農業の６次産業化・スマート化を担うＡＩ技術者育成プログラムの開発・実施

テキスト開発報告書

１．概要

１．１　背景

本事業では、農業ＡＩのカリキュラムに応じたテキストを開発する。本年度は、その中でも、初年次教育の核のひとつであるプログラミング教育の教材を取り上げて開発する。

ＡＩのシステム開発では圧倒的に Python 言語が用いられており、大きな開発コミュニティとライブラリ群があるため、今後もその傾向は変わらないと予想される。ＡＩ開発においては、単にコンピュータ言語としての Python を学べばよいわけでなく、アルゴリズムを習得し、さらにアルゴリズムと人間の思考との関係、言い換えれば人間が目的を理解しやすいプログラムの記述方法についても学んでおく必要がある。

イスラエルでは、「CodeMonkey」というプログラミング教材が開発されており、アメリカとイスラエルを中心に利用が進んでいる。すでに日本語版も開発されており、日本の小中学校などで利用されている。本事業に参加している清風情報工科学院と九州技術教育専門学校は、我が国でこの教材を先駆的に採用して利用している専門学校である。

「CodeMonkey」は CoffeeScript というプログラミング言語を通じてアルゴリズムを習得する教材である。CoffeeScript の文法は Python に似ており、AI の基礎教育に適した側面がある。

１．２　目的

ＡＩの基礎として、Python 言語をもとにアルゴリズムと人間の思考との関係を整理して習得できるテキストを開発する。その為に、既存の STEM 教材である「CodeMonkey」について整理する。

次年度以降、参加校等で活用しながら教材の改善が進められるよう重要な情報から順にリリースしていく。なお、後に述べるように「CodeMonkey」が日本向けに供給されないなどの事態に備えて、代替となる教材を開発するのに必要な情報も収集しておくことにする。

１．３　実施概要

（１）CodeMonkey の概要の整理

- CodeMonkey の学習上の特徴・学習者数・サブシステム・教師用管理画面などについてインターネット・資料・カタログなどから情報収集
- CodeMonkey の今後について日本代理店へのヒアリング

（２）CodeMonkey の授業実施結果の整理

- CodeMonkey 授業実施講師へ授業でのつまずき箇所やその対策についてのヒアリング
- CodeMonkey の学生の実施結果の整理
- ヒアリング記録をもとに、CodeMonkey のチャプター別につまずき箇所とその対策をまとめる

（3）CodeMonkey のカリキュラムの整理

- 画面および教師用ガイドをもとに、CodeMonkey の学習項目とその意味を整理
- CodeMonkey の学習項目について、委員会から指定された他のコンピュータ言語（C 言語、Java）との言語仕様の違いについて整理
- CodeMonkey の学習項目や予約語等について、英語と日本語と相違・機能の組み合わせの視点でつまずき箇所を整理
- つまずき箇所について対策を整理
- 委員会から指定された代替システムの候補（Pyxel）および実施方法の候補（日本語プリプロセッサ、Moocs への接続）について情報収集

１．４　実施成果

CodeMonkey を用いたプログラミング教育は、実施２校において良好な学習結果を挙げている。従来の講師が板書とプリントとコンピュータ演習を組み合わせて教育するのに比べて、はるかに高い習得結果が得られている。

しかしながら、日本語の語順とコンピュータ言語や英語の語順の違い、関数名などが英文法の影響を受けている影響、CodeMonkey の教育課程のステップの粗さなどが原因で習得の歩留まりが下がっていることがわかった。

また、他の授業で行っている C 言語や Java 言語などとの文法の違いから、せっかく CodeMonkey で言語を習得できても、それが応用できていない者もいることがわかった。同様に、CodeMonkey で習得しているアルゴリズムが他の授業で行う流れ図などで表現されるアルゴリズムと同じものであることが理解できていないものもいることがわかった。

これらのつまずきポイントについて橋渡しを行えば、CodeMonkey によりコンピュータ言語の習得の歩留まりを飛躍的に改善できる可能性がある。本開発において、この橋渡し教材の基礎をつくった。次年度に実証授業を行う。

また、CodeMonkey は、2018 年 12 月中国の TAL グループに 22 億円で買収されたと発表された。日本代理店の J21 によれば当面日本語版のサポートが行われ日本での流通には変化なしとのことであった。しかし、より長期的に見れば、CodeMonkey に依存することは教材供給ストップなどのリスクを抱えることになる。

本開発において、Pyxel 上に CodeMonkey クローンを構築するという仮説のもと、その基本的な情報を整理した。次年度その開発可能性を探る。

２．CodeMonkey の概要

２．１　CodeMonkey とは

（１）コンセプト

「コードモンキーのストーリーは、サルの「モンタ」がゴリラにバナナを奪われてしまうところから冒険が始まります。友だちのカメやヤギと協力して道中様々な障害物に遭遇しながらバナナを獲得していきます。例えば、茂みで道がふさがれていたり、ネズミに邪魔されたり、ときにはカメの背中に乗って川の対岸を目指します。しかし、モンタは人間の言語を理解できません。効率よくバナナを獲得するには、問題点と解決策を見つけ、コンピューター言語の「コード」で表現しなければなりません。関数や繰り返し処理を使ってモンタに指示を出しながら、プログラミングの基礎を学習します。」（CodeMonkey カタログより）

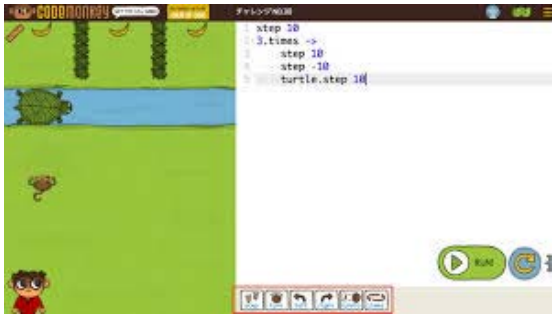


図１　基本画面

図２　指示画面

（２）教材としての CodeMonkey

CodeMonkey は、ゲーム仕立てのプログラミング学習教材である。ゲームとして表現すれば、RPG の一種である。「ゲームの攻略を通してコーディング演習を行っていくタイプ」

