O'REILLY®
オライリー・ジャパン

無料お試し版

Head First はじめてのプログラミング

頭とからだで覚えるPythonプログラミング入門

ご自由にお持ちください

Eric Freeman 著
嶋田 健志 監訳 木下 哲也 訳

こんにちは！

この小冊子は、プログラミングの基礎をやさしくていねいに学べる書籍『Head First はじめてのプログラミング』の無料サンプル版です。

実は書籍全体では全636ページのボリュームを誇りますが、何もおそれる必要はありません。これはプログラミングについてわかりやすく説明しようと、本文に写真やイラストをたくさん盛り込んだためです。その全体から第1章を中心に、エッセンスを抽出してまとめました。

入門書は薄くて文字も少ないほうがわかりやすいと考えていないでしょうか? それは正しい考えですが、同時に誤った考えであると私たちは考えます。薄い入門書は、目標までの道のりを最短経路で教えてくれますが、みんながみんな最短経路で無駄なく学習できるわけではありません。より多くの解説や、さまざまな例、また練習問題などがあったほうが、じっくりと身につけられるという方は多いのではないでしょうか。この点については、ぜひ学校での勉強を思い出してみてください。教科書だけで、どこまで頭に入りましたか?

本書は英語版からの翻訳によるものですが、英語圏ではすでに多くの読者がこの書籍でプログラミングを学んでいます。また、本書もその一冊である「Head First」シリーズは、これまで約30点が刊行されている、長らく実績のあるシリーズです。

このお試し版をぱらぱらと眺めていただき、「あっ、これは他の本とちょっと違うな、面白そう！」と思われたら、ぜひ実物の『Head First はじめてのプログラミング』を手に取っていただけますと幸いです。

Head First

はじめてのプログラミング

頭とからだで覚えるPythonプログラミング入門

Eric Freeman 著
嶋田 健志 監訳
木下 哲也 訳

O'REILLY®
オライリー・ジャパン

目次（要約）

はじめに

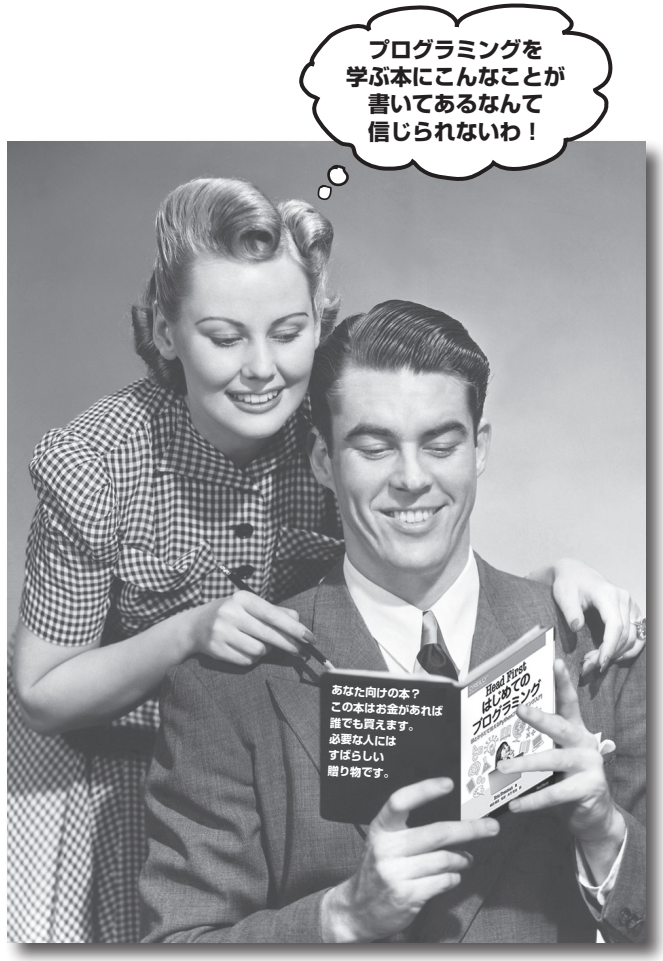
脳をプログラミングに向けましょう。あなたは何かを**学ぼう**としていますが、**脳**は学習内容を**定着**させないようにしています。脳は「どの野生動物を避けるべきかや裸でスノーボードをすることは悪いことかどうかなど、もっと重要なことに考える余地を残しておくほうがよい」と考えています。そこで、プログラミングの方法を学習することであなたの人生が大きく変わると脳が考えるように仕向けるためにどのような策を講じればよいでしょうか？

1章 コンピュータ的に考える

コンピュータ的な考え方をすることにより自分をコントロールすることができます。世の中は、ますます多くのものがつながり、設定可能になり、プログラム可能になり、いわゆる**コンピュータ的**になっています。受動的に関わることもできますが、**コードの書き方を学ぶ**こともできます。そしてコードを書くことができれば主導権を握ることができます。あなたのためにコンピュータを実行させることができるのです。コードが書けると、自分の運命を変えることができます（少なくともインターネットに接続された芝生のスプリンクラーのプログラミングができるようにはなるでしょう）。しかし、どうやってコードの書き方を学ぶのでしょうか。まず、**コンピュータ的な考え方**を学びます。次に、**プログラミング言語**を覚えます。すると、コンピュータ、モバイルデバイス、CPUを備えたデバイスが使う言語と同じ言語を使えるようになります。これにはどんなメリットがあるのでしょうか？ 本当にやりたいことをする時間が増え、能力が向上し、より創造性豊かな可能性が広がります。さあ、始めましょう。

この本の読み方

はじめに



この章では、「プログラミングを学ぶ本にどうして
こんなことが書いてあるの？」という疑問に答えます。

この本が向いている人

次の 2 つの項目すべてに該当すれば、この本はあなたに向いています。

1. プログラミング方法を**学び、理解し、覚えたい**。
2. **無味乾燥で退屈な学校の授業よりも、ディナーパーティでの刺激的な会話**が好き。

「マーケティング担当者からのコメント：
この本はクレジットカードを持っている人にオススメです。」

これはリファレンス本ではありません。『Head First はじめてのプログラミング』はプログラミングの学習を目的としています。プログラミングの真相に関する百科事典ではありません（それには Google がありますよね）。

この本が向いていない人

次の 3 つの項目のいずれかに該当する場合には、この本はあなたに向いていません。

1. コンピュータについてまったく何も知らない。

   ファイルやフォルダの管理方法、アプリケーションのインストール方法、ワープロソフトの使い方などコンピュータについて詳しくなければ、まずはそれを学ぶべきでしょう。

2. **リファレンス本**を探しているプログラミングの達人である。
3. **変わったものを試すのが怖い**。

   変に見られるよりは苦痛に耐えるほうがよい。プログラミングの学習を楽しんでいるような技術本は真面目ではないと思う。

あなたの考えはわかっています

「この本のどこが真面目なの？」
「どうして絵や写真ばかりなの？」
「こんなので本当に**学べる**わけ？」

あなたの「脳」がどう考えるかもわかっています

人間の脳は常に目新しいことを求めています。そして、いつも何か珍しいことがないかを**探し求めて**います。人間が生き延びるため、生来、脳はそのように作られているのです。

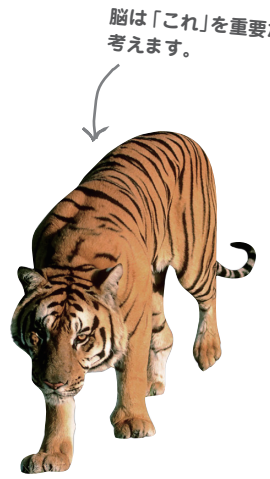
今日では、トラのえさになることはほとんどありません。しかし、それでも脳は注意しています。自分では気付いていないだけです。

では、決まりきったありきたりの、平凡なことに対して、脳はどのように対処するのでしょうか？**重要なこと**を記録するという脳の**本来の**仕事を平凡なことに邪魔されないように全力を尽くすのです。脳は退屈なことはわざわざ記憶しません。「たいして重要じゃないな」とフィルタが働くからです。

では、脳はどのようにして重要なことを**判断する**のでしょうか？例えば、ハイキングに出掛けたときに突然トラが襲いかかってきた場合、頭と体の中では何が起こるでしょうか。

そんなとき、脳の神経細胞が燃え上がり、感情が昂り、**化学物質が活性化します**。

そして、脳はこう判断するのです。



これは重要だ！絶対に忘れるな！

しかし、家や図書館という、トラに襲われる心配がない安全で暖かい場所で、試験勉強か、あるいは上司から1週間から10日間くらいかけて目を通しておくように言われた難しい技術テーマを学習しているとします。

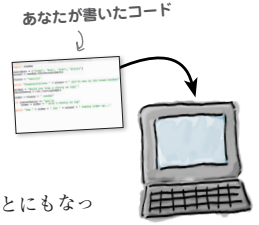
ここで1つだけ問題があります。脳はあなたの役に立とうとします。つまり**明らかに**重要でないことに脳の貴重なリソースが使われないようにするのです。脳のリソースは本当に**重要な**こと、例えばトラが襲ってくる、火事の危険がある、二度と短パンをはいてスノーボードをしてはいけない、といったことを覚えておくことに使うほうがいいのです。

「脳さん、いつも本当にありがとう。だけど、この本は退屈でいまはまったく魅力もないんだけど、何とか頭の中に入れておいてよ」と脳に伝える簡単な方法はないのです。

「Head First」の読者は学ぶ人です。

学習には何が必要なのでしょうか？まず**習得**し、その上で**忘れない**ようにする必要があります。知識を頭に詰め込むことではありません。認知科学、神経生物学、教育心理学の最新の研究によると、**学習**には文字以外のものが必要なのです。必要なのは脳のスイッチを入れることです。

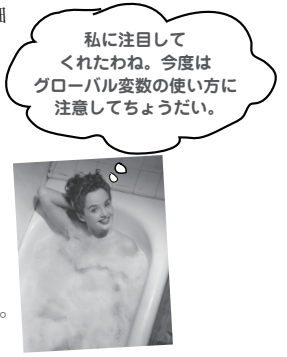
Head Firstシリーズには学習についての方針があります。



ビジュアル重視。画像は、文字だけよりもずっと記憶されやすいので、学習効果が高まります（最大89%向上）。また、内容がより理解しやすくなります。この本では、**関連する絵や図、写真の中や近くに説明の文章を配置しています。**ページの下や別のページにまとめて配置してしまうと、内容に関連する問題を2度考えることにもなってしまいます。

会話のような親密な感じの文体。最近の研究では、無愛想な硬い文体よりも一人称を使って読者に直接語りかける会話的な文体のほうが、学習後のテストで学生の成績が最大40%も向上することがわかっています。講義ではなく、物語を語るような砕けた文体をわれわれは使っています。あまり深刻に受け取らないでください。ディナーパーティで仲間と会話するほうが講義よりも話に身が入りますよね？

考えを深めながら学べるようにする。つまり、自らの脳細胞を活性化させない限り、頭の中は何も変わらないのです。読者に必要なのは、目的をはっきり持ち、熱心に楽しみながら問題の解決に集中し、結論をひねり出し、新しい知識を習得することです。そのためには、脳と感覚の両方を駆使する課題、練習問題、Q&Aなどに取り組みましょう。



読者の関心を引きつけ、飽きさせない。「本当に学びたいんだけど、1ページも読み終わらないうちに眠くなってしまう」という体験は誰にでもあるでしょう。人間の脳は、ありきたりでないもの、面白く、変わっていて目を引く、予想に反したものに関心を集めます。新しくて難しい技術的なテーマを学ぶことは必ずしも退屈ではないのです。退屈でなければ、脳はかなり短時間で学習できます。

感情に訴える。何かを記憶する能力は、感情の動きに大きく左右されることがわかっています。気になることや何かを感じたことは記憶しやすいのです。何も「少年と愛犬の涙と感動の物語」のことを言っているのではありません。驚き、好奇心、楽しみ、「これは何だ？」といった感情や、パズルを解いたときに起こる「やったぞ！」という感覚、誰もが難しいと思っていることを習得したとき、エンジニアのボブより技術に詳しいことがわかったときなどの感情のことを指しているのです。

メタ認知：思考について考える

何かを心から学習したいと願っていて、しかも効率よく深く学習したいのであれば、関心の持ち方に注目するといいでしょう。考え方について考え、学び方について学ぶのです。

ほとんどの人は大人になるまでにメタ認知や学習理論の授業を受けたことがありません。私たちは学ぶことを**要求**されますが、学び方を**教わる**ことはまずありません。

