

Table des matières

Prologue	5
1 Paradigme orienté objet	33
1.1 Introduction	33
1.2 Objets et méthodes	34
1.3 Notion de classe	37
1.4 Héritage, surcharge et polymorphisme	39
1.4.1 Héritage	39
1.4.2 Surcharge	40
1.4.3 Surcharge des opérateurs usuels	43
1.4.4 Polymorphisme	44
1.5 Agrégation et composition	44
1.6 Méthodes d'instances, méthodes de classes et méthodes statiques	46
1.7 Langage UML	47
2 Mots réservés et fonctions internes orientés objet	51
2.1 Mots réservés orientés objet	51
2.2 Données internes (module <code>builtins</code>)	52
2.3 Données internes orientées objet	54
3 Types offerts par Python	55
3.1 Introduction	55
3.2 Catégories I et III	56
3.3 Catégorie II : les types d'exceptions	57
3.4 Catégories IV et V	57
3.5 Interfaces offertes par Python	58
3.6 Hiérarchie officielle des types internes	59
3.7 Introspection	60
4 Objets	63
4.1 Introduction	63
4.2 Valeur	64
4.2.1 Définition	64
4.2.2 Valeur des objets de type interne	65
4.2.2.1 Nombres	65
4.2.2.2 Chaînes	65
4.2.2.3 Booléens	66

4.2.2.4	Conteneurs	66
4.2.2.5	Objets de type <code>object</code>	66
4.2.3	Valeur d'une instance de classe	67
4.2.4	Mécanisme derrière « <code>==</code> »	68
4.3	Identité	70
4.3.1	Définition	70
4.3.2	Bizarrie en coulisse	70
4.3.3	Cas des objets mutables	72
5	Classes	75
5.1	Classes chez Python	75
5.2	Créer une classe	75
5.3	Variables d'une classe (attributs)	76
5.3.1	Définition	76
5.3.2	Attributs-données et attributs-méthodes	78
5.3.3	Identificateurs recommandés	79
5.3.4	Dictionnaire <code>__dict__</code>	80
5.4	Instancier une classe	81
5.5	Méthodes	82
5.6	Filiation et héritage	83
5.7	Attributs <code>name</code> et <code>qualname</code>	86
5.8	Fonction <code>type</code>	88
5.9	Docstring	89
5.10	Attribut <code>slots</code>	90
5.11	Décorer une classe	93
5.12	Introspection	93
5.12.1	Afficher les parents d'une classe	93
5.12.2	Retour au dictionnaire	94
5.12.3	Une bizarrerie avec les attributs privés	95
6	Méthodes	97
6.1	Méthodes d'instances	97
6.2	Méthodes de classes	98
6.3	Méthodes statiques	99
6.4	Cas d'utilisation	100
6.4.1	Généralités	100
6.4.2	Exemples internes	101
6.4.2.1	<code>float.fromhex</code>	101
6.4.2.2	<code>int.from_bytes</code>	102
6.4.2.3	<code>str.maketrans</code>	102
6.4.2.4	<code>dict.fromkeys</code>	103
6.4.2.5	<code>bytearray.fromhex</code>	103
6.4.2.6	<code>bytes.fromhex</code>	104
6.4.3	Cas des fonctions d'un module	104
6.4.4	Exemples avec des matrices	105
6.4.5	Exemples avec des dates	108

6.5	Introspection	110
6.5.1	Explorer <i>method</i>	110
6.5.2	Explorer <code>classmethod</code>	111
6.5.3	Explorer <code>staticmethod</code>	113
6.6	Bilan	115
7	Type <code>object</code>	117
7.1	Introduction	117
7.2	Exploration rapide	118
7.3	Attributs basiques	119
7.3.1	Attribut <code>__name__</code>	119
7.3.2	Attribut <code>__qualname__</code>	119
7.3.3	Attribut <code>__bases__</code>	119
7.3.4	Attribut <code>__dict__</code>	120
7.4	Méthodes pour instancier	121
7.4.1	Méthode <code>__new__</code>	121
7.4.2	Méthode <code>__init__</code>	123
7.5	Méthodes pour afficher	125
7.5.1	Méthode <code>__repr__</code>	125
7.5.2	Méthode <code>__str__</code>	125
7.5.3	Méthode <code>__format__</code>	125
7.6	Méthodes pour comparer	126
7.7	Attributs pour introspection	127
7.7.1	Attribut <code>__class__</code>	127
7.7.2	Attribut <code>__doc__</code>	128
7.7.3	Méthode <code>__dir__</code>	128
7.7.4	Méthode <code>__sizeof__</code>	129
7.8	Méthodes accesseur, mutateur et effaceur	129
7.9	Attributs liés à l'héritage	130
7.9.1	Méthode <code>mro</code>	130
7.9.2	Attribut <code>__mro__</code>	130
7.9.3	Méthode <code>__subclasses__</code>	131
7.10	Méthode pour hacher	131
7.11	Curiosités	132
7.11.1	Mise en abyme	132
7.11.2	Bound ou unbound ?	133
7.11.3	Retour à l'espace des noms de <code>object</code>	135
7.11.4	Retour à <code>__subclasses__</code>	136
7.11.5	Retour à <code>mro</code>	137
7.11.6	Dernier coup d'œil	138
8	Type <i>function</i>	139
8.1	Paramètres	139
8.1.1	Introduction	139
8.1.2	Paramètres de position sans valeur par défaut	140
8.1.3	Paramètres nommés sans valeur par défaut	140