の学習コンテンツである。中心となる「コードの冒険」は次のような構成となっている。

ゲームを進んでいくために、コンピュータ言語を段階的に習得していく。半完成のコードが学習者に示され（図１）、システム側から指示がある（図２）。学習者はそれをヒントにプログラムを書いて実行すると、画面上の主人公がプログラムの通りに動く。目的は主人公が画面上にあるすべてのバナナを取ることである。目的を達するとそのチャレンジ（問題のこと）はクリア（合格）になる。プログラムはシステムによって評価され（図３）、適切なアドバイスが与えられる（図４）。for や if といった一つの学習項目について、約１０題の問題（チャレンジ）を解いていく。問題は徐々に難しくなるがステージの全チャレンジを解き終わると、次の学習項目に移る。チャレンジの中には、学生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いるかを確認するアセスメント（評価）チャレンジもある。全部で 17 の学習項目があり、合計２１０題の問題（チャレンジ）を解く。チャレンジの地図（図５）があり、自分の進捗が確認できる。応用問題は「スキルモード」と呼ばれている。スキルモードまで解けば、全部で４２０題以上の問題を解くことになる。なお、教師が学習者を集中的に管理できる管理画面（図６）があり、進捗状況の確認や評価の確認などができる。

図３　システムによる評価

図４　アドバイス

図５　ステージマップ

図６　管理画面

「コードの冒険」の１チャレンジは３から５分で解けるように設計されており、１レッスンは約１０チャレンジ（メインモード）なので、メインモードだけなら３０から５０分、スキルモードもするなら６０から１００分である。１７レッスン全体にかかる学習時間は合計で最短 10 時間半から 35 時間である。

（3）コンピュータシステムとしての CodeMonkey

コードモンキーの対象としている言語や学習項目、動作環境などについては、表 1 の通りである。

表 1　CodeMonkey の概要

言語／ 学習スキル	CoffeeScript ／オブジェクト、引数、変数、配列、ループ、FORループ、UNTILループ、関数、IFとIF ELSE文、ブーリアン、AND/OR、ファンクションコールなど
対応機器	インターネットに接続可能なWindows、Mac、タブレット ※スマートフォンには未対応。
推奨ブラウザ	Windows、Mac：Firefox、Chrome、Safari、Internet Explorer タブレット：Chrome、Safari ※タブレットは、古いモデル（2014年以前発売）では十分なスピードが出ない場合がある。
対応言語	日本語、英語、ロシア語、中国語（簡体字）、メキシコスペイン語、中国語（繁体字）、アルゼンチンスペイン語、スペイン語、韓国語、ユダヤ語、アラビア語、ポルトガル語、タイ語、ベトナム語、インドネシア語、タガログ語、ミャンマー語、トルコ語、カンボジア語、ラオ語、全20言語。この他にコミュニティによって翻訳された10言語がある。
構成	メインモード（本編）／スキルモード（練習編） チャレンジ・ビルダー（オリジナルのコーディング課題を作成） ゲーム・ビルダー
レッスン数	レッスン１７、チャレンジ420以上（タブレットでは、チャレンジ165、スキルモード8-15まで対応。）
レベル／対象年齢	未経験者から中級者／ 9歳から
学習時間（目安）	1つのチャレンジは3〜5分程度で解けるように設計されている。 ※１レッスン　メインモードだけなら30から50分、スキルモードもするなら60から100分。合計17レッスンで構成されており、最短10時間半から35時間程度。
サポート	3つ星の解答例ページの提供。

「CoffeeScript」は、Ruby 風・Python 風のオブジェクト指向プログラミング言語である。JavaScript を自動生成するためのスクリプト言語。JavaScript の半分近いコード量でセキュアかつ高性能な JavaScript のコードに変換して使用できる。

（4）CodeMonkey による学習の特徴

a. 学習進度がひと目で分かる

210 以上ある学習内容をカテゴライズしてマップ上に「見える化」しているので、学習進度がひと目で分かる。進捗が一目で分かりやすいと、目標も立てやすく、効率的な学習が期待できる。

b. 結果が分かりやすい

書いたプログラムは、モンタの動きとなって即座に画面上に出力される。ミスのあるコードを書けば、ゲームをクリアすることができないため、簡単に「問題点」や「改善策」を見つけられる環境になっている。

すぐに結果を見ることができることは、すぐに答え合わせができるという意味でもあり、効率的に学習を進めることができる。もちろん、正答のコードも用意されている。

c. 学習コンテンツの追加

CodeMonkey の「コードの冒険」には、現時点で「210」以上のチャレンジ（問題）がある。今後も新しいチャレンジが追加されていくとアナウンスされている。「コードの冒険」以外に、「ゲームビルダー」というコンテンツも提供されており、その追加・拡充もアナウンスされている。

d. 独学だがひとりではない

モンタやそれ以外の個性的なキャラクター達がミスを正し、コードの品質を評価し、アドバイスしてくれる。

e. 環境を選ばない

CodeMonkey はブラウザ上で動作するので、コンピュータにソフトウェアなどをインストールする必要はない。また、CodeMonkey のチャレンジはそれぞれ 3 ～ 5 分程度に設計されているので、日々の隙間時間にプログラミングが学べる。但し、スマートフォンには対応していない。

２．２　運営主体

（１）開発元

コードモンキー・スタジオは、イスラエルのテルアビブ市で 2014 年に設立されたスタートアップ企業で、学校教育用のプログラミングゲームの開発と運営を行ってきた。CodeMonkey の開発元である。CodeMonkey は、同国の小中学校 1700 校に標準教材として導入され、米国、英国、フランス、中国など全世界でユーザー数は 800 万人を超えた（CodeMonkey HP、2019 年 3 月確認）。年齢層は小学生から 70 歳まで様々である。

（２）日本代理店

ジャパン・トゥエンティワン（以下、J21）は、CodeMonkey の日本代理店である。1992 年 9 月に創業し、”イノベーションを市場化する”を掲げ、イスラエルを中心に世界最先端のハイテク企業の技術や製品のビジネス展開を日本で行っている。

（３）中国の TAL グループによる買収

コードモンキー・スタジオは 2018 年 12 月に、中国の北京に本社をおく NASDAQ 上場企業である教育サービス会社の「TAL Education Group」によって買収された。買収金額は 2000 万ドル（約 22 億円）だった。TAL Education Group は、中国の 40 以上の都市でラーニングセンターを運営する教育技術会社で、230 万人以上の学生にサービスを提供している。TAL は、1,800 万人以上のユーザーを擁するオンライン学習プラットフォームも運営している。同社は時価総額 167 億ドルの会社である。買収後も、CodeMonkey は引き続き独立した子会社として機能し、中国およびその他の市場向けに研究開発を行うとしている。

出典：TAL Education Group Buys Codemonkey for $20 Million
https://www.calcalistech.com/ctech/articles/0,7340,L-3751307,00.html
China's TAL Education buys Israeli learning start-up Codemonkey
http://global.chinadaily.com.cn/a/201812/05/WS5c0765a8a310eff30328f371.html

