はじめに

こんにちは。

本書は、Python で実装した豊富なプログラム例を通じて、**アルゴリズムとデータ構造**の基礎を身につけるためのテキストです。

Python そのものの学習だけではなく、アルゴリズムとデータ構造の学習が必要なのは、次のような問題にぶつかった際に、**問題を簡単に解決する能力が要求されるからです**。

- データの集まりの中に、ある特定の値が入っているかどうかを調べたい。
- 配列の要素を小さいほうから順に並べたい。
- 常に 50 音順に並ぶように、データの集合を構造化したい。

本書は、基本的なアルゴリズムとデータ構造に始まって、目的とするデータを見つける**探索**、データの並びを一定の順序で並びかえる**ソート**、そして、**スタック・キュー・再帰的アルゴリズム・線形リスト・2分探索木**などを学習します。

学習にあたっては、高度な数学の知識は不要ですが、論理的な思考能力は必要です。そのため本書では、**難しい理論や概念を視覚的なイメージで理解できるように、213 点もの図表を示しています。**すべての解説を見開きの 2 ページ単位とすることによって、図表やプログラムと対比しながら解説を読み進めていただけるように工夫しています。

本書は、アルゴリズムやデータ構造を、ただ“紹介する”だけの本ではありません。アルゴリズムやデータ構造の基礎を学習した上で、それを使った実用的なプログラムを作る技術を身につけるための本です。本書に示している 136 編ものプログラムは、アルゴリズムやデータ構造を紹介するための、単なる“サンプル”ではなく、実際に動作するものばかりです。すべてのプログラムを読破すれば、かなりのコーディング力が身につくでしょう。

本書を活用して、アルゴリズムとデータ構造の基礎的な知識や、それらを用いたプログラムの技術等を習得していただければ幸いです。

2019 年 11 月

柴田 望洋

本書の構成

本書は、基礎的なアルゴリズムとデータ構造を学習するためのテキストです。章の構成は、次のとおりです。

第1章　基本的なアルゴリズム
第2章　データ構造と配列
第3章　探索
第4章　スタックとキュー
第5章　再帰的アルゴリズム
第6章　ソート
第7章　文字列探索
第8章　線形リスト
第9章　木構造と2分探索木

おおむね難易度の低いほうから高いほうへと並んでいますので、章の順序に沿って学習を進めていただくとよいでしょう。

▶ 第1章と第2章は、それ以降のすべての章の基礎です。また、第3章の「線形探索」は、それ以降の多くの章で応用されます。第4章の「スタック」の知識は、第5章と第6章で必要です。
　なお、第3章の「ハッシュ法」では第8章の「線形リスト」の知識が必要となり、第6章の「ヒープソート」では第9章の「木構造」の知識が必要となります。

以下、本書を読み進める上で、知っておくべきこと・注意すべきことをまとめています。

▪ 数字文字ゼロの表記について

数字のゼロは、中に斜線が入った文字“Ø”で表記して、アルファベット大文字の“O”と区別しやすくしています。ただし、章・節・図表・ページなどの番号や年月表示などのゼロは、斜線のない0で表記しています。

なお、数字の1（いち）、小文字のl（エル）、大文字のI（アイ）、記号文字の｜（たてせん）も、識別しやすい文字を使って表記しています。

▪ 逆斜線記号＼と円記号￥の表記について

Pythonのプログラムで用いられる逆斜線記号＼は、環境によっては円記号￥に置きかえられます。必要に応じて、すべての＼を￥に読みかえるようにしましょう。

▪ スクリプトプログラムについて

本書は、136編のスクリプトプログラムを参照しながら学習を進めていきます。ただし、掲載プログラムを少し変更しただけのプログラムなどは、一部あるいはすべてを割愛しています。具体的には、本書に示すのは104編で、32編は割愛しています。

すべてのプログラムは、以下のホームページからダウンロードできます。

柴田望洋後援会オフィシャルホームページ　http://www.bohyoh.com/

なお、掲載を割愛しているプログラムリストに関しては、（`'chap99/****.py'`）という形式で、フォルダ名を含むファイル名を本文中に示しています。

▪ Python に関する基礎知識について

本書で学習するアルゴリズムやデータ構造がおおむね理解できても、プログラムの実装が理解できないのでしたら、Python そのものの知識が不足しているということです。もしそうでしたら、いったん入門書に戻ってから学習を進めていただくとよいでしょう。

なお、明らかな誤りが書かれているテキストがあるので注意が必要です。たとえば、次のような解説です。

① 変数には記憶寿命（記憶域期間）がある。
② 論理演算子 and 演算子と or 演算子は、True あるいは False の論理値を生成する。
③ 代入演算子は右結合の演算子である。
④ 関数の引数の受渡しは「値渡し」あるいは「参照渡し」で行われる。

簡単に解説します。

① Python では、（他の一部の言語とは異なり）関数への出入りに伴ってオブジェクトが生成されたり破棄されたりすることはありません。当然、記憶寿命（記憶域期間）といった概念が存在し得る余地は、まったくありません（この点は **Column 1-14**：p.34 で学習します）。

② 論理式 “`x and y`” や “`x or y`” の評価で得られるのは、`True` や `False` ではなく、`x` もしくは `y` です（**Column 8-2**：p.290 で学習します）。また、そのことを利用したコーディングは、Python の世界では常識的なことです（たとえば **List 8-5 [A]**：p.294 などです）。

③ 代入文で使われる `=` は演算子でありません。“`a = b = 1`” が “`a = (b = 1)`” とみなされるという解説を見受けますが、そもそも “`a = (b = 1)`” は、エラーとなるため動作しません（**Column 2-1**：p.46 で学習します）。また、`=` が右結合の演算子ではないことを知らなければ、落とし穴に陥る可能性があります（**Column 8-3**：p.305）。

④ Python では、関数間の引数の受渡しは、「実引数が仮引数に代入される」という極めてシンプルな規則に基づいて行われます。また、引数がイミュータブルな型であるかどうかで、見かけ上の挙動が変わるものの、引数の型や性質に応じて、「値渡し」と「参照渡し」が使い分けられる、といったこともありません（**Column 2-6**：p.64 で学習します）。

これらの点すべてを誤って解説しているテキストがあります。また、ここに指摘した点以外にも、気になる解説がいろいろと見受けられます。