CodeMonkey

プログラミング学習ゲーム コードモンキー

カリキュラムガイド

【PART-A：チャレンジNo.0-100】

サンプルページ

目 次

イントロダクション

指導者の皆さんへ

プログラミング学習教材に CodeMonkey を選んでいただきありがとうございます。皆さんと一緒に、教育者として、子どもたちの生活を豊かにしていきましょう。かわいいキャラクターときれいなアニメーションの楽しいチャレンジ、CodeMonkey のユニークなユーザー・エクスペリエンスに満ちあふれたオンライン教材。21 世紀社会を生きていく子どもたちに必須となるコンピュータサイエンスの基本知識を、うまく身につけられることを願って CodeMonkey を開発しました。また、指導される皆さんにコンピュータサイエンスの経験や予備知識がなくても、だれでもが、子どもたちがプログラミング知識を学べるように指導できること、また皆さんも子どもたちと一緒に学べるようにと、このカリキュラム・ガイドを作りました。

カリキュラム・ガイドは次からできています。

- 始めるにあたってのイントロダクションとグループ管理の概要
- 20 レッスンのカリキュラム・ガイド　（1 レッスン ＝ 45 分を想定）
- CodeMonkey のリファレンス・カード
- メインモードとスキルモード 全チャレンジの解答例
- サポート情報

教室で CodeMonkey を使うにあたり、次のことを理解しておいてください。

CoffeeScript（コーヒースクリプト）

- CodeMonkey は、CoffeeScript というプログラミング言語を使います。これは JavaScript に変換されるプログラミング言語で、JavaScript と同様に Web アプリケーションの作成に一般に使われています。
- プログラミング言語として CoffeeScript を使うことにした理由はいろいろありますが、その一番の理由は、ほかのプログラミング言語と比べてとてもフレンドリーな文構造で、自然な英語を書くのに似ているからです。
- CoffeeScript を詳しく知りたい方は、https://ja.wikipedia.org/wiki/CoffeeScript　を参照してください。

クラスをはじめる前に

- 最初のいくつかのチャレンジをまず自分でやってみてください。できる人は全部やってみましょう。そうすることで、クラスの生徒たちをうまくヘルプできるようになります。
- CodeMonkey は HTML5 で作られています。ブラウザーだけあればよいので、教室のインターネットに接続された PC で問題なく動くはずです。
- CodeMonkey のプロになる必要はないですが、そうなるとチャレンジがとても面白くなります。
- 生徒の数だけ充分な PC がなければ、生徒たちがペアで PC を使うようにしてください。一人一台の PC があっても、となり同士で相談しても良いようにしてください。その時にはどうやって問題を解けば良いか、ディスカッションして一緒に協力し合うように仕向けてください。ペアで取り組むことで、ほかの人と解決のプロセスをディスカッションすることになり、学習効果がより高まることが期待できます。
- 指導者は、毎回のクラスをはじめる前にその日のレッスン計画に目を通し、その日のトピックをしっかりと理解してください。
- このガイドを、必要に応じていつも見直すようにしてください。
- あなたの学校で Google や Office 365 などのログイン手段を持っていない場合は、CodeMonkey の ID を、このステップガイドにしたがって作ってください。ガイドには生徒の登録とグループ管理が含まれています。生徒のユーザー名とパスワードのリストを今後のためにきちんと管理しておいてください。

ペアあるいは数名で一緒にチャレンジをすることは、とても良い効果を出します。一人一台 PC を使える環境であっても、隣同士で相談し合える雰囲気を作りましょう。

クラスで

- 最初のクラスでは、CodeMonkey にどうやってログインするかを、生徒に教えてください。
- その日のレッスン計画にしたがってください。でも、それにとらわれず状況に応じて柔軟に対応してください。あなたから生徒に、たくさんの質問をして生徒たちが自らたくさん話し出すような雰囲気づくりに心がけてください。
- 生徒たちが問題に取り組んでいるあいだ、教室内を歩き回って生徒たちのいろいろな質問に答えてあげてください。
- ついて行けずに遅れている生徒たちや、うまく行っていないペアには、特に注意をはらってください。
- 特に、「星の数」に注目するように生徒たちに指示してください。
- ときどき生徒たちに、別のソリューションがないかを考え、それをディスカッションするようにガイドしてください。
- 教師のダッシュボードでは、生徒たちが実際に書いたコードが見られます。クラスのディスカッションを盛り上げるために、この機能を使って実際に書かれたコードを活用して下さい。でも、だれがそのコードを書いたかは分からない様に気をつけてください。
- チャレンジのいくつかは、それまでに学んだことを確認する場となっています。そこでは、生徒にゼロからコードのすべてを書かせることで、確かに学んだということを確認できる様にしています。

スキルチャレンジ

- コーディングできるようになるには、とにかく練習をすることです。そうすることでコードがどんどん書けるようになっていきます。でも、コーディングがうまくなることが、必ずしもプログラミングを学習し終えた、ということになる訳ではありません。それは単に、より高度なチャレンジに立ち向かえるようになったと、いうだけに過ぎません。
- まさにその理由のために、コードモンキーの「スキルチャレンジ」をつくりました。コードモンキーであれ実生活であれ、より高度なトピックに対応できるようにそれぞれのチャレンジの内容をつくりました。

- スキルチャレンジは、生徒がチャレンジをクリアするたびにアンロックされて行き、生徒が学んだだトピックをさらに深めて学べる、さらなる機会を提供するものです。画面右上の [スキルモード] タブをクリックすると、今どのステージのスキルモードにいて、どれにチャレンジできるかをマップで確認できます。
- 生徒の学年やその他の理由で、このガイドのレッスンの一部は 45 分かからずに、早く終わることがあるでしょう。 その場合は、先のレッスンには進まず、生徒たちには、アンロックされたスキルチャレンジを解かせてください。また、スキルチャレンジは、皆より早くチャレンジをクリアできてしまった生徒のために使うこともできます。

生徒が困っている時に

Gordo（ゴード）

- 生徒が直面する問題のほとんどは、指示やコード自体を正しく読んでいないことが原因です。指示やコードを、よく注意をしてしっかり読むように、生徒に指導してください。 画面左下のサル（名前は Gordo）をクリックすると、指示をもう一度表示できます。
- いくつかのチャレンジは、コードにわざと間違いを入れてあって、生徒がそれに気付いて修正する必要のある「デバッグ（バグ取り・虫取り）」課題になっています。生徒には、とにかくまず [RUN] を押して、そこに書かれているコードがどんな動きをするかを見てから、それぞれのチャレンジを解くようにさせてください。 これは、どこに問題があるかを明らかにするのに役立ちます。
- 生徒を励まし、ポジティブに力づけてあげてください。「良いじゃないの、その調子、がんばってチャンレンジしていこう！」という風に。
- 「先生も答えは分からないの、一緒に考えよう！」と対応するのも良いことです。生徒がどこにつまずいているのか分からない時は、それをとても良い学習レッスンの素材としてクラスで利用してみましょう！「テクノロジーは必ずしもいつも私たちが望むようにしてくれるわけではありません。みんなで一緒に考えましょう。私たちは皆、一緒に学ぶ仲間たちです！」
- 教師のダッシュボードで、すべてのチャレンジの星３つの解答例を、いつでも参照することができます。また本書の巻末に、チャレンジマップと、メインモードとスキルモードの全解答例を掲載しました。これは、コードモンキーの全体像を指導者の皆さんにご理解いただき、自身の勉強と指導

の準備に活用頂くことを願って掲載することにしたものです。星１つでも２つでも生徒が一生懸命考えてバナナをとったコードはすべて立派な解答です。掲載した解答と同じコードを一発で書けた生徒が優秀ということでは決してありません。そのようにこの解答例が使われるのであれば、それは私達が意図する使い方ではなく、とても不本意なことです。正解は一つだけではありません。間違いの繰り返し、遠回りすること、それらのプロセスすべてが生徒たちの、そして指導者の皆さんの学びの場です。私たち CodeMonkey チームの想いを理解して活用していただけることを願っています。

原著のカリキュラムガイドには回答例は掲載されていません。

日本の指導者の方々からの要望も踏まえた上で、日本語版には掲載することにしたものです。

早く進んでいく生徒にはどう対応すればい良いでしょうか？

- すでにクリアしたけれど星１つか２つしか取れていないチャレンジに戻って、すべて星３つを取らせましょう。
- それができたら、同じトピックでさらに練習ができる「スキルチャレンジ」をやらせましょう。
- それも終わってしまった生徒には、よく分からず困っていたり、クリアするのに手こずっているクラスメートをヘルプするように頼みましょう。

クラスが終わったら

教師用のダッシュボードをつかって

- 生徒の進捗状況をモニターする
- 生徒の書いた実際のコードを見る
- それぞれのチャレンジで生徒が取ったスコアを見る
- やり直しが必要なトピックがあるかどうかを判断する
- クラスの成果の統計を見る
- それぞれのチャレンジの星３つのソリューションを見つけ出す

