巻頭言

何かをゼロから始める人にとって、入門書選びはとても難しい問題です。その分野のことを全く知らない状態のため、使われている言葉が1つ分からないというだけでつまずくことや、誤解して何時間もハマることがよくあります。だから「分かりやすく、丁寧に説明されている本で基礎から学びたい」と考えるかもしれません。

私も最近、電子工作の世界に足を踏み入れましたが、その分野の常識を知らずにハマって試行錯誤しています。それでも、基礎をしっかり身に付けてから作り始めようという気にはなりません。だって、何時間学んでも、そこから何が作れるようになるのか全然見えてこないんです。基礎を学ぶことはとても重要ですが、基礎を全部身に付けてから実践に進む方法では途中で疲れて挫折してしまいます。プログラミングでも電子工作でも、「簡単でも実践的なもの」を作って完成までの流れを把握してから、そこで必要になった基礎知識をその都度寄り道しながら学ぶほうが楽しめます。

本書『最短距離でゼロからしっかり学ぶ Python 入門』は、「必修編」で“実践に必要な基礎知識”がひと通り網羅されています。それはつまり、楽しいゲームを作るのも、仕事に役立つ道具を作るのも、必修編を修了すればあとはアイディア次第で始められるということです。そして「実践編」は、何かを作る全体像をなぞりながら必要な知識を学ぶのにピッタリです。原書名は『Python Crash Course』で、「Python短期集中講座」という意味合いです。クラッシュコースのハードな解釈には「水に投げ込んで泳げるか確認する（やってみてできないところを練習する）方式」というものもあるので、この解釈に沿って「実践編」から読みはじめるのもおすすめです。私のように実践から学びたい人は、「実践編」から始めて分からない部分にマーカーをひきつつ、「必修編」の当該箇所で基礎を押さえながら読み進める方式がおすすめです。日本語版で2冊に分かれたことで並行して読み進めやすくなったと言えるでしょう（笑）。翻訳した2人は日常的にPythonを書いている現役プログラマーとベテランWeb制作者というコンビです。異なる2つの視点で最新情報や日本特有の事情などが補完されていることにより、本書をより価値の高いものにしています。

本書の構成についてまとめましょう。**「必修編」**ではPythonの文法や基本的なデータ型が分かりやすく丁寧に説明されているため、基礎から足場をしっかり固めてから一歩ずつ学びたい人はこちらから読みはじめるとよいでしょう。読者自身で解決が難しい問題が起こりそうなところには、すぐに復帰できるように解決方法が配置してあります。これによって、「本のとおりにコードを書いたつもりなのにうまく動作しない」という書籍にありがちな問題にハマりづらい構成になっています。また、本書では例外とユニットテストについても触れていて、実践に必要な基礎知識がひと通り網羅されています。ユニットテストは作ったプログラムの入出力を明確にする設計の一部であり、その後の修正で意図しない変更（バグ）を起こさないためにも必要な、現在のプログラミングではほぼ必須となる大事な要素です。各所に配置された演習問題「やってみよう」は理解度を確認するのによいでしょう。

「実践編」では「エイリアン侵略ゲーム」の他、データ可視化ツールやWebアプリなどをプログラミングしていきます。プログラミングは、シンプルに書いて必要になったらより良い書き方に修正していく「リファクタリング」を行うことがよいとされていますが、実践編でもはじめから完璧なコードを目指すのではなく、愚直なコードをリファクタリングして整理されたコードを発掘していく流れを体験できる構成になっています。実践編を終えたら、いずれかのプロジェクトを足がかりに自分のアイディアを盛り込んでプログラムを進化させるのも良い腕試しになると思います。

2020年8月 清水川貴之

清水川貴之

株式会社ビープラウド所属。一般社団法人PyCon JP Association会計理事。2003年からPythonを使いはじめた。Python関連イベントを運営しつつ、カンファレンスや書籍、OSS開発を通じてPython技術情報を発信している。最近電子工作を始めました。

著書／訳書『自走プログラマー（2020年 技術評論社）』『Pythonプロフェッショナルプログラミング第3版（2018年 秀和システム）』『エキスパートPythonプログラミング 改訂2版（2018年 アスキードワンゴ）』『独学プログラマー（2018年 日経BP社）』『Sphinxをはじめよう第2版（2017年 オライリー・ジャパン）』

日本語版に寄せて

『Python Crash Course』（訳注：本書の原書）を執筆したことで得られている楽しみの1つは、この本が多くの異なる言語に翻訳され、プログラミングを学ぶ世界中の人々の目標達成を手助けできることです。ここのところ何年も多くの日本の読者からメールをいただいていたので、日本の読者がこの本を母国語で読めるようになることを知って、興奮を覚えています。

