

科学技術計算のためのPython入門

開発基礎、必須ライブラリ、高速化

本書について	III
謝辞	IV
本書の構成	V
本書の想定読者について	V
動作確認に使用したOSとPythonのバージョン等について	VI
本書のサポートページについて	VI

第1章	科学技術計算とPython	1
1.1	テータて見るPythonの今	2
	Pythonの台頭	2
	教育用言語としてのPython	5
	日本におけるPythonの利用は?	6
1.2	Pythonの基礎知識	7
	Pythonの開発経緯	7
	Pythonの特徴	8
	<u>Column</u> Pythonの誕生	8
	①可読性とメンテナンス性	9
	②インタープリタ言語	9
	③スクリプト言語	9
	④グルー言語	10
	⑤バッテリー内蔵	10
	⑥充実したエコシステム	10
	パッケージ管理システム	12
	Python 2系とPython 3系	13
	<u>Column</u> Pythonのライセンス	13
1.3	科学技術計算とPythonの関わり	15
	Pythonが科学技術計算で使われる理由	15
	Pythonの普及度	15
	利用のしやすさ	16
	なぜPythonを使うのか	16
	SciPy Stack	18
	民間企業とコミュニティ	20
	Pythonの活用事例	21
	Pythonの実行速度は本当に遅いのか	22
	今回のベンチマークの詳細	23
	<u>Column</u> スタックメモリとヒープメモリ	26
1.4	まとめ	28

第2章	ゼロからのシミュレータ開発	29
2.1	シミュレータを設計する	30
	ロケットシミュレータ「PyRockSim」	30
	機能構成	31
	プログラム構築の手順	31
2.2	機能分割とファイル分割	32
	機能分割	32
	機能実装上の留意点	33
	ファイル分割とimport	34
2.3	コーディング	34
	処理の流れ	34
	ライブラリのimport	35
	ロケット諸元の設定	35
	状態量の設定と積分計算	36
	メイン実行コード	39
	計算結果の確認	42
2.4	静的コード解析	44
	静的コード解析の目的	44
	静的コード解析用ツール	44
2.5	単体テスト	46
	ソフトウェアのテスト	46
	単体テスト用のツール	47
	doctest	47
	unittest	50
	nose	52
2.6	デバッグ	53
	pdb	53
	pdbが不要なデバッグ	54
	pdbが必要なデバッグ	56
2.7	プログラムの最適化	59
	まずはプロファイリング	59
	先人の成果を活用する	61
	さらなる高速化へ	63
2.8	まとめ	64
第3章	IPythonとSpyder	65
3.1	IPython	66
	IPythonとは	66
	IPythonを使うには	67
	Column Jupyter	68

	Jupyter Notebook上での利用	69
	IPythonの基本 入力と出力の関係から	71
	オブジェクトの中身を確認する	71
	関数の内容表示	72
	関数の詳細な内容表示	73
	オブジェクト名を検索する	73
	マジックコマンド ラインマジック、セルマジック	74
	OSとの連携	76
	履歴の利用 history	77
	履歴の検索	78
	タブ補完	78
	スクリプトファイルの実行	79
	IPythonによるデバッグ デバッグpdb	79
	事後解析デバッグ	80
	スクリプト指定でデバッグ起動	82
	指定の箇所でデバッグ起動	83
	プロファイリング	84
	実行時間計測	84
	プロファイリングの準備	85
	実行時間のプロファイリング	86
	メモリ使用量のプロファイリング	89
	メモリプロファイリング対象のソースコード内指定	90
3.2	Spyder	92
	Spyderとは	92
	Spyderの主要な機能	92
	プログラムエディタ	94
	プログラムの実行	96
	Docstringやヘルプの表示 Object inspector	96
	ワークスペース内の変数を表示 Variable explorer	97
	データファイルの入出力	98
	UMD Spyderの隠れた重要な機能	99
3.3	まとめ	100

第4章

	Pythonの基礎	101
4.1	記述スタイル	102
	スクリプトの記述ルール	102
	エンコーディング	102
	インテント	103
	コメント	104
	PEP	104
	スクリプトの構成	105
4.2	オブジェクトと型	108
	オブジェクト	108
	識別子	110
	Column 予約済みの識別子	110
	データ型(組み込みのデータ型) 重要な型の一覧から	111