この本を手に取った方は、プログラムやアプリケーションをコーディングする方法を本気で勉強したい人でしょう。でも、勉強にあまり時間をかけたくありませんよね。また、読んだ内容を**記憶**し、使えるようになりたいでしょう。そのためには、**理解**する必要があります。この本に限らず本や学習体験を最大限に活用するためには、自分の脳に対して責任を持つようにします。

学習の秘訣は、学んでいる新しい物事が「本当に重要」なものだと脳に思わせることです。つまり、学習している内容がトラに関することと同様に、あなたが幸せになるために重要なことであると認識させるのです。そうしないと、新しい内容を脳に留めるために脳と常に格闘することになります。

では、飢えたトラと同じようにコーディングを脳に扱わせるにはどうしたらいいのでしょうか？

そのためには退屈で時間のかかる方法と、効率的で時間のかからない方法があります。退屈で時間のかかる方法とは、ひたすら繰り返すことです。どのくらいつまらなく関心のないことでも、何度も繰り返し学べば覚えられるという経験は誰でもあるでしょう。十分に繰り返せば、脳は「これは重要とは**思えない**のだが、これだけ**繰り返して**いるのだから重要だと思うことにしよう」と判断するのです。

効率的で時間のかからない方法とは、**脳の働きを活性化させる**ことです。特に、さまざまな**種類**の脳の働きを活性化させるのです。前のページに示した項目はそのための重要な役割を担い、脳を望みどおりに働かせるのに役立つことがわかっています。例えば、絵や写真の**中に**内容を説明する言葉を入れておくと、見出しや本文のような別の箇所に入れるよりも言葉と絵や写真との関係を理解しようとして神経細胞がより活発に働くという研究報告もあります。神経細胞が活発に働けば働くほど、関心を払い記憶するにふさわしいものであると脳が考える可能性が高くなります。

会話のような文体が効果的なのは、人は自分が会話の中に入っていると感じると、話に遅れないようにして会話における責任を果たすことが期待されるため、より多くの関心を払うからです。面白いのは、会話が本との間で行われていても脳は一向に**構わない**という点です。逆に、文体がかしこまった無味乾燥なものであると、脳は大勢の受け身な聞き手と一緒に座って講義を受けている状態と同じように受け止めます。居眠りしてもかまわないと考えるのです。

しかし、ビジュアル化や会話的なスタイルは最初の一歩にすぎません。

この本で工夫したこと

この本では**絵や写真**を多用しています。これは、人間の脳が文字よりも視覚要素に反応するからです。脳の観点から見れば、1つの絵や写真は1000文字の言葉に匹敵します。また、テキストと絵や写真を組み合わせるときは、関係するテキストを絵や写真の**中に**埋め込みました。テキストが関連するものの**中に**あるほうが、表題や本文のどこかに埋もれているよりも脳がずっと効果的に働くからです。

また、同じことを**繰り返し**説明しています。同じことを**さまざま**な表現や素材で表現して**複数の意味**を持たせることで、学んだ内容が脳のさまざまな領域に記憶される可能性が高くなります。

Pythonのインタプリタになってみよう

脳は目新しいものに向かうようになっているので、概念と絵や写真を**予想外の**方法で使うようにしました。また、脳は感情の動きに注目するという特性があるので、少なくとも何らかの**感情に訴えるような**形で使いました。その感情がちょっとした**ユーモア**、**驚き**、**興味**にすぎなくても、何かを**感じさせた**内容は記憶される可能性が高くなるのです。

読者ひとりひとりに**話しかけるような文体**を使いました。脳は、受け身の姿勢で説明を聞いているときより会話をしているときのほうが注意を払うようになっているからです。こうした脳の働きは本を**読む**場合でも同じです。

重要ポイント

脳は何かを**読んで**いるときより何かを**行っている**ときのほうが学習効果が高いので、120問以上の**練習問題**を収めました。練習問題のレベルは「難しいけれども実行可能」というくらいにしてあります。ほとんどの人がそのくらいのレベルの問題を好むからです。

複数のアプローチで学べるようにしています。基礎から順を追って学習するほうがいいという人もいれば、ともかくまずだいたいの概要をつかみたいという人、例だけが見たいという人もいるからです。好きなアプローチはそれぞれに異なっていても、同じ内容を複数のアプローチで表現していれば**誰もが**納得するでしょう。

クロスワードパズル

左脳と右脳の両方を使う内容を盛り込んでいます。脳を使う箇所が多いほど、学習や記憶の効果が高まり、集中力も持続します。一方の脳が働いているときには他方の脳を休ませられるので、長い目で見ると学習の生産性が上がります。

複数の観点を示す**話題**や練習問題を含めるようにしています。脳は、評価や判断を強いられたときのほうがより深く学習するようになっているからです。

彼らが付き合ってくれます。

練習問題を用意したり、必ずしも簡単な答えが出ないような**質問**をしたりすることで、**課題**が生じるようにしています。なぜなら、脳は何かに**取り組む**必要があるときに学習し記憶する特性があるからです。ジムで人を**見ている**だけでは**体**を鍛えることはできません。しかし、努力しているときは、**確実に力が付く**ように最善を尽くしているのです。難解な例を挙げたり、難しい専門用語を多用したり、逆にあまりに当たり前すぎる説明をしたりして、**脳細胞を余計に使わせることはありません**。

説明、例、絵や写真などに**人物**を登場させています。**あなた**もやはり人なので、脳は**物**よりも**人**に関心を持つからです。

80/20のアプローチを採用しています。プログラミングの達人になるつもりなら、この本だけでは十分ではないと考えています。この本では**すべて**を取り上げてはいません。本当に**必要**になることだけを取り上げています。

ここから下を切り取って、冷蔵庫にでも貼っておくといいでしょう。

脳を思いどおりにさせるためにできること

この本は最善を尽くしていますので、あとはあなた次第です。次のヒントは第一歩にすぎません。脳に耳を傾け、何が役に立ち何が役に立たないかを把握してください。新しいことに挑戦してみましょう。

1. **時間をかけて読みましょう。理解すればするほど、覚えなければならないことの数は少なくなります。**

   ただ**読む**だけでなく、ときどき読むのを止めて考えましょう。本の中で問題が出されても、すぐに答えを見ないでください。誰かから本当に質問されていると思いましょう。脳に深く考えさせればさせるほど、学習や記憶の効果が高まるのです。

2. **問題を解きましょう。自分のノートに書き込んでください。**

   この本には「自分で考えてみよう」「エクササイズ」「コードマグネット」といった練習問題を載せていますが、私たちが問題を解いてあげてしまったら、あなたのために他人がトレーニングしているようなものです。練習問題を**見て**いるだけではいけません。**実際に鉛筆を使って取り組んでください。**学習**中**に体を動かすと学習効果が高まります。

3. **「素朴な疑問に答えます」を読みましょう。**

   このコーナーはとても大切です。これは内容を補足するものではなく、むしろ**核心となる内容の一部なのです！**読み飛ばさないようにしましょう。

4. **この本を読んだあとは寝るまで他の本を読まないようにしましょう。少なくとも、難しいものは読まないようにしましょう。**

   学習の一部、特に学習内容の長期記憶への転送は、本を閉じた**あとに**行われます。脳が次の処理を実行するには時間が必要です。この処理中に新たに別のことを学習すると、前に学習したことが一部失われてしまいます。

5. **水をたくさん飲みましょう。**

   脳は十分な水分がある状態で最もよく働きます。脱水状態（のどが渇いたと感じる前に起こります）になると認知機能は衰えます。

6. **内容をはっきりと声に出してみましょう。**

   話すことは脳の別の部分を活性化します。何かを理解したい場合やあとで思い出しやすくしたい場合には、はっきりと声に出してください。さらによいのは、それを他の誰かに明確に説明してみることです。そうすると、学習の効率が上がり、読んだだけではわからなかった概念がはっきりするかもしれません。

7. **脳に耳を傾けましょう。**

   脳に負担をかけすぎないように注意しましょう。内容を表面的にしか理解できなくなったり、読んだばかりのことを忘れるようになったりしたら休憩してください。ある限界を超えると、それ以上詰め込もうとしても学習の効率は上がらず、かえって学習を妨げることもあります。

8. **感情を持ちましょう。**

   脳は**重要**であるかどうかを判断する必要があります。話に集中しましょう。写真や絵などに自分なりの注釈を入れるのもよい方法です。くだらないジョークに文句を言うだけでも、何も感じないよりはるかにいいのです。

9. **何か作ろう。**

   計画中の新規案件を作成するか、昔のプロジェクトを改良しましょう。この本の練習問題や作業以外にも**何か**をやってみて経験を積んでください。必要なのは鉛筆と解決すべき課題だけです。プログラミングの恩恵を受ける課題です。

10. **よく寝よう。**

    プログラミングを学ぶには新たに多くの脳神経の結合が必要です。よく寝ましょう。睡眠が役立ちます。

初めに読んでね

この本は順を追って読みながら学習を進めていく本であり、辞書のようなリファレンスではありません。そのため、学習の障害になりそうな内容についてはあえて割愛しています。この本を初めて読む場合には、1章から順番に読んでください。それまでの章で学んだことを前提に話を展開しているからです。

プログラミングの背後にある思考プロセスを学んでほしい。

その思考プロセスをコンピュータ科学と呼ぶ人もいますが、少し秘密があります。コンピュータ科学は科学ではなく、それほどコンピュータに関連しているわけでもありません（天文学が望遠鏡に関連していないのと同じです）。コンピュータ科学は考え方です（最近ではコンピュータ的思考とも呼ばれます）。コンピュータ的な考え方がわかると、課題、環境、プログラミング言語に適用しやすくなります。

この本ではPythonを使う。

自動車なしで運転を学ぶのはあまり実用的ではありません。プログラミング言語なしでコンピュータ的な考え方を学ぶのは、売りになるスキルというより思考実験です。そこで、この本では人気のPythonを使います。1章ではPythonのよさを詳しく説明しますが、趣味で使うのか高収入を得るソフトウェア開発者になりたいのかにかかわらず、手始めとしては（そして、おそらく最終的にも）Pythonが適しています。

Python言語のあらゆる側面を包括的に取り上げているわけではない。

包括的には程遠い内容です。Pythonについて学べることはたくさんあります。この本はリファレンス本ではなく学習するための本なので、Pythonについて知っておくべきことをすべて取り上げているわけではありません。この本の目的はプログラミングとコンピュータ的な考え方の基本を教え、**どのプログラミング言語**の本を選んでも途方にくれないようにすることです。

さらに、Mac、Windows PC、またはLinuxを使える。

この本では主にPythonを手段として使います。Pythonはクロスプラットフォームなので、使い慣れたOSで利用できます。この本のスクリーンショットの大部分はMacのものですが、みなさんのWindows PCやLinuxマシンでも同じように見えるはずです。

この本ではベストプラクティスに基づいた適切な構造の読みやすいコードを推奨している。

自分や他者が読んで理解でき、将来の新しいバージョンのPythonでも機能するコードを書きたいでしょう。この本では、最初からわかりやすい適切な構造のコードを書く方法を教えます。満足のいく、額に入れて壁に飾りたいようなコードを書く方法です（デートに連れてくる前に外してください）。プロが書くコードとの唯一の違いは、この本ではコードの隣に手書きの注釈を付けてそのコードが何をするかを説明している点です。学習のための本では、そのほうが従来のコード内のコメントよりも効果的であることに気付いたからです（何を言っているのかわからなくてもすぐにわかります。何章かで試してみてください）。しかし、コードにコメントを付ける方法を教え、コードへのコメントの例も示すので心配しないでください。とはいえ、最も簡単にコードを書き、やるべきことをやってさらに優れた処理に進めるようにしたいと思っています。

プログラミングは大事な作業である。時には懸命に取り組まなければいけない。

プログラマは異なるものの見方をし、異なる考え方をします。コーディングがとても論理的だと思うこともあれば、そこまでショッキングではないにしてもかなり抽象的だと感じることもあります。理解するのに時間のかかるプログラミング概念もあります。実際に、一晩考えないと理解できないものもあります。しかし、心配しないでください。この本では脳に優しい方法でそのすべてに取り組みます。マイペースで取り組み、時間をかけて概念を理解し、必要ならこの本の内容を何度も繰り返し読んでください。