最後に、コードモンキーの楽しさを思い出してください！ 生徒たちには、コーディングが楽しくて、それが退屈で怖いものではないことを分かってもらうのが一番大切です。

質問がありましたら、いつでも codemonkey_support@japan21.co.jp にご連絡ください。
Twitter や Facebook でもお待ちしています。

- Twitter: @codemonkey_jp　　https://twitter.com/codemonkey_jp
- Facebook: codemonkey.jp　　https://www.facebook.com/codemonkey.jp/

では、がんばって行きましょう！

CodeMonkey チーム一同

レッスンガイドの構成

コードモンキーのチャレンジは全部で 200 まであますが、本書はその前半の No.0 から No.100 までを使うレッスン（Part-A）です。

CodeMonkey の対象年齢は９歳からを目安としていますが、幼稚園児でもほぼ全部クリアした子もいれば、苦戦しながらも楽しんでいる大人からお年寄り、親子でワイワイやりながら楽しむ家族まで、幅広く皆さんに楽しんでいただける、プログラミング学習ゲーム教材です。

このカリカリキュラムガイドは、小学校中学年から中学校のクラスで行うことを想定して記述しています。しかし、本書をベースに説明方法やアクティビティーの内容、ディスカッションの話題などを対象の生徒と目的に合わせて修正することで、幅広い年齢層に対応可能です。

１レッスンは 45 分を想定しています。、各レッスンガイドには、そのレッスンの概要を示すまえがきと、クラスで実施するレッスンの内容を記載しています。一回のレッスンは３つのパートから構成されています。

まえがき

- 各レッスンのタイトルと、そのレッスンで行うチャレンジの番号。
- レッスンの概略紹介
- **目的：** レッスンで生徒たちが学ぶこと
- **コンポーネント：** このレッスンを構成する主要な要素。プログラミングとして学ぶ**命令**、新しく学ぶ**用語**、レッスンのハイライトや**主演**となるものなど。
- **レッスンの配分時間：** クラスで実施する 45 分のレッスン３つのパートの時間配分（目安）。棒グラフで視覚的に表示。

Part １： イントロダクション（*introduction*）

授業の導入部分です。新しく学ぶ内容の説明や、その導入となるアクティビティー、前回の振り返り、ディスカッションなどで、指導者が中心に生徒と一緒に行います。

- **説明：** 操作方法や、レッスンで学ぶことを解説。
- **ディスカッション：** 生徒たちに考えたり議論させたりすることで、これから行うレッスンへの関心を高めたり予備知識を得ることで、スムーズな導入を可能にするために行う。
- **アクティビティー：** 生徒からボランティアを募ってロールプレーをさせるなどして、これから学ぶプログラミングの位置づけや意味などを理解したり気づかせるために行う。
- **ウォークスルー：** 指導者がコードモンキーの画面を操作し、それを生徒たちに見せながら説明や解説をして、生徒たちの理解を深めるために行う。
- **レビュー：** 以前に習ったことの復習や振り返りを行い、知識の定着を図る。

Part ２： レッツゴー！（*Let's Go*）

学生たちが各自でコードモンキーを操作して、コードを書きバナナをキャッチしてチャレンジをクリアしていく、授業のメインとなるところです。`

- **ログイン：** 生徒たちが各自のユーザーID とパスワードで、コードモンキーにログインする。
- **説明：** 教師が操作の説明や事前注意などをする。
- **プレータイム：** 生徒たちが各自でコードモンキーを操作し、指定されたチャレンジをクリアする。
- **ウォークスルー：** 生徒は手を動かすのを止め、指導者がコードモンキーの画面を操作してそれを見せながら説明や解説をして、生徒たちの理解を深めるために行う。
- **ディスカッション：** 生徒への質問や問題提起を中心に、対話形式でディスカッションして生徒たちの理解を深める。
- **アセスメント：** 特定のチャレンジは、生徒がコードをゼロからすべて自分で書かせることで、そ

れらを覚え理解してるかを確認する、アセスメント（評価）チャレンジとなっています。

- **プラクティス：** プレータイム中に、他の生徒より早く終えてしまう生徒たちに習慣として実践してもらいたいこと。

Part 3： ディブリーフィング（*debriefing*）

今日のレッスンで行ったことや学んだことなどの振り返りと議論を、教師と生徒で一緒に行うことでフィードバックを提供し、学習内容の固定化と、自律的な思考と将来の行動の変容を促すものとして行います。

- **レビュー：** レッスンで学ぶことを期待された内容を理解し、それが使えるようになったことを確認する。
- **ディスカッション：** 生徒への質問や問題提起を中心に対話形式でディスカッションして、生徒たちの学習内容の定着化を深める。
- **ウォークスルー：** 選んだチャレンジを教師が操作し、生徒に画面を見せながら説明し、それをもとに質疑応答などを行って、理解の確認と定着化を促進する。
- **宿題：** 学んだことの定着をより促進するために、次回までに各自でやってきてほしい課題を出す。

準備は整いました
さあ、子どもたちと楽しいプログラミングの世界に出かけましょう！

序章

レッスン 1 - さあ始めましょう *Let´s Get Started*

チャレンジ No. 0 – 5

はじまりの森

このレッスンでは、魅惑的なコンピュータの世界とコードモンキー・プラットフォーム（ゲームをする基本となる画面）について紹介します。 生徒の中には、「コーディング」や「プログラミング」という用語に精通していて、コンピュータを快適に操作できる人もいます。 一方で、プログラミングを学ぶことは苦手でプレッシャーだ、辛くて嫌だという人もいるでしょう。教育者として私たちの目標は、コンピュータサイエンスを含むさまざまな科目の学習や探索・探求（explore・エクスプローラー）の手助けをすることです。 生徒たちが間違いを繰り返すことから学び、新しいものを創造するための基礎知識を築き上げていくことができる、そんな環境を提供しサポートしたいと考えています。

目的

このレッスンで、生徒は以下のことを学びます。

- 「コーディング」と「コンピュータプログラミング」の定義
- コードモンキーのプラットフォームに慣れる
- コードモンキーのチャレンジ No. 0 - 5 をクリア

コンポーネント（レッスンに出てくる主要な構成要素）

- 命令： `step` 、`turn`
- 用語： チャレンジ（challenge）、CoffeeScript

レッスンの時間配分

Part 1: Introduction 10	Part 2: Let's Go 25	Part 3: Debriefing 10

Part 1： イントロダクション（授業の導入部） 【 10 分 】

ディスカッション	2 分
1. コンピュータを使ったことがある人、何人いるかな？ 2. コンピュータで、プレゼンテーション・お絵かき・ゲームなどを作ったことがある人は？ 3. クラスの 2-3 人の生徒に、どんなものを作ったことがあるか話してもらいましょう。	
説明	1 分
みんなの好きなアプリケーションやゲームがコンピュータで動くためには、コンピュータに命令をすることが必要です。コンピュータは私たちが言ったとおりに動くだけで、自分で考えることはできません。コンピュータにこうしろと命令を与えることを、コンピュータプログラミングまたはコーディングといいます。	
アクティビティー	3 分
命令する、ということを理解するために、生徒とちょっとしたゲームをします。教室のどこかに何か物を置いてください。そして生徒に、あなたが立っている場所からそこにどうやって行けばよいか、あなたに指示をするよう頼んでください。 ● 生徒はどんな指示をしましたか：（例）前に進む、右に曲がる、左に曲がる	
ディスカッション	2 分
コンピュータは人間と同じ言葉がわかりますか？ コンピュータには独自の言語があって、人間の言葉は理解できません。HTML、JavaScript、Python などと呼ばれるものが、コンピュータが理解できる言葉の一例です。それぞれの言葉は違っていますが、	

どれも共通するところがあります。それは、特定の考え方があること、明確な指示が必要なこと、そして構造があることです。 コンピュータの言語を学ぶことは、基本的には新しい外国語を学ぶことと同じです。

注目	2 分

今日から皆さんは、コードモンキーというゲームを通じて、プログラミング（コーディング）の原則を体験し学習します。私たちが使うのは CoffeeScript と呼ばれるプログラミング言語ですが、この言語を覚えることが目的ではなく、それを手段として使って、プログラミングの楽しさ、コンピュータサイエンスの基礎や、論理的思考などを会得するのがゴールです。

生徒たちと一緒に、コードモンキーの紹介動画 https://youtu.be/o3geZ_0r_3Q を見ましょう。また、コードモンキーでプログラミングを学習しているイスラエルの小学生たちの紹介ビデオ https://youtu.be/NS0Z_V4cUEE も見てください。

Part 2： レッツゴー！（授業のメイン） 【 25 分 】

説明　　：　教師が説明	3 分

コードモンキーのウェブサイトに移動します。 playcodemonkey.com
コードモンキーアカウントにログインする方法を生徒に教えます。

ログインするためのユーザー名とパスワードは、忘れないように保管するよう、生徒にしっかり指導してください。ユーザーID とパスワードを印刷したカードを渡すのも、一つの方法です。