8.1.4	Paramètres positionnels ou nommés sans valeur par défaut	140
8.1.5	Paramètres avec valeur par défaut	140
8.1.6	Paramètre <i>var-positional</i>	141
8.1.7	Paramètre <i>var-keyword</i>	141
8.1.8	Signatures	142
8.1.9	Valeur par défaut et effet de bord	145
8.2	Arguments	146
8.3	Exemple : émulation de <code>max</code>	148
8.4	Expérience	150
8.5	Passage par valeur, variable ou référence ?	150
8.6	Attributs spécifiques	151
8.6.1	<code>__name__</code>	152
8.6.2	<code>__qualname__</code>	152
8.6.3	<code>__doc__</code>	152
8.6.4	<code>__annotations__</code>	153
8.6.5	<code>__module__</code>	153
8.6.6	<code>__globals__</code>	153
8.6.7	<code>__defaults__</code>	153
8.6.8	<code>__kwdefaults__</code>	154
8.6.9	<code>__closure__</code>	154
8.6.10	<code>__code__</code>	154
8.7	Instanciation	154
8.8	Ajouter des attributs à une fonction	156
8.9	Fermeture d'une fonction	158
8.9.1	Notion de variable libre	158
8.9.2	Contenu de la fermeture	159
8.9.3	Calcul de la fermeture	160
8.9.4	À quoi sert la fermeture ?	162
8.9.5	Quelques exemples amusants	163
8.9.6	Retour à une ancienne énigme	165
8.9.7	Émuler une classe avec une fonction	166
8.9.8	Conclusion	168
8.10	Annotations	168
8.10.1	Introduction	168
8.10.2	Module <code>typing</code>	169
8.10.3	Annotations contraignantes	171
8.10.3.1	Fonction <code>checktypes</code>	171
8.10.3.2	Fonction <code>ensure_annotations</code> du module <code>ensure</code>	172
8.10.3.3	Fonction <code>validate</code>	172
9	Constructeurs et initialiseurs	175
9.1	Préliminaire	175
9.2	Mécanisme d'instanciation	175
9.3	Constructeurs	178
9.4	Pourquoi les constructeurs sont des méthodes statiques ?	180
9.5	Initialiseurs	183

9.5.1	Introduction	183
9.5.2	Appel de l'initialiseur	184
9.5.3	Utilisation courante	184
9.5.4	Exemple	185
9.5.5	Dérivation et Coopération	185
9.5.6	Confusion fréquente	188
9.6	Manipulations	188
9.7	Instanciation des types internes	193
9.7.1	Cas des objets mutables	193
9.7.2	Cas des objets non mutables	194
9.7.3	Cas des classes	195
9.8	Cas courants où l'on surcharge le constructeur	196
9.8.1	Dérivation d'un type interne d'objets non mutables	196
9.8.2	Dérivation de la classe <code>type</code>	198
9.9	Application à des <i>design patterns</i>	200
9.9.1	Singleton	200
9.9.2	Borg	203
9.9.3	Compteur d'instances	206
10	Destructeurs et ramasse-miettes	209
10.1	Mot réservé <code>del</code>	209
10.2	Destructeur (<i>finalizer</i>)	210
10.3	La commande <code>del</code> est-elle utile ?	215
10.4	Exemple : retour au design pattern compteur	216
11	Espaces de noms	219
11.1	Introduction	219
11.2	Opérations sur les variables	220
11.3	Espace des noms internes	221
11.4	Espace des noms globaux	221
11.5	Espace de noms locaux	222
11.5.1	Introduction	222
11.5.2	Variable globale et variable locale pointant vers le même objet	223
11.5.3	Paramètres avec valeur par défaut	224
11.6	Résolution d'un nom dans le corps d'une fonction	225
11.7	Espace des noms associé à une classe	228
11.8	Espace des noms associé à une instance	229
11.9	Vie et mort des variables d'une classe et d'une instance	231
11.10	Résolution d'un nom dans le corps d'une classe	233
11.10.1	Énoncé des règles	233
11.10.2	Exemples	235
11.10.3	Une fonction à l'intérieur d'une classe	236
11.10.4	Curiosités	238
11.11	Outils pour l'introspection	240
11.11.1	Fonction <code>dir</code>	240
11.11.2	Attribut <code>__dict__</code>	241