練習問題を省略しない。

「自分で考えてみよう」「エクササイズ」「コードマグネット」といった練習問題は付けたしではありません。この本の重要な本文の一部です。記憶や理解を促すものもあれば、学んだことを応用する際に役立つものもあります。練習問題を飛ばすと、この本の大半をみすみす見逃してしまうことになります（おそらくかなり困惑するでしょう）。飛ばしてもいいのはクロスワードパズルだけですが、用語を別の見地で考える機会を脳に与えてくれるものです。

あえて冗長にしている。これは重要。

Head Firstシリーズは、読者に内容を本当に理解してほしいという点が他の本と異なります。一冊を読み終えたときに、学習したことを覚えていてほしいのです。ほとんどのリファレンス本は、記憶することや思い出すことを目的としていません。本書は学習に目的を置いているため、同じ概念を何度も取り上げることがあります。

例はできるだけ簡潔にしてある。

理解する必要のある2行のコードを探すのに、200行の例を苦労して読まなければいけないのはイライラすると読者から言われたことがあります。この本のほとんどの例は最小限の文脈だけを示しているので、学習したい部分が明確で単純になっています。すべての例が完璧であるとは思わないでください。学習しやすいように作成したものなので、必ずしも完全に機能するわけではありません。一方、長いコードの例では、友達や家族に見せたくなるような楽しく、興味を引く、素晴らしい例になるようにも努めています。

サンプルコードのファイルはすべてWebからダウンロードできます。https://wickedlysmart.com/hflearntocode（原書のサイト）またはhttps://www.oreilly.co.jp/books/9784873118741/（日本語版のサイト）から入手してください。

オペレータは待機していませんが、必要なコードとサンプルファイルはhttps://wickedlysmart.com/hflearntocodeから入手できます。

能力発揮には通常は答えがない。

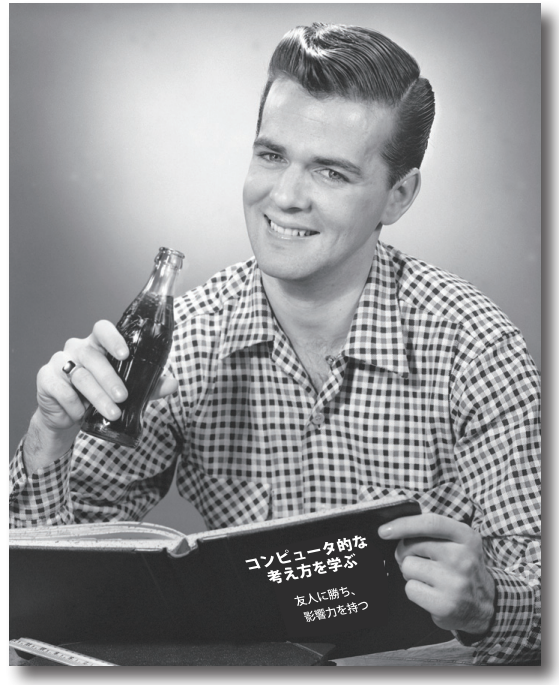
「能力発揮」のセクションには正しい答えがないものもあります。また、答えが正しいかどうかを判断したり、どのような場合に正しいかを判断することもあります。一部には、正しい方向へ導くためのヒントもあります。

サンプルコード、ヘルプ、解説をオンラインで入手する。

コード例のファイルや追加の補足資料など、この本で必要なものはhttps://wickedlysmart.com/hflearntocodeあるいはhttps://www.oreilly.co.jp/books/9784873118741/から入手できます。

1章　コンピュータ的に考える

始めよう



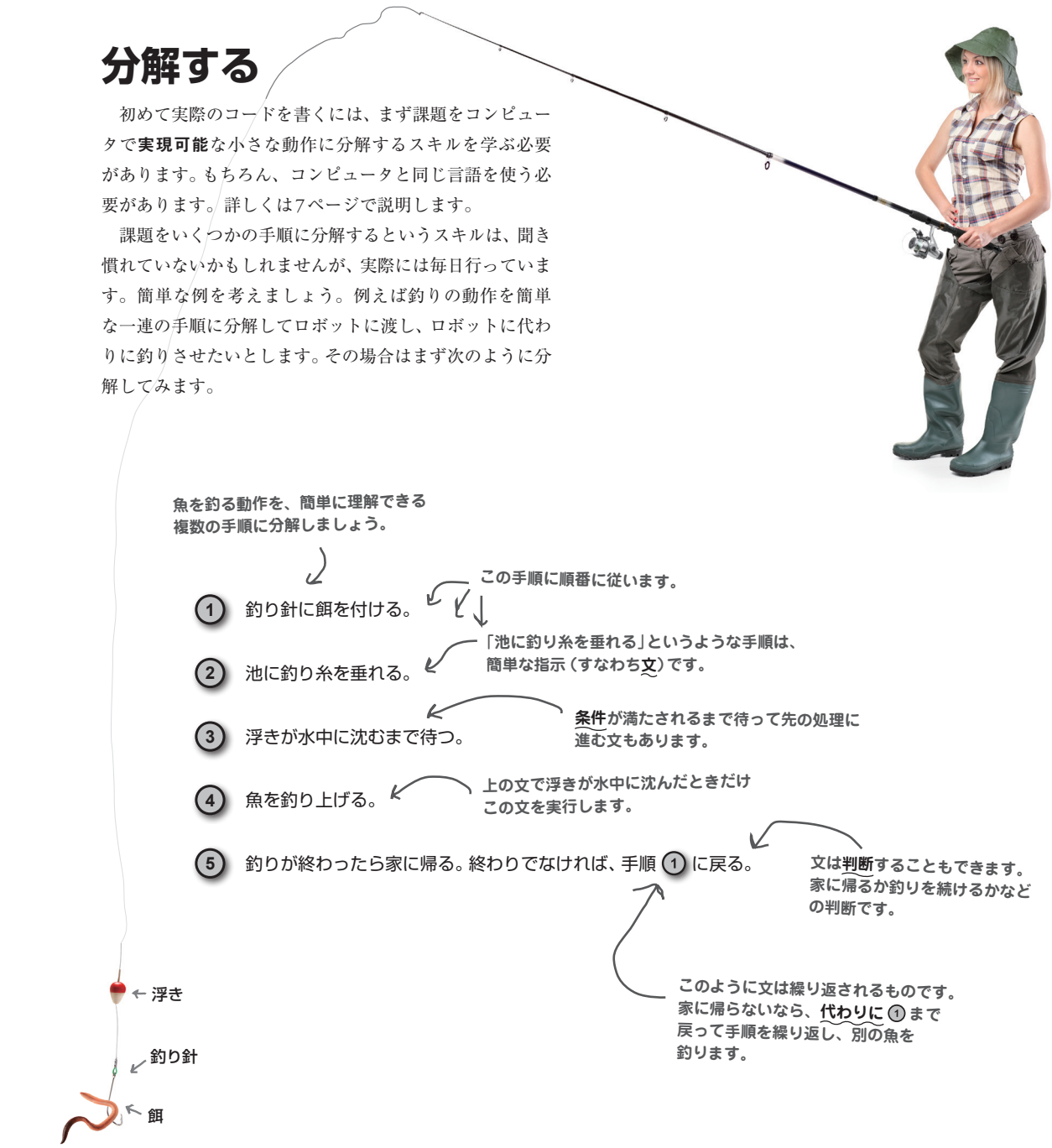
コンピュータ的な考え方をすることにより自分をコントロールすることができます。

世の中は、ますます多くのものがつながり、設定可能になり、プログラム可能になり、いわゆる**コンピュータ的**になっています。受動的に関わることもできますが、**コードの書き方を学ぶ**こともできます。そしてコードを書くことができれば主導権を握ることができます。あなたのためにコンピュータを実行させることができるのです。コードが書けると、自分の運命を変えることができます（少なくともインターネットに接続された芝生のスプリンクラーのプログラミングができるようにはなるでしょう）。しかし、どうやってコードの書き方を学ぶのでしょうか。まず、**コンピュータ的な考え方**を学びます。次に、**プログラミング言語**を覚えます。すると、コンピュータ、モバイルデバイス、CPUを備えたデバイスが使う言語と同じ言語を使えるようになります。これにはどんなメリットがあるのでしょうか？ 本当にやりたいことをする時間が増え、能力が向上し、より創造性豊かな可能性が広がります。さあ、始めましょう。

分解する

初めて実際のコードを書くには、まず課題をコンピュータで**実現可能**な小さな動作に分解するスキルを学ぶ必要があります。もちろん、コンピュータと同じ言語を使う必要があります。詳しくは7ページで説明します。

課題をいくつかの手順に分解するというスキルは、聞き慣れていないかもしれませんが、実際には毎日行っています。簡単な例を考えましょう。例えば釣りの動作を簡単な一連の手順に分解してロボットに渡し、ロボットに代わりに釣りさせたいとします。その場合はまず次のように分解してみます。



前ページの文は釣りをするための親切な**レシピ**と考えられます。他のレシピと同様に、このレシピも一連の手順を示し、順番どおりに従うことで何らかの結果を生み出します(この例では、うまくいけば魚が釣れます)。

ほとんどの手順は、「池に釣り糸を垂れる」や「魚を釣り上げる」のような簡単な指示です。しかし、少し異なる指示もあります。例えば、3番目の文は「浮きが水面の上にあるか下にあるか」という条件に依存します。最後の「釣りが終わっていなければ、最初に戻って釣り針に餌を付け直す」などは、レシピの流れを指示しています。「終わったら家に帰る」は釣りを止める条件を示しています。

このような簡単な文や指示がコーディングの土台になることがだんだんとわかっていきます。実際に、これまで使ったことのあるアプリケーションやソフトウェアプログラムは、どれもコンピュータに実行すべきことを伝える簡単な(場合によっては多くの)指示なのです。

エクササイズ

実際のレシピは**何をすべきか**という手順だけではなく、料理を作るために使うオブジェクト(計量カップ、泡立て器、フードプロセッサ、そしてもちろん材料など)も示します。釣りのレシピで使うオブジェクトとは何でしょうか。前ページの釣りのレシピでオブジェクトに該当するものに丸を付け、30ページで答えが合っているかを確認してから先に進んでください。

練習問題ですから、直接書き込んでかまいません。むしろ実際に書き込むことをお勧めします。

自分で考えてみよう

あらかじめ理解しておいてもらいたいことがあります。コンピュータは**指示したとおり**にしか動作しません。それ以上でも以下でもありません。前ページの釣りのレシピを考えてください。ロボットになってこの指示に正確に従うとしたら、このレシピで本当にうまくいくでしょうか。どのような問題が起こり得るでしょうか。

- ☐ A. 魚がいなければ、長時間(もしかしたら永久に)釣りをすることになってしまう。
- ☐ B. 釣り針から餌が外れても気付かず、付け直すこともない。
- ☐ C. 餌がなくなった場合は？
- ☐ D. 魚を釣り上げたら何をするかを指定したか？
- ☐ E. 釣り竿は？
- ☐ F. _____

他にどんな問題が起こり得るでしょうか？

釣りについてよく知らない人のために説明しておくと、これが浮きです。魚が餌に食い付くと、水中に沈みます。

この問題の答えは、30ページにあります。

コーディングの勉強をしようと思って
高い技術書を買ったのに、レシピの話を
聞かなくちゃならないの？ あまり期待
できそうにないし、技術的でもないわね。

実際、レシピはコンピュータへの指示を表す方法として最適です。

この本よりも高度なプログラミング書籍でも「レシピ」という用語が特に断りなく使われていることもあります。クックブックと呼ばれる一般的なソフトウェア開発技法に関する本でもよく使われています。もちろん、「レシピ」という言葉を使わずに技術的に説明することもできます。コンピュータ科学者やプロのソフトウェア開発者ならレシピのことを**アルゴリズム**と呼びます。アルゴリズムとは何でしょうか。レシピとほとんど変わりありません。アルゴリズムは、最初は**擬似コード**と呼ばれる非公式なコードで書くことが多いでしょう。

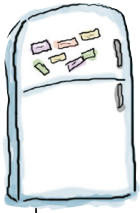
擬似コードについては2章から詳しく説明します。

1つ覚えておいてほしいことがあります。レシピ、擬似コード、アルゴリズムについて説明する際は、まず問題の解決方法の概要を示してから、コンピュータが理解して実行できる実際のコードの詳細を検討するようにします。