ウォークスルー（1）　：　教師が画面を見せながら説明	4 分

コードモンキーの基本的な概要を、生徒達とウォークスルーして（教師が操作して生徒たちと同じ画面を見ながら、説明や解説をして理解を深めることをして）理解します。

1. ホームページ playcodemonkey.com ［スタート（PLAY NOW）］ボタンをクリックする。
2. 表示される紹介アニメーションを見る。
3. 声を出して指示を読む。
4. コードモンキーゲームは、いろいろなレベルから構成されており、それらはチャレンジと呼ばれます。まさに、それに「挑戦」して解決方法をみつけ「クリア」するのです！
5. 画面右側の文字をタイプして編集できるところは、プログラムのコードを書く場所です。キーボードから文字をタイプする代わりに、画面下にあるボタンを押すことで簡単に入力することもできます。タブレットを使う時は、画面下のボタンをタッチして入力します。
6. 画面左側は、書いたプログラムにしたがって登場キャラクターが動く舞台です。ゴールは、おサルのモンタにバナナをキャッチさせ、すべてのチャレンジをクリアすることです。
7. 画面左下隅のモンキーは Gordo（ゴード）で、チャレンジの内容や、時には、立ち往生した時にヒントを出してくれるガイド役です。Gordo をクリックすると、いつでもそのチャレンジの指示を再表示することができます。

ウォークスルー (2)	3 分

8. [RUN] ボタンを押してコードを実行し、そのコードがどうなるかを見ます。
9. 右側のコードは `step 15` と書かれていています。 [RUN] ボタンをクリックすると、おサルのモンタは 15 歩前に歩きます。
10. [RUN] をクリックします。
11. これで最初のチャレンジをクリアしました。チャレンジをクリアすると、その解答に星で点数がつけられます。星３つが最高点で、それはすべてのバナナを取って新しいトピックを学習し、コードを短く書けたご褒美です。もし星３つより少ない場合は、星３つを取るためのヒントが出ます。チャレンジは何回でも好きなだけやり直すことができます。何度失敗してやり直しても、最後に取る星の数には影響しません！
12. [やり直し] をクリックすると、ソリューションが再表示されます。
13. コードを
    `step 15`
    から
    `step 5`
    `step 15`
    に書き換えます

ボタンをクリックする代わりに、キーボードの Control を押しながら Enter（改行）を押してもいいです。慣れれば、その方がマウス操作を省略できて早く快適に実行できます。

キーボードが半角英数ではなく全角になっていると、プログラムの編集がうまくできません。操作に慣れていない子は、それだけでつまずいて取り残され、やる気を失ってしまう恐れがあります。キーボードの基本操作を生徒と確認しておいてください。

14. そして [RUN] をクリックしてください。この解答が、星２つであることを示して、星３つを取るにはどうすればよいかというヒントになるように、生徒の気を引くようにしてください。
15. [やり直し] をクリックし、星３つの取れるコードに修正し、[RUN] をクリックしてください。

ウォークスルー (3)	３分

16. [次のチャレンジ] をクリックして、 次に移りましょう 。
17. 声を出して指示を読みましょう。
18. 右側のコードは `step 10` と書かれています。[RUN] をクリックして、何が起こるか見ます。
19. モンタはバナナまで届かなかったですね。そして「`step 15` にしてみよう」 とヒントが出ています。数字の 10 を 15 に変更して [RUN] をもう一度クリックします。
20. チャレンジを初めた時に最初から書かれているコードを利用するのは良いアイディアです。バナナが取れないコードを何度実行してもかまいません。というよりも、そのコードはそれなりの意味があって最初から書かれているので、それを全部消すことはしないで、そのコードをどう直していけばよいかを考えます。
21. 立ち往生した時の一つの良い方法は、もう一度最初からやり直すことです。その時は、[やり直し] ボタンをクリックすることで、コードを最初のコードに戻すことができます。
22. [やり直し] ボタンをクリックして、最初のコードからやり直しする方法を生徒に教えます。
23. そしてバナナが取れるようにコードを編集して、[RUN] をクリックします。
24. マップアイコン（画面右上）をクリックし、番号の 0（ゼロ）をクリックしてチャレンジ No.0 に戻ります。こうやって、以前に行ったチャレンジに戻れるということを生徒に教えてください。生徒は、まだクリアしていないチャレンジを超えて、その先にスキップすることはできません。

プレータイム　　：　生徒が各自で行う	10 分

チャレンジ No.0 - 5 を、すべての生徒が少なくとも星２つでクリアします（12 歳以上の生徒は、すべて星３つを取るようにしてください）。生徒の進捗状況を、教師のダッシュボードを使って、生徒の進捗状況を常に確認してください。

生徒が、右に向くか左か混乱している場合は、モンタの左手につけている時計に注目させます。その時計の方向に向くのが左だと教えてください。

レビュー　　：　教師と生徒が一緒に行う	２分

チャレンジ No.2 を開くと、定規のアニメーションが自動的に表示されます。 指示に従って、モンタとバナナの距離を測り、その距離を使ってコードを修正します。生徒が定規の使い方を理解していることを確認してください。

やり直しボタン

定規の使い方のヒント：
クリック＆ドラッグで計測しようとする生徒は混乱します。まず定規をクリックして定規の操作を開始。①計測を初めたいところへマウスを移動してクリック、②測りたいところまで移動してクリック。これを必要なだけ繰り返し、左上の元の位置に定規を戻してクリック、で終了。これが基本だが、途中いつでも画面右のプログラム編集画面にマウスを移動すれば計測を終了できる。

タブレットの場合は異なる：
定規をタップしたら、計測開始点から計測終了点までを指でなぞる。

Part 3： ディブリーフィング（今日の授業の振り返り） 【 10 分 】

ディスカッション　：　教師と生徒が一緒に行う	５分

- 今日はどんなプログラムの命令を学びましたか？
- コードモンキーのどこが一番好きでしたか？
- 命令以外に、今日何を学びましたか？
- コードモンキーのチャレンジをどうやって星３つでクリアしますか？ 一発でクリアして星３つ取ることと、何度も間違えてやりなおすこと、その間に違いはあるのでしょうか？（答えは NO）
- 立ち往生したとき、あなたはどうしますか？（以下の２つの質問が関連しています）
- コードモンキーで、もう一度チャレンジの指示を表示したい時はどうすればいいですか？
- コードモンキーで、チャレンジを開始した最初の状態にコードを戻すには、どうしますか？

レビュー　　：　教師と生徒が一緒に行う	３分

チャレンジ No.6 を開き、クラスのみんなでそれを解きます。次のレッスンでは、生徒がそれぞれ自分自身で解きます。

宿題	2 分

次のレッスンまでに、あなたの家から学校までの道順の地図を、今日学んだように、コンピュータへの指示として書いてきてください。さらに、あなたの家の中で自分の部屋から別の部屋へとか、学校の自分の教室から別の部屋や場所まで、などの道順を書いてきてもいいです。

例をホワイトボード書いて示します。

レッスン 2 - 方向転換 *Turn Around*

チャレンジ No. 6 – 10

このレッスンでは、さらに 5 つのチャレンジをクリアすることで、コードモンキーの使い方の理解を深めます。授業を始める前に、教師のダッシュボードを使って、クラスの生徒たち全員がが最初の 5 つのチャレンジを星 3 つで完了していることを確認します。すべての生徒が同じレベルにあり、誰も遅れていないことが重要です。

目的：

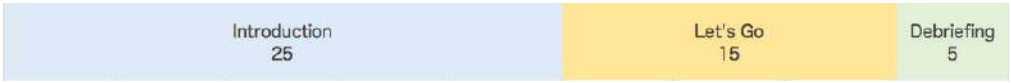
このレッスンで生徒は以下のことを学びます。

- 前のレッスンで学んだ内容の確認
- `turn` 命令の、別の使い方の理解
- コードモンキーのチャレンジ No. 6 – 10 をクリア

コンポーネント：

- 命令： 角度を使った `turn` 、後方への `step`
- 用語： プログラム（program）、関数（かんすう、function・ファンクション）、
  引数（ひきすう、argument・アーギュメント）、ステートメント（statement）、
  オブジェクト（object）

レッスンの時間配分



Part 1： イントロダクション 【 25 分 】

レビュー	5 分

宿題を集める。

前のレッスンで何を学んだか、クラスで簡単なディスカッションをしましょう

- コーディングとは何ですか？
- これまで使用してきた命令は何ですか？ （`step`、`turn`）
- プログラミング言語って何？ コードモンキーには何を使いますか？ （CoffeeScript）

アクティビティー	5 分

3 人のボランティアを募り、それぞれに役割を与えます。一人は「プログラマー」、もう一人は「コンピュータ」で、三人目は「キャラクター」です。そして「プログラマー」に、「キャラクター」が教室に置かれた物を取りに行くように「コンピュータ」に命令をしてもらいます。「プログラマー」役の生徒が正しく命令していることを確認します（`step` 数字、`turn` 右、`turn` 左）。学んだ事をクラスの生徒と確認できるように、ホワイトボードに命令を書いていきます。