11.11.3	Fonction <code>globals</code>	241
11.11.4	Fonction <code>locals</code>	241
11.11.5	Fonction <code>vars</code>	242
11.12	Notion de portée (<i>scope</i>)	242
11.13	À propos du tiret bas (<i>underscore</i>)	243
12	Méthodes spéciales pour l’affichage	245
12.1	Méthode <i>repr</i>	245
12.1.1	Introduction	245
12.1.2	Penser à l’héritage	246
12.1.3	Représentation basée sur l’identité	246
12.1.4	Représentation des objets de type interne	247
12.1.4.1	Cas de <code>object</code>	247
12.1.4.2	Cas de <code>classmethod</code> , <code>enumerate</code> , <code>filter</code> , <code>map</code> , <code>property</code> , <code>reversed</code> , <code>staticmethod</code> et <code>zip</code>	248
12.1.4.3	Cas de <i>function</i>	248
12.1.4.4	Cas de <code>memoryview</code>	249
12.1.4.5	Cas de <code>super</code>	249
12.1.4.6	Cas de <code>type</code>	250
12.1.4.7	Cas de <i>method</i>	251
12.1.5	Représentation par défaut d’une instance de classe	252
12.1.6	Une surcharge pratique pour débbugger	253
12.2	Méthode <i>str</i>	254
12.2.1	Introduction	254
12.2.2	Lien entre <i>str</i> et <i>repr</i>	255
12.2.3	Cas des objets conteneurs de type interne	257
12.3	Méthode <i>format</i>	259
12.3.1	Introduction	259
12.3.2	Mécanisme derrière la fonction <code>format</code>	260
12.3.3	Mécanisme derrière <code>str.format</code>	260
12.3.4	Surcharger <i>format</i>	261
12.4	Exemple amusant	262
13	<code>None</code>, <code>Ellipsis</code> et <code>NotImplemented</code>	265
13.1	Objet <code>None</code>	265
13.1.1	Introduction	265
13.1.2	Affichage	266
13.1.3	Attributs	266
13.1.4	Retour par défaut	267
13.1.5	Utilisé comme valeur par défaut	267
13.1.6	Utilisé pour désactiver une méthode héritée	268
13.2	Objet <code>Ellipsis</code>	268
13.2.1	Introduction	268
13.2.2	Affichage	269
13.2.3	Utilisation	269
13.2.4	Exemple	270

13.3	Objet <code>NotImplemented</code>	272
13.3.1	Introduction	272
13.3.2	Usage général	272
13.3.3	Usage dans les opérateurs d'arité 2	273
13.3.4	Ne pas confondre avec <code>NotImplementedError</code>	275
14	Opérateurs d'arité 2	277
14.1	Notion d'opérateur symétrique	277
14.2	Méthodes spéciales d'opérateurs	278
14.3	Symétrie d'une méthode d'opérateur	278
14.4	Mécanisme derrière les opérateurs arithmétiques	279
14.5	Mécanisme derrière les opérateurs de comparaison	280
14.5.1	Égalité et non égalité	281
14.5.2	Ordre	284
14.6	Opérateurs in situ (<i>in place</i>)	286
14.7	Priorités	288
14.8	Expériences	288
14.8.1	Expérience 1	288
14.8.2	Expérience 2	289
14.8.3	Expérience 3	289
14.8.4	Expérience 4	290
14.8.5	Expérience 5	291
14.8.6	Expérience 6	291
15	Objets subscriptables	293
15.1	Définition	293
15.2	Méthode <i>getitem</i>	294
15.3	Méthode <i>missing</i>	295
15.4	Méthode <i>setitem</i>	297
15.5	Méthode <i>delitem</i>	297
15.6	Opération « <code>for k in x</code> »	298
15.7	Opération « <code>k in a</code> »	299
15.8	Méthode spéciale <i>index</i>	301
15.9	Exemple : matrices	302
16	Slicing	305
16.1	Rappels	305
16.1.1	Rudiments	305
16.1.2	Syntaxes	305
16.1.3	Robustesse du slicing	306
16.2	Mécanisme derrière le slicing	307
16.3	Objets de type <code>slice</code>	309
16.4	Reconditionnement	310
16.5	Méthode <code>slice.indices</code>	311
16.6	Annexe : <i>index</i>	312