こうするとコーディングの作業が簡単になり、間違いが起こりにくくなります。

この本では、「レシピ」「擬似コード」「アルゴリズム」を必要に応じて切り替えています。次の就職面接では、**アルゴリズム**や**擬似コード**という言葉を使ってより多くの報酬を確保するといいでしょう（しかし、**レシピ**という言葉でも問題はありません）。

同じ料理を作るためのたくさんのレシピがあるように、同じ問題を解決するための数多くのアルゴリズムがあります。そして、中でも特に魅力的な方法があります。



コードマグネット

~~レシピ~~アルゴリズムの練習を少ししてみましょう。Head Firstレストランの卵を3個使ったオムレツのアルゴリズムを忘れないように冷蔵庫のドアに貼っていましたが、誰かが順番を崩してしまいました。マグネットを正しい順番に直し、アルゴリズムを使えるようにしてください。なお、Head Firstレストランのオムレツには、プレーンとチーズの2種類があります。**31ページで必ず答え合わせをしてください。**

マグネットを並べ直してアルゴリズムが正しく機能するようにしてください。

注文がチーズオムレツなら:

上にチーズを加える

卵がよく混ぜ合わされていなければ:

卵をお皿に移す

卵に火が十分通るまで:

フライパンを火から下す

卵をかき混ぜる

客に提供

フライパンを火にかける

卵を溶く

3個の卵を割ってボールに入れる

卵をフライパンに流し入れる

コーディングの手順

コンピュータに何かタスク（コンピュータが処理する仕事のこと）を実行してもらいたい場合は、そのタスクをコンピュータが理解できる複数の指示に分割する必要があることはわかりましたが、**実際にどのようにして**コンピュータに何かを実行するように**指示**するのでしょうか。しかし、プログラミング言語に深く足を踏み入れる前に、実際にコードを書く際の手順を調べましょう。

❶ アルゴリズムを作成する

ここで解決したい問題やタスクをレシピ、擬似コード、アルゴリズムに変換します。これは、必要な結果を実現するためにコンピュータが実行する必要がある手順を表します。

この段階ではプログラミング言語に変換する前に解決策を立てます。

1. 釣り針に餌を付ける。
2. 池に釣り糸を垂れる。
3. 浮きが水中に沈むまで待つ。
4. 魚を釣り上げる。
5. 釣りが終わったら家に帰る。終わりでなければ、手順 ① に戻る。

❷ プログラムを書く

次に、そのレシピをプログラミング言語で書いた特定の命令に変換します。これが**コーディング**と呼ばれる作業で、その結果を「プログラム」または「コード」（さらに正式な名称は「ソースコード」）と呼びます。

```
def hook_fish():
    print('魚が釣れた！')
def wait():
    print('待つ')
print('餌を取り出す')
print('釣り針に餌を付ける')
print('釣り糸を垂れる')
while True:
    response = input('浮きが水中に沈んだか？')
    if response == 'yes':
        is_moving = True
        print('魚が食いついた！')
        hook_fish()
    else:
        wait()
```

これが「コーディング」の段階です。アルゴリズムをコード（ソースコードの略）に変換し、次の段階で実行できる状態にします。

❸ プログラムを実行する

最後に、ソースコードをコンピュータに渡し、コンピュータがその命令の実行を開始します。この段階は、言語によってコードの**解釈**（interpreting）、**実行**（running、executing）、**評価**（evaluating）などと呼びます。

われわれもこの3つを同じ意味として使うことがあります。

ソースコードが完成したら、実行の準備が整ったことになります。コードが適切に設計できていれば、期待した結果が得られるでしょう。

同じ言語で話している?

プログラミング言語とは、コンピュータにタスクを明確に指定するために作成した専用言語だと考えてください。つまり、コンピュータが理解できるようにわかりやすく正確にレシピを表す手段なのです。

プログラミング言語を学ぶには、2つのことをはっきりさせておきます。言語を使って何が表せるのかと、その意味です。コンピュータ科学者は、それぞれを言語の**シンタックス(構文)**、そして**セマンティクス**と呼んでいます。この2つはいまの段階では頭の奥深くにしまっておいてください。2章で両方とも詳しく説明します。

世界中にさまざまな言葉がさくさんあるように、プログラミング言語も**数多く**あります。もう気付いたかもしれませんが、この本ではPythonを使います。プログラミング言語とPythonについてもう少し理解を深めましょう。

ほっと一息

心配しないでください。まだコードの読み書きができなくても大丈夫です。いまの段階ではコードに親しみ、コードの表現や動作に慣れるだけで十分です。この章で重要なのは全体を見渡すことだけです。

この本で学ぶテクニックは、将来使うことになるどんなプログラミング言語にも適用できることが、そのうちにわかります。

何て言う?

左側は人間が使う言語で書いた文、右側はプログラミング言語で書いた文(コード)です。人間の使う文と、それに対応するコードを線で結んでください。1番目だけは線を引いておきました。32ページで答え合わせをしてから先に進みましょう。

画面に「やあ」と出力する。

気温が22℃以上なら、画面に「短パンをはく」と出力する。

パン、ミルク、卵といった食料品の覧。

5杯の飲み物を注ぐ。

ユーザに「あなたのお名前は?」と聞く。

```
for num in range(0, 5):
        pour_drink()

name = input('あなたのお名前は? ')


if temperature > 22:
        print('短パンをはく')

grocery_list = ['パン', 'ミルク', '卵']


print('やあ')
```

プログラミング言語の世界

この本には、さまざまなプログラミング言語の名前が登場します。また、街中の書店のプログラミング本のコーナーでも、(一部の例を挙げるだけでも) Java、C、C++、LISP、Scheme、Objective-C、Perl、PHP、Swift、Clojure、Haskell、COBOL、Ruby、Fortran、Smalltalk、BASIC、Algol、JavaScript、そしてもちろんPythonという名前を見かけるでしょう。このようなプログラミング言語の名前は何に由来しているのか不思議に思うかもしれません。実は、プログラミング言語の名前はロックバンドの名前のようなものです。言語開発者にとっては意味があるのです。例えばJavaは、予想どおりコーヒーにちなんでいます(開発者が気に入っていた名前のOakは先に取られてしまいました)。Haskellは数学者ハスケル・カリーから、Cはベル研究所で開発された言語AとBの後継であったことに由来します。しかし、なぜこんなに多くの言語が存在しているのでしょう。それにそれぞれどういったものなのでしょうか。複数の人に、自分が使用している言語についての意見を聞いてみましょう。

僕のお気に入りはObjective-C。一日中iPhoneアプリを作っているんだ。Objective-CはCに似ているけどずっと動的だしオブジェクト指向なところが好きだな。Appleの新しい言語Swiftも勉強しているよ。

Javaではオブジェクトのレベルで考えられるし、メモリ管理やスレッドといったことも任せられるの。

僕はWordPressの世界で生きているよ。WordPressはPHPで書かれているから、PHPが僕の主力言語だね。PHPをスクリプティング言語と言う人もいるけど、必要なことは何でもできるよ。

主にC言語を使っている。OSの一部を書く必要があるから、効率性がすごく求められるんだ。実際の仕事ではあらゆるCPUサイクルとメモリ位置が重要なんだ。

研究者の私はSchemeやLISP系の言語が好きだな。私にとっては高階関数と抽象化が重要なんだ。Clojureのような関数型言語が実際に産業界で使われるようになっているのは嬉しいね。

Web開発者の私はJavaScriptを主に使うの。JavaScriptはすべてのブラウザのデファクト言語だし、バックエンドWebサービスを書くのにも使われているわ。

私はシステム管理者です。よくPerlを使ってさまざまなシステムスクリプトを書いています。たった数行のコードで多くの仕事をこなすことができます。

選択、選択、また選択

このように、多くの言語があり、多くの意見があります。いまの私たちは言語についてほんの少しかじっているだけです。また、言語には多くの専門用語が付随しているので、いまは戸惑うことも多いかと思いますが、学習を進めていくうちにその用語のことがよく理解できるようになるでしょう。現時点では、世の中にはすでに多種多様な言語があり、毎日のように多くの言語が生まれていることを覚えておいてください。

では、どの言語を選べばよいのでしょうか。いまは何よりもまず、**コンピュータ的**に考える方法を私たちは学びたいのです。そうすれば、将来どのような言語に出会っても習得しやすいからです。とにかく**何らかの言語**を使わなければならないので、私たちはPythonを使うことにします。なぜでしょうか？ 上の友人たちがその理由をはっきりと述べています。Pythonはとても読みやすく一貫性があるので、初心者に最適な言語の1つと考えられています。また、強力な言語でもあります。Pythonを使って（この本や他のことで）何を実行するにしても、コード拡張（**モジュール**や**パッケージ**と呼びます）というかたちでサポートがあり、開発者のコミュニティがサポートしてくれます。それに、Pythonは他の言語よりも**楽しい**と言う開発者もいます。Pythonを選んでおいて間違いはないでしょう。

自分で考えてみよう

まだPythonのことをよく知らなくても、コードがどのように動作するかを十分推測できると思います。次のコードがそれぞれ何をするか推測して、予想を書いてみましょう。わからなければ、次のページの答えを見てください。1行目の答えは書いておきました（各単語の下に日本語の意味を記したので参考にしてください）。

Pythonの簡単さを確認する

```
customers = ['ジミー', 'キム', 'ジョン',
顧客                'ステイシー']

winner = random.choice(customers)
勝者          ランダムに選ぶ

flavor = 'バニラ'
味

print('おめでとう ' + winner +
出力      ' アイスクリームサンデーが当たりました！')

prompt = 'サクランボを乗せますか？ '
プロンプト

wants_cherry = input(prompt)
欲しい   チェリー     入力

order = flavor + 'サンデー '
注文

if wants_cherry == 'yes':
    order = order + 'サクランボ乗せ'

print(winner + 'の' + order + '1つ '  +
      ' すぐにできます')
```

顧客リストを作成する。

Pythonの出力

```
おめでとう ステイシー アイスクリームサンデーが当たりました！
サクランボを乗せますか？ yes
ステイシーのバニラサンデーサクランボ乗せ1つ すぐにできます
```

このコードの出力です。これがヒントになるでしょう。このコードはいつ実行しても同じ出力となるでしょうか？

まだPythonのことをよく知らなくても、コードがどのように動作するかを十分推測できると思います。次のコードがそれぞれ何をするか推測して、予想を書いてみましょう。

Pythonの簡単さを確認する

```
customers = ['ジミー', 'キム', 'ジョン',
             'ステイシー']

winner = random.choice(customers)


flavor = 'バニラ'


print('おめでとう ' + winner +
      ' アイスクリームサンデーが当たりました！')

prompt = 'サクランボを乗せますか？ '


wants_cherry = input(prompt)


order = flavor + 'サンデー '

if wants_cherry == 'yes':
    order = order + 'サクランボ乗せ'

print(winner + 'の' + order + '1つ '  +
      ' すぐにできます')
```

顧客リストを作成する。

顧客の 1 人をランダムに選ぶ。

変数に数値や文字列を設定することをプログラミングでは「代入」と言います。

flavorという名前（変数）にテキスト「バニラ」を設定。

画面に客の名前が入ったおめでとうメッセージを出力する。例えば、キムが当選した場合、このコードは「おめでとう キム アイスクリームサンデーが当たりました！」と出力する。

promptという名前（変数）にテキスト「サクランボを乗せますか？」を設定する。

ユーザにテキストを入力するように要求し、そのテキストをwants_cherryに設定する。ユーザに入力を求めると、（Pythonの出力で示したように）まずプロンプトが表示される。

orderにテキスト「バニラ」と「サンデー」をつなげて設定する。

ユーザが「サクランボを乗せますか？」という問いにyesと答えたら、orderにテキスト「サクランボ乗せ」を追加する。

注文がすぐにできあがると出力する。

Pythonの出力

```
おめでとう ステイシー アイスクリームサンデーが当たりました！
サクランボを乗せますか？ yes
ステイシーのバニラサンデーサクランボ乗せ1つ すぐにできます
```

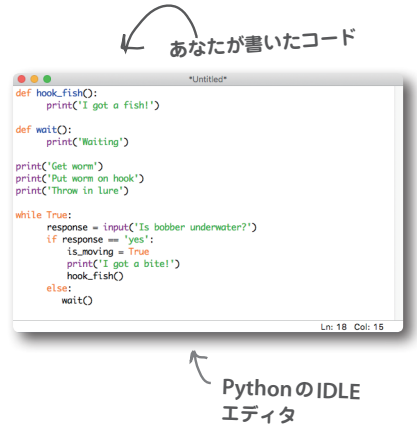
追伸：このコードを入力して試したいという誘惑にあらがえない場合は、ファイルの先頭にimport randomを追加してから実行してください。この文の意味は後で説明します。断っておきますが、現段階でこのコードはあまり参考にならないし、実行する必要はありません。でも試してみずにいられないという気持ちもわかります。自分のことは自分がよくわかっていますからね！