J21 によると、TAL による買収は日本市場へは影響がなく、今後も少なくとも数年はこれまで同様のサポートや商品の供給が行われるとのことである。

２．３　CodeMonkey の学習コースの種類

（１）コードの冒険

CodeMonkey の中心的なコンテンツである。コンピュータ言語である CoffeeScript の言語とアリゴリズムを段階的に学ぶ。基本を学ぶ「メインモード」と、応用問題を解く「スキルモード」がある。コンピュータプログラマを目指す場合、両モードを学んでおくのがよい。

１７レッスンは次のように構成され、それぞれ約１０チャレンジで構成されていえる。

表２　コードモンキーのコードの冒険のレッスン構成

No	タイトル（概要）	メインモード	スキルモード
1	はじまりの森（コード入力の基本）	10	11
2	オブジェクトの平原（オブジェクトの概念）	10	10
3	ループ・アイランド（loop）	10	10
4	バリアブル・バレー（variable）	20	20
5	アレイの沼（array）	10	10
6	フォーの森（for）	15	15
7	ファンクション牧場（function）	15	15
8	アンティル砂漠（until）	15	15
9	イフの北国（if）	10	11
10	雪の町オアエルス（or else）	8	8
11	ブーリアンの雪山（boolean）	20	21
12	城下町ノット（not）	6	6
13	コンパリソン・ストリート（comparison）	8	9

14	リターン・アベニュー（return）	15	15
15	成功への鍵	14	14
16	ネズミを逃がしたのは誰？	6	6
17	クリックしてね！	15	15
計		210	214

チャレンジでは、右側にコーディング画面、左側にバナナと主人公モンタが表示される。同時に何をすればよいのか、指示やヒントが表示される。

図７　チャレンジの画面

図８　すこし複雑になったチャレンジの画面

（２）ゲームビルダー

「チャレンジをつくる」は自前のチャレンジをつくれるコース。トードーの算数は、初歩的な算数の確認ができるコースである。距離や角度などでつまずく学生の学習に利用でき

る。トリビアチャットボットは、Python でチャットボットをつくるコースである。メッセージが英語なので利用勝手が悪い。Moon Lander では物理が学べるが、まだ英語モードしかない。その他のコースはゲームづくりの方法を学ぶコースである。ゲームづくりそのものが目的の場合、情報系専門学校では他にも代替的な手段があり、あえて使う価値は低い。

a. チャレンジをつくる

内容

「コードの冒険」のチャレンジを作成することができる。作成したチャレンジは友達や先生と共有できる。

b. トードーの算数　（基礎、距離、角度、掛け算）

内容

加算、減算、測定、コーディングを組み合わせたコース。基本の 1 時間のコース、距離について学ぶコース、分度器で角度を測るコース、掛け算の基礎を練習するコースがある。いずれもゲーム仕立てになっている。各コースは２０チャレンジ。

c. トリビアチャットボット

内容

基本的なチャットボットのプログラミングを学べる、およそ 1 時間のコース。Python 言語によるコース。サルのモンタ・チャットボットが出すトリビア（豆知識）を問う質問に、ユーザーが回答、チャットボットが正否を判定して、チャットの最後に正答数に応じたメッセージを自動的に返す、という簡単なチャットボットの作り方を学ぶ。コード・チャットボットの簡略版。演習数１６。

コースは日本語化されているものの、チャットのメッセージの本体が英語で、日本語には対応していない。

d. コード・チャットボット

内容

単語当てゲームをしてくれるチャットボットをプログラミングする。人気のプログラミング言語 Python をマスターできる。演習数 70 以上。ライセンスの関係で評価できなかった。

e. Moon Lander

内容

コーディングしながら物理学を学ぶ。 Moon Lander は、物理ベースのゲームを構築する方法を学ぶコース。このゲームでは、宇宙船を安全に月面に着陸させるために、コー

ドと 6〜9 年生（日本の小 6 から中学）の物理学の概念を使用する。演習数 17。まだ日本語版がない。

f.プラットフォーマー

内容

衝突時のアクションなど、ゲーム作成の基本を学ぶ。演習数３５。プラットフォーマーコースを簡略化した「ゲームデザイン」コースもある。演習数１５。

学習スキル

キーボードで自機スプライトを動かす、ゲーム背景の編集、ゲームを横スクロール可能にする、敵キャラを無限ループで動かす、スコアの計上、パワーアップアイテムの作成、など

g.フロッガー

内容

タブレットやスマートフォンでも遊べるゲームを作成する。演習数３０。

学習スキル

スワイプで自機スプライトを動かす、敵キャラが画面外に消えていくような効果、自機の強制移動、関数によるコードの整理、ライフの減少、アイテムによるライフの追加、など

h.スプライトアニメーション

内容

画像と、ビジュアルの連動性を学習できる。演習数２６。

学習スキル

新しいスプライトシートの作成、フレームの複製、アニメーションの定義、アニメーションの開始・停止、フレームを自分で描く、フレームレートの設定、など

i.ゲームをつくる

内容

f, g, h の 91 のステップで学習したスキルを使って、自分でオリジナルゲームを作る。作成したゲームは友達や先生をはじめ、インターネットにつながる全ての人と共有可能で、PC やタブレット、スマートフォンでプレイできる。テンプレート 6 種。

２．４　CodeMonkey の管理画面（管理者用ダッシュボード）

（１）グループ

学習者をグループ化して管理できる。クラス単位で学生を登録して管理する。

図９　グループ画面

a. Lessons

最近増えた画面。教案と標準的な学習時間が表示される。コンテンツのロックとその解除ができる。各チャレンジへのショートカットがある。リファレンスカード、登場するキャラクターのレビューがある。教案などは英語である。