このアクティビティーを、もう一組別の 3 人のボランティアグループで繰り返します。

ディスカッション	3 分

生徒たちに、「なぜ、コンピュータとキャラクターの両方が必要なのですか？ なぜ一人で両方のことができないのですか？」と質問してください。

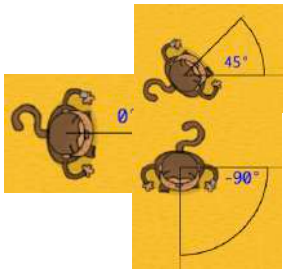
もしプログラミングを人の体にたとえるならば、プログラマーは身体のさまざまな部分に命令を送る脳です。コンピュータの役割は、身体のいろいろな部分（キャラクター）が、命令の指示どおり正確に動いていることを確認することです。

説明	2 分

生徒に「ステートメント（文）」という新しい用語を説明します。これは、何をしたいのか、というアクションを表現したものです。コンピュータプログラムとは、コンピュータに与えるシンプルな作業命令の集まりです。 これらの命令は「ステートメント」と呼ばれます。さっきのアクティビティーで、「プログラマー」が「コンピュータ」に与えた指令が「ステートメント」です。 ステートメントは、単純な一行のコードから、複雑な条件と数式でできた複数の行に至るまで、さまざまな形態があります。

説明	3 分

このレッスンでは、向きを変え、そして後ろにバックして進みます。キャラクターの向きを変えるのに 3 つの方法があります。最初の方法は、前のレッスンで学んだように `turn right` や `turn left` を使うことです。このレッスンでは別の方法を紹介します。



右・左に向きを変えるのではなく、角度を使って向きを変えることができます。生徒が 360 度の回転や 90 度の回転といった角度の基礎知識を持っている場合は、その知識を生徒たちと復習してください。そうでなければ、角度という知識の導入をしてください。分度器を使用するとよいでしょう。

説明	3 分

「オブジェクト」とは、画面左の舞台上にある、コンピュータで操作できるものすべて、「モンタ」「バナナ」「茂み」「橋」「カメ」などです。

それぞれのオブジェクトには、実行できるアクション一式が備わっています。それは、例えば `step` 、`turn` 、`turnTo` （`turnTo` は次のレッスンで学習します）などです。これらのアクションは「関数（かんすう）」と呼ばれます。その関数が実行するときに追加する入力を「引数（ひきすう、argument・アーギュメント）」と呼びます。 たとえば `turn 10` では `turn` が関数で、`10` がその関数の引数です。

ディスカッション	2 分

- クラスの生徒に、「ステートメント」の例をあげて、それをホワイトボードに書いてもらいます。（回答例： `step 10` 、`step 15` 、`turn right` 、`turn left`）
- このステートメントの中で、「関数」が何であるか聞きます（`step`、`turn`）
- 「引数」がどれであるかを聞きます（`10`、`15`、`right`、`left`）

説明	2 分

後にバックして歩くという概念の理解は比較的容易です。 前に 15 歩進みたい時は `step 15` 、15 歩後ろにバックしたい時には、`step -15` をタイプします。`-15` を、コンピュータはこの文脈では「15 ステップ後方に進む」と解釈します。もし、あなたの生徒が中学生以上ならば、数直線上の負の数について話をする良い機会です。

Part 2： レッツゴー！ 【 15 分 】

ログイン	1 分

生徒は、コードモンキーのウェブサイト playcodemonkey.com で、自分のアカウントにログインします。ログイン情報を覚えていない場合は、教師の持っているアカウントリストかそれを記入したカードを使って、ユーザーID とパスワードを生徒に教えてください。ID とパスワードはなくさないように、きちんと保管しておくよう生徒に徹底してください。

プレータイム	2 分

チャレンジ No.6 - 9 を、すべての生徒が少なくとも星 2 つでクリアします（12 歳以上の生徒は、すべて星 3 つを取るようにしてください）。生徒の進捗状況を、教師のダッシュボードを使って常に確認してください。

角度を使って向きを変える時に、生徒が困難にぶつかるかもしれないことを意識しておいてください。チャレンジ No.7 と No.8 では、ヘルプをする必要があるかもしれません。下記のウォークスルーを活用してください。

ウォークスルー	2 分

チャレンジ No.7 を開き、角度に関するアニメーションを表示します。 定規は、モンタとバナナの距離を測るとともに、分度器でもあることを示します。45 という数字は、モンタからバナナの方向の角度です。これがコードの step の後ろの数字と同じであることを示します。生徒が定規の分度器としての

使い方も理解していることを確認してください。

説明	2 分

チャレンジマップを開き、生徒に [スキルモードタブ] を示します。スキルモードは、もっと多くの課題にチャレンジして、コードモンキーのスキルをさらに完璧にするためのものであることを説明します。 これらの追加のチャレンジはとても素晴らしい練習で、一定のチャレンジをクリアした後にのみプレイできます。ロックされたチャレンジにカーソルを合わせると、ロックを解除するためのヒントが表示されます。最初のスキルチャレンジは、チャレンジ No.6 をクリアした時に現れます。

スキルモード

プレータイム中に全部のチャレンジを早くクリアした場合には、スキルモードに行って、アンロックされているスキルチャレンジが実行できることを生徒に知らせます。

プレータイム	5 分

生徒は、引き続きチャレンジ No. 6 - 9 を続けます。

チャレンジ No.8 では `turn left` または `turn 90` のどちらでも星３つをとることができます。 生徒の何人かは `turn left` を使うでしょう。その生徒には `turn 90` でも同じ結果になることを強調してください。

アセスメント（評価）	3 分

チャレンジ No.10 は、これまでにコードモンキーで学んだことすべてを確認する、アセスメント・チャレンジです。

Part 3： ディブリーフィング（レッスンの振り返り） 【 5 分 】

レビュー	2 分

生徒が角度を使って向きを変えるのを理解したか、チェックします。

生徒全員に立ち上がってもらい、「ターン 90」 「ターン 120」 「ターン 360」と向きを変えるように指示をします。

レビュー	2分

後ろへバックで進むことを理解したか確認します。教師はドアを背にして少し前に立って、「私がモンタだとして、私がドアのところに行くための適切な命令は何でしょうか？」と質問してください。それは一回だけの命令で、方向使わないことを強調してください。 その答えは `step -`（マイナス）`X` であることを確認します。

説明を続けます。指定したステップ数だけバックで進むには、数字の前にマイナス記号（ - ）をつけます。例： `step -10` コンピュータは `-10` を「10 歩後退」と解釈します。

宿題	2 分

次のレッスンまでに、あなたの家から学校までの道順を、角度を使った命令で作ってきてください。

第１章

レッスン ５ - 繰り返し *In the Loop*

チャレンジ No. 21 – 25

ループ・アイランド

おめでとう！ コードモンキーの導入部（序章）を通過しました。さあこれで生徒たちは、基本的なプログラミングのスキルを身につけました。このレッスンでは、ループ（繰り返し）に焦点を当てます。ループには `for` ループ（特定の条件のあいだ繰り返す）と、`until` ループ（条件が成り立つまで繰り返す）の異なる種類がありますが、最初に単純なループの使用方法を学習します。

目的

このレッスンで、生徒は以下のことを学びます。

- ループ（繰り返し）をプログラミング用語として定義
- プログラミングでループを使うと、より効率的である理由を理解
- コードモンキーのチャレンジ No. 21 – 25 をクリア

コンポーネント

- 命令： `x .times ->`
- 用語： ループ（loop・繰り返し）
- 機能： タブ（tab、indent・インデント/段落付け/字下げ）

レッスンの時間配分

Introduction	Let's Go	Debriefing
25	15	5

Part 1： イントロダクション 【 25 分 】

説明	5 分

プログラミングの学習では、正しい順序で正しいステートメント（命令）を書くだけではなく、分かりやすく短いコードで書く方法を学ぶこともとても大事です。

モンタが 100 段の長い階段を登る簡単なプログラムを書くとします。ただし、一度に階段を一段だけ昇る `stepUp` という関数しか使えません。

生徒に質問してください。「プログラマーはすべて階段の段ごとにコードを書くと思いますか？」と。100 行のコードがどれほど長いコードとなるか想像してみましょう！」

だから、コードをこんな風に（100 回繰り返して）書く代わりに

```
stepUp
stepUp
stepUp
stepUp
…
```

短く書くのは素晴らしいとは思いませんか？ 生徒たちに、短く書く方法を提案するように問いかけてください。こんなのはどうでしょう？

```
stepUp 100 times
```

幸いなことに、このような記述が可能なのです。これと全く同じではありませんがとても似ています。このような方法で書かれたコードを、ループ/繰り返し（Loop）と呼びます。

説明	5 分

生徒たちに、「単純ループ」は、指定した回数だけ命令をくり返すものだと説明してください。それとは別に、特定の条件が成り立つまで繰り返しを続ける `for` ループ、`until` ループ という種類もありますが、それらについてはあとから学習します。