<i>Column</i>	イミュータブルとは何か? Pythonの実義はどのようにメモリを用いるか	112
	数値の型	113
	文字列型	113
	リスト	114
	タプル	114
	バイトおよびバイト配列	114
	辞書型	115
	集合型	115
	リテラル	116
	文字列リテラル	116
	文字列のエスケープシーケンス	117
	数値リテラル	118
	コンテナ型のリテラル	119
4.3	シーケンス型の操作	119
	インデキシング	120
	スライシング	120
	データ(値)の更新	121
	リスト内包表記	123
4.4	集合型と辞書型の操作	124
	集合型の操作	124
	辞書型の操作	126
4.5	変数とテータ	126
	変数の新規作成 Pythonの場合	127
	C言語の場合	128
	変数の再定義 Pythonの場合	129
	C言語の場合	130
	参照の割り当て 基本的な参照割り当ての例から	131
	参照割り当て後の再定義	133
	2変数への同一リストの割り当て	133
4.6	浅いコピーと深いコピー	135
	浅いコピー	135
	複合オブジェクトでない場合	135
	複合オブジェクトの場合	137
	深いコピー	138
4.7	演算子と式評価	139
	ブール値判定とブール演算	139
	比較演算子	140
	数値の型の演算	140
	整数のビット演算	142
4.8	フロー制御	142
	if文	143
	for文	145
<i>Column</i>	スペース(空白文字)の使い方	146
	while文	147
	try文	147
	with文	149

4.9	関数の定義	150
	関数定義の基本	151
	オプションパラメータ	152
	可変長引数とキーワード引数	153
	lambda式	154
	ジェネレータ関数	155
	デコレータ	156
	手続き型言語	158
4.10	モジュールとパッケージ	159
	ライブラリ、モジュール、パッケージ	159
	importの基本	160
	パッケージのimport	163
	ファイル検索の順番	163
4.11	名前空間とスコープ	164
	名前空間	164
	スコープ	165
	関数におけるスコープと名前空間	166
	名前空間と変数操作	166
	global文とスコープ拡張	167
	nonlocalとスコープ拡張	168
	クロージャ	168
4.12	まとめ	169

第5章

クラスとオブジェクトの基礎

171

5.1	クラス定義	172
	本章における解説項目について	172
	クラス定義の基本形	172
	クラス属性とインスタンス属性	174
	コンストラクタとデストラクタ	175
5.2	継承	176
	基底クラスと派生クラス	177
	継承後の属性の再定義と新規追加	177
5.3	スタティックメソッドとクラスメソッド	178
	スタティックメソッド	179
	クラスメソッド	180
5.4	隠ぺいの方法	180
	情報隠ぺいとカプセル化	181
	プライベートメンバの指定	181
5.5	クラスと名前空間	182
	名前空間とスコープの生成	182
	クラス属性とインスタンス属性	183
5.6	まとめ	186

第6章	入力と出力	187
6.1	コンソール入出力	188
	コンソール入力	188
	コンソール出力	189
6.2	ファイル入出力の基本	189
	open関数	189
	open関数のモード	190
	ファイルの読み込みとクローズ	190
	ファイルへのデータ書き出し	191
6.3	データファイルの入出力	192
	入出力によく使われるデータ形式	192
	CSVファイルの入出力	193
	標準モジュールcsv ..	194
	NumPyのCSV読み込み用関数	195
	Excelファイルの入出力	196
	XLS形式の入出力	197
	OOXML形式の入出力	197
	pickleファイルの入出力	199
	単一変数のpickle化	199
	複数変数のpickle化	200
	その他のバイナリファイルの入出力	202
	NumPyのnumpy/npz形式	202
	HDF5形式	202
	MAT-file形式	203
	Column HDF5	204
6.4	pandasのデータ入出力機能	205
	pandasのデータ入出力関数	205
	データ形式と入出力速度	207
	テキストデータの入出力	208
	読み込み処理の詳細	212
6.5	Web入力	214
	urllibパッケージを用いたHTMLデータの読み出し	214
	Python 2系とPython 3系のurllib関連情報	214
6.6	まとめ	215
	Column 入力設定の試行錯誤	216
第7章	NumPy	217
7.1	NumPyとは	218
	NumPyの機能全貌	218
	NumPyの各種関数群	219
	NumPyはなぜ速い?	220

	Column 線形代数の数値演算ライブラリ	221
7.2	NumPyのデータ型	222
	細分化されたデータ型	222
	NumPyの組み込みデータ型	222
	NumPyのスカラー	222
7.3	多次元配列オブジェクトndarray	224
	配列と行列	224
	ndarrayの生成	226
	データ型の指定	228
	ndarrayの属性	229
	ndarrayのメソッド	231
	ndarrayによる行列計算	232
	ndarrayのインデキシング	233
	基本インデキシングによる参照	233
	応用インデキシングによる参照	235
	ビューとコピー	237
	データとメモリの関係	239
7.4	ユニバーサル関数	240
	ユニバーサル関数「ufunc」の機能	240
	Python関数のufunc化	241
7.5	フロートキャスト	242
	ブロードキャストの仕組み	242
	ブロードキャストの具体例	243
	次元に関する注意事項	243
7.6	まとめ	246
第8章	SciPy	247
8.1	SciPyとは	248
	SciPyの概要	248
	NumPyとの関係	248
	最適化で一步先を行くSciPy	250
	SciPyとNumPyの差を調べる	250
8.2	実践SciPy	251
	統計分布関数	251
	離散フーリエ解析	254
	Column Pythonの統計処理	254
	ボード線図	256
	データの内挿	257
	デジタル信号フィルタの設計	260
	行列の分解	261
8.3	まとめ	264