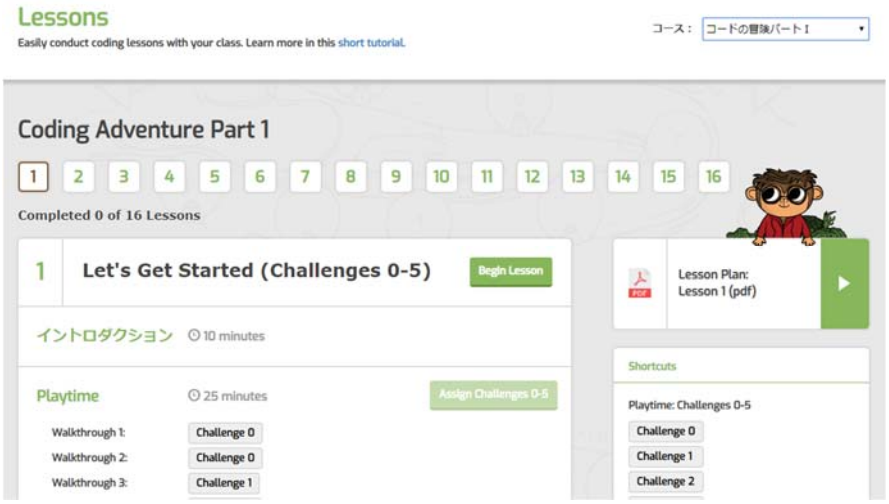


図１０　レッスン画面

b. 生徒管理

学生のアカウント管理をする画面。アカウントの作成、CSV による一括アップロード、グループコードの共有、パスワードのリセット、アカウントの ON/OFF（アーカイブ、学習記録は保存されるが学生はコンテンツにアクセスできなくなる)、アカウントの移動、アカウントの削除、ができる。

図１１　生徒管理画面

c. コース

どのコースを有効化・無効化するか選択できる。

図１２　コース画面

d. 進捗管理

学生の進捗の管理ができる。授業中にどのレッスンまで学生に進捗を許可するか「進捗制限（ロック）」ができる。ステージは学習項目別にレッスンとしてまとめられている。学生がコードをゼロから書くことで定着状態が確認できるアセスメントチャレンジがどれなのかも明示されている（チャレンジ番号が黄色で明示）。チャレンジ番号をクリックすると模範解答が表示される。学生の学習状況は、「★」によって表され、色とアイコンにより評価が５段階でわかる（星３，星２，星１、失敗、トライしていない）。

図１３　進捗管理画面

学生の進捗表は、学生の名前順、進捗順にソートできる。進捗表を CSV でダウンロードすることも可能である。「★」をクリックすると、学生の書いたコードを確認できる。これにより、より詳細に何でつまずいたのか確認できる。