先ほどの階段を例にすると、コードモンキーでそれを書く方法はこのようになります。

```
100.times ->
    stepUp
```

数字（`100`）は、ループの内側のコードを実行する回数を表します。

特別な構文に注目してください。数字と `times` という単語の間にドット（.）があることと、`->` の前にスペースがあること、そしてくり返されるコードは、左端から右側に何文字か字下げ（Indent・インデント）されています。この例では、`stepUp` がループの内側のコードです。

コードの先頭位置を下げる（右にずらす）ために、キーボードの［tab キー］の使い方を生徒が分かっていることを確認します。tab キーを使う代わりの方法として、スペースキーを４回押してもかまいません。

画面下部の［times］アイコンをクリックすることで、構文を気にすることなくループが書けることを生徒たちに思い出させてください。

アクティビティー	15 分

単純なループの使い方をきちんと理解するために別の例を示します。ホワイトボードの左側に次のように書きます：

```
step 10
turn left
step 10
turn left
step 10
turn left
step 10
turn left
```

生徒たちに、コードの中で**くり返されているパターン**が何であるかを聞きます。識別されたパターンは次のようになります。

```
step 10
turn left
```

そして、ホワイトボードの右側に、次のように書きます

```
4.times ->
    step 10
    turn left
```

生徒たちに、左右のそれぞれのコードについてどう思うか質問します。

右側のコードはループを使って書かれているところが違うけれど、どちらのコードも実行結果が同じであることを説明します。左側の中の繰り返しパターンを発見したら、私たちがやるべきことは、それを一度だけ書き、`4.times ->` を書き足すことです。その結果できたコードは同じことを行いますが、より短くより読みやすくなっています。

右側のコードの意味は、`step 10` 、`turn left` を 4 回繰り返したら、そのループが終了する、ということです。ループが終了すると、コンピュータは次の行の文に移動します。

Part 2： レッツゴー！ 【 15 分 】

ログイン	1 分

生徒は、コードモンキーのウェブサイト playcodemonkey.com で、自分のアカウントにログインします。

プレータイム	10 分

チャレンジ No. 21 - 25 を、すべての生徒が少なくとも星２つでクリアします（12 歳以上の生徒は、

すべて星３つを取るようにしてください)。生徒の進捗状況を、教師のダッシュボードを使って、常に確認してください。

アセスメント	４分

チャレンジ No. 24 は、コードモンキーで最近学んだ事をすべて使って解く、アセスメント・チャレンジであることに注目してください。

プラクティス	

早く終わった生徒には、画面右上のマップアイコンをクリックし、[スキルモード] タブをクリックしてスキルチャレンジを開き、ロック解除されたチャレンジをクリアするように推奨します。

Part３：　ディブリーフィング　【５分】

ウォークスルー	３分

No.25

チャレンジ No.25 を開き、[やり直し] ボタンをクリックしてコードをリセットします。そして生徒といっしょにコードを書きます。コードを書いたらゆっくり明快に声に出して読み上げましょう。このウォークスルーは、どうやってコードを正しく読むかを、生徒に示すことを意図しています。

バナナの配置 **パターン** が **L** 字型であることを発見し、それを以下のような一連のコードにするウォークスルーを行います。

```
turn left
step 5
turn right
step 5
```

そして、表示されているループ内側の文を、この発見したコードに一致するように修正して [RUN] ボタンをクリックします。うまくバナナを取りに行きましたが、でもループは４回ではなく３回なので、最後のバナナは取れませんでした。

生徒には、L パターンを何回繰り返せばよいの？と聞きます。このチャレンジを解くために必要な最後のヒントをだします。例えば、「３を４にかえてみたら？」 と。

説明	２分

５ブロック先に行く指示をする場面を想像してください。あなたはこんな風に言いますか？

「１ブロック行って、さらに１ブロック行って、そして１ブロック行って、もう１ブロック行って、さらにもう１ブロック行って」

違いますよね？ あなたはシンプルに、通りを５ブロック先まで行ってください、と言いますよね。なぜなら同じアクションを何度も言うのは大変だからです。

それはコーディングでも同じであることを、生徒たちに認識させてください。実行すべきパターンがくり返される場合、ループを使うことはプログラムを短く分かり易くするための良い方法です。繰り返されるパターンを見つけて、一度だけそれを書き、何回繰り返すのかをコンピュータに指示するコードを追加するのです。

第２章

レッスン ７ - 変数の谷 *Variable Valley*

チャレンジ No. 31 – 35

バリアブル・バレー

コードモンキーの第２章へようこそ！ ここでは、「Variable（変数・バリアブル）」や「`for` ループ」といった高度なトピックを学んで行きましょう。

目的

このレッスンで、生徒は以下のことを学びます。

- 「変数（へんすう、Variable・バリアブル）」というプログラミング用語を定義
- 変数をどうやって使うか、なぜ使うのか、をディスカッション
- コードモンキーのチャレンジ No. 31 – 35 をクリア

コンポーネント

- 命令： `X =`
- 用語： 変数（variable・バリアブル）、代入（assignment・アサインメント）

レッスンの時間配分

Introduction	Let's Go	Debriefing
20	15	10

Part 1： イントロダクション 【 20 分 】

アクティビティー	5 分

生徒２人にボランティアをお願いします。彼らを、たとえばアリスとボブと呼びましょう。クラスの皆さんはプログラマーです。さて、アリスは前に進む必要がありますが、何歩進めばよいのか、まだ知りません。

あなたは、１から１０までの数字を書いて折った紙をボブに渡します。そして、あなたが「スタート」と言ったらボブはその紙をアリスに渡します。その紙には、アリスが前進すべき歩数が書かれています。

そして、これからどんなアクティビティーをするのか、クラスの生徒に説明します。プログラマー役がスタートと言ったら、まず、ボブがアリスに紙を渡す。そうすると、アリスは紙に書かれている数字の歩数だけ前に進みます。ここで強調すべきことは、アリスは、どれだけ進むかは分かっていないけれど、前進しなさい、という指示をすでに理解している、という事実です。では、始めましょう。

これをしばらく行ったら、生徒に「このアクティビティーにどんな意味があると思う？」とたずねます。

アリスに前進しなさいという指示を与えた時点では、何歩進むかはまだ不明だったけれど、その状態であっても、何をしなさいという指示を与えることは可能であった、という事実についてディスカッションします。

指示を与えた時点で不確定の要素があっても、指示を実行する時点でそれが確定していれば、命令として成立する。

アクティビティー	5 分

このアクティビティーを、あと２名の生徒で繰り返します。ただし、次の点を変えて行います。

今回は、ボブが紙に番号を記入します。プログラマーのあなたもアリスの進む歩数を知りません。しかしそれでも、アリスに行うべき指示を与えることができた、という事実を指摘してください。

説明	10 分

前のレッスンでは、特定の値を指定して関数を呼び出す方法を学んだことを説明します。たとえば、`step 10` とか `step 20` で、この `10` や `20` は関数の「引数（ひきすう）」と言います。これとは違って、特定の値ではなく「変数（へんすう）」を使って呼び出す方法があります。

変数とは、何かを保管しておく入れ物のようなものです。その中にデータを入れておき、それが必要となった時に中身を使います。後から使う値がその中に保管されているという点で、いま行ったアクティビティーで使った紙に似ています。

変数の中に情報を入れるために、等号記号（=）を使った文を書きます。これをプログラムでは「代入（だいにゅう、assignment・アサインメント）といいます。ちょうど、紙に数字を書き込むようなものです。

識別子： いろいろな対象から特定の一つを識別・同定するのに用いられる名前や符号など。プログラミング言語では変数や関数等に対して指定する名前。

プログラムの代入文は、「識別子（しきべつし・identifier・アイデンティファイヤー）」と「値（あたい、value・バリュー）の２つの要素からできています。

```
x = 20
```

この代入文の先頭の `x` が識別子です。これには `x` 以外の他の文字や単語を使うことができます。識別子は変数の名前です。変数の中に入っている値を使用する場合、その名前を書きます。例えば、モンタに変数 `x` の中に入っている値と同じだけ前進させたい時は `step x` と書きます。この例では 20 です。名前と値を分離することは、名前を、それが示す情報から独立して使用することを可能にします。コードの実行時にその値がどうなるかをまだ知らなくても、`x` を使ってプログラムを書くことができます。

オプション： さきほどのアクティビティーに戻り、ボブに新しい数字を紙に書いてもらいます。そして何度かさっきと同じことを数回繰り返し、このポイントの理解を深めます。

このアクティビティー、変数がプログラムでどのように機能するかを示すことを意図しています。この例では、アリスは行うべき行動は分かっているが、ボブから紙を受け取るまでは具体的な値が分かりません。

Part 2： レッツゴー！ 【 15 分 】

ログイン	1 分

生徒は、コードモンキーのウェブサイト playcodemonkey.com で、自分のアカウントにログインします。

プレータイム	14 分

チャレンジ No. 31 - 35 を、すべての生徒が少なくとも星２つでクリアします（12 歳以上の生徒は、すべて星３つを取るようにしてください）。生徒の進捗状況を、教師のダッシュボードを使って、常に確認してください。

