Outils de profilage	125
3.4	Règles d'optimisation	126
3.4.1	Pourquoi optimiser ?	126
3.4.2	Règles générales	127
3.4.3	Profiler un algorithme	128
3.4.4	Optimiser l'utilisation de la mémoire	137

Partie 2

Programmation, système, réseau et scientifique

Chapitre 2.1

Programmation système

1.	Présentation	139
1.1	Définition	139
1.2	Objectifs du chapitre	140
2.	Appréhender son système d'exploitation	140
2.1	Avertissement	140
2.2	Système d'exploitation	140
2.3	Processus courant	141
2.4	Utilisateurs et groupes	142
2.5	Mots de passe et authentification	144
2.6	Constantes pour le système de fichiers	145
2.7	Gérer les chemins	146
2.8	Utilitaires	146
2.8.1	Comparer deux fichiers	146
2.8.2	Utilitaire de sauvegarde	148
2.8.3	Lire un fichier de configuration	149
2.8.4	Pickle	150
2.9	Compresser et décompresser un fichier	152
2.9.1	Tarfile	152
2.9.2	Gzip	154
2.9.3	Bz2	155
2.9.4	LZMA	155

2.9.5 Zipfile	155
2.9.6 Interface de haut niveau	157
3. Gestion d'un fichier	158
3.1 Changer les droits d'un fichier	158
3.2 Changer de propriétaire ou de groupe	160
3.3 Récupérer des informations sur un fichier	161
3.4 Supprimer un fichier	162
4. Alternatives simples à des commandes bash usuelles	162
4.1 Répertoires	162
4.2 Fichiers	165
4.3 Module de haut niveau	166
4.4 Recherche d'un fichier	168
5. Exécuter des commandes externes	168
5.1 Exécuter et afficher le résultat	168
5.2 Exécuter et récupérer le résultat	169
5.3 Pour Python 3.5 et supérieur	170
6. IHM système	170
6.1 Présentation	170
6.2 Travailler avec des arguments	171
6.3 Fermeture du programme	176
6.4 Entrée standard	177
6.5 Sortie standard	177
6.6 Sortie d'erreur	178
6.7 Taille du terminal	179
7. Curses	180
7.1 Présentation	180
7.2 Gérer le démarrage et l'arrêt	180
7.3 Affichage de texte	181
7.4 Gestion des attributs	182
7.5 Gestion des couleurs	182
7.6 Gestion des événements	182
8. Accès aux périphériques	183
8.1 Introduction	183
8.2 Voir les partitions	183
8.3 Voir les clés USB	184
8.4 Liste des informations disponibles	185

8.5	Déetecter le branchement d'une clé USB	185
8.6	Communiquer via un port série.....	186
8.7	Communiquer via un port USB.....	187

Chapitre 2.2

Programmation réseau

1.	Écrire un serveur et un client	189
1.1	Utilisation d'une socket TCP.....	189
1.2	Utilisation d'une socket UDP	193
1.3	Création d'un serveur TCP	195
1.4	Création d'un serveur UDP	197
1.5	Un peu plus loin sur le sujet	198
2.	Utiliser un protocole standard	199
2.1	Client HTTP.....	199
2.2	Serveur HTTP.....	200
2.3	Proxy	203
2.4	Cookies	204
2.5	FTP et SFTP.....	204
2.6	SSH	206
2.7	POP et POPS	208
2.8	IMAP et IMAPS	209
2.9	SMTP et SMPTS.....	210
2.10	NNTP	215
2.11	IRC.....	216
3.	Wake-on-LAN	219
3.1	Prérequis	219
3.2	Mise en œuvre.....	219

Chapitre 2.3

Programmation web

1.	Services web	221
1.1	REST	221
1.2	SOAP	222
1.3	Pyro	224
2.	Client web	225
3.	Web scrapping	228

Chapitre 2.4

Programmation scientifique

1.	Calcul scientifique	231
1.1	Présentation	231
1.2	Python, une alternative libre et crédible	231
1.3	Vue d'ensemble de quelques bibliothèques	232
2.	Tableaux multidimensionnels	233
2.1	Création	233
2.2	Déterminer la composition d'un tableau	234
2.3	Générateurs de tableaux	235
2.4	Opérations basiques	236
2.5	Opérateur crochet	238
3.	Matrices	240
4.	Génération de graphiques	241
4.1	Syntaxe MATLAB	242
4.2	Syntaxe objet	243
4.3	Mise en forme	243
4.4	Graphiques 3D	248
5.	Introduction à Pandas	251
5.1	Présentation	251
5.2	Séries	251
5.3	Dataframes	255

Partie 3 Programmation concurrente

Chapitre 3.1 Initiation à la programmation concurrente

1.	Notion de performance	263
1.1	Programmation bloquante	263
1.2	Utilisation de ressources CPU	264
1.3	Organisation de l'application	265
2.	Terminologie	266
2.1	Processus	266
2.2	Fil d'exécution	267
2.3	Programmation non bloquante	268
2.4	Notion de GIL	268
3.	Présentation des paradigmes	269
3.1	Programmation asynchrone	269
3.2	Programmation parallèle	269
3.3	Programmation distribuée	270
3.4	Programmation concurrente	271