17 Séquences	315
17.1 Définition	315
17.2 Les séquences sont itérables	316
17.3 Les séquences supportent l'opération « <code>x in a</code> »	316
17.4 Slicing	317
17.4.1 Rappels	317
17.4.2 Exemple consistant	318
17.4.3 Une autre version de cet exemple	322
17.5 Concaténation et répétition	323
17.6 Méthodes spéciales	328
17.6.1 Tour d'horizon	328
17.6.2 Méthode <code>len</code>	328
17.6.3 Lien entre <code>len</code> et <code>reversed</code>	329
17.6.4 Méthode <code>reversed</code>	331
17.7 Méthodes typiques	332
17.8 Exemple complet : séquences géométriques	335
17.8.1 Introduction	335
17.8.2 Slicing	336
17.8.3 Changer la longueur et la raison	338
17.8.4 Surcharge de <code>setitem</code>	339
17.8.5 Surcharge de <code>delitem</code>	340
17.8.6 Surcharge de méthodes typiques	342
17.8.6.1 Surcharge de <code>append</code>	342
17.8.6.2 Surcharge de <code>extend</code>	343
17.8.6.3 Surcharge de <code>copy</code>	344
17.8.6.4 Surcharge de <code>index</code>	344
17.8.6.5 Surcharge de <code>reverse</code>	345
17.8.6.6 Surcharge de <code>pop</code>	345
17.8.7 Surcharge de <code>add</code> et <code>iadd</code>	346
17.9 Exemple complet par dérivation	347
17.9.1 Introduction	347
17.9.2 Slicing	349
17.9.3 Changer la longueur et la raison	349
17.9.4 Surcharge de <code>setitem</code>	350
17.9.5 Surcharge de <code>delitem</code>	350
17.9.6 Surcharge de méthodes typiques	352
17.9.6.1 Méthodes à éliminer	353
17.9.6.2 Surcharge de <code>append</code>	353
17.9.6.3 Surcharge de <code>extend</code>	353
17.9.6.4 Surcharge de <code>copy</code>	354
17.9.6.5 Pas besoin de surcharger <code>index</code>	354
17.9.6.6 Pas besoin de surcharger <code>reverse</code>	355
17.9.6.7 Surcharge de <code>pop</code>	355
17.9.7 Surcharge de <code>add</code> et <code>iadd</code>	356
18 Itérables	359

18.1	Définition	359
18.2	Mécanisme derrière « <code>for k in x</code> »	359
18.3	Cas particulier d'un objet muni de <i>getitem</i>	360
18.4	Fonction <code>iter</code>	363
18.5	Fonction <code>next</code>	364
18.6	Surcharger <i>iter</i> et <i>next</i> : usage courant	365
18.7	Itérateur versus pseudo-itérateur	367
18.8	Exemples complets	369
18.8.1	Suites de Syracuse	369
18.8.2	Itérer sur un entier	370
18.9	Émulation	371
18.10	Décompression	373
18.11	Opérateur <code>splat</code>	373
18.12	Introspection	374
18.13	Compléments sur <code>iter</code> et <code>next</code>	375
18.13.1	Appel de <code>iter</code> avec une fonction et une sentinelle	375
18.13.2	Appel de <code>next</code> avec valeur <code>default</code>	377
18.14	Expériences	377
18.15	En résumé	378
19	Conteneurs	381
19.1	Définition	381
19.2	Mécanisme derrière « <code>k in x</code> »	381
19.3	Conteneurs de type interne	382
19.4	Surcharge de <i>contains</i> : exemples simples	383
19.4.1	Conteneur universel	383
19.4.2	Conteneur très exotique	383
19.4.3	Idéaux de \mathbb{Z}	384
19.5	Cas particulier des itérables infinis	385
19.6	Émulation	386
19.7	Exemples	387
19.7.1	Retour à Syracuse	387
19.7.2	Divisibilité chez les entiers	388
19.7.3	Intervalles de \mathbb{R}	390
20	Tables de hachage	393
20.1	Introduction	393
20.1.1	Définition	393
20.1.2	Implémentation naïve des ensembles finis	394
20.1.3	Ensemble naïf versus <code>set</code>	395
20.1.4	Implémentation naïve des dictionnaires	398
20.2	Hachage-adressage quand l'univers est fini	399
20.2.1	Fonctionnement	399
20.2.2	Émulation	401
20.3	Hachage-adressage quand l'univers est infini	403
20.3.1	Principe de base	403

