2023 年 5 月 22 日
綾　皓二郎

書籍の補足資料

本書に収めることのできなかった事項を掲載しています。

【NB】 課題解決に求められるマインド（mind）

それらには，探求心，好奇心，センス・オブ・ワンダー（sense of wonder），リジリエンス（resilience, 逆境力），創造性などがあります。

リジリエンスは，ストレスや困難をはねのけるたくましさです。戸塚滝登『子どもたちの未来を作ったプログラミング教育』技術評論社（2022）。

センス・オブ・ワンダーは，自然の神秘さや不思議さに目を見はる感性です。ただし，本書では対象を自然に限らず数理的な事象も含めています。レイチェル・カーソン，上遠恵子（訳）『センス・オブ・ワンダー』新潮社（1997）。

【NB】 コンピューターの利用環境： GUI と CUI

普段使っているコンピューターでは，OS (Operating System) の種類 (Windows OS，macOS など）に関係なく，ウィンドウやアイコンなどの画像をマウスで操作してコンピューターに指示して結果を得ています。これを **GUI （Graphical User Interface）**のシステムといいます。これに対して，Linux や Unix などの OS で，キーボードによる文字入力でコンピューターに指示して結果を得るシステムを **CUI （Character User Interface）**と呼んでいます。実は現在のコンピューターの OS は，GUI と CUI が併存し，ユーザーの使用目的に応じて GUI と CUI が使い分けられています。プログラミングでは，コーディングの大半は CUI の利用環境で行います。プログラムはコードをキーボード入力により作成し，処理結果をディスプレイに出力します。Colab のノートブックでは，プログラムの操作過程で Linux OS のコマンド（command，短い文字列からなる命令）を入力する必要も起きます。

【NB】 コンピューターを用いた課題解決の問題点

コンピューターによる解が直ちに課題解決になるわけではありません。課題解決には解の**評価**が必要です。解に疑問符が付く場合として，次のようなことがあります。

・データに改竄などがあり信頼できないものであれば，またデータが正統な実験や統計的調査から得られたものでなければ，いくらコンピューターで処理しても課題解決にはなりません。

・処理手順は正しくても，結果が実験方法や統計調査法の適用範囲を外れていれば，信頼できる解とは

なりません。
・実行結果を自分に都合のよいように意図的に解釈することがありますので，第三者による解の評価が求められます。
・課題解決として正しい結果が得られたとしても，物事を多角的に捉えて評価しなければ問題の解決とはならないことがあります。

以上のことは，アプリケーションソフトを使用して解を求める場合にもあてはまります。

【NB】 N 進法： N を底とする位取り記数法と漢数字表記

N 進法の表記で，アラビア数字を用いると読みが曖昧になるので，漢数字を用いるべきとする考え方があります。たとえば，10 進法と書くと，10 が 10 進数の 10 なのか，2 進数の 10 なのかが曖昧になるとし，十進法と書くと曖昧さがなくなる，という理由からです。

【NB】パスワードの作成と使用は自己責任

本書ではパスワードを生成するプログラムを作成していますが，生成されたパスワードが安全であることを保証するものではありません。自作したパスワードはあくまで自己責任でお使いください。パスワードは，可能ならば，文字種は英大文字，英小文字，数字，記号の 4 種とし，長さも 15 文字以上とするとよい。パスワードの強度（安全性）を無料でチェックしてくれる Web サイトがありますので，試してみるのも一案です。

【NB】 データ公開の原則，公開データが入手できるサイト，利用者の責任

公的データは，誰もが使いやすい形式で永続的に公開されることが民主主義的なディジタル社会の基本原則です。
・一次データが公開されること。二次データでは必要な分析ができないことがある。
・データには意図的な抜けがないこと，改竄がないこと。
・データはテキスト形式，CSV 形式でダウンロードできること。
・分析結果の図やグラフは出典を表示し，必要があれば原データまで遡れること。
・個人情報の範囲が定められ，保護されていること。
・データの利用は無料で，利用登録を必要とせず，使い方が自由であること。
・テキストを無料で公開するシステム（Project Gutenberg，青空文庫，ウィキペディアなど）への支援が求められている。

ここで，公的データとは，政府や自治体などの行政機関が保有するデータだけではありません。民間企業のデータであっても，たとえば，水俣病や東京電力福島第一原子力発電所の事故のように，多数の市民に深刻な被害を及ぼす可能性のある場合には，データは可能な限り公開される必要があります。

統計資料では，政府統計の総合窓口 e-Stat に全般的なデータが，気象庁のサイトに気象・地震などのデータがあり，無料で利用できます。一般的な資料では Wkisource，Wikitionary，Wikipedia などのサイ

トでデータが公開されており，自由に二次利用できます。利用に関する著作権などの注意事項を当該サイトで確認し，出典を明示することを忘れないようにしてください。データの利用は自由であっても，利用者による分析結果の公開には責任が伴うことにも注意してください。

【Math】　数学における問題の区分と発見学の小事典

数学者の G.ポリアは問題解決のプロセスを次の 4 つにまとめています。「問題を理解すること」「計画をたてること」「計画を実行すること」「 振り返ってみること」。発見学の小事典の章では，解析，分解と結合，発見学，一般化，再利用，帰納，類推，などの項目があります。これらは CT の概念でもあります。ポリア，G., 柿内賢信（訳）『いかにして問題をとくか』丸善（1954）。.