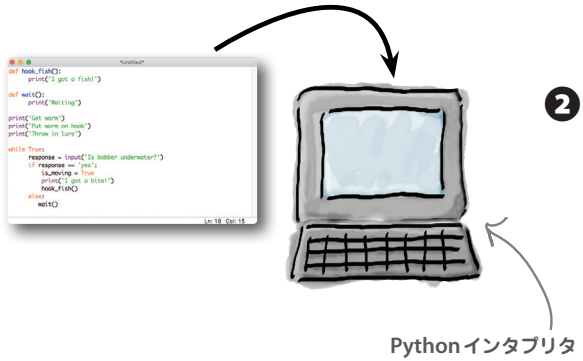
Pythonを使ったコードの記述と実行

少しコードのことがわかってきたと思います。いよいよ本物のコードを実際に書いて実行してみましょう。前にも言ったように、コードの書き方と実行方法は言語や環境によってさまざまなやり方があります。どのようにPythonのコードを書いて実行するかをおおまかに理解しましょう。

❶ コードを書く

まず、エディタにコードを入力して保存します。Windowsのメモ帳やMacのテキストエディットなど、普段使っているテキストエディタを使ってコードを書くことができます。ですが、ほとんどの開発者はIDE（Integrated Development Environment：統合開発環境）と呼ばれる特殊なエディタを使います。なぜでしょうか？ IDEはワープロソフトに似ています。一般的なPythonキーワードの自動補完、言語構文（または構文エラー）のハイライト表示などの多くの優れた機能に加え、テスト機能も備えています。また、Pythonは都合のよいことにIDLEというIDEを備えています。IDLEについては18ページから説明します。





❷ コードを実行する

コードの実行は簡単で、コードをPythonインタプリタに渡すだけです。Pythonインタプリタは、書いたコードを実行するために必要な処理を行うプログラムです。すぐに詳しく説明しますが、インタプリタはIDLEや、コンピュータのコマンドラインから直接利用できます。

❸ コードの解釈方法

Pythonは、人間とコンピュータの両方が理解できる言語だと主張してきました。これまで学んだように、インタプリタはコードを読んで実行する役割を果たします。そのために、インタプリタは実は水面下でコードをマシンコードに変換し、そのマシンコードはコンピュータハードウェアが直接実行します。インタプリタがどのように変換するかを気にする必要はありません。インタプリタがPythonのコードを実行することを知っていればよいのです。

素朴な疑問に答えます

Q：なぜ英語を使ってプログラミングしないのですか？ そうすればこのように特殊なプログラミング言語を学ばなくていいのに。

A：そうできればいいのですが、英語は曖昧な点だらけで、前述のような変換器を作るのが極めて困難です。この分野に取り組んでいる研究者もいますが、プログラミング言語として英語や他の話し言葉を使うにはほど遠い状態です。また、英語に似たものにしようとしている言語もありますが、プログラマは話し言葉にはあまり似ていない、もっとコーディングに効率的な言語を好みます。

Q：プログラミング言語はなぜ1つだけではないのですか？

A：技術的には、最近のプログラミング言語はすべて同じ演算ができるという点で同等なので、理論的には1つのプログラミング言語であらゆるニーズを満たすことができます。しかし、話し言葉と同様に、プログラミング言語はそれぞれ表現力が異なります。つまり、ある種のプログラミングタスク（例えば、Webサイトの構築）には他の言語よりも適した言語があるのです。また、好みや特定の手法を利用するためや、さらには勤務先が採用しているからという理由だけでプログラミング言語を選ぶ場合もあります。しかし、プログラミング言語は進化し続けるので、ますますその数が増えるのは間違いないでしょう。

Q：Pythonは初心者向けのトイ言語（おもちゃの言語）にすぎないのですか？ ソフトウェア開発者になりたい場合、Pythonの学習は役に立ちますか？

A：Pythonは本格的な言語で、みなさんが愛用している多くの製品に使われています。さらに、Pythonは初心者にも適した言語と考えられている数少ないプロ用言語の1つです。なぜでしょうか？ 既存の多くの言語と比べ、Pythonは簡単な一貫性のある方法で物事に対処します（このことは、徐々にPythonや他の言語の経験を積むにつれよく理解できるようになります）。

Q：コードの書き方を学ぶこととコンピュータ的に考えることは何が違うのですか？ コンピュータ的に考えるとは単なるコンピュータ科学なのですか？

A：コンピュータ的な考え方は問題解決に関する考え方で、コンピュータサイエンスから生まれました。コンピュータ的に考えると、問題を分解して解決するためのアルゴリズムを作成し、その解決策を一般化する方法が身につくので、大きな問題も解決できます。コンピュータにそのアルゴリズムを実行させることがコーディングの役割です。コーディングは、コンピュータ（または、スマートフォンなどのすべてのコンピュータデバイス）にアルゴリズムを指定する手段です。そのため、この2つは実に密接に関連しています。コンピュータ的な考え方はコーディングしたい問題の解決策を考案する方法であり、コーディングはその解決策をコンピュータに指定する手段です。コーディングしない場合でもコンピュータ的な考え方をすることは大切です。

脳力発揮

Pythonという名前は何から由来すると思いますか？ 最もあてはまりそうな項目をチェックしてください。

- ☐ A. 開発者はヘビが好き。彼は過去にもあまり出来のよくない言語Cobraを開発したことがある。
- ☐ B. 開発者がパイ（pie）が好きだったので、Piethon。
- ☐ C. イギリスのシュールなコメディグループの名前に由来している。
- ☐ D.「Programming Your Things, Hosted On the Network」の頭字語。
- ☐ E. ランタイムシステムAnacondaに由来する。AnacondaはPythonをビルドする基盤環境。

C.「モンティ・パイソン」（Monty Python）は、「空飛ぶモンティ・パイソン」（Monty Python's Flying Circus）というテレビ番組を制作したイギリスのコメディグループ。下調べをしてせひ彼らの番組を見てください。気に入るかどうかにかかわらず、Pythonコミュニティではモンティ・パイソンはよく登場します。

Pythonのざっくりした歴史

Python 1.0

オランダ国立情報数学研究所（National Research Institute for Mathematics and Computer Science：CWI）では大きな問題に悩まされていました。この研究所の科学者たちにとって、プログラミング言語の習得が研究の障害となっていました。高度な教育を受けた科学者たちにとっても、最新のプログラミング言語はわかりにくく、一貫性がなかったのです。研究所はその救済策として、新たな言語「ABC」（このときはまだ「Python」ではありません）を開発しました。ABCは、学びやすいように設計されていたので、ある程度成果は上がっていたのですが、週末にモンティ・パイソンの再放送を一気に見たグイド・ヴァン・ロッサム（Guido van Rossum）という野心的な若い開発者は、ABCをさらに進化させることができると考えたのです。GuidoはABCから学んだことを生かして、Pythonを開発しました。その後の話はあなたもご存知のとおりです。

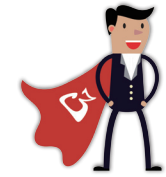
実は、「週末に一気に見た」の部分は話を盛っています。

編集者からのコメント：「その後の話」は次の段落に書かれています。

Python 2.0

Pythonは Python 2.0に進化し、成長を続ける開発コミュニティを支えることを目的とした新機能を備えました。例えば、Pythonは真のグローバル言語であると認められ、2.0では英数字以外の多くの文字セットを扱うように拡張されました。また、言語の多くの技術面も改善されました。メモリ処理の改良や、リストや文字列などの一般的なデータ型サポートの改善などです。

さらに、開発メンバーはPythonを開発者コミュニティ全体にオープンにするように努め、開発コミュニティが言語や実装の改善に貢献できるようにしました。



Python 3.0

完璧な人はいないように、Pythonの開発者たちも完璧ではありません。開発者たちがPythonを見直して一部を改良すべき時期が来ました。Pythonは簡潔さを保っていることで知られていましたが、使用が広がるにつれ一部の設計を改善し、古くなった部分を削除する必要が出てきました。

このような変更は、Python 2の一部の機能がサポートされなくなることを意味しました。しかし、Pythonの開発者は2.0コードを引き続き動作させる方法を保証したので、Python 2で書かれたコードでも問題はありません。Python 2はまだ十分使うことができますが、将来はPython 3になることを覚えておいてください。

空飛ぶ車がPython対応になることが予想できます。

1994　2000　2008　未来！

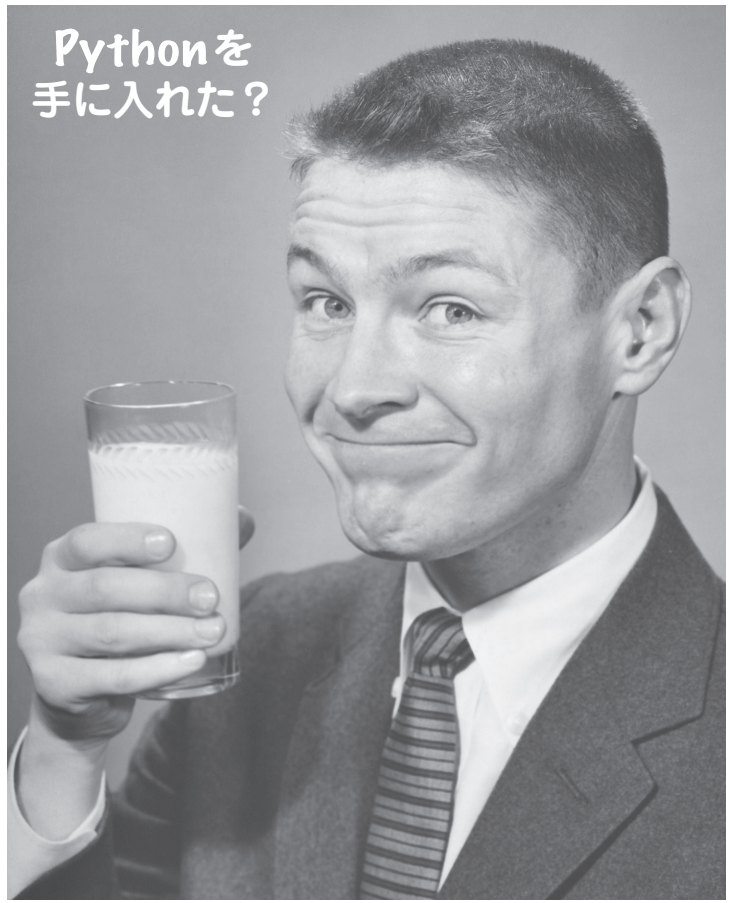
じゃあ、Python 2とPython 3の2つのバージョンがあるんだね。どっちを使うの？Python 2とPython 3はどこが違うの？

いい質問ですね。あなたの言うとおり、Pythonには異なる2つのバージョンがあります。より詳しく言えば、この本の出版時点でのバージョンはPython 3.7とPython 2.7です。

バージョンについては次のように考えるとわかりやすいでしょう。例えば、約3,000キロメートル上空からPython 2とPython 3を眺めると、同じように見えます。違いがまったくわからないかもしれませんが、実際には異なります。使用するバージョンに注意しないと痛い目に遭うでしょう。この本ではPythonの最新バージョン、つまりPython 3を使います。将来に引き継がれるバージョンで始めてもらいたいからです。

世の中にはすでにPython 2で書かれたコードがたくさんあります。インターネットからダウンロードしたモジュールには必ずといっていいほど、Python 2のコードが含まれています。また、ソフトウェア開発者であれば、古いコードの一部に残っているPython 2のコードをメンテナンスすることになるでしょう。この本を終わりまで読めば、Python 3とPython 2の小さな違いがわかるようになるでしょう。

Python 3、あるいはPython 2と言う場合、それぞれの最新バージョンのことを意味します（この本の翻訳時点ではPython 3.7とPython 2.7）。