この時間を使い、クラスの中を歩きまわって、行き詰まっている生徒への助言をします。

プラクティス	

早く終わった生徒には、画面右上のマップアイコンをクリックし、[スキルモード] タブをクリックしてスキルチャレンジを開き、ロック解除されたチャレンジをクリアするように推奨します。

Part 3： ディブリーフィング 【 10 分 】

ウォークスルー	7 分

チャレンジ No.35 を開きます。ここでは初めて、自分自身で変数を定義します。 [やり直し] ボタンをクリックしてコードを最初の状態にします。

No.35

問題の解決を始めます。

ここで、星３つの解決策をいくつか示します。

1. 変数の名前を `x` から何かほかの文字に変更して実行します。それを何度か繰り返します。結果が同じことから、変数の名前は必ずしも `x` でなくても良いことを強調します。
2. 代入文を `x = 1 + 1` のような算術式に変更します。そして、コンピュータは数式や他の操作の計算ができることを強調します（それが、コンピュータとか計算機と呼ばれる由縁です）。
3. 一つの変数から別の変数に代入します。例えば

```
y = 15
x = y
```

これでは星３つはもらえませんが、ちょうど一つの紙から別の紙に数字をコピーするように、変数から変数に代入できることを示します。

アクティビティー	３分

以下の点を変更して、今日のレッスンの最初に行ったアクティビティーを繰り返します。

ボブに数字を書いた紙を渡します。ボブは、まずその数字を別の紙に書き写し、それをアリスに渡すと、アリスは前進します。それが次のプログラムと同じであることを説明します。

```
y = 15
x = y
step x
```

第３章

レッスン 14 - 関数ファーム *Function Farm*

チャレンジ No. 71 – 75

ファンクション牧場

コードモンキーの第 3 章が、チャレンジ No.71 から始まります。ゴリラが橋を壊したので、モンタは川を渡れなくなってしまいました。ラッキーなことに、モンタが橋を直すのを手伝うためにネズミが来てくれるので、生徒たちにはネズミを助けてもらいます！ この章では、新しいキャラクターとクールな新機能、そしてループを紹介します。

目的

このレッスンで、生徒は以下のことを学びます。

- プログラミング用語として、「関数」と「コメント」の定義を再確認
- 主コード（メイン・コード）を定義
- 関数操作の実施
- コメント記述の実施
- コードモンキーのチャレンジ No. 71 – 75 をクリア

コンポーネント

- 命令： `grab()` 、 `drop()` 、 `# コメント`、 `関数名 = (引数) ->`
- 用語： 関数（function・ファンクション）、引数なし関数、関数の定義と呼び出し、
  コメント/注釈（comment）

レッスンの時間配分

Introduction	Let's Go	Debriefing
20	15	10

Part 1： イントロダクション 【 20 分 】

説明	8 分

このレッスンでは、新しい登場キャラクターのネズミとプレーを始めます。すでにネズミには出会っていますが、このレッスンからはプログラムでネズミを制御し、ネズミに物を集めてもらいます。

このレッスンの主要なトピックは、関数を読むことと書く方法を学ぶことです。「関数（かんすう）」という用語はレッスン２の「方向転換」で出てきましたが、ここではさらに掘り下げて学びます。

どうして「関数」が必要なのでしょうか？

自転車に乗るためのコード一式（例： 自転車のサドルにすわる、ペダルをこぐ、交通に気をつける、ベルを鳴らす、など）を生徒たちに想像させてください。

そして、このコード一式を、ほかの場所で別の自転車で使うことを考えます。このように。

```
go to mountain
sit on saddle of mountain bike, pedal, watch out for traffic, use bell
go to the city
sit on saddle of city bike, pedal, watch out for traffic, use bell

山に行く
マウンテンバイクのサドルに座る、ペダルをこぐ、交通に気をつける、ベルを鳴らす
街に行く
シティバイクのサドルに座る、ペダルをこぐ、交通に気をつける、ベルを鳴らす
```

毎回、何度も何度も自転車に乗るためのコード一式（`sit on saddle …` ）を書く必要がありますか？

コンピュータに一回どうやって「自転車に乗る」かを伝えたら、次からは「乗る」と一言で言う方が良いと思いませんか？ 次のように。

```
ride bike means:
    sit on saddle of bike
    pedal
    watch out for traffic
    use bell
go to mountain
ride mountain bike
go to the city
ride city bike
```

```
自転車に 乗る ということは：
    自転車のサドルにすわる
    ペダルをこぐ
    交通に気をつける
    ベルを鳴らす
山に行く
マウンテンバイクに 乗る
街に行く
シティバイクに 乗る
```

注意： 「**自転車**に 乗る」 を定義した時、具体的にどんな自転車かは言及していません。それが「自転車」というオブジェクトであって他の別物ではない限り、何であっても構わないのです。

説明： 「関数」とは、特定のタスクを実行する命令のひとかたまりです。私たちがそれを呼び出した時にのみ、コンピュータはその関数を実行します。関数を使う事を、関数を呼び出す（ファンクション・コール）と言い、関数を作ることを、関数を定義すると言います。これまでに私たちはいろんな関数を呼んで（使って）きましたが、今日はどうやって関数を定義する（作る）かを学習します。

関数を使ってコードを書くこと（さっきの後の例）が、関数を使わないで書くこと（最初の例）とは対照的に、なぜすぐれているのかという理由を、生徒たちに考えさせます。

正解の例：

- コードを短くします
- コードを読みやすくします
- 「乗る」という作業の一部が変更された場合（例えば、曲がる時にハンドルを使う、を追加する場合)、 関数を定義しているところを一度だけコードを変更すればよい
- チームメンバーの仕事を、関数を書くメンバーと、それを使用してコード書くメンバーに分けることができます

生徒たちの答えが、これらをカバーしていることを確認します。

ウォークスルー	8 分

引数を伴う関数（引数付きの関数）はこのよう書かれます。

```
function_name = (argument) ->
    first_statement
    second_statement
    etc ...
```

```
関数の名前 = (引数) ->
    最初の文
    2 番目の文
...
```

これは、関数の定義と呼ばれます。

先頭行が関数の定義の始まりで、その下に、 [RUN] ボタンを押した時にコンピュータに実行してほしいコードを書きます。これで、関数が定義され、それ以降この関数を呼び出して使えるようになります。

例 ：

```
goto = (t) ->
    turnTo t
    step distanceTo t
goto banana
```

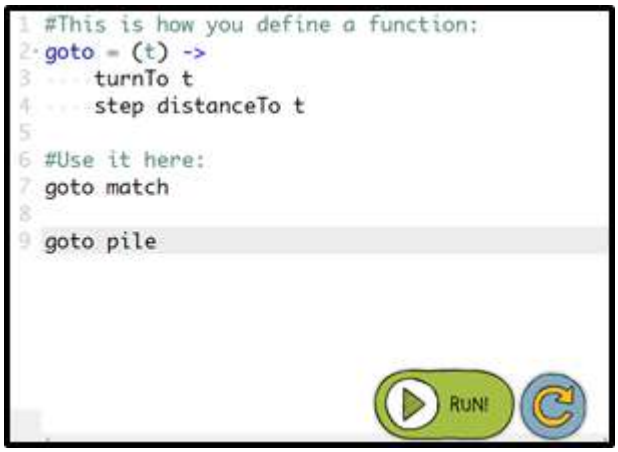
[RUN] ボタンを押すと、コンピュータは `goto banana` の行に直行してそれを実行することに注意してください。それから、その上にある関数の定義部分を呼び出します。

質問： コンピュータはどうやって、関数を定義する文と、そうでない文とを区別するのでしょうか？
答え： インデント。レッスン 5 でならった単純ループのように。

関数の名前は識別子として使用されます。

プログラマーは、関数の名前には、それが何をするかを表現してコードが読みやすくなる名前を付けます。

関数を呼び出すとき、関数の名前と、関数が使用するオブジェクトの名前をペアにする必要があります。次の例を見てください。



最初は混乱するかもしれませんが、最も重要なことは、ゆっくりとこのコードを読むことです。ホワイトボードにこのコードを書いて、生徒たちと一緒にそれを読み上げましょう。

関数の名前は `goto` で、引数が `t`

そして、7行目の `goto match` から読み始めます。この行とそれ以降の行は「主コード/メインコード」と呼ばれます。[RUN] ボタンを押した時に、コンピュータが実行を開始する場所がここであることを理解するのが非常に重要です。それより上の行で、関数を定義している部分は一旦記憶され、メインコードで関数を使った時に初めて、コンピュータが実行します。

7 行目で、`goto` 関数を呼び出し、その引数は `match` です。したがって `t=match` となり、関数定義の中の `t` が `match` に置き換えられます。そしてコンピュータは３行目に戻り、そこを「`turnTo match, step distanceTo match` 」と読み替えます。

match: マッチ棒（最近の子供はマッチを知らない可能性があります）

pile：積み上げられた山

コンピュータが `goto` 関数を定義する文の実行を終えると、メインコード（9 行目）に戻って、実行を続けます。次に、同じ関数を呼び出す別の文があります。今回は引数が `pile` なので、コンピュータは３行目に戻り、`t = pile` で `goto` 関数が再び実行されます。