e. 評価一覧

学生の評価を 10 段階でレッスン別に一覧表示できる。評価一覧はダウンロード可能。

図１４　評価一覧画面

f. ショールーム

学生が作成したゲーム作品やチャレンジ作品が表示される。

g. My Classrooms

グループ別に学生数、週間および月間アクティブユーザー数が表示される。クラスデータの CSV 出力、学生の一括移動、クラス名の変更ができる。

図１５　マイクラスルーム画面

（２）My Activities

教師個人がコンテンツへアクセスするショートカット。進捗状況も表示されている。

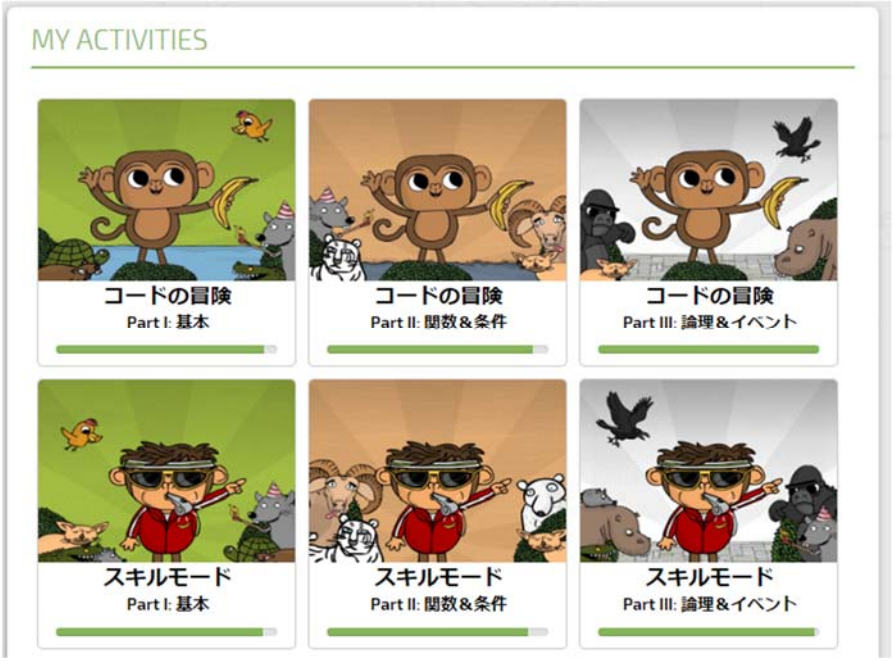


図１６　マイアクティビティ画面

（３）My Games

教師個人が作成したゲームが表示される。

（４）My Challenges

教師個人が作成したチャレンジが表示される。

（５）アチーブメント

教師個人が獲得したバッジが表示される。バッジの獲得には各レッスンの全チャレンジを「★」３つでクリアしなければならない。

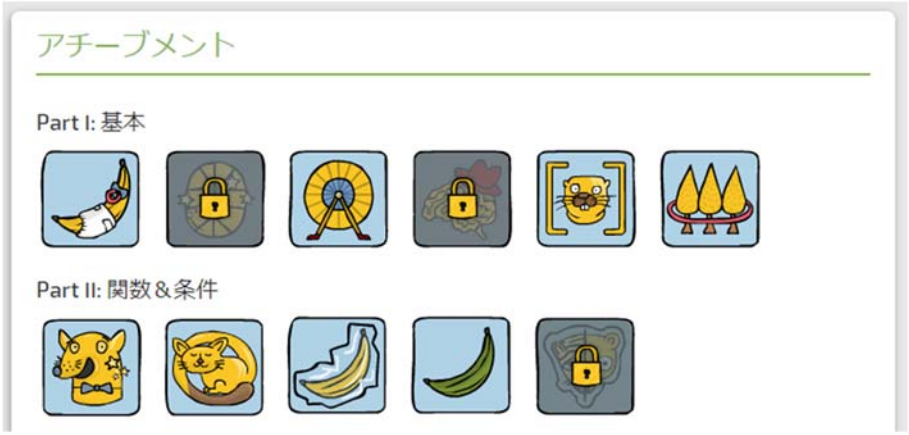


図１７　アチーブメント画面

（６）Discover

世界中のユーザーが作成したゲームやチャレンジが表示される。自分がつくったものも、ここで公開することができる。

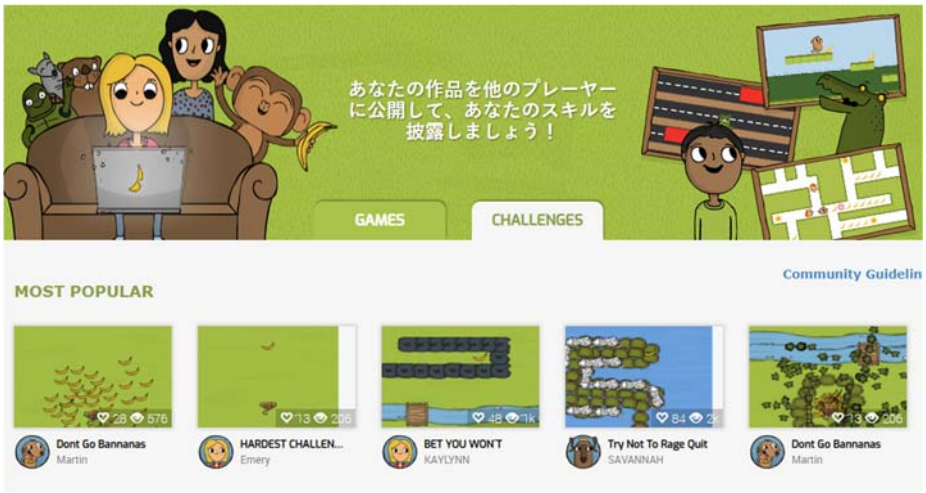


図１８　ディスカバー画面

（７）教師用のツール

次のツールが用意されている。

a. 英語の資料

以下の８つのツールがあるがいずれも英語の資料である。

① Getting Started（コードモンキー全体のイントロダクション、英語）

② Lesson Plans（教案、英語）

③ Hour of Code（アワーオブコード、各コンテンツのイントロダクション、英語）

④ ビデオチュートリアル（動画による解説、英語）

⑤ New Features（最新情報、英語）

⑥ Certificates（修了証書、英語）

⑦ Questions（FAQ、英語）

⑧ Media（マスメディアに載った記録、教室に貼るポスター、いずれも英語）

b. カリキュラムガイド

英語資料の①、②、③に対応すると考えられる。カリキュラムガイドは日本語である。内容はチャレンジ 105 までである。コンピュータ言語がわかっている教師であれば、一通り目を通しておけばよい。内容は、小中学校での展開が想定されている。ソフトウェア開発の専門家を目指す場合、このガイドでは不十分である。

目次は以下の通り。

図１９　カリキュラムガイドの目次

c. リファレンスカード

カリキュラムガイドに含まれている。

d. 登場キャラクターガイド

カリキュラムガイドに含まれている。

e. 模範解答

カリキュラムガイドには含まれておらず、進捗管理画面からアクセスする。

（8）学習者用のツール

a. ヘルプ（学習スキルの解説）

主要な関数や予約語の解説。

b. よくあるご質問

学習前のよくある質問への応答。

c. ブログ

日本語版の最新情報。

https://codemonkey.jp/cm_news/

d. Twitter

日本語版の最新情報。

https://twitter.com/codemonkey_jp

e. Facebook

英語版の最新情報。

https://www.facebook.com/CodeMonkeySTU/

（9）その他の情報

a. 攻略動画

Youtube を「CodeMonkey」で検索すると、プレイ画面が多数アップされている。

例えば、1-200 の全チャレンジを★3 でクリアした動画。

https://www.youtube.com/watch?v=wVSv1dw2aDM

b. CoffeeScript の情報

コードモンキーに使われている CoffeeScript の情報については、この記事が参考になる。

教育向けプログラミング言語としての CoffeeScript

https://qiita.com/Gord_4/items/5415c90671c8cc3c92e4

この記事によると、CodeMonkey に登場する「整数.times」というループは、CoffeeScript の言語仕様にはなく、CodeMonkey 専用の命令である。

２．５　CodeMonkey の費用

（１）個人ライセンス

管理用画面は付属しない。５ライセンス以上の購入で割引がある。

a. コードの冒険

30 ステージまで無料

年間ライセンスは初年度 6,480 円、2 年目以降は 4,320 円

プレイ可能なコース

・ コードの冒険 Part 1 & 2 & 3（全 420 ステージ）

・ チャレンジを作る

b. ゲームビルダー

年間ライセンス 6,480 円

プレイ可能なコース

・ プラットフォーマー

・ フロッガー

・ スプライト作成

・ ゲームをつくる

c. コードの冒険&ゲームビルダー

年間ライセンス 12,960 円

プレイ可能なコース

・ a,b のすべてのコース

d. コード・チャットボット

年間ライセンス 3,240 円

プレイ可能なコース

・ コード・チャットボット（全 74 エクササイズ）

（２）法人向けライセンス

予備校、学習塾、パソコンスクール、語学学校、その他一般企業などが対象。管理者用ダッシュボード（教師アカウント）が付属する。

エントリー	アドバンス
￥2,430（税込/ 1 名）	￥6,480（税込/ 1 名）
6 カ月	1 年
145 ステージ	420 ステージ

コードの冒険 Part 1 & 2 （メインモードのみ）	コードの冒険 Part 1 & 2 & 3 チャレンジをつくる
教師アカウント ＋ 生徒アカウント	同左
最低購入数：生徒 10 名以上	同左

※4 カリキュラムガイド（￥2,160/税込・冊）

（3）学校向けライセンス

学校法人や地方行政の教育組織などが対象。管理者用ダッシュボード（教師アカウント）が付属する。

ライセンス内容は法人向けと同様だが、価格がオープン価格となっている。

３．CodeMonkey の授業の実施結果

本事業に参加している清風情報工科学院と九州技術教育専門学校は、我が国でこの教材を先駆的に採用して利用している専門学校である。この度、その実施結果を整理し、良好な結果が得られているのか、問題があるとしたら何が原因でどう対策すればいいのか、その方向性を明らかにする。

実施結果をまとめるに先立って、CodeMonkey の学習項目を簡単に整理しておく。そのうえで、実施校の担当講師にインタビューし、実態を探る。

３．１　CodeMonkey の学習項目

実施結果をまとめるに先立って、CodeMonkey の学習項目を簡単に整理した。

表１　CodeMonkey の学習項目

チャレンジ	レッスン	名称	主な処理	意味
0～10	1	step，turn	step 10 turn 45 turn right	10歩進む 45°向きを変える 右を向く
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	バナナの方に向きを変える カメがバナナの方に向きを変える カメが10歩進む
21～30	3	timesループ	3.times ->　　turn left　　step 5	左を向く、5歩進む を3回繰り返す
31～50	4	変数	a = 10 step a b = distanceTo banana	=の値を代入する
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	ビーバー[0]が10歩進む バナナ[1]の距離進む
61～75	6	forループ	for b in bananas　　turnTo b　　step distanceTo b	バナナの方に向きを変える バナナの距離進む をbananas[0]～順に繰り返す
76～90	7	関数	goto = (a) ->　　turnTo a　　step distanceTo a	関数の定義
91～105	8	untilループ	until near match　　turnTo match　　step 1	マッチに近づくまで マッチの方に向きを変える 1歩進むを繰り返す