第9章	Matplotlib	265
9.1	Matplotlibとは	266
	Matplotlibの概要	266
	Matplotlibのモジュール	266
	Matplotlibのツールキット	267
	pylabとpyplotとNumPyの関係	268
9.2	Matplotlibの設定	269
	2つの設定方法	269
	設定の確認と、設定コマンドによる変更	269
	設定ファイルへの記述	271
	スタイルシート	272
	グラフに日本語を使う	274
9.3	実践Matplotlib	277
	基本の描画	277
	サブプロット	279
	等高線図	283
	3次元プロット	285
	<i>Column</i> カラーマップについて	285
9.4	その他の作図ツール	287
	Matplotlib以外のおもな作図ツール	287
9.5	まとめ	288
第10章	pandas	289
10.1	pandasとは	290
	pandasの概要	290
	PyData	290
	pandasで何ができる?	290
10.2	pandasのデータ型	291
	基本のデータ型	292
	シリーズ	293
	データフレーム	295
	パネル	298
10.3	データの処理	300
	pandasのAPI	300
	NumPyとの連携機能 ユニバーサル関数、データ型の変換	302
	部分データを取り出す	303
	基本的な演算規則	306
	比較演算	309
	基礎的な統計関数	310
	関数の適用	312

	NaNの処理	315
	プロット機能	318
	ビューとコピー	321
10.4	まとめ	322
第11章	プログラムの高速化	323
11.1	プログラムの高速化の基本	324
	高速化への4つのアプローチ	324
11.2	ボトルネックの解消	325
	ボトルネックの解消	325
	コーディング方法による高速化	326
	先入観を持たずに試してみる	326
	極力Pythonの組み込み関数や標準ライブラリを使う	326
	ループ計算(for, while)を極力避ける	327
	メモリ利用の効率化	328
	メモリのマネジメント	328
	ndarrayに関する省メモリ化	329
	プロファイラの有効活用	332
	IPythonを使わない関数プロファイリング	332
	プロファイリング結果のグラフィカル表示...	335
	IPythonを使わないラインプロファイリング	335
11.3	処理の並列化	337
	CPUの性能向上	337
	GIL	338
	<i>Column</i> Intel Xeon Phi	338
	SIMD	339
	IntelのSIMD拡張命令	339
	PythonにおけるSIMD活用	340
	<i>Column</i> Intel MKL	340
	スレッドとマルチスレッド化	341
	マルチスレッドのプログラム	341
	並行だが並列じゃない	342
	マルチスレッド化による処理速度向上について	343
	マルチプロセス利用	343
	<i>Column</i> 注目を集めるGPU	344
	マルチプロセス利用の利点	345
	ProcessPoolExecutor	345
	<i>Column</i> Blazeエコシステム	347
11.4	まとめ	348

第12章	プログラム高速化の応用例	349
12.1	高速ライブラリ(他言語)の活用	350
	他言語ライブラリのパッケージ	350
	Cython Cythonは拡張言語	350
	Cythonの機能	351
	Cythonの使い方	351
	Cythonコード作成からコンパイルまで	352
	Cythonコードの実行時コンパイル	354
	Cythonによる並列プログラミング例 NumPyプログラムのCythonコード化から	354
	setupスクリプトの作成例	355
	高速化の効果検証	356
	自作のC/C++ライブラリの活用	357
	自作C/C++ライブラリのコンパイル	358
	ライブラリのimport方法	359
12.2	JITコンパイラ利用	360
	Numba	360
	<i>Column</i> Julia	361
	基本的な使い方	362
	どのようなプログラムに使えるか	363
	Numbaのデコレータ	364
	Numbaの利用例(@jitclass)	365
	Numbaの利用例(ufunc作成)	366
	Numbaの利用例(マルチスレッド化)	366
	Numexpr	368
12.3	まとめ	370
	Appendix	371
	Appendix A 参考文献&学習リソース	372
	Appendix B 組み込み関数と標準ライブラリ	375
	Appendix C NumPyの関数リファレンス	380
	索引	392