Pythonがはじめて世に出たとき、クリーンな言語であるとすぐに評判になりました。それは、他の言語と同等にパワフルでありながら扱いが簡単なことによります。そして、Pythonは長年にわたってシンプルさを維持しながら、着実に新機能を追加してきました。Pythonは開発されてから30年経ちましたが、完璧に現代的なプログラミング言語でもあります。Pythonは最初に学習するのに最適なプログラミング言語であり、多くのアプリケーションにおいてまさに最高のプログラミング言語です。多岐にわたる科学分野でも重用されており、データサイエンスのあらゆる局面で頼りになるプログラミング言語の1つでもあります。また、世界で最も人気のあるWebサイトのいくつかは、PythonベースのWebフレームワークであるDjangoで作られています。著名なテック系企業のほぼすべてが、何らかの形でPythonを使用しています。

Pythonは、他のプログラミング言語よりも一歩踏み込んだ形で人々に愛されています。これを物語る言葉があります。

I came for the language, and I stayed for the community.

——言語目当てでやって来て、コミュニティのおかげで居着くことになった。

Pythonコミュニティは、あらゆるバックグラウンドを持つ人々が世界中から参加することに対してとても積極的です。PyConイベントは50か国以上で開催されており、そのほとんどの国にPythonのユーザーグループがあります。オンラインとオフライン（対面）のPythonコミュニティには一貫したコード・オブ・コンダクト（行動規範）があり、職業や地位を問わず、すべての人々の学びと貢献が歓迎され、サポートされます。

クリーブランドで開催されたPyCon 2019で、翻訳者の1人である鈴木たかのりさんにお会いできたのは嬉しい出来事でした。この本の日本語版への翻訳にあたって、翻訳者のお二人とレビュアーの方々が注意深く丁寧な仕事をしてくださったことに深く感謝しています。鈴木さんとの再会を楽しみにしていますし、近いうちに家族で日本を訪れてみたいです。

皆さんのPythonの旅の成功を祈っています。もしこれがあなたにとってはじめてのプログラミング言語なら、あなたの前にはまったく新しい世界への扉が開かれようとしています。もしあなたが別のプログラミング言語の知識をお持ちなら、Pythonの持つシンプルさとパワーをお楽しみいただけるでしょう。

Happy coding!
Eric Matthes

はじめに

すべてのプログラマーには、最初にプログラムを書くことを学んだときの物語があります。私は子どもの頃にプログラミングを始めました。そのとき、父親はDECという近代コンピューティング時代の先駆的な企業で働いていました。私の最初のプログラムは、家の地下で父親が組み立てたコンピューターキット上で作成されました。そのコンピューターは、ケースがなくむき出しのマザーボードにキーボードを接続したもので、モニターはむき出しのブラウン管でした。私のはじめてのプログラムは単純な数字当てゲームで、次のようなものです。

```
数字を考えたよ！ぼくが考えた数字を当ててね: 25
小さすぎる！もう一度: 50
大きすぎる！もう一度: 42
正解！もう一度遊びますか？(yes/no) no
遊んでくれてありがとう！
```

私が作成したゲームが想定どおりに動き、それを家族が遊ぶところを見て、いつも満足していたことを覚えています。

この幼い頃の経験はいつまでも影響を及ぼしました。目的や課題を解決するために何かを作成することは満足感をもたらします。現在、私が作成しているソフトウェアは子どもの頃よりも重要なものですが、正しく動作するプログラムを作成することで得られる満足感は、同じくらい大きなままです。

この本の対象読者は?

本書の目的は、Pythonをできるだけ早く使いこなせるようになることです。そのために、「実践編」では動作するプログラム（ゲーム、データの可視化、Webアプリケーション）を構築しながら、今後の人生で役立つプログラミングの基礎を習得します。『最短距離でゼロからしっかり学ぶ Python 入門』は、Pythonのプログラムを書いたことがない、または全くプログラムを書いたことがないあらゆる世代の人に向けて書かれています。興味のあるプロジェクトに集中するためにプログラミングの基礎を学びたい人や、新しい概念を理解するために意味のある課題を解きたい人におすすめです。また、『最短距離でゼロからしっかり学ぶ Python 入門』は生徒に対してプロジェクトベースでプログラミングを導入したい中学校や高校の先生にとっても最適です。大学で受講しているPython講座のテキストよりもわかりやすい入門書が必要な場合は、本書が助けとなります。

なにを学ぶことができるのか?