【Math】 数学における共通原理

数学のどの分野にも一貫して走っている「ヨコ糸」としてのいくつかの原理で，「分析と総合」「構造の科学」「記号と図式」「拡張への衝動」「機能と写像」 が挙げられています。遠山 啓『現代数学対話』岩波書店（1967）。

【Prg】 テキストエディタ

Colab のエディタとは別のテキストエディタも使えるようにしておいてください。たとえば, Windows, Mac で使えるものに Visual Studio Code，Atom などがあります。Windows 用に，Notepad++，サクラエディタなどがあります。いずれも無料です。文字コードは, UTF-8（BOM 無し）に設定します。Windows OS に付属のメモ帳は機能が足りないのでお勧めしません。ワープロソフトはテキスト形式でファイルを保存することもできますが，プログラミングでは使いません。

【Prg】リストのインデックスが `0`（ゼロ）から始まる理由

本来的に離散的なものを数えるときには，たとえば，リンゴの個数を数えるときには，`1` から始まります。リンゴが `0` 個というときは，リンゴがないときです。他方，本来は連続的なものをある単位で区切って離散的にする場合には，特定の場所を基準にして `0` から始める考えがあります。たとえば，建物はイギリス式では建物の階数を，地上を基準にして空間を区切って，地上（ground `0`）からどれだけ離れているかというオフセットの意味合いで定めています。

リストや配列は，本来連続的なメモリーをデータ型に応じて区切って離散的に扱っていますが，基準となるのはリストや配列の先頭アドレスです。そこで，先頭アドレスからどれだけ離れているかという考えに基づいて，インデックスを `0` から始める通し番号にしたと考えられます。

【Prg】 データ構造とアルゴリズム構築，プログラミング

Python には**リスト，タプル，集合，辞書，文字列**が，Numpy には**配列**というデータ構造があります。データ構造は，シーケンス型か／非シーケンス型か，ミュータブルか／イミュータブルか，型の異なる要素データの混在や重複が許されているか／許されていないか，どのようなメソッドが用意されているか，などの特徴に注意して，使い分けと相互変換ができるようになる必要があります。

アルゴリズムの構築・プログラミングでは，データ構造の選択がアルゴリズムの考案とプログラムへの変換に大きな影響を与えます。適切なデータ構造の選択が柔軟で分かりやすい，効率的なアルゴリズムとプログラムを生み出します。第 5 章以降のプログラムの例題で，適切なデータ構造が選択され，データ構造の相互変換が巧みに用いられていることを確認してください。特にリストの威力に，配列を用いる効用に気づいてください。。

【Prg】 ColabTurtlePlus によるタートルグラフィックス

Colab でタートルグラフィックスをやりたい人が多いかと思います。Python の標準ライブラリの `turtle` モジュールは，通常のグラフィックスでは意識することはあまりないのですが，Tkinter を使用し GUI を必要とするため，ブラウザベースの Python 実行環境である Colab では使えません。しかし，Colab でも，次のようにしてタートルグラフィックスをやることができます。

(1) 外部ライブラリ `ColabTurtlePlus` をインストールします（`ColabTurtlePlus` は，`ColabTurtle` よりも機能が強化されていますので，これを使ってください）。

```
!pip install ColabTurtlePlus
```

https://pypi.org/project/ColabTurtlePlus/

(2) モジュール `ColabTurtlePlus.Turtle` をインポートします

```
from ColabTurtlePlus.Turtle import *
clearscreen()
```

をプログラムの冒頭に入れます。

`ColabTurtlePlus.Turtle` でも手続き型と OOP の両方が使えるようになっています。OOP では

```
object = Turtle()
```

で，オブジェクトを作ります。

`ColabTurtlePlus.Turtle` は `turtle` モジュールとほぼ同じように使えます。ただし，いくつかの関数（メソッド）で仕様が異なりますので，`turtle` モジュールをインポートしたプログラムをそのまま移植すると正常に動かないことがあります。たとえば，関数 `speed()` の引数が異なります。`ColabTurtlePlus` の関数についてはドキュメンテーションを参照してください。

https://larryriddle.agnesscott.org/ColabTurtlePlus/documentation2.html

【Prg】オブジェクト指向開発

オブジェクト指向プログラミングは，手続き型プログラミングと同様に，分析と設計，実行，評価を含むと述べましたが，これを一般化すれば，ソフトウェア工学（SE）でのオブジェクト指向開発の

プロセス― オブジェクト指向分析（OOA），オブジェクト指向設計（OOD），オブジェクト指向プログラミング（OOP）―となります。

【Col】プログラミングの哲学 The Zen of Python（PEP20）

Python には PEP（Python Enhancement Proposals，Python 強化のための提案）という連番の資料集があり，PEP20 は The Zen of Python となっています。これは Python プログラミングの哲学について簡潔に述べたもので，その中にプログラムを書くときに基本とすべき考え方でいくつか示されています。これらはいずれもプログラムの可読性を高める働きをします。

