

Table des matières

1	Introduction	1
1.1	Pourquoi Python	1
1.2	À qui s'adresse ce livre	3
1.3	Un peu de méthodologie	5
2	Objets dans Python. Structures et fonctionnalités	9
2.1	Propriétés essentielles. Exemples	9
2.1.1	Variables et dictionnaires	11
2.1.2	Classe comme environnement	13
2.2	Surcharge des opérateurs	13
2.2.1	Arithmétique et héritage	13
2.2.2	Héritage des classes numériques. Surcharge de <code>__new__</code>	16
2.2.3	Surcharge de l'indexation	18
2.2.4	Opérateurs formés par le programmeur?	19
2.2.5	Un exemple plus long : différentiation automatique	21
2.3	Digression : héritage multiple	24
2.3.1	L'ordre de résolution des méthodes (MRO)	25
2.4	Fonctions et objets	28
2.4.1	Fonctions et persistance	28
2.4.2	Fermetures	30
2.4.3	Modification d'un programme pendant son exécution	31
2.4.4	Décorations des fonctions	32
2.4.5	Mémoïsation	33
2.4.6	Exemple. Distance de Levenshtein-Damerau entre deux textes	34
2.4.7	Outils de mémoïsation existants	36
2.5	Exceptions comme objets	38
2.5.1	Traitement et héritage des Exceptions	39
2.5.2	Rapport <i>post mortem</i> (traceback)	41
2.5.3	Exceptions comme structures de contrôle	42
2.6	Instruction <i>with</i> et <i>Context Managers</i>	45
2.7	Slots	47
2.8	Exercices	49

3	Conteneurs, séquences et itérateurs	57
3.1	Structures standard, codage pas toujours	57
3.1.1	Itérateurs implicits. Structures « virtuelles » prédéfinies	59
3.1.2	Exemple. Séries de Taylor	61
3.2	Librairie <i>Collections</i>	65
3.2.1	Compteurs	65
3.2.2	Tuples nommés	66
3.2.3	Deque	68
3.3	Itérateurs, leurs combinaison et contrôle	72
3.3.1	Suite logistique; un exemple animé	73
3.3.2	Exemples de générateurs	77
3.3.3	Itertools	79
3.3.4	Générateurs - managers et autres utilités : librairie Contextlib	82
3.3.5	Pièges d'évaluation paresseuse	83
3.4	Générateurs et non-déterminisme logique	84
3.4.1	<i>Backtracking</i>	85
3.4.2	Itérateurs et génération combinatoire	86
3.5	Générateurs étendus : coroutines	90
3.5.1	L'expression <code>yield</code> et la méthode <code>send</code>	93
3.5.2	Générateurs-coroutines et un simple modèle de concurrence	95
3.6	Itérations, coroutines et non-déterminisme	97
3.6.1	Coroutines et programmation par flots de données	99
3.6.2	Blocs élémentaires et composites; exemples de circuits	106
3.6.3	Exemple évolué. Un oscillateur atténué décomposé.	111
3.7	Exercices	113
4	Librairie NumPy. Usage avancé des tableaux	123
4.1	Tableaux comme objets mathématiques	123
4.1.1	Solution de l'équation de Laplace	125
4.2	Création des tableaux	128
4.2.1	Sous-classes de <code>ndarray</code>	132
4.2.2	Sous-classes standard	134
4.3	Fonctions universelles	136
4.3.1	Vectorisation	138
4.3.2	Sélection conditionnelle	139
4.4	Indexation avancée	141
4.4.1	Indices booléens composites	142
4.4.2	Indices entiers composites	142
4.4.3	Indices tabulaires et géométrie	143
4.4.4	Exemple. Arbre de Feigenbaum, et fonctions de histogrammation	146
4.5	Traitement d'images et visualisation avec Python	148
4.5.1	Génération des motifs géométriques. Couleurs	149
4.5.2	Filtrage des contours et équilibrage de l'histogramme	154
4.5.3	Distorsion des images	157
4.5.4	Visualisation des champs vectoriels, technique LIC	161
4.5.5	Projections cartographiques	164

4.6	Simulation en physique statistique : modèle d'Ising 2D	170
4.6.1	Implémentation	172
4.7	Exercices	174
5	Quelques objets « sous le capot »	183
5.1	Ajout dynamique des méthodes	183
5.2	Gestion avancée des attributs	187
5.2.1	Décodage des méthodes spéciales	190
5.2.2	<i>Properties</i>	192
5.2.3	Exemple de <i>properties</i> : températures	194
5.2.4	<i>Properties</i> spécifiées par décorateurs	195
5.3	Descripteurs	197
5.3.1	Propriétés essentielles des descripteurs	197
5.3.2	Implémentation des méthodes comme descripteurs	200
5.3.3	Implémentation de <i>properties</i> et d'autres descripteurs de données	202
5.3.4	La construction <i>super(...)</i> et les descripteurs	203
5.3.5	Héritage multiple et <i>super(...)</i>	204
5.3.6	Une catégorie-exemple de descripteurs : les <i>proxies</i>	206
5.4	Décorateurs avancés	208
5.4.1	Paramétrisation des décorateurs	208
5.4.2	Classes décoratrices, et classes décorées	209
5.5	Métaclasses	211
5.5.1	A-t-on besoin de ce concept ?	212
5.5.2	Usage des métaclasses	213
5.5.3	Création dynamique des classes	215
5.5.4	La « préparation » de la classe	216
5.6	Exercices	217
6	Programmes concurrents et asynchrones	221
6.1	Parallélisme et concurrence	221
6.1.1	Digression méthodologique : « lois » de Amdahl et de Gustafson	226
6.1.2	Threads et processus : quelle différence ?	228
6.2	Multi-traitement et processus	229
6.2.1	Module <i>multiprocessing</i> et son usage de base	229
6.2.2	Partitionnement des données	233
6.2.3	Échange des données et communication	235
6.2.4	Partage des données et verrous	238
6.2.5	Autre exemple de communication : tri parallèle	239
6.2.6	MPI	243
6.3	Threads	246
6.3.1	Exemple, une variante du tri	250
6.3.2	Timers et événements	252
6.4	Asyncio	259
6.4.1	Boucle événementielle	261
6.4.2	Constructions <i>async</i> et <i>await</i> , et les coroutines	265
6.4.3	Interaction et communication	270

6.4.4	Flux	271
6.5	Exercices	274
7	Accélération des programmes	281
7.1	Quelques avertissements	281
7.2	Numba	283
7.2.1	Configuration et paramétrage de Numba	285
7.2.2	Compilation anticipée	286
7.2.3	Optimisation des classes	287
7.2.4	Vectorisation et fonctions universelles	288
7.3	Cython	290
7.3.1	Directive <i>cdef</i> et types C dans un programme Python	292
7.3.2	Paramétrage avancé de Cython	294
7.4	PyPy et autres outils	295
7.5	GPU et Python	298
7.5.1	Logiciel de support	299
7.5.2	CUDA, la première leçon	300
7.5.3	Interface PyCuda	304
7.5.4	Encore quelques outils d'interfaçage	308
7.6	Parallélisme et Numba	311
7.6.1	Quelques éléments d'interfaçage CUDA	311
7.7	Exercices	314
8	Structures de données sérialisées	321
8.1	Pickle	321
8.1.1	La « machine » Pickle	324
8.1.2	Pickle : outils de support	326
8.1.3	<i>Shelve</i>	327
8.1.4	<i>Dill</i>	327
8.2	JSON	329
8.3	Autres outils	330
8.4	Exercices	331
A	Solutions des exercices choisis	335
	Liste des exercices	386
	Index	387