本書の目的は、読者に一般的な良いプログラマー、もしくは特に優れたPythonプログラマーになってもらうことです。一般的なプログラミングの概念の基礎を説明することによって、プログラムを効率的に学び、よい習慣を身につけることができます。『最短距離でゼロからしっかり学ぶ Python 入門』の全体を通して学んだあとは、より高度なPythonのテクニックを学ぶ準備ができているでしょう。また、別のプログラミング言語を理解することがより簡単になっているはずです。

『最短距離でゼロからしっかり学ぶ Python 入門』の「必修編」では、Pythonでプログラムを書くために必要な基本的なプログラミングの概念を学んでいます。この概念は、ほとんどのプログラミング言語を学びはじめるときに共通のものです。次のことを学びます。

- データのさまざまな種類について
- データをリストや辞書に格納する方法
- データの集まりを作成し、そのデータの集まり全体に対して効率的に処理を行う方法
- whileループとif文で特定の条件をチェックし、成功した場合はコードの特定の箇所を実行し、失敗した場合は他の箇所を実行する方法（処理を自動化する場合に非常に役に立ちます）

また、対話的なプログラムを作成するためにユーザーからの入力を受け取る方法と、有効な状態の間のみプログラムを実行しつづける方法を学びます。プログラムの一部を再利用するための関数の書き方を知ることにより、特定の処理を行うコードのブロックを一度書くだけで何度も繰り返し使用できるようになります。この考え方を拡張し、より複雑な振る舞いをするクラスを使用することで、さまざまな状況にシンプルなプログラムで対応できるようになります。加えて、一般的なエラーを適切に処理するプログラムの書き方を学びます。これらの基本的な概念を実践したあとに、いくつかのわかりやすい問題を解くための短いプログラムを作成します。最後に、コードのテストの書き方を学ぶことで中級プログラミングの第一歩を踏み出します。テストを利用することでバグの混入を心配せずにプログラムを開発できます。必修編の情報はすべて、大規模で複雑なプロジェクトに取り組むための準備に必要なものです。

『最短距離でゼロからしっかり学ぶ Python 入門』の本書「実践編」では、必修編で学んだことを3つのプロジェクトに適用します。これらのプロジェクトのいずれか、またはすべてを好きな順番で進めてください。最初のプロジェクト（第1章から第3章）では、「エイリアン侵略ゲーム」というスペースインベーダーのようなシューティングゲームを作成します。このゲームはレベルが上がるとだんだん難しくなります。このプロジェクトを完了すると、2Dゲームを開発できるようになります。

2番目のプロジェクト（第4章から第6章）はデータの可視化を紹介します。データサイエンスは、大量の情報にさまざまな可視化の技術を適用することでデータを理解することを目標としています。コードから生成したデータセット、インターネット上からダウンロードしたデータセット、またはプログラムで自動的にダウンロードしたデータセットを使用します。このプロジェクトを完了すると、大量のデータを詳しく調査し、格納された情報を可視化して表現するプログラムを書けるようになります。

3番目のプロジェクト（第7章から第9章）は「学習ノート」という名前の小さなWebアプリケーションを構築します。このプロジェクトでは特定のトピック（話題）に関するアイデアやコンセプトを日記にして保管します。異なるトピックに対して別々のログで保存し、他のユーザーがアカウントを作成して自分の日記を書きはじめられるようにします。誰もがインターネット上のどこからでもアクセスできるように、プロジェクトをデプロイする方法も学びます。

この章ではエイリアン侵略ゲームにエイリアンを追加します。最初は1匹のエイリアンを画面の上部に追加し、その後エイリアンの艦隊を生成します。艦隊が横と下に移動できるようにし、弾で撃たれたエイリアンを削除します。最後にプレイヤーの宇宙船の数に上限を設け、プレイヤーがすべての宇宙船を失うとゲームが終了するようにします。
この章では、Pygameと大規模プロジェクトの管理方法についてより深く学びます。また、弾とエイリアンのようなゲーム上のオブジェクト同士の衝突を検出する方法についても学びます。衝突を検出することで、ゲーム内の複数の要素間における相互作用を定義できます。相互作用の例としては、迷路の壁にキャラクターを閉じ込めたり、2つのキャラクター間でボールをパスしたりするようなことが考えられます。当初の計画に基づいて作業を続け、作業の焦点がずれないように計画を再確認しましょう。
エイリアンの艦隊を画面に追加するコードを書く前に、プロジェクト全体を見渡して計画を更新しましょう。

プロジェクトをレビューする

大規模プロジェクトの新しい開発フェーズを始めるときには、計画を再確認してこれから作成するコードで達成したい目的を明確にしましょう。この章では次のことを行います。

- コードを調べ、新しい機能を実装する前にリファクタリングが必要かを判断する
- 画面の左上の角に適切な余白をつけて1匹のエイリアンを配置する
- 最初のエイリアンの余白を使用して、画面サイズから適切なエイリアンの数を決定する。画面上部をエイリアンで埋めるループ処理を記述する
- エイリアンの艦隊が全滅するか、エイリアンが宇宙船に衝突するか、エイリアンが画面の一番下に到達するまで、艦隊を横と下に移動する。全艦隊が撃ち落とされた場合は、新しい艦隊を作成する。エイリアンが宇宙船に衝突するか、画面の一番下に到達した場合は、宇宙船を破壊して新しい艦隊を作成する
- プレイヤーが使用できる宇宙船の数を制限し、プレイヤーが割り当てられた宇宙船をすべて使い切ったらゲームを終了する