20.3.2	Traitement des collisions	404
20.3.3	Exemple à la main	405
20.3.4	Émulation	406
20.3.5	Défauts	410
20.3.5.1	Nécessité d'une bonne fonction de hachage	410
20.3.5.2	Modélisation	410
20.3.5.3	Paradoxe des anniversaires	411
20.3.5.4	Clusters	415
20.3.6	Variantes	415
20.4	Tables de hachage de type interne	417
20.4.1	Taille dynamique	417
20.4.2	Collections d'objets ou de valeurs ?	418
20.4.3	Objets hachables	420
20.4.3.1	Généralités	420
20.4.3.2	Récurtivité	421
20.4.3.3	Introspection	421
20.4.4	Propriétés requises chez <i>hash</i>	423
20.4.4.1	Être entier	423
20.4.4.2	Être constant dans le temps	423
20.4.4.3	Être invariant modulo <i>eq</i>	424
20.4.4.4	Expérience de pensée : un hachage non invariant modulo <i>eq</i>	424
20.4.4.5	Suivre une distribution uniforme	427
20.4.5	Problème avec les mutables de type interne	427
20.4.6	Classe-identités versus classe-valeurs	429
20.4.7	Problème avec les classes-identités	430
20.4.8	Problème avec les classes-valeurs	432
20.4.9	Les 5 commandements	436
20.5	Expériences	436
20.5.1	Expérience 1	436
20.5.2	Expérience 2	437
20.5.3	Expérience 3	437
20.5.4	Expérience 4	438
20.5.5	Expérience 5	439
20.5.6	Expérience 6	439
20.5.7	Expérience 7	440
20.6	Fonction interne <i>hash</i>	441
20.6.1	Évolution de <i>hash</i> au fil du temps	441
20.6.2	Python interdit qu'un hachage donne -1	443
20.6.3	Définition de <i>hash</i> (<i>n</i>) pour <i>n</i> entier	444
20.6.4	Émulation du <i>hash</i> de <i>int</i>	445
20.6.5	Résidus et sommes digitales	445
20.6.6	Émulation du <i>hash</i> de <i>int</i> avec des sommes digitales	446
20.6.7	Compléments sur les opérateurs logiques (base 2)	447
20.6.7.1	Répunit	447
20.6.7.2	Opération AND	447
20.6.7.3	Opération OR	448

20.6.7.4	Opération XOR	448
20.6.7.5	Décalage logique vers la gauche (<i>left shift</i>)	448
20.6.7.6	Décalage logique vers la droite (<i>right shift</i>)	449
20.6.8	Émulation du <i>hash</i> de <code>int</code> avec des opérateurs logiques	449
20.6.9	Une remarque sur le nombre M_{61}	450
20.6.10	Émulation de la fonction interne <code>hash</code>	451
20.6.11	Émulation du <i>hash</i> de <code>str</code>	452
20.6.12	Émulation du <i>hash</i> de <code>tuple</code>	454
20.6.13	Émulation du <i>hash</i> de <code>Fraction</code>	454
20.6.14	Émulation du <i>hash</i> de <code>float</code>	455
20.6.15	Émulation du <i>hash</i> de <code>complexe</code>	456
20.6.16	Bibliothèque de hachage <code>hashlib</code>	456
20.6.17	Expérience	457
21	Ensembles	459
21.1	Introduction	459
21.2	Méthodes typiques	459
21.2.1	Méthodes de base	459
21.2.2	Opérations	460
21.2.3	Tests	461
21.2.4	Mises à jour	461
21.2.5	Autre	462
21.3	Méthodes spéciales	463
21.3.1	Méthodes spéciales de base	463
21.3.2	Opérations	464
21.3.3	Opérations in situ	465
21.3.4	Comparaisons	466
21.4	Exemple complet	467
21.4.1	Méthodes de base	467
21.4.2	Méthodes spéciales de base	468
21.4.3	Opérations typiques	469
21.4.4	Tests typiques	470
21.4.5	Mises à jour	470
21.4.6	Méthodes <code>pop</code> , <code>clear</code> et <code>copy</code>	471
21.4.7	Opérateurs arithmétiques	472
21.4.8	Opérateurs in situ	472
21.4.9	Comparaisons	472
21.5	Exemple avec des ensembles infinis	473
22	Tables de correspondance	477
22.1	Introduction	477
22.1.1	Première présentation	477
22.1.2	Deuxième présentation	478
22.1.3	Troisième présentation	479
22.1.4	Quatrième présentation	479

22.2	Méthodes courantes chez une table de correspondance	479
22.2.1	Introspection dans <code>dict</code>	480
22.2.2	Méthodes de base	480
22.2.3	Mutations typiques	481
22.2.3.1	Méthode <code>update</code>	481
22.2.3.2	Méthode <code>clear</code>	481
22.2.3.3	Méthodes <code>pop</code> et <code>popitem</code>	482
22.2.4	Longueur et itération	482
22.2.5	Méthodes <code>items</code> , <code>keys</code> et <code>values</code>	482
22.2.6	Méthodes <code>fromkeys</code>	483
22.2.7	Méthode <code>copy</code>	483
22.3	Exemple complet	484
23	Objets appelables	491
23.1	Définition	491
23.2	Mécanisme derrière un appel	491
23.3	Cas typiques	493
23.3.1	Fonctions et méthodes	493
23.3.2	Décorateurs gestionnaires de contexte	493
23.3.3	Classes (émulation de <code>call</code> dans <code>type</code>)	493
23.4	Émuler les fonctions numériques	494
23.4.1	Fonctions réelles d'une variable réelle	494
23.4.2	Fonctions réelles de plusieurs variables réelles	497
23.5	Émuler les variables aléatoires	499
23.5.1	Support fini	499
23.5.2	Support dénombrable	502
23.5.3	Support compact	503
23.5.4	Support non compact	505
23.5.4.1	Inversion de la fonction de répartition	505
23.5.4.2	Cas de la loi normale	506
24	Objets algébriques	509
24.1	Introduction	509
24.2	Type <code>bool</code>	509
24.3	Types <code>int</code> (et <code>bool</code>)	511
24.3.1	Attributs typiques	511
24.3.2	Méthodes typiques	511
24.3.3	Méthode spéciale <code>index</code>	512
24.3.4	Opérateurs typiques d'arité 2	512
24.3.5	Opérateurs typiques d'arité 1	513
24.3.6	Opérateurs bit à bit d'arité 2	513
24.3.7	Opérateur bit à bit d'arité 1	513
24.3.8	Opérateurs de comparaison	513
24.3.9	Méthodes spéciales pour fonctions internes	513
24.3.10	Méthodes spéciales pour fonctions du module <code>math</code>	514
24.3.11	Implémentation	514