・`Beautiful is better than ugly.`
・`Simple is better than complex.`
・`Flat is better than nested.`

【CS】 知の階層モデル DIKW

統計処理の課題解決では，プログラミングにより**データ（Data）**から価値のある**情報（Information）**が抽出され，得られた情報は**知識（Knowledge）**として統合され，知識をわがものとすれば**知恵（Wisdom）**として生かされます。これを**知の階層モデル DIKW** と呼んでいます。

【Eng】 スタック（stack）における push，pop，append のイメージと意味

プログラミングでは一般に**スタック（stack）**にデータを追加する操作を**プッシュ（push）**，スタックからデータを取り出す操作を**ポップ（pop）**と呼んでいます。英単語 push には物（オブジェクト）を力で押し込む，pop には物を急にポンと取り出す，イメージがあります。Python のスタックでは push の代わりに a**ppend** が用いられ，pop は同じです。ここで，append の意味は to add sth to the end of a piece of writing ということで，単なる `add` とは異なります。データの追加操作を，リストではメソッド `append()`，集合では `add()`と使い分けています。このように，Python ではメソッド名などの命名において，一語で細かな意味を区別をする単語を用いています。

【Eng】 `for` 文における `in` 演算子と後置されるオブジェクト

キーワード `in` は値が存在するかどうかを問う演算子ですが，前置詞として「～の中に」という意味で使われています。英文では `in` の後には前置詞の働きを受ける目的語（`object`，オブジェクト）が置かれます。`for` 文の `in` の後に置かれるイテラブル／イテレータは，プログラミングでもオブジェクトですから，`in` の目的語と考えてかまいません。英語プログラミング言語では `for` 文のような前置詞句を作って繰り返しのコードを簡潔に書くことができます。

【Eng】 オブジェクト指向プログラミングにおける基本的な用語の原義を理解する

OOP のクラス，メソッド，オブジェクトという基本用語を，学校英語のクラスを学級，メソッドを方法，オブジェクトを物体・目的語という訳語で類推しても，何か掴みどころがないと感じるのではないでしょうか。こういうときには，英英辞典で単語の原義を理解すれば，これらの語がなぜ術語として選ばれているかがよくわります。COD によれば，

class: a set or category of things having some property or attiribute in common and differentiated from others by kind, type , or quality.

method：a particular procedure for accomplishing or approaching something.

object：a material thing that can be seen and touched. Philosophy a thing external to the thinking mind or subject. a person or thing to which an action or feeling is directed.

です。英英辞典で意味が汲み取れないときには，DeepL 翻訳などのサービスを使ってかまいません。

【Prg】 オブジェクト指向プログラミング（OOP）と英語の視点・認識

OOP ではプログラム作成者（プログラマー）は画面上に生成したオブジェクトの状態を外から俯瞰的に見て，それぞれに固有なメソッドを使ってオブジェクトを操作して課題を解決していく方略をとると説明しましたが，この OOP の視点には日本語とは異なる英語の視点が強く反映されています。すなわち，英語はモノ志向，場面外視点，客観的観点，分析的把握に，日本語はコト志向，場面内視点，主観的観点，体験的把握に特徴があると言われています。池上嘉彦『「する」と「なる」の言語学―言語と文化のタイポロジーへの試論』大修館書店，1981。濱田英人『認知と言語: 日本語の世界・英語の世界』開拓社，2016。

英語では，モノを場面外から客観的に観て分析的に認識します。たとえば，川端康成の小説『雪国』の冒頭の文が，よく引用されます。「国境の長いトンネルを抜けると雪国であった」は，サイデンステッカーの英訳では「The train came out of the long tunnel into the snow country」となっています。英文では，原文にはない汽車というモノが主語として出てきて，トンネルから抜け出る汽車を，視点を汽車の外に置いて，それも上方から見下ろして認識していることが分かります。金谷武洋『英語にも主語はなかった』講談社，2004。

次の文は，OOP の発想が英語特有の認識に基づくものであることを強く示唆しています。“ものごとを抽象化・理念化 ＝「物」化してとらえる思考（&志向）は英語（等欧米語）の本性である。「物」と「物」の（さまざまな「組み合わせ」としてものごとをとらえ，あるいは構成していく思考は，機械，機械文明に最も端的に現れている。” 岩谷 宏『にっぽん再鎖国論―ぼくらに英語はわからない』ロッキング・オン社，1982。

以上から，日本人プログラマーが OOP を難しいと感じる背景に，英語と日本語の視点・認識に基本的な違いがあり，日本人プログラマーが OOP をマスターするには，英語の視点からのオブジェクトの理解が欠かせないことがお分かりいただけるかと思います。

ここで，日本人の中にもオブジェクト指向の立場から指揮をとる専門家がすでにいることに気づいてください。たとえば，将棋の棋士や舞台の演出家，野球の監督は，オブジェクト指向で場面全体を場面外から俯瞰的に観て分析して，多数のオブジェクト（駒，俳優，選手など）に指示を出し，それらを

協調的に振舞わせることで，その職分を果たしています。