説明	4 分

別の種類として、引数のない関数があります。このタイプの関数も機能を実行しますが、何のインプットも関数に渡す必要はありません（引数は不要）。このような関数は、関数の後ろに空のカッコ `()` を付けて呼び出します。

() が付いていないと、変数なのか関数なのか区別ができない。

前に言ったように、ネズミはものを集めるのが大好きです。

次のいくつかのチャレンジで、私たちはネズミにマッチ棒を集めてもらいます。そのために、単純な関数（引数のない関数）である `grab()` と `drop()` を使います。

grab:つかむ　drop:落とす/置く

生徒たちに質問します。「コードモンキーで、これ以外の関数にはどんなのがありますか？」

答え： `step` 、`turn` 、`turnTo` はすべて関数です！ これらの関数の引数は、数字、方向、オブジェクトなどで、関数の後ろにそれを付け足します。たとえば、`step 20`（`step` は関数名で `20` が引数）、`turn left`（`turn` が関数名で `left` が引数）、`turnTo banana`（`turnTo` が関数名で `banana` が引数）などです。これらの関数は、コードモンキーを作ったプログラマーが、私たちがこれらの基本的な動作ができるように、事前に定義して使えるようにしておいてくれたものです。プログラミングでは、誰か他の人が定義した関数を使うのは、ごく一般的なことです。

最後に、このレッスンに出てくる新しい概念はコメント（注釈）です。コメントとは、先頭にシャープ記号（#）をつけたプログラミングコードの一行です。コンピュータは、この#から後ろは読み飛ばして命令としては扱いません。これは、コードを書いたり読んだりするプログラマーが、互いに理解するために使われます。誰か別の人が私たちのコードを読むときに、その人を助けるための注釈として主に使用します。第３部でコメントはとても有用です。生徒たちに、コードモンキーのコメントは、コードをどう修正すればよいかというヒントになっている、ということを教えましょう。

Part 2： レッツゴー！ 【 15 分 】

ログイン	1 分

生徒は、コードモンキーのウェブサイト playcodemonkey.com で、自分のアカウントにログインします。

プレータイム	5 分

チャレンジ No.71 - 75 を、すべての生徒が少なくとも星２つでクリアします（12 歳以上の生徒は、すべて星３つを取るようにしてください）。生徒の進捗状況を、教師のダッシュボードを使って常に確認してください。

クラスの中を歩き回り、困っている生徒、戸惑っている生徒をアシストしてください。

このレッスンのトピックは複雑なので、必然的に生徒たちがより多くのヘルプを必要としてきます。戸惑っている生徒に気付いた時は、彼らを小グループに集めて、より詳しい説明をします。

ヒント： `grab()` 、`drop()` を使う時には、括弧 `()` を忘れないように！

ウォークスルー	4 分

チャレンジ No.74 を開き、[やり直し] をクリックしてコードをリセットします。これは、関数定義が出て来る最初のチャレンジです。生徒にはコメントに注目するように言い、一緒に関数の定義を読み上げます。一度コードを実行して動きを確認し、それを修正してチャレンジをクリアします。

「 [RUN] をクリックしたら、コンピュータはどの行から実行を開始しますか？」と、生徒に尋ねます。答えは、７行目（`goto match`）です。

プレータイム	10 分

生徒はチャレンジ No. 71-75 を続けます。

プラクティス	

早く終わった生徒には、マップ上のスキルチャレンジを開き、ロック解除されたチャレンジをクリアするように推奨します。

No.74

Part 3： ディブリーフィング 【 10 分 】

ウォークスルー	10 分

チャレンジ No.75 を開き、コードを消します。 そして `goto` を使わない次のコードを書きます。

```
turnTo bridge
step distanceTo bridge
turnTo match
step distanceTo match
grab()
turnTo bridge
step distanceTo bridge
turnTo pile
step distanceTo pile
drop()
```

生徒たちは、このコードは星１つしか取れないことに気づくでしょう。星３つを取るためには、次のようにコードを書く必要があります。

```
goto = (t) ->
    turnTo t
    step distanceTo t
goto bridge
goto match
grab()
goto bridge
goto pile
drop()
```

ホワイトボードに、このコードを書きます。生徒たちがコードの読み方を理解していることを確認するために、一緒にコードを見ていきます。ヒント：コンピュータがコードを読んでいく順序が分かるように、矢印を描いていきます。

1. goto bridge
2. goto 関数定義の中の t を bridge に置き換えて実行
3. goto match
4. goto 関数定義の中の t を match に置き換えて実行
5. grab()
6. goto bridge
7. goto 関数定義の中の t を bridge に置き換えて実行
8. goto pile
9. goto 関数定義の中の t を pile に置き換えて実行
10. drop()

[RUN] をクリックしてコンピュータがコードの実行を開始すると、その時に実行されている行がオレンジ色でハイライトされることを、生徒たちに気づかせます。

最終章

レッスン 20 - 星 300 パーティー！ *Three hundred stars party !*

チャレンジ No. 1 – 100 と スキルチャレンジ

このレッスンで、生徒たちはすでにクリアしたけれど星が１つか２つしかとれなかったチャレンジにもう一度取り組みます。レッスン 20 の終了時に生徒たちは、コードモンキー の No.1 から No.100 までの全部のチャレンジを星３つでパーフェクトにクリアしていることでしょう。

目的

このレッスンで、生徒は以下のことを学びます。

- 星１つ、または星２つだったチャレンジに再挑戦
- 全部のチャレンジで星３つをゲット

レッスンの時間配分

Introduction 15	Let's Go 25	Debriefing 5

Part 1： イントロダクション 【 15 分 】

クラスを始める前に： 教師のダッシュボードで、星３つを取れない生徒が多いチャレンジの番号をチェックします。教師アカウントでグループを選び、[評価一覧] タブ画面の一番下、統計の「チャレンジの難易度」を使用して、どのチャレンジがあなたの生徒たちにとって難しかったかを確認します。大きな円ほど、より多くの生徒が苦戦したことを意味します。

ウォークスルー	15 分
青や赤の星が比較的多いチャレンジを２つ３つ選び、それらをクラスの生徒たちと一緒に解決します。	

Part 2： レッツゴー！ 【 25 分 】

プレータイム	25 分
生徒は、コードモンキーのウェブサイト playcodemonkey.com で、自分のアカウントにログインします。 ログインしたら、右上のマップアイコンをクリックし、左右の矢印をクリックしてこれまで取り組んできたチャレンジを確認し、星１つまたは星２つのしか取れていないチャレンジを見つけるように指示します。 クラスの終了時には、すべての生徒が、チャレンジ No.1 から １００まで、すべて星３つを取ることを目指します。 最初の 100 チャレンジですべて星３つ取った生徒は、[スキルモード] タブをクリックし、スキルチャレンジのアンロックされた課題に取り組むか、すでにクリアしたけれど星３つが取れていないスキルスキルチャレンジに再挑戦して全部星３つを取るようにします。 100 チャレンジとスキルチャレンジすべて星３つを取った生徒は、他のクラスメートをヘルプするように依頼します。	

Part 3： ディブリーフィング 【 5分 】

説明	5分

生徒たちと、コードを短く簡潔に書くことの重要性を議論します。

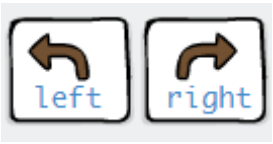
コードモンキーで星２つだった時は、同じ結果に到達するのに、より短かく書ける方法があることを意味します。最終結果に到達するのに対して不要な行があるか、ループを使うなどの短く書く方法があります。

例えば一つの授業を終えて次の授業に行く時に、毎回一度家に帰り、それからまた学校に戻ってきて教室に行かなければならない、という場面を想像してみてください。そうするのは無意味ですよね。長いコードを書くことも同じです。同じことを実現するのに、より短い方法がある場合は、それを長い道のりでやることに意味がありません。

おめでとう！
レッスン 20 - チャレンジ No.100 まで終了しました！

リファレンス・カード

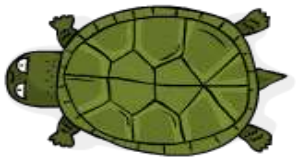
こちらも参考にしてください　https://www.playcodemonkey.com/help

キーワード / ボタン	説明
	`step 10` のように、`step` の後ろに数字で前進させたいステップ数をつけて、モンタを指定した距離だけ前進させます。
	向きを変える `turn` は、方向を示す `left` や `right`、または角度を示す `45 90 180` などと一緒に書くことが必要です。 例： `turn right` 、`turn 90`
	モンタを希望する方向に向けるために、`turn` の後ろに `left` または `right` をつけます このボタンをクリックすると、ボタンに応じてコードに単語 `left` または `right` がタイプされます。
	`turnTo` は向きを変える別の方法です。方向や角度を使う代わりに、`turnTo banana` のように、モンタを特定のオブジェクトの方向に向かせます。