表１　CodeMonkey の学習項目（つづき）

チャレンジ	レッスン	名称	主な処理	意味
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	もしバナナが凍っている場合は ヤギがバナナの場所に行き ヤギが氷を砕く
117～124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	もしバナナが緑色の場合は ヤギがバナナの場所に行き そうでない場合は サルがバナナの場所に行く
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	クマとトラが眠るまで 待つ クマが眠るか、遊ぶまで 待つ
146～151	12	not	if not banana.rotten() 　　goto banana	もしバナナが腐っていない場合は バナナの場所に行く
152～160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	もし体力が70より少ない場合は 回復場所に行き 体力が100になるまで 待つ
161～175	14	戻り値	return no return yes return not crow.watching()	定義された値を返す
176～189	15	onKey	onKey = (key) -> 　　if key == 'w' 　　　　step 1	Wキーを押した場合 1歩進む
190～195	16	onMouseMove	onMouseMove = (pos) -> 　　bat.setX pos.x 　　bat.setY pos.y	マウスのX軸とY軸の定義
196～210	17	onClick	hippo.onClick = () -> 　　monkey.turnTo hippo 　　monkey.toss()	カバをクリックすると サルがカバの方を向き サルが投げる

３．２　九州技術教育専門学校

（１）九州技術教育専門学校

ａ．概要

九州技術教育専門学校（きゅうしゅうぎじゅつきょういくせんもんがっこう）は、熊本県人吉市と熊本市に学校を置く専門学校。略称「KTEC」「九州技専」。

コンピュータ技術をベースとした情報処理システム・組込みシステム・CG デザイン・医療情報・医療事務の分野で活躍する人材を育成・輩出している。

毎年多くの検定試験合格実績を持ち、特に情報処理技術者国家試験（基本情報技術者試験等）には力を入れており、県下専門学校内でも常に上位の成績を上げている。TOPPERS プロジェクトの特別会員になっている。

設置法人	学校法人赤山学園
理事長	赤山武興
副理事長・校長	赤山聖子

ｂ．沿革

1954 年（昭和 29 年）- 赤山講師養成所として創設。
1961 年（昭和 36 年）- 熊本県知事認可、人吉女子専門学校となる。
1976 年（昭和 51 年）- 熊本県で最初の専修学校認可校となる。家政専門課程被服学科、家政高等過程被服学科設置。
1986 年（昭和 61 年）- 工業専門課程情報処理科、コンピュータ秘書科設置。
1986 年（昭和 61 年）- 九州技術教育専門学校に名称変更。
1991 年（平成 3 年）- 学校法人赤山学園に設置者変更認可、学科名変更（工業専門課程は情報システム科、情報処理秘書科、家政専門課程はファッションデザイン科となる。
2005 年（平成 17 年）- 熊本市細工町に「熊本校」を開校（4 階建)。
2008 年（平成 20 年）- 学科名変更（総合情報マルチメディア科を情報システム工学科、マルチメディア科を CG デザイン学科、情報処理秘書科を医療情報学科に)。
2016 年（平成 28 年）-熊本校に高等課程（情報システム科）を設置

ｃ．設置学科

専門課程

情報システム工学科 - 情報システムコース、組込みシステムコース
情報ビジネス学科 - Ｗｅｂビジネスコース
医療情報学科 - 情報医療秘書コース

高等課程

情報システム科

（２）実施状況

九州技術教育専門学校では、平成 29 年度から CodeMonkey を利用しており、平成 30 年度は、次の学科で実施した。いずれも、祝竜児講師が務めている。

高等課程１年　　生徒１４名

専門課程１年　　学生　４名

（３）インタビュー結果

表２　九州技術教育専門学校での問題点

チャレンジ	レッスン	名称	主な処理	問題点（九州）
0～10	1	step，turn	step 10 turn 45 turn right	
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う）
21～30	3	timesループ	3.times -> 　turn left 　step 5	
31～50	4	変数	a = 10 step a b = distanceTo banana	
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	変数と配列の違いがわからない（bananaとbananasの「s」の有無で変数か配列かが異なるということへの戸惑い）
61～75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b	
76～90	7	関数	goto = (a) -> 　turnTo a 　step distanceTo a	goto以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。

表２　九州技術教育専門学校での問題点（続き）

チャレンジ	レッスン	名称	主な処理	問題点（九州）
91～105	8	untilループ	until near match 　　turnTo match 　　step 1	関数の戻り値をそのまま繰り返しの判定としているため、ここを曖昧にして進んだ場合に125からのbooleanで再度同じようにひっかかってしまう。
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	
117～124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	
146～151	12	not	if not banana.rotten() 　　goto banana	
152～160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	
161～175	14	戻り値	return no return yes return not crow.watching()	
176～189	15	onKey	onKey = (key) -> 　　if key == 'w' 　　　　step 1	
190～195	16	onMouseMove	onMouseMove = (pos) -> 　　bat.setX pos.x 　　bat.setY pos.y	
196～210	17	onClick	hippo.onClick = () -> 　　monkey.turnTo hippo 　　monkey.toss()	

（4）考察

・レッスン 2 の問題は、オブジェクト指向言語の仕様に由来する問題である。

・レッスン 5 の問題は、変数と配列の違いと、英語の文法の 2 つの問題の複合問題である。

・レッスン 7 の問題は、CodeMonkey の中の goto 以外関数がいまひとつピンときにくいという問題がひとつである。もうひとつの問題は、CodeMonkey の関数の引数が単なる変数なのかオブジェクトなのかわかりにくいという問題である。その上で、関数の引数が配列などのオブジェクトである場合の処理を明確化していないという問題である。

・レッスン 8 の問題は、ループの継続・終了条件の判断において、このレッスンのチャレンジが関数の戻り値を継続・終了条件にするものから始まっており、ほぼそれに終始するからである。

３．３　清風情報工科学院

（１）概要

ａ．概要

清風情報工科学院（せいふうじょうほうこうかがくいん）は、大阪府大阪市阿倍野区丸山通に所在する専門課程の専修学校。前身は、大阪ガスが経営していたオージスコンピュータ学院専門学校。

設置法人	学校法人清風明育社
理事長	平岡龍人
副理事長・校長	平岡憲人

ｂ．沿革

1988 年 3 月 11 日	オージー総合学園、オージスコンピュータ学院専門学校が大阪府より設立認可を受ける
2001 年	学校法人清風学園専務理事である平岡龍人が理事長に就任。
2002 年	校名を清風情報工科学院に変更。
2003 年	日本語学科を増設。