24.4	Type <code>float</code>	514
24.4.1	Attributs typiques	514
24.4.2	Méthodes typiques	514
24.4.3	Opérateurs typiques d'arité 2	514
24.4.4	Opérateurs typiques d'arité 1	515
24.4.5	Opérateurs de comparaison	515
24.4.6	Méthodes spéciales pour fonctions internes	515
24.4.7	Méthode spéciale pour la fonction <code>trunc</code> du module <code>math</code>	515
24.4.8	Instances particulières	515
24.4.9	Implémentation	516
24.5	Type <code>complex</code>	516
24.5.1	Instanciation	516
24.5.2	Méthode typique	517
24.5.3	Opérateurs typiques d'arité 2	517
24.5.4	Opérateurs typiques d'arité 1	517
24.5.5	Opérateurs de comparaison	517
24.5.6	Méthodes spéciales pour fonctions internes	517
24.6	Type <code>decimal.Decimal</code>	517
24.6.1	Intérêt du type <code>Decimal</code>	517
24.6.2	Notion de contexte	519
24.6.3	Signaux, <code>flags</code> et <code>traps</code>	520
24.6.4	Méthodes d'un objet de type <code>Context</code>	521
24.6.5	Méthodes d'un objet de type <code>Decimal</code>	521
24.6.6	Instances particulières	521
24.7	Type <code>fractions.Fraction</code>	522
24.7.1	Introduction	522
24.7.2	Attributs et méthodes	522
24.8	Classe abstraite <code>Number</code>	523
24.9	Priorités entre opérateurs	524
24.10	Classe algébrique en général	524
24.11	Dériver un type numérique	526
24.12	Exemple : quaternions	527
24.13	Exemple : permutations	533
24.14	Exemple : vecteurs	537
24.15	Exemple : fonctions et variables aléatoires	541
24.15.1	Retour aux fonctions réelles d'une variable réelle	541
24.15.2	Retour aux variables aléatoires réelles	541
25	Gestionnaires de contexte	545
25.1	Introduction	545
25.2	Signature de <code>enter</code> et <code>exit</code>	546
25.3	Gestionnaire customisé	548
25.4	Gestionnaire-décorateur	549
26	Héritage	553
26.1	Taxonomie et notion d'héritage	553

26.2	Ordre de résolution d'un attribut (MRO)	556
26.2.1	Linéarisation	556
26.2.2	Position du problème	557
26.2.3	Ordre local	558
26.2.4	Depth first	558
26.2.5	From left to right, depth first : LRDF	559
26.2.6	Monotonie	560
26.2.7	Algorithme LRDF amélioré : KEEP LAST LRDF	561
26.2.8	Exemple fâcheux	562
26.2.9	Solution adoptée par Python 3	563
26.3	Algorithme C3	564
26.4	Superclasse	569
26.4.1	Ne pas confondre les superclasses avec la superclasse	569
26.4.2	Fonction <code>super</code>	570
26.5	Un peu d'histoire	571
26.5.1	Python 2.1	571
26.5.2	Python 2.2 : arrivée des classes modernes	572
26.5.3	Python 2.3 : adoption de l'algorithme C3	575
27	Fonction <code>super</code>	577
27.1	Notion de superclasse	577
27.2	Fonction <code>super</code>	578
27.3	Utilisation élémentaire	580
27.4	Utilisation un peu moins élémentaire	582
27.5	Classes coopératives	583
27.5.1	Introduction	583
27.5.2	Signature robuste	586
27.5.3	Stopper la chaîne des délégations	587
27.6	Adaptateur	588
27.7	Exemple complet	589
27.8	Introspection	594
27.9	Émulation	598
28	Projet : dans la savane	601
28.1	Objectif	601
28.2	Population globale	601
28.3	Classe origine	602
28.4	Êtres vivants, faune et flore	603
28.5	Programme principal	606
28.6	Enrichissements	607
29	Méthodes spéciales derrière les accesseurs	609
29.1	Introduction	609
29.2	Fonctions <code>getattr</code> , <code>setattr</code> , <code>delattr</code> et <code>hasattr</code>	610
29.3	Méthode <code>getattrattribute</code>	611