Pythonをインストールしないことには、先に進むことはできません。

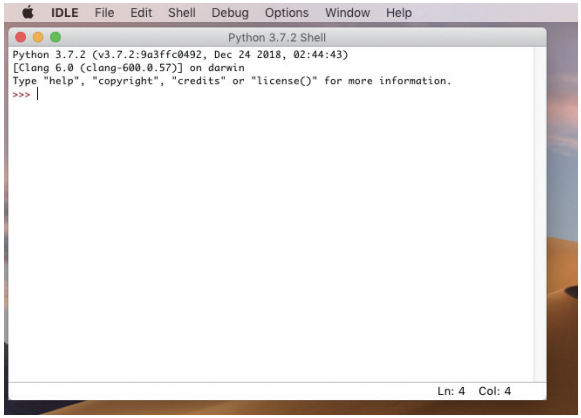
時間がなくて、まだインストールできていないのなら、いますぐインストールしてください。「はじめに」の「Pythonをインストールする」（xxxii～xxxiiiページ）が参考になります。MacやLinuxには、すでにPythonがインストールされているでしょう。でもPython 3ではなくPython 2の可能性が高いので、Mac、Windows、Linuxのいずれであっても、ここでPython 3をインストールする必要があるかもしれません。

Python 3がインストールできたら、本物のコードを書く準備は完了です。

Pythonを試す

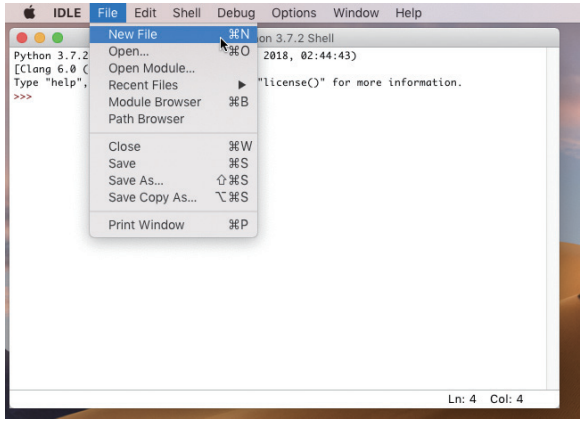
Pythonがインストールできたのでさっそく使ってみましょう。問題なく動くかを確認する小さなテストプログラムの作成から始めます。まず、エディタを使ってプログラムのコードを入力します。ここではPythonエディタ*IDLEの出番です。「はじめに」でも説明しましたが、まずはIDLEを開きます。Macでは[アプリケーション]→[Python 3.x]フォルダにIDLEがあります。Windowsでは[スタート]ボタン→[すべてのプログラム]→[Python 3.x]を選ぶとメニューにIDLEが表示されます。

＊ IDE (Integrated Development Environment)とも言います。



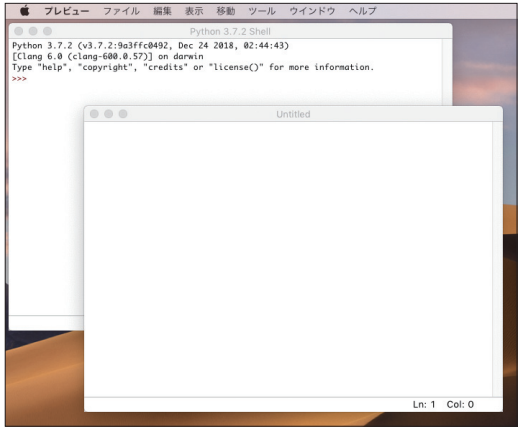
IDLEを開くと、対話型インタプリタのPython Shellが表示されます。興味があれば、このインタプリタに1+1(1足す1)と入力し、リターンを押してください。詳しくは2章で説明します。

IDLEを開くと、まずPython Shellと呼ばれる対話型ウィンドウが表示されます。Python Shellには Pythonの文を入力することができます。IDLEメニューの[File]から[New File]を選ぶと、何も入力されていない編集ウィンドウが現れるので、ここに入力します。



[File]→[New File]を選択すると、新しいウィンドウが開くので、コードを入力します。

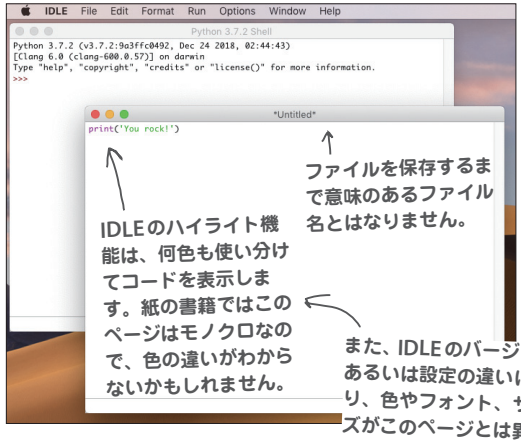
IDLEはワープロソフトと同様の機能を備えていますが、理解できるのはPythonのコードだけです。また、Python言語のキーワードハイライトやフォーマットなど、入力をサポートする機能を備えています。さらに必要に応じてPythonキーワードを自動補完するなど、入力の負担を軽減してくれます。



[New File]を選択すると、新しいウィンドウが表示されます。

まだ何も入力されていない編集ウィンドウにコードを1行入力して試してみましょう。さっそく次のコードを入力してください。

```
print('You rock!')
```



ファイルを保存するまで意味のあるファイル名とはなりません。

IDLEのハイライト機能は、何色も使い分けてコードを表示します。紙の書籍ではこのページはモノクロなので、色の違いがわからないかもしれません。

また、IDLEのバージョン、あるいは設定の違いにより、色やフォント、サイズがこのページとは異なる可能性があります。

スペルや句読記号は間違えないようによく注意してください。Pythonだけでなく、他のプログラミング言語でも句読記号の間違いに容赦せずにエラーとなることが多いです。

注意！

現在のIDLEで日本語を入力するのは不便です。ここではmacOSで日本語を入力する例を紹介します。Pythonのコマンドや予約語は、すべて半角で入力します。処理したい文字列に全角の日本語（漢字やひらがな）を利用したい場合は、まずメニューバーで日本語入力になっていること、つまりメニューバーに「あ」が表示されていることを確認して行います。
本書の説明のように、IDLEの「File」メニューから「New File」を選びます。本の例では、

```
print('rock!')
```

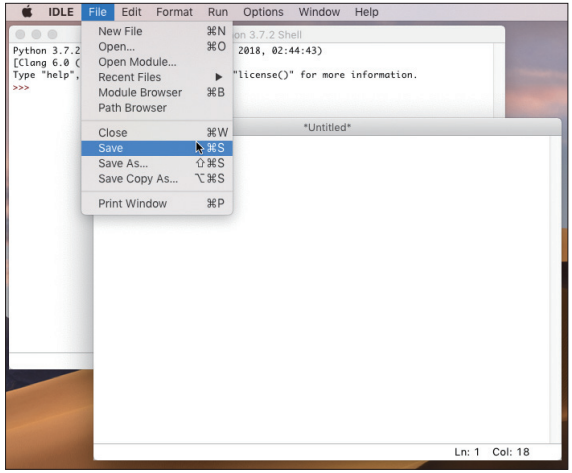
のように、そのまま半角英字と記号を続けて入力（書くことが）できます。「rock!」の代わりに、全角文字で「ロック！」にしたいときは、まず半角で

```
print('')
```

と入力してから、シングルクォート（''）の間に日本語を入力します。クォート記号の間にカーソルを移動したら、メニューが全角の入力ができる状態であること、メニューバーに「あ」が表示されている、を確認します。ここで、ローマ字かな漢字変換でタイプしても、何も現れません。しかし、そのままスペースバーで確定します。すると、漢字が現れます。
IDLEの右端の「Help」メニューから「Python Docs」F1を選ぶと英語のマニュアルが表示されます。日本語のマニュアルはhttps://docs.python.org/ja/3/を見てください。「IDLE Help」の日本語訳はhttps://docs.python.org/ja/3/library/idle.htmlにあります。現段階では翻訳途中のようです。
翻訳されていない英語の文章の意味を知りたいときは、英語に得意な友達に尋ねたり、Google翻訳などを利用するとよいでしょう。

作業を保存する

入力した1行目のコードを保存しましょう。保存するには、IDLEメニューの[File]から[Save]を選びます。

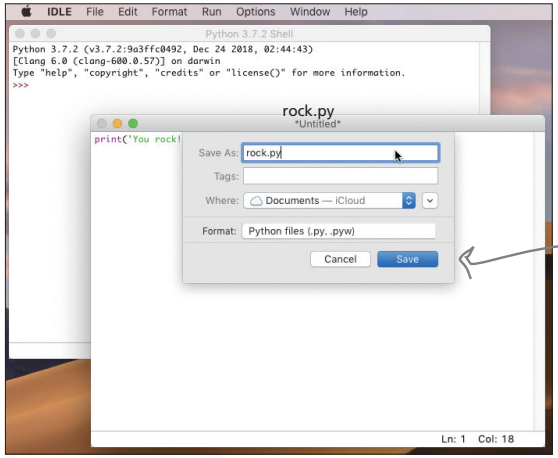


コードを実行する前に、保存する必要があります。IDLEでは、[Save]は[File]メニューから選択します。[Save]を選んでソースコードに名前を付けるだけで保存できます。Pythonのコードは、ファイル名の最後に拡張子「.py」を付けます。

ソースコード、ソースファイル、コード、プログラムという名前は、いずれもコードが保存されているファイルを指します。

そして、コードに名前を付け拡張子 .py を末尾に付けます。ここではrock.pyという名前にします。ここでは示しませんでしたが、ch1という1章用のフォルダも作成しました。みなさんも作成することをお勧めします。

この本と同じフォルダ名とファイルになるようにしてください。詳しくは、xxxivページの「コードのまとめ方」を参照してください。



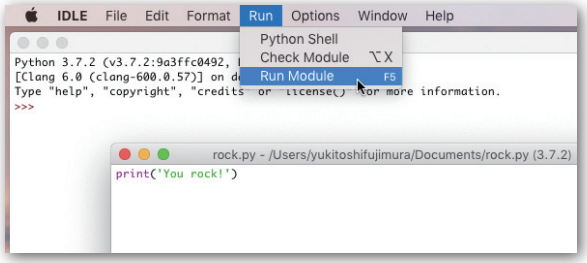
コード用のフォルダができたら、[Save]をクリックします。

試運転

ここですべてうまくいっていることを確認しましょう。コードを保存したら、メニューの［Run］から［Run Module］を選ぶと、Python Shellウィンドウにプログラムの出力が表示されます。

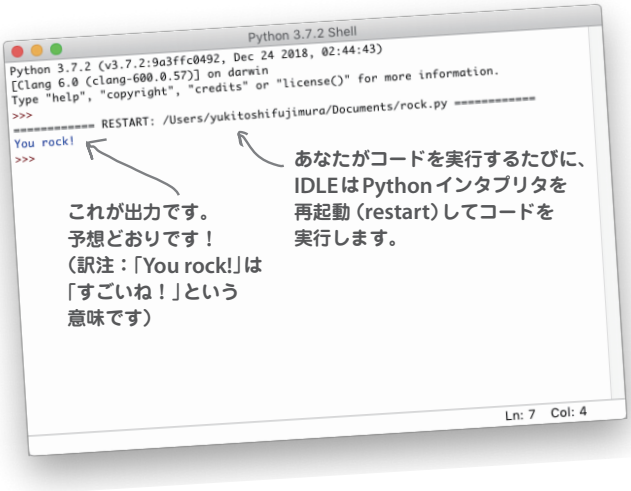
「実行」　「モジュール実行」

メニューの［Run］から［Run Module］を選ぶとコードが実行されます。コードが保存されていなければ、IDLEから保存するように言われます。



おめでとう! 最初のPythonプログラムが書けました!

Pythonをインストールして、IDLEを使って短い実際のコードを入力して初めてのプログラムを実行することができました。次は、いきなり複雑なコードではなくてもよいのですが、何かを始めなければいけません。幸い、ここまでで準備はすべて整っているので、いつでも本格的なビジネスアプリケーションのコードを書き始めることができます。



これが出力です。予想どおりです！（訳注：「You rock!」は「すごいね！」という意味です）

あなたがコードを実行するたびに、IDLEはPythonインタプリタを再起動（restart）してコードを実行します。

注目！

「You rock!」以外に表示されたものは？

コードの記述とテストは、間違いが起こりやすい工程です。1回でうまくいかないのは当たり前です。われわれ開発者は、常にコードの間違いを修正しています。普段は次のことを心がけてください。

- `SyntaxError: invalid syntax`のようなエラーが出たら、かっこが片方ないといった句読記号の間違いがないか探します。IDLEの場合はコードがハイライトされるので、すぐに場所が特定できます。
- `NameError: name 'prin' is not defined`のようなエラーが出たら、スペルが間違っていないかといったタイプミスを調べるようにします。この例では`print`の`t`が抜けています。
- `SyntaxError: EOL while scanning`のようなエラーが出たら、通常は文字列のシングルクォートが片方ないという意味です。`'You rock!'`のように2つのシングルクォートで囲まれていることを確認します。

素朴な疑問に答えます

Q：なぜ［Run Module］を使ってコードを実行するのですか?