ｃ．設置学科

工業専門課程

デザイン・コンピュータ学科

研究コース（4 年制）

応用コース（3 年制）

基礎コース（2 年制）

グローバル教養学科（2 年制）

キャリア専攻科（1 年制）

文化・教養専門課程

日本語学科（1.5／2 年制）

（２）実施状況

平成 30 年度から CodeMonkey を利用しており、平成 30 年度は次の学科で実施した。

デザイン・コンピュータ学科１年	A クラス	学生　32 名	担当　山田茂
	B クラス	学生　37 名	担当　林耕平
デザイン・コンピュータ学科２年	ハードコース	学生　15 名	担当　林耕平

デザイン・コンピュータ学科１年には、留学生（ベトナム、中国、インドネシアなど）も在籍しているが、学生らは必要に応じて表示言語を母国語に変えて受講していた。

担当講師は進度（レッスンのロック）の管理を行い、必要に応じて個別指導する形式。

（3）インタビュー結果

授業は極めて好評だった。８割以上の受講生がメインモードの２１０チャレンジをクリアできた。従来の同程度の理解度への到達が３～４割程度に比べるとはるかに高かった。授業中も私語もなく、ゲームなどで遊ぶものもなく、必要に応じて隣同士教えあって進捗できた。

大きな問題点は、他の授業との接続である。具体的には、他の授業で行っているアルゴリズムとの連携がとられなかった、また他の授業で行っている C 言語の授業との連携がとられなかった。そのため、単に CodeMonkey または CoffeeScript が上達しただけという学生がかなりの数いた可能性がある。これについては、今後検証する。

各レッスンにおける問題点を次表にまとめた。これは、担当講師が授業中に学生の質問などか受けた印象に加えて、学生の書いたコードに対するシステムの評価（★の数）をたよりに、星の数の少ない学生が実際に書いたコードから洗い出したものである。

表３　清風情報工科学院での問題点

チャレンジ	レッスン	名称	主な処理	問題点（清風）
0～10	1	step，turn	step 10 turn 45 turn right	定規の正しい使い方
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	turnとturnToの違い 一人称（monkey）を省ける事
21～30	3	timesループ	3.times ->　 　　turn left 　　step 5	アルゴリズムの構築（バナナを取るまでの手順の明確化）
31～50	4	変数	a = 10 step a b = distanceTo banana	変数を使う意味が理解できない 代入した値をイメージできない
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	手順を間違えやすい
61～75	6	forループ	for b in bananas 　　turnTo b 　　step distanceTo b	forが繰り返しであるという理解 Tab（４つスペース）の意味
76～90	7	関数	goto = (a) ->　 　　turnTo a 　　step distanceTo a	goto以下はあくまで関数であるという理解 引数の理解
91～105	8	untilループ	until near match 　　turnTo match 　　step 1	条件を満たすまでの繰り返しであることの理解

表３　清風情報工科学院での問題点（続き）

チャレンジ	レッスン	名称	主な処理	問題点（清風）
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	条件の定義
117～124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	条件の定義
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	andとorの違い
146～151	12	not	if not banana.rotten() 　　goto banana	notを書く位置（文法的な）
152～160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	大なりと小なりの違い 以上と以下と違い、該当の数字は含まない事
161～175	14	戻り値	return no return yes return not crow.watching()	値を返すという意味自体がわからない
176～189	15	onKey	onKey = (key) -> 　　if key == 'w' 　　　　step 1	有利に進めるキーの設定（キーをどれだけ増やしても星３とれる）
190～195	16	onMouseMove	onMouseMove = (pos) -> 　　bat.setX pos.x 　　bat.setY pos.y	マウスの動きに関する設定である事
196～210	17	onClick	hippo.onClick = () -> 　　monkey.turnTo hippo 　　monkey.toss()	対象物をクリックした場合の命令であること

（４）考察

・CoffeeScript を学ぶのでなく、アルゴリズムを学ぶのであるという意識が必要

・流れ図との接点が必要

・C 言語など他の言語との接点の認識が必要

・レッスン２，６，８，１２については、命令語や関数名が英語であることから来ている可

能性がある。
・レッスン１，１３については、算数・数学の問題から来ている可能性がある。
・レッスン３，４，５，６，７，８，１４についてはコンピュータ言語一般・アルゴリズムに由来する問題である。
・レッスン２，１５，１６，１７については、オブジェクト指向やイベントドリブンなどを許容するモダンコンピュータ言語に由来する問題である。

３．４　まとめ

（１）レッスン別の問題点のまとめ

両校とも良好な結果を得ている。その上で、学生のつまずきポイントが明らかになった。

表４　両校の問題点のまとめ

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）
0～10	1	step，turn	step 10 turn 45 turn right	定規の正しい使い方
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う） turnとturnToの違い 一人称（monkey）を省ける事
21～30	3	timesループ	3.times -> 　turn left 　step 5	アルゴリズムの構築（バナナを取るまでの手順の明確化）
31～50	4	変数	a = 10 step a b = distanceTo banana	変数を使う意味が理解できない 代入した値をイメージできない
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	変数と配列の違いがわからない（bananaとbananasの「s」の有無で変数か配列かが異なるということへの戸惑い） 手順を間違えやすい
61～75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b	forが繰り返しであるという理解 Tab（４つスペース）の意味

表４　両校の問題点のまとめ（続き）

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）
76～90	7	関数	goto = (a) -> 　　turnTo a 　　step distanceTo a	goto以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。 goto以下はあくまで関数であるという理解 引数の理解
91～105	8	untilループ	until near match 　　turnTo match 　　step 1	関数の戻り値をそのまま繰り返しの判定としているため、ここを曖昧にして進んだ場合に125からのbooleanで再度同じようにひっかかってしまう。 条件を満たすまでの繰り返しであることの理解
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	条件の定義
117～124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	条件の定義
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	andとorの違い
146～151	12	not	if not banana.rotten() 　　goto banana	notを書く位置（文法的な）
152～160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	大なりと小なりの違い 以上と以下と違い、該当の数字は含まない事

表４　両校の問題点のまとめ（続き）

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）
161～175	14	戻り値	return no return yes return not crow.watching()	値を返すという意味自体がわからない
176～189	15	onKey	onKey = (key) -> 　if key == 'w' 　　step 1	有利に進めるキーの設定（キーをどれだけ増やしても星３とれる）
190～195	16	onMouseMove	onMouseMove = (pos) -> 　bat.setX pos.x 　bat.setY pos.y	マウスの動きに関する設定である事
196～210	17	onClick	hippo.onClick = () -> 　monkey.turnTo hippo 　monkey.toss()	対象物をクリックした場合の命令であること