Chapitre 3.2 Programmation asynchrone : initiation

1.	Utilité de la programmation asynchrone	273
2.	Introduction à l'asynchrone	274
2.1	Notion de coroutine	274
2.2	Exécution d'une coroutine	275
2.3	Précisions sur le mot-clé await	276
2.4	Exécuter plusieurs coroutines en séries	278
2.5	Tâches asynchrones	278
2.6	Exécuter les tâches asynchrones de manière concurrente	280
2.7	Notion d'awaitable	280
2.8	Précisions sur asyncio.sleep	281

3.	Éléments de grammaire	281
3.1	Gestionnaire de contexte asynchrone	281
3.2	Générateurs asynchrones	282
3.3	Compréhensions asynchrones	282
3.4	Itération asynchrone.	282
4.	Notions avancées	282
4.1	Introspection	282
4.2	Gestion du résultat ou de l'exception	286
4.3	Annuler une tâche.	287
4.4	Se prémunir d'une annulation	289
4.5	Timeouts	290
4.6	Gestion globale fine de l'attente	290
4.7	Module contextvars	293
5.	Boucle événementielle asynchrone	296
5.1	Gestion de la boucle	296
5.2	Débogage	297
5.3	Notion de future	299
5.4	Annulation	301
5.5	Gestion des exceptions	302
6.	Utiliser la boucle événementielle	306
6.1	Utilisation des fonctions de retour (callbacks)	306
6.2	Planifier des appels à des fonctions classiques	309
6.3	Utiliser des signaux.	310
6.4	Synchronisation	313
6.5	Communication entre coroutines	317
6.6	Exemple : lecture de l'entrée standard	321
6.7	Programmation parallèle et asynchrone	322
6.8	Exécuter du code bloquant	322
7.	L'asynchrone suivant les versions de Python.	323
7.1	Python 3.7	323
7.2	Python 3.6	324
7.3	Python 3.5	324
7.4	Python 3.4	325
7.5	Python 3.3 et inférieur	325
7.6	Python 2.7	326

8.	Cas concret	326
8.1	Exemple de travail	326
9.	Exemple retravaillé en utilisant des générateurs	329
9.1	Télécharger en asynchrone	330
9.2	Parser en asynchrone	331
9.3	Faire plusieurs traitements en asynchrone	332
9.4	Utiliser un exécuteur	333
9.5	Pour aller plus loin	334

Chapitre 3.3

Programmation asynchrone : avancée

1.	Transports et protocoles	335
1.1	Introduction	335
1.2	Transports	335
1.3	Protocoles	339
1.4	Mise en œuvre	340
1.5	Exemple pour le protocole TCP	342
1.6	Exemple pour le protocole SSL	350
1.7	Exemple pour le protocole UDP	353
1.8	Traiter d'autres types de protocoles réseau	356
1.9	Exemple pour Subprocess	356
2.	Flux	363
2.1	Introduction	363
2.2	Exemple avec le protocole TCP	363

Chapitre 3.4

Programmation asynchrone : alternatives

1.	Gevent	367
1.1	Introduction	367
1.2	DNS	367
1.3	Appels système	368
1.4	Programmation asynchrone	369

2.	Twisted	370
2.1	Introduction	370
2.2	Client/serveur TCP	370
2.3	Serveur/client UDP	372
2.4	AMP	373
2.5	Autres protocoles	376
3.	Trollius	376
4.	HTTP asynchrone avec aiohttp	378
4.1	Côté serveur	378
4.2	Côté client	380

Chapitre 3.5

Programmation parallèle

1.	Utilisation d'un fil d'exécution	381
1.1	Gestion d'un fil d'exécution	381
1.1.1	Présentation	381
1.1.2	Création	381
1.2	Gestion de plusieurs fils d'exécution	384
1.2.1	Lancement et contrôle	384
1.2.2	Opportunité d'utiliser un fil d'exécution	386
1.3	Résolution des problématiques liées	388
1.3.1	Synchronisation	388
1.3.2	Synchronisation conditionnelle	390
1.3.3	Sémaphore	392
2.	Utilisation de processus	394
2.1	Gestion d'un processus	394
2.1.1	Présentation	394
2.1.2	Création	394
2.2	Gestion de plusieurs processus	397
2.2.1	Synchronisation	397
2.2.2	Paralléliser un travail	397
2.3	Résolution des problématiques liées	399
2.3.1	Communication interprocessus	399
2.3.2	Partage de données entre processus	401
2.4	Opportunité d'utiliser les processus	402

2.5	Démon	402
3.	Exécution asynchrone	404
3.1	Introduction	404
3.2	Présentation	405

Chapitre 3.6

Programmation distribuée

1.	Définitions	411
2.	ØMQ	411
2.1	Présentation générale	411
2.2	Pair	413
2.3	Client/serveur	414
2.4	Publish/subscribe	415
2.5	PUSH/PULL	419
2.6	Patron pipeline	420
3.	AMQP avec RabbitMQ	422
3.1	Installation	422
3.2	Configuration	422
3.3	Introduction	423
3.4	Notions importantes	425
3.5	Publish/subscribe	426
3.6	Routage et sujets	427
4.	Kafka	429
4.1	Présentation	429
4.2	Principes généraux	429
4.3	Client Kafka	430
4.4	Faust	430
5.	Celery	431
5.1	Présentation	431
5.2	Dans quel cas utiliser Celery ?	432
5.3	Installation	432
5.4	Configuration	433
5.5	Utilisation	434
5.6	Monitorer	435

6.	Crossbar	436
6.1	Présentation	436
6.2	WebSocket	436
6.3	Publish/subscribe	443
	Index	445