29.3.1	Introduction	611
29.3.2	Exemple de surcharge	612
29.3.3	Attention aux appels récursifs	613
29.3.4	Accesseur de <code>object</code>	613
29.3.5	Émulation de l'accesseur de <code>object</code>	617
29.4	Méthode <code>getattr</code>	618
29.5	Méthode <code>setattr</code>	619
29.5.1	Fonctionnement	619
29.5.2	Émulation du mutateur de <code>object</code>	621
29.6	Méthode <code>delattr</code>	623
29.6.1	Fonctionnement	623
29.6.2	Émulation de l'effaceur de <code>object</code>	624
29.7	Surcharger <code>dir</code>	625
30	Descripteurs	627
30.1	Introduction	627
30.1.1	Définition	627
30.1.2	Méthode <code>get</code>	628
30.1.3	Méthode <code>set</code>	630
30.1.4	Méthode <code>delete</code>	631
30.1.5	Méthode <code>set name</code>	631
30.1.6	Protocole descripteur	632
30.2	Cas des fonctions et méthodes	632
30.2.1	Fonctions	632
30.2.2	Méthodes d'instances	634
30.2.3	Méthodes statiques	636
30.2.4	Méthodes de classes	637
30.3	Cas d'un <i>member descriptor</i> (<i>slots</i>)	639
30.4	Cas d'un <i>getset descriptor</i>	640
30.5	Utilisations élémentaires	643
30.5.1	Première expérience	643
30.5.2	Deuxième expérience	644
30.5.3	Troisième expérience	645
30.6	Utilisation de <code>set name</code>	646
30.7	Validateurs	647
30.8	Propriétés	648
30.9	Émulation totale	649
30.10	Retour au problème de <i>bound versus unbound</i>	658
31	Type <code>property</code>	661
31.1	Introduction	661
31.2	Application simpliste	663
31.3	Méthodes <code>getter</code> , <code>setter</code> et <code>deleter</code>	663
31.4	Émulation	665
32	Métaclasses	667

32.1	Définitions	667
32.2	Dériver <code>type</code>	668
32.2.1	Usage courant : surcharger <code>new</code>	668
32.2.2	Alternative : surcharger <code>__init__</code>	669
32.3	Instancier une métaclasse	669
32.3.1	Appel direct	670
32.3.2	Mot-clé <code>class</code> avec paramètre <code>metaclass</code>	670
32.4	À propos du paramètre <code>metaclass</code>	671
32.4.1	Avec quoi alimenter ce paramètre ?	671
32.4.2	Avec un objet possédant une méthode <code>call</code>	671
32.4.3	Avec une fonction	672
32.5	Exemple	673
32.6	Attributs <code>mro</code> et <code>mro</code>	674
32.7	Émulation du <code>call</code> de <code>type</code>	674
32.8	Méthode <code>__init_subclass__</code>	675
32.8.1	Introduction	675
32.8.2	Signature	676
32.8.3	Exemple amusant	677
32.8.4	Nature	677
32.8.5	Émulation	678
32.9	Méthodes <code>__instancecheck__</code> et <code>__subclasscheck__</code>	679
32.9.1	Méthode spéciale derrière <code>isinstance</code>	679
32.9.2	Méthode spéciale derrière <code>issubclass</code>	680
32.9.3	Cas typique	681
32.9.4	Introspection	682
33	Classes abstraites (<code>abc</code>)	685
33.1	Définition	685
33.2	Métaclasse <code>ABCMeta</code> du module <code>abc</code>	685
33.3	Classe <code>ABC</code>	687
33.4	Décorateur <code>abstractmethod</code>	688
33.5	Décorateurs obsolètes	689
33.6	Méthode <code>register</code>	690
33.7	Surcharge de <code>__subclasscheck__</code> dans <code>ABCMeta</code>	692
33.8	Surcharge de <code>__instancecheck__</code> dans <code>ABCMeta</code>	693
33.9	Méthode <code>__subclasshook__</code>	693
33.10	Cas courants	695
33.11	Exemple : classe abstraite des validateurs	699
33.12	Module <code>collections.abc</code>	701
33.13	Module <code>numbers</code>	703
34	Aide-mémoire méthodes spéciales	705
34.1	Instanciation et destruction	705
34.2	Dérivation	705
34.3	Espaces de noms	705
34.4	Affichage	705