この計画は、機能を実装しながらさらに改良していきますが、プロジェクトを開始するには十分な内容です。

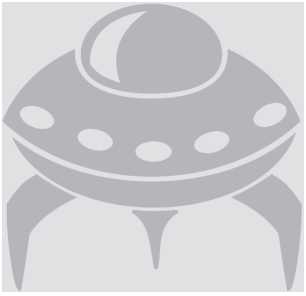
また、プロジェクトに一連の新機能を追加する作業を始める際には、事前に既存のコードを見直す必要があります。新しいフェーズに入るとプロジェクトはより複雑になるので、その前に乱雑で非効率なコードは整理

しておくべきです。エイリアン侵略ゲームについてはこれまでもリファクタリングを実施しているので、現時点でリファクタリングの必要なコードはありません。

最初のエイリアンを生成する

宇宙船を画面に配置したときと同じように、1匹のエイリアンを画面に配置します。各エイリアンの動作はAlienクラスで制御します。このクラスはShipクラスと似た構造になっています。単純化のために引きつづきビットマップ画像を使用します。自分の好きなエイリアン画像を用意してもよいですし、**図2-1**の画像を使用してもよいです。この画像はサポートサイト（https://gihyo.jp/book/2020/978-4-297-11572-2/support）からダウンロードできます。この画像の背景色はグレーなので画面の背景色とも合っています。画像ファイルをimagesフォルダーに保存します。

図2-1　艦隊に使用するエイリアン



Alienクラスを作成する

次のようにAlienクラスを作成してalien.pyに保存します。

alien.py

```
import pygame
from pygame.sprite import Sprite

class Alien(Sprite):
    """艦隊の中の1匹のエイリアンを表すクラス"""

    def __init__(self, ai_game):
```

図 3-6　エイリアン侵略ゲームの完成した得点システム

やってみよう

3-5. 全期間のハイスコア

ハイスコアはプレイヤーがエイリアン侵略ゲームを閉じて再スタートするとリセットされます。この問題に対応するために、sys.exit()を呼び出す前にハイスコアをファイルに書き込むようにし、GameStatsでハイスコアの値を初期化するときにファイルから読み込むようにします。

3-6. リファクタリング

2つ以上のタスクを行っているメソッドを探し、そのコードをリファクタリング、整理して効率化します。たとえば_check_bullet_alien_collisions()内のコードの一部を移動し、艦隊を全滅させたときに新しいレベルを開始するstart_new_level()関数を作成します。同様に、Scoreboardの__init__()メソッドの中で呼び出す4つのメソッドを移動し、__init__()の中ではprep_images()メソッドだけを呼び出すようにします。prep_images()メソッドは_check_play_button()をシンプルにするのに役立ちます。すでに_check_play_button()をリファクタリングしている場合は、start_game()もprep_images()メソッドを使ってシンプルにすることができます。

プロジェクトのリファクタリングを試みる前に付録の「A バージョン管理にGitを使う」（292ページ）を参照し、リファクタリング中にバグが発生したときにプロジェクトを正常な状態に戻す方法を学んでください。

3-7. ゲームを拡張する

エイリアン侵略ゲームを拡張する方法を考えます。たとえば、エイリアンが宇宙船を弾で撃つようにしたり、宇宙船が隠れる盾を追加したりできます。その盾はどちら側からでも弾で破壊できます。またはpygame.mixerモジュールなどを使用して爆発音や弾を撃つ効果音を追加できます。

3-8. 横向きのシューティングゲーム（最終バージョン）

横向きのシューティングゲームの開発を継続し、このプロジェクトで扱ったすべての機能を実装します。「Play」ボタンを追加し、適切な箇所でゲームをスピードアップし、得点システムを開発します。作業中には必ずリファクタリングを行い、この章で説明した内容以外にもゲームをカスタマイズできる場所を探してください。

まとめ

この章では次のことを学びました。

- 新規ゲームを開始するための「Play」ボタンの実装方法
- ボタンに対するマウスイベントを検知する方法
- ゲームがアクティブな状態のときにカーソルを非表示にする方法

ゲームの遊び方の説明を表示する「Help」ボタンなど、ゲームに他のボタンを作成する際にもここで学んだことを活かせます。

また、次のことについても学びました。

- ゲームの進行状況によってスピードを変更する方法
- 現在の状況を表示する得点システムの実装方法
- テキストまたは画像による情報の表示方法