A：Pythonでは、コードが保存されているファイルを**モジュール**と呼びます。つまり、［Run Module］は「ファイル内のすべてのPythonのコードを実行する」という意味です。また、モジュールはコードの高度な構成手段でもあります。詳しくは3章で説明します。

Q：入力と出力とは具体的には何なのですか?

A：現時点での入力と出力はまだ簡単なものだけです。出力はプログラムが作成してPython Shellウィンドウに表示するテキストのことです。入力はプログラムがPython Shellウィンドウから受け取るテキストのことです。さらに広い意味では、マウス入力、タッチ入力、グラフィックスや音声出力などのあらゆる種類の入力と出力が可能です。

Q：`print`はテキストをユーザに出力する方法だと理解しています。なぜ「print」という名前なのですか? 初めて見たときには、プリンタで印刷するための何かかと思いました。

A：昔は画面よりもプリンタに出力することの方が多かったのです。その当時は「print」という名前が現在よりしっくり来ていました。Pythonは比較的若い言語なので`print`という名前でなくてもよかったのですが、`print`は従来から多くの言語で画面に出力する手段だったからです。Python Shellウィンドウには、`print`を使って出力します。一方、Python Shellからユーザ入力を取得するのが、先ほどの例で登場した`input`です。

Q：Pythonから出力する手段は`print`だけですか?

A：いいえ。最も基本的な方法が`print`であって、ほかにもあります。コンピュータもプログラミング言語も出力（および入力）は得意です。Pythonも例外ではありません。Pythonをはじめ、ほとんどの言語では、Webページ、ネットワーク、ストレージデバイス上のファイル、グラフィックスデバイス、音声デバイスなどに出力することができます。

Q：なるほど。では、`print('hi there')`と入力すると、一体何が起こっているのですか?

「やあ」という意味です。

A：Pythonに組み込まれた出力機能を使います。より具体的に言うと、`print`関数に、クォート（引用符）内のテキストをPython Shellに出力するように指示しています。関数やテキストとはどういうものかについては後で詳しく説明しますが、いまの段階ではPython Shellに出力する際は`print`関数を使うことを覚えておいてください。

ブール型の真実

今週のインタビュー：

本気なの？

Head First：ようこそ、Pythonさん！ あなたが一体何者なのかその正体を解き明かすのを楽しみにしていましたよ。

Python：お招きいただき光栄です。

Head First：さて、コメディグループにちなんで名付けられ、初心者用の言語として有名なPythonですが、正直、あなたのことを本格的な言語とみなしてよいものでしょうか？

Python：私はありとあらゆるものに使われています。シリコンチップ生産ラインの管理から有名な映画製作をサポートするアプリケーション（いままで黙っていましたが、「ジョージ・ルーカス」とか）、航空管制システムのインタフェースとかにね。他にもありますよ。本格的だと思いませんか？

Head First：なるほど。では、そのように本格的な言語なのに、一体どうして初心者が簡単に使えるのですか？ つまり、あなたがいま挙げたプロジェクトはどれも複雑そうですよね。普通、そのような複雑なプロジェクトを行うには、本格的で複雑な言語が必要だと思うじゃないですか。

Python：私は初心者だけでなく**エキスパート**にも高く評価されているんですよ。その理由の1つは、コードがとても単純で読みやすいからなんです。例えばJavaのコードを見たことがありますか？ Javaでは「Hello World ！」と表示させるだけでも大変ですが、Pythonなら1行で済むんです。

Head First：なるほど。読みやすいのですね。それは素晴らしい。でも、一体それが何だと言うのですか？

Python：せっかくJavaの話が出てきたので、簡単な例を紹介しましょう。例えば「Hello!」と表示させたいとき、Javaでは次のように書きます。

```
class HelloWorldApp {
  public static void main(String[] args)
{
     System.out.println("Hello!");
     }
  }
```

なんと5行も必要です。それに読みにくいと思いませんか？ 初心者にとっては特につらいでしょうね。一体なぜこんなに必要なのでしょうね。本当に全部必要なのでしょうか？ では、私の場合を見てください。もちろんPythonで書いてますよ。

```
print('Hello!')
```

こっちの方がずっとシンプルで読みやすいでしょう？ それにこのコードが何をするか、誰にでも伝わります。これはほんの一例ですが、Pythonはわかりやすくて英語に似ていて一貫性があるという印象を持たれているんです。

Head First：一貫性があるとはどういうことですか？

Python：一貫性とは、意外性があまりないというのが1つです。つまり、言語の一部を理解したら、他の部分もだいたい予想できて、そのとおりに使えることが多いのです。しかし、すべての言語がそういうわけではありません。

Head First：以前におっしゃっていたことに戻りたいのですが、航空管制、チップ製造、スペースシャトル用のメインソフトといった例を挙げていました。これはどれもとても工業的で特殊な目的のように思えます。Pythonがわれわれの読者にとって最適な言語であるのかよくわからないのですが。

Python：スペースシャトルなんて言っていませんよ。でも航空管制とチップ製造はそのとおり。**本格的**だと思っていただけそうな例として挙げました。Pythonは、Webサイトの作成、ゲームの製作、デスクトップアプリケーションの作成などでよく使われています。

Head First：話題を変えてもいいですか？ 今メモを渡されたのですが、情報筋によると、Pythonには**2つのバージョン**があって、さらにはなんと、何て言えばいいんだろう、**それぞれ互換性がない**ということなのですが。一体どこに一貫性があるというのですか？

Python：何事も同じですが、言語は成長し進化するものです。確かにPythonにはPython 2とPython 3の2つのバージョンがあります。Python 3にはPython 2にはない新しい機能がありますが、下位互換性を取る方法がありますよ。ここで読者に説明させてください。

Head First：残念ですがもう時間がありません。次回の奇襲、えっと、つまりお話しする機会を楽しみにしています。

Python：ありがとうございます。次回も喜んで…、たぶん。

新製品のフレーズ・オ・マチック
（Phrase-O-Matic：自動フレーズ作成機）
を試せば、上司やマーケティングの連中
みたいに弁が立つようになるだろうよ。

では、まじめに実際のビジネスアプリケーションをPythonで書いてみましょう。この フレーズ・オ・マチックのコードを見てください。よくできていると思いませんか？

ほっと一息

はい、本当に一息入れてください！ いまの段階では全体を少しずつ理解することが大事です。コードを1行ずつ読んで、何をしているかの説明を確認し、脳に記録しましょう。右のコードには「コメント」も追加されています。コメントとは#記号の後に書かれたメモ的なもので、コメントから何をするコードなのかがはっきりします。コメントは、参考になる擬似コードと考えてもよいでしょう。コメントについて詳しくは3章で説明します。コードを何となく理解できたと思ったら、次のページに進んでください。1行ずつ詳しく説明していきます。

```
# ランダム機能を使うことをPythonに知らせるために、
# randomモジュールをインポートする
import random

# 3つのリストを作成する。動詞のリスト、形容詞のリスト、
# そして名詞のリスト
verbs = ['Leverage', 'Sync', 'Target',
         'Gamify', 'Offline', 'Crowd-sourced',
         '24/7', 'Lean-in', '30,000 foot']
adjectives = ['A/B Tested', 'Freemium',
              'Hyperlocal', 'Siloed', 'B-to-B',
              'Oriented', 'Cloud-based',
              'API-based']
nouns = ['Early Adopter', 'Low-hanging Fruit',
         'Pipeline', 'Splash Page', 'Productivity',
         'Process', 'Tipping Point', 'Paradigm']

# リストから動詞、形容詞、名詞をそれぞれ1つ選ぶ
verb = random.choice(verbs)
adjective = random.choice(adjectives)
noun = random.choice(nouns)

# 単語を「つなぎ合わせて」フレーズを作る
phrase = verb + ' ' + adjective + ' ' + noun

# フレーズを出力する
print(phrase)
```

動詞のリスト

形容詞のリスト

名詞のリスト

フレーズ・オ・マチック

このプログラムを簡単に説明すると、3つの単語リストからランダムにそれぞれ単語を1つの選び、その単語を組み合わせて（次の新規事業のスローガンにふさわしい）フレーズを作成して出力します。このプログラムに理解できないところがあっても大丈夫です。600ページもある本のまだほんの25ページ目に来たばかりです。いま大切なのはコードにある程度慣れることです。

1. import文は、Pythonのrandomモジュールにある追加の組み込み機能を使うことをPythonに知らせます。import文は、コードが実行できることを拡張すると考えてください。この例では、ランダムに選ぶ機能を追加します。importがどのように機能するかについては、後で詳しく説明します。
2. 次に、リストを3つ用意します。リストの宣言は簡単です。次のように、リストの各要素をクォート（'）で括り、全体を角かっこで囲みます。

```
verbs = ['Leverage', 'Sync', 'Target',
         'Gamify', 'Offline', 'Crowd-sourced',
         '24/7', 'Lean-in', '30,000 foot']
```

   リストに名前を付けておくと（ここではverbs）、後でコード内から参照できます。
3. 次に、リストからランダムに単語を1つ選びます。ここではリストから1つの要素をランダムに選ぶrandom.choiceを使います。後で参照できるように、それぞれの要素を対応する名前（verb、adjective、noun）に代入します。
4. 次に、3つの要素（動詞、形容詞、名詞）をつなぎ合わせてフレーズを作ります。Pythonではプラス記号（+）を使います。また、単語と単語の間に空白も挿入します。空白がないと、「Lean-inCloud-basedPipeline」のようになってしまいます。
5. 最後に、printを使ってPython Shellにフレーズを出力します。これでマーケティングフレーズの一丁上がりです！

Pythonから作られた次期新規事業のスローガン

24/7 Freemium Productivity
（フリーミアム生産性の年中無休化）

Lean-in Hyperlocal Splash Page
（超地域密着型のスプラッシュページに乗り出す）

Gamify Siloed Early Adopter
（サイロ化した新し物好きのゲーム化）

Offline API-based Process
（APIベースのプロセスのオフライン化）

Crowd-sourced Cloud-based Pipeline
（クラウドベースのパイプラインのクラウドソース化）

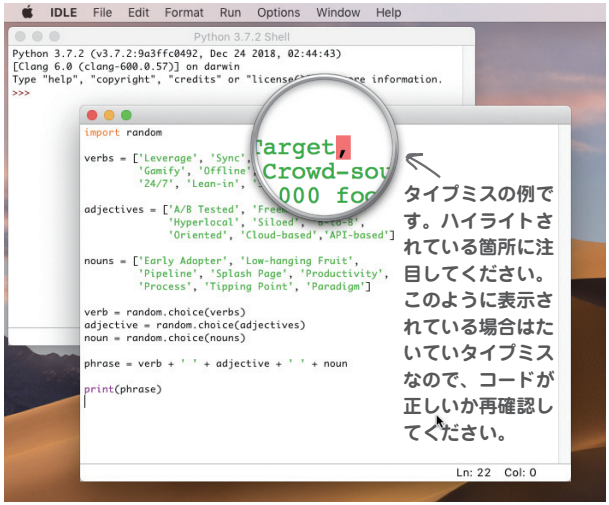
random.choiceは、Pythonの標準ライブラリの関数です。

「結合」とも呼びます。

コンピュータ科学者は、テキストをつなぎ合わせることを「連結」と呼びます。この言葉は便利なので、この本でもちょくちょく登場します。

コンピュータにコードを読ませる

あなたはすでにプログラムを1つIDLEに入力し実行できていますが、ここでもう一度、手順を順番に説明しましょう。メニューの[File]から[New File]を選択し、新しいファイルを作成します。次に、24ページのコードを入力します。