	単純ループは、一連の命令を指定した回数だけ繰り返します。
	例：
	`3.times ->`
	`····step 5`
	`····turn left`
	この例では、モンタは `step 5` と `turn left`（5 歩前進し、左折）を 3 回繰り返します。ループの中に書かれる繰り返し実行される命令は、インデント（文字下げ ···）して書く必要があります。インデントは、スペース（半角）４文字文で、キーボードの Tab キーをタイプすることでも可能です。
	このボタンをクリックすると、コードに単純ループの先頭行が以下のようにタイプされます。
	`1 functionName = (argument) ->`
	`2 ····# Your code here`

`x = 10` `step x`	変数へ値を代入します。変数は記憶装置のようなもので、そこにデータを格納し、必要な時にそれを使用します。この変数への代入文は、変数の名前である識別子と、代入する値から構成されています。命令が実行される時点での値が分からなくても、*X* を使ってプログラムを書くことが可能になります。

登場キャラクター

登場キャラクター	説明
	アメリカの宇宙開発プロジェクトで、1958 年にロケットで宇宙飛行をした世界最初のサルの Gordo（ゴード）にちなんで名付けられました。あなたをヘルプするガイド役で、ヒントとチャレンジの目的を出してくれます。彼の発言は面白く、役に立ちます。 この Gord をクリックすると、チャレンジの目的をいつでも、もう一度表示できます。
	主人公のおサルのモンタです。あなたがプログラムコードを書いて、彼がバナナを集めるのをヘルプします。ご存知の通り。サルは水に濡れるのが大嫌い、そしてとてもフレンドリーです。
	チャレンジ No.13 で、信頼できる仲間のカメに出会います。カメ（turtle・タートル）は、あなたが面倒なバナナをキャッチするのをヘルプしてくれます。カメに `turn` や `step` 命令をする時には、その命令の前に `turtle.` を付ける（カメをクリックしてからコードを書く）ことが必要です。実施したいアクションとその対象としたいオブジェクトを、ドット（ . ）でつないで書きます。 例： `turtle.step 10`
	チャレンジ No.50 で、ビーバー（beaver）に出会います。ビーバーは木が大好きで、水の上のその木の上を渡ってたくさんのバナナを取れるように、あなたを助けることを約束してくれました。ビーバーは `step` しかできません。ビーバーに助けてもらうためには、彼らの名前とやってもらいたいアクションの命令の間にドット（ . ）をつけてコードを書くことが必要です。 `beavers[0].step 10`

	ワニは、チャレンジ No.66 から登場します。モンタがバナナをとるのを助けるために、水の上の橋の役割をしてくれます。ワニは `turn` しかできません。たくさんのワニが通常いるので、このように `for` ループと一緒に使うことが多いです。
	`for c in crocodiles`
	`····c.turn right`

	ネズミは何度もゲームに登場してきますが、彼を制御できるのはチャレンジ No.71 からです。ネズミは物を集めるのが大好きで、マッチ棒を集めるのをヘルプします。そのために、引数のないシンプルな関数 `grab()` と `drop()` を使います。
	アリは、チャレンジ No.89 から登場し、`until` ループ の使い方を学びます。アリは大事なマッチ棒を引きずって行くので、その後を追いかけてマッチ棒を取り戻さなければなりません。
	ネコは、チャレンジ No.96 から登場し、`until` ループ と `wait()` 関数を一緒に使うやり方を学びます。ネコはネズミを見つけると攻撃をしかけるので、ネコが眠りに落ちるまで待ってからマッチ棒を取りに行かなければなりません。

すべてのチャレンジの解答例

	メインモード		チャレンジ No.
1	はじまりの森	*FIRST STEPS*	0 - 10
2	オブジェクトの平原	*OBJECTS AND FRIENDS*	11 – 20
3	ループ・アイランド	*LOOP LAND*	21 – 30
4	バリアブル・バレー	*VARIABLE VALLEY*	31 – 45
5	アレイの沼	*ARRAY INDEXING*	46 – 55
6	フォーの森	*FOR FOREST*	56 – 70
7	ファンクション牧場	*FUNCTION FARM*	71 – 85
8	アンティル砂漠	*THE SANDS OF UNTIL*	86 – 100
9	イフの北国	*IF IT SNOWS*	101 – 111
10	雪の町オアエルス	*OR ELSE !*	112 – 119
11	ブーリアンの雪山	*BOOLEAN OPERA*	120 – 135
12	城下町ノット	*NOT MY CUP OF TEA*	136 – 141
13	コンパリソン・ストリート	*COMPARISONS*	142 – 149
14	リターン・アベニュー	*RETURN TO SENDER*	150 – 165
15	成功への鍵	*KEY TO SUCCESS*	166 – 179
16	ネズミを逃したのは誰？	*WHO MOVED MY MOUSE ?*	180 – 185
17	クリックしてね	*CLICK ME !*	186 - 200

	スキルモード		チャレンジ No.
1	はじまりの森	*FIRST STEPS*	1 – 11
2	オブジェクトの平原	*OBJECTS AND FRIENDS*	1 – 10
3	ループ・アイランド	*LOOP LAND*	1 – 10
4	バリアブル・バレー	*VARIABLE VALLEY*	1 – 15
5	アレイの沼	*ARRAY INDEXING*	1 – 10
6	フォーの森	*FOR FOREST*	1 – 15
7	ファンクション牧場	*FUNCTION FARM*	1 – 15
8	アンティル砂漠	*THE SANDS OF UNTIL*	1 – 15

お問合せ先

j21Corporation
ジャパン・トゥエンティワン株式会社 CodeMonkey サポートデスク


codemonkey_support@japan21.co.jp


@codemonkey_jp https://twitter.com/codemonkey_jp


codemonkey.jp https://www.facebook.com/codemonkey.jp/

■ 日本語訳・解説 著者紹介

髙嶋 孝明 （たかしま・たかあき）

豊橋技術科学大学（Toyohashi University of Technology）教授

国立岐阜工業高等専門学校・電気工学科卒業、国立豊橋技術科学大学・情報工学修士課程を修了後、1982 年から 2013 年末まで 32 年間 IBM に勤務。1973 年、高専入学直後の 15 歳からプログラミングの自学を始め、大型汎用機からミニコン・パソコン・マイコン・タブレットなどでプログラミングを趣味と勉学・研究・仕事で行ってきた。コンピュータやプログラミングそのものではなく、人間とコンピュータの界面とそこに内在する人間的要因に強く興味を持ち始め、大学ではタッチタイピング練習システム(TUT Touch Typing)、日本語 2 ストローク入力システム (TUT-Code)、思考を整理してプログラムの作譜に至る過程の整理手法、読みやすいプログラムの文書化などを修士論文テーマとして研究。IBM では当時日本では取り組みの少なかったヒューマンファクター（人間工学・ユーザーインタフェース）部門を大和製品開発研究所に立ち上げ、IBM5550 や ThinkPad のクリック感が良く使いやすいキーボードの開発をはじめとして、ハードウェアとソフトウェアおよびマニュアルのさまざまな使いやすさ設計と評価を推進し、ユーザー・エクスペリエンスの重要性を啓蒙。その後、エクゼクティブアシスタント、ハードディスク営業技術、海外駐在、グローバルオペレーション、エンジニアリング&テクノロジーサービスのビジネス開発、研究開発コンサルティング・営業、基礎研究部門のテクノロジー＆知的財産ビジネス開発などの役職に従事。*"People Helping People through Technology"* を自分の大切な言葉としてきた。豊橋技術科学同窓会会長、同大学経営協議会学外委員を経て、2014 年 1 月より教授として豊橋技術科学に転籍し、大学のグローバル化を推進。

2016 年 CodeMonkey に出会った瞬間に、子どもたちにプログラミングの楽しさを伝えたい！ プログラミングの基礎概念を楽しみながら体験を通じて身につけてもらいたい！ コンピュータサイエンスの経験や予備知識がない指導者でも生徒たちにプログラミングの知識を教えられるようにしたい！ 指導者と生徒たちの能動的なアクティブラーニングを実現したい！… CodeMonkey を開発したスタートアップの若者たちの溢れ出るすごい熱意と、そのための創意工夫と仕込まれた仕掛けが次から次へビシビシと伝わって来た。自分が子供の頃に CodeMonkey に出会っていたらどれほど嬉しくて楽しかっただろうか、この素敵な体験を多くの子供たちに届けたいとの想いがこみ上げ、一瞬にして CodeMonkey に魅了されてしまった。

著者がプログラミングを自学した当時の環境

コードモンキー・カリキュラムガイド

2017 年 1 月 27 日　初版

原著 …… CodeMonkey Studios,
195 Montague, 14th floor, Brooklny NY, U.S.A.

日本語訳・解説 …… 高嶋 孝明

発行者 …… ジャパン・トゥエンティワン株式会社

〒150-0021 東京都渋谷区恵比寿西 1-26-7
TEL：03-5456-8540
URL：https://codemonkey.jp/