34.5 Comparaisons	706
34.6 Opérateurs arithmétiques	706
34.6.1 Arité 1	706
34.6.2 Arité 2	706
34.7 Valeur booléenne	707
34.8 Subscriptables	707
34.9 Conteneurs et séquences	707
34.10 Itérables	707
34.11 Hachables	707
34.12 Ensembles	707
34.13 Tables de correspondance	708
34.14 Appelables	708
34.15 Objets algébriques	708
34.16 Gestionnaires de contexte	708
34.17 Accesseurs	709
34.18 Descripteurs	709
34.19 Introspection	709
34.20 Autres méthodes	709
34.20.1 Méthode <i>bytes</i>	709
34.20.2 Méthode <i>length hint</i>	710
34.20.3 Méthode pour objets asynchrones	710
34.20.4 Divers	710
34.21 Créer ses propres méthodes spéciales	711
35 Threading	715
35.1 Notion de fil d'exécution	715
35.2 Fils d'exécution simultanés	717
35.3 Dériver la classe <code>Thread</code>	718
35.4 Attribut de type <code>Thread</code>	721
35.5 Synchroniser des fils avec des instances de <code>Lock</code>	721
35.6 Méthode <code>join</code>	724
35.7 Fonctions utiles	726
35.8 Introspection	727
35.9 Parallélisme	728
36 Réseaux	731
36.1 Protocoles et ports	731
36.2 Notion de socket	732
36.2.1 Connecteur	732
36.2.2 Famille et genre	733
36.2.3 Signature de <code>socket</code>	733
36.2.4 Adresses	734
36.3 Principes de base	734
36.4 Émettre et recevoir	736
36.5 Mise en pratique	737
36.5.1 Manipulations élémentaires	737

36.5.2	Conseils pratiques	741
36.5.3	Connecter deux ordinateurs à travers le réseau Internet	742
36.6	Envoyer une requête à un serveur HTTP	743
36.7	Expérience avec le navigateur	746
36.8	Recommandations officielles	748
36.8.1	Aléas du réseau	748
36.8.2	Messages de longueurs fixes	748
36.8.3	Messages indiquant leurs longueurs	749
36.8.4	Méthode <code>sendall</code>	751
36.8.5	Conclusion	751
36.9	Chat rudimentaire (ping-pong)	751
36.9.1	Client	752
36.9.2	Serveur	752
36.10	Les sockets sont des gestionnaires de contexte	753
36.11	Fonction <code>select</code>	755
36.12	Messagerie instantanée	757
36.12.1	Position du problème	757
36.12.2	Chat avec <code>select</code> et <code>setblocking</code>	758
36.12.3	Chat avec <code>Thread</code>	761
36.12.4	Chat avec <code>Thread</code> 100 % orienté objet	764
36.12.5	Conclusion	768
36.13	Fonctions pour retrouver un IP ou une adresse	768
36.14	Compléments	770
36.14.1	Attributs d'un socket	770
36.14.2	Méthode <code>fileno</code>	771
36.14.3	Méthodes <code>getsockname</code> et <code>getpeername</code>	771
36.14.4	Méthode <code>sendfile</code>	772
36.14.5	Méthodes <code>gettimeout</code> et <code>settimeout</code>	773
36.14.6	Méthode <code>getblocking</code>	773
36.14.7	Méthode <code>shutdown</code>	773
36.14.8	Erreurs de connexion	775
37	Annexe : listes chaînées	777
37.1	Pointeurs	777
37.2	Notion de liste simplement chaînée	781
37.2.1	Définition	781
37.2.2	Méthodes courantes	782
37.2.3	Implémentation classique (Pascal)	783
37.3	Émulation des listes simplement chaînées	784
37.4	Émulation du type <code>list</code>	789
37.5	Notion de liste doublement chaînée	792
38	Annexe : complément à 2	795
38.1	Préliminaire	795
38.2	Complément à 2 dans le cas de 4 bits	795
38.3	Trouver la valeur représentée	796

38.4	Généralisation à n bits	796
38.5	Arithmétique	797
38.6	Émulation	797
38.7	Involution	799
38.8	Petite astuce	800
38.9	Retour à Python	800
38.10	Opérateur \sim d'inversion	801
38.11	Réciproques de <code>bin</code> , <code>oct</code> et <code>hex</code>	801
39	Annexe : module <code>urllib</code>	805
39.1	Présentation du paquet	805
39.2	Notion de URL	805
39.3	Récupérer une ressource sur le web	806
39.4	Requête HTTP GET	806
39.5	Requête HTTP POST	808
39.5.1	Introduction	808
39.5.2	Requête écrite à la main	809
39.5.3	Fonction <code>Request</code>	810
39.6	Exemple amusant	811
39.7	Entête	813
39.8	Attributs d'un objet requête	814
39.9	Attributs d'un objet réponse	816
39.10	Gestion des erreurs	817
39.11	Autres modules	819
	Bibliographie	821
	Index	823