.,;!?: のような文字のことです。

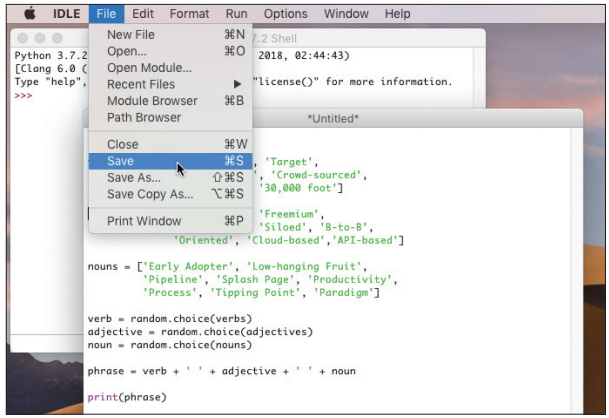
単語と句読記号には特に注意を払ってください。詳しくは後の章で取り上げますが、いまの段階では慣れておくだけで十分です。

IDLEは一般的なエラーをハイライトしてくれます。エラーの種類によって、コード入力中にハイライトされるものもあれば、実行時にハイライトされるものもあります。エラーがあったら、コードを再確認して修正しましょう。

タイプミスの例です。ハイライトされている箇所に注目してください。このように表示されている場合はたいていタイプミスなので、コードが正しいか再確認してください。

ここでは意図的にエラーを入れてIDLEの反応を説明しました。24ページのコードを忠実に入力すれば、このエラーは表示されません。

コードを最後まで入力したので、保存します。IDLEメニューの[File]から[Save]を選び、ファイル`phraseomatic.py`として保存します。



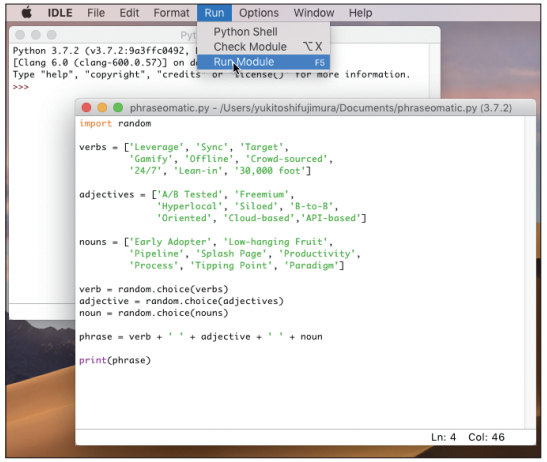
IDLEはコードの役割によってコードに色を付けます。

さらにリターンキーを押した後に必要に応じてインデントを挿入してくれます。

空白や改行を追加すると、さらにコードが読みやすくなります。Pythonはほとんどの空白や改行を無視するので実行には影響しません（例外については9章で取り上げます）。

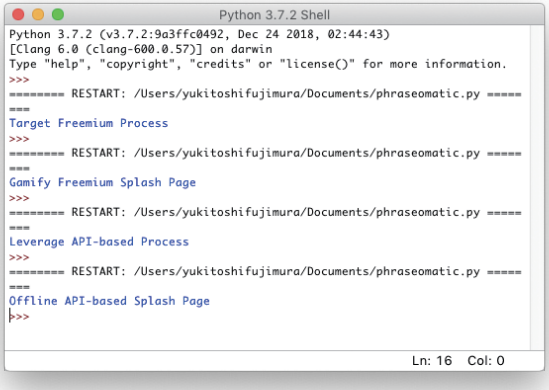
フレーズ・オ・マチックを実行してみましょう。もう一度順を追って説明しますね。まずコードを保存して、メニューの[Run]から[Run Module]を選んで実行してください。Python Shellウィンドウで出力を確認すると、ほら、次期新規事業のスローガンが作成できています！

[Run]から[Run Module]を選んでコードを実行します。コードが保存されていないと、保存するようにIDLEから言われます。



A/Bテストしたスプラッシュページをゲーム化するよ！

フレーズ・オ・マチックを何度か実行しました。出力を確認してください。



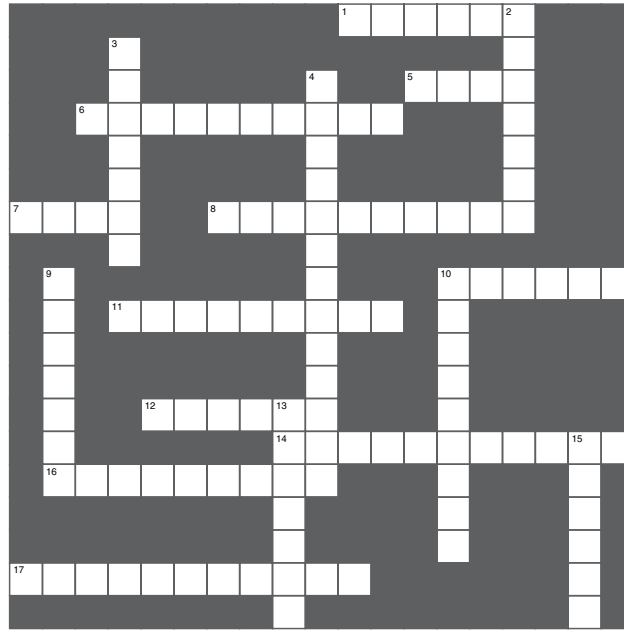
フレーズ・オ・マチックを再実行するには、まずコードのウィンドウをクリックし、もう一度[Run]から[Run Module]を選びます。

重要ポイント

- コードを書くには、まず問題を解決する簡単な一連の手順に問題を分解する。
- この一連の手順を問題解決のためのアルゴリズム、または少しくだけてレシピと呼ぶ。
- この手順は「文」という形式を取り、簡単なタスクの実行、判断、コードの一部を繰り返すことによるアルゴリズムのフロー制御を実行できる。
- コンピュータ的な考え方は、コンピュータ科学から生まれた問題解決の考え方である。
- コーディングは、アルゴリズムの手順をコンピュータが実行できるプログラミング言語に変換する行為のこと。
- アルゴリズムは、人間が読みやすい擬似コードで表してから実際のプログラミング言語に変換することもある。
- プログラミング言語は、コンピュータにタスクを示すために特別に開発された専用言語である。
- 英語など、人の言葉はかなり曖昧なため、プログラミング言語としては不十分。
- 多くのプログラミング言語があり、それぞれ利点と欠点がある。
- Pythonという名前はヘビからではなく、開発者がコメディグループのモンティ・パイソンを好きだったことに由来する。
- 初心者でも経験豊富なプログラマでも、Pythonのわかりやすく一貫性のある設計を高く評価している。
- Python 2とPython 3という異なるバージョンがある。この本ではPython 3を使う（ただし、その違いはほとんどの場合ごくわずか）。
- Pythonのコードはインタプリタが実行する。インタプリタは、Pythonのコードをコンピュータが直接実行できるマシンコードに変換する。
- Pythonには、専用のIDLEというエディタがある。
- Pythonでは、空白や改行を使ってプログラムを読みやすくする。
- `input`と`print`はどちらもコマンドラインで使うことができる、Pythonの入力および出力の関数。

コーディングクロスワード

さあ、右脳を働かせましょう。
よくあるクロスワードパズルですが、答えの単語はすべて1章で登場しています（クロスワードの答えはアルファベットで記入してください）。



ヨコのカギ

1. アルゴリズムに対する一般向けの用語。
5. PythonのIDE。
6. インタプリタやコンパイラへの入力。
8. 人間が読めるコード。
10. Flying _____
11. レシピの技術的な言い方。
12. 最高の初心者用言語の1つ。
14. コンピュータが直接実行できるコード。
16. Pythonはこの種の言語。
17. Pythonはその1つ。

タテのカギ

2. プログラムの実行。
3. ソースコードの別名。
4. この本で教える考え方の種類。
9. コードには曖昧すぎる。
10. Pythonの支援者。
13. Head Firstレストランがこれを出す。
15. Pythonを使っている企業。

エクササイズの答え

実際のレシピは**何をすべきか**という手順だけではなく、特定の料理を作るために使うオブジェクト（道具、調理器具、材料など）も示します。釣りのレシピで使うオブジェクト（対象物）は何でしょうか。

1. 釣り針に餌を付ける。
2. 池に釣り糸を垂れる。
3. 浮きが水中に沈むまで待つ。
4. 魚を釣り上げる。
5. 釣りが終わったら家に帰る。終わりでなければ、手順 1 に戻る。

自分で考えてみようの答え

あらかじめ理解しておいてもらいたいことがあります。コンピュータは**指示したとおり**にしか動作しません。それ以上でも以下でもありません。2ページの釣りのレシピを考えてください。ロボットになってこの指示に正確に従うとしたら、このレシピで本当にうまくいくでしょうか。

- ☑ A. 魚がいなければ、長時間（もしかしたら永久に）釣りをすることになってしまう。
- ☑ B. 釣り針から餌が外れても気付かず、付け直すこともない。
- ☑ C. 餌がなくなった場合は？

「A～Eの全部」が答えのようです。

- ☑ D. 魚を釣り上げたら何をするかを指定したか？
- ☑ E. 釣り竿は？
- ☑ **F. 上手な投げ入れとは具体的にどのようなものですか？ 餌がスイレンの葉に乗っかってしまったら、もう一度投げ入れる必要がありますか？ 通常は浮きが水中に沈んだら、魚を「ひっかけて」から釣り上げます。ここではそれについて何も述べていません。釣りが終わったかどうかはどのように判断するのでしょうか？ 時間ですか？ 餌がなくなったときですか？ 別の条件でしょうか？ このレシピには推測することがたくさんあります。きっと上に挙げたほかにも規定されていないたくさんの指示を思い付くでしょう。**

コードマグネットの答え

Head Firstレストランの卵を3個使ったオムレツの(~~レシピ~~)アルゴリズムを忘れないように冷蔵庫のドアに貼っていましたが、誰かが順番を崩してしまいました。マグネットを正しい順番に直し、アルゴリズムを使えるようにしてください。なお、Head Firstレストランのオムレツには、プレーンとチーズの2種類があります。

これが元どおりにしたマグネットです！

フライパンを火にかける

3個の卵を割ってボールに入れる

卵がよく混ぜ合わされていなければ:

卵を溶く

卵をフライパンに流し入れる

卵に火が十分通るまで:

卵をかき混ぜる

注文がチーズオムレツなら:

上にチーズを加える

フライパンを火から下す

卵をお皿に移す

客に提供

ほかにも正しい並べ方はいくつかあるでしょう。自分の答えと違っていても、この本の答えを理解し、自分の答えが論理的に筋が通っていることを確認すればOKです。

最初に必要なものを準備します。フライパンを火にかけ、卵を割ります。

そして、卵をよく溶きます。

卵を溶く部分をインデントしています。卵が混ざり切るまで溶き続けるという意味です。このことを別の方法で示してもOKです。

次に火にかけておいたフライパンに卵を流し入れます。

そして、十分火を通します。

卵をかき混ぜる部分をインデントしています。火が通るまで卵をかき混ぜ続けるという意味です。このことを別の方法で示してもOKです。

注文がチーズオムレツなら、チーズを追加します。

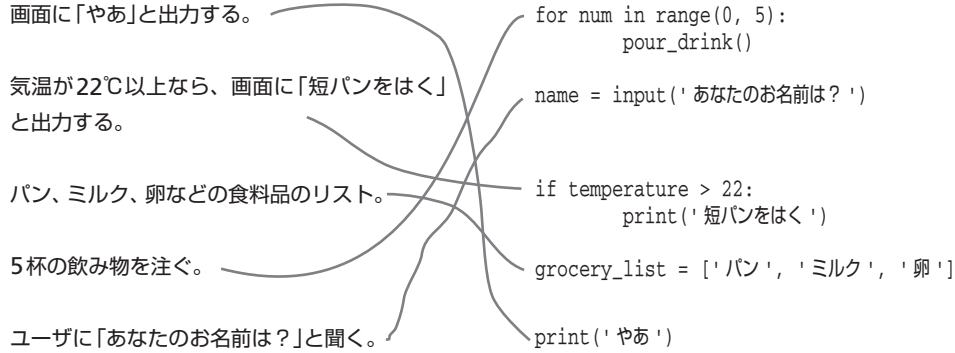
チーズの追加もインデントしています。注文がチーズオムレツの場合にだけ追加するからです。

どちらの場合でも、フライパンを火から下ろしてお皿に移します。

最後に、オムレツを客に出します。

何て言う？の答え

左側は人間が使う言語で書いた文、右側はプログラミング言語で書いた文（コード）です。人間の使う文と、それに対応するコードを線で結んでください。1番目だけは線を引いておきました。



コーディングクロスワードの答え