（２）考察のまとめ

CodeMonkey をソフトウェア開発能力の獲得の基礎と位置づけ、情報処理関係の他の授業と協調して成果を上げる、という観点に立つと以下の改善点が見いだせる。

・CoffeeScript を学ぶのでなく、「アルゴリズムを学ぶのである」という意識が必要。
・命令語や関数名が英語であることからつまずく可能性の検討が必要。
・算数・数学の理解に問題があってつまずく可能性の検討が必要。
・CodeMonkey のレッスンの構成・ステップが不十分であることなら問題が生じる可能性の検討が必要。特に、以下の問題はすでに認識された：
　－変数と配列の違い
　－goto 以外関数がいまひとつピンときにくい
　－関数の引数が配列などのオブジェクトである場合の処理を明確化していない
　－ループの継続・終了条件の判断が関数の戻り値から始まっている
・コンピュータ言語一般・アルゴリズムの習得時につまずきやすい点は CodeMonkey でもつまずきやすい点である。例えば、変数の概念や関数の概念である。
・オブジェクト指向やイベントドリブンなどを許容するモダンコンピュータ言語の習得時につまずきやすい点は CodeMonkey でもつまずきやすい点である。
・流れ図との接点が必要。例えば、CodeMonkey 内のコードを流れ図で記述させるなどの工夫が必要。
・C 言語など他の言語との接点の認識が必要。とりわけ、他の言語と言語仕様が異なる部分には橋渡しが必要。例えば、for ループのカウンタの扱い、Until の扱いなど。
・より高度な理解のため、「コードの目的と手段」の意識が必要。

４．CodeMonkey のカリキュラムとその改善方法

本節では、第３節から得られた視点に基づき、CodeMonkey を用いた授業において、つまずきを軽減し、ソフトウェア開発能力をより確実かつ高度に獲得させる方法について整理する。

次年度、この方法を各校で授業に反映し、その効果を検証する。

４．１　CodeMonkey の学習項目（詳細）

（１）整理の観点

３．４節にて、つまずき箇所の解明の視点およびその改善の方向性についての視点が得られたので、それを再掲する。

- CoffeeScript を学ぶのでなく、「アルゴリズムを学ぶのである」という意識が必要。
- 命令語や関数名が英語であることからつまずく可能性の検討が必要。
- 算数・数学の理解に問題があってつまずく可能性の検討が必要。
- CodeMonkey のレッスンの構成・ステップが不十分であることなら問題が生じる可能性の検討が必要。特に、以下の問題はすでに認識された：
  - －変数と配列の違い
  - －goto 以外関数がいまひとつピンときにくい
  - －関数の引数が配列などのオブジェクトである場合の処理を明確化していない
  - －ループの継続・終了条件の判断が関数の戻り値から始まっている
- コンピュータ言語一般・アルゴリズムの習得時につまずきやすい点は CodeMonkey でもつまずきやすい点である。例えば、変数の概念や関数の概念である。
- オブジェクト指向やイベントドリブンなどを許容するモダンコンピュータ言語の習得時につまずきやすい点は CodeMonkey でもつまずきやすい点である。
- 流れ図との接点が必要。例えば、CodeMonkey 内のコードを流れ図で記述させるなどの工夫が必要。
- C 言語など他の言語との接点の認識が必要。とりわけ、他の言語と言語仕様が異なる部分には橋渡しが必要。例えば、for ループのカウンタの扱い、Until の扱いなど。
- より高度な理解のため、「コードの目的と手段」の意識が必要。

（２）分析の方針

本年度は時間の関係で、それぞれの視点について悉皆的に調査・分析することはできない。次年度の Monkey の授業に役立つ教材を生み出すことを暫定的な目標として分析する。次年度の授業で有効性を見極めて、必要性が高いものは次年度に悉皆調査する。

（１）の問題点について、分析の方針は以下の通りである。

ａ．命令語や関数名が英語

英語の意味と日本語の意味のズレ、英語の語順と日本語の語順のズレ、英文法特有の表現（複数の「s」など）に注目してコードや識別子を分析する。

b．算数・数学の理解

学生の算数などにおける基礎学力の不足に起因する問題である。例えば、定規や分度器の使い方、座標軸と回転の関係、座標の概念、ベン図の概念の有無、大小関係（不等式）である。またこの周辺のスキルとして、物事を丁寧に図化して整理したり、途中のプロセスを段階を追って見える化するスキルがある。これらを意識して分析する。

c．CodeMonkey のレッスンの構成・ステップが不十分

画面上で主人公の猿のモンタを動かしてアイテムをとってゆくというコンテンツのコンセプトに関わる問題と、学習項目同士の組み合わせが不適切であったり、十分な練習量が与えられていない、という問題であると推測される。前者に関しては、CodeMonkey 外で補足する方針でゆく。後者に関しては、学習項目の組み合わせに注目して分析する。

d．コンピュータ言語一般の問題

CodeMonkey の学習範囲から言えば、変数、代入、配列、関数、関数の引数、関数の戻り値、変数のスコープが主要なつまずき箇所であるので、そこに注目して分析する。

e．モダンコンピュータ言語の問題

「オブジェクト.メソッド」という語順が従来型のコンピュータ言語の語順に比べて日本人にはわかりやすいという利点がある。本年度は、引数にオブジェクトが渡されるという点に注目することにし、それ以外の点については本年度は分析しない。

f．流れ図との接点

CodeMonkey の学習範囲から言えば、ループ処理と判断処理のところが主である。この点に注目して分析する。

g．C 言語など他の言語との接点

CodeMonkey にあって他の言語にない言語仕様の部分が問題になる。とりわけ、ループとループカウンタの部分が大きく相違しているので、この点に注目して分析する。

４．２　CodeMonkey のカリキュラムの分析結果

（１）命令語や関数名が英語

中学校・高等学校で英語に苦手意識をもったり、英文法について十分な知識がない場合、日本語と英語の違いが原因となって、つまずく可能性がある。コードの冒険に登場するプログラムの予約語や関数、変数名などの識別子について、英語の意味と日本語の意味のズレ、英語の語順と日本語の語順のズレ、英文法特有の表現（複数の「ｓ」など）に注目して分析した。

表１　英語と日本語の相違に注目した予約語と識別子の主たる問題点

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（英語）
0〜10	1	step，turn	step 10 turn 45 turn right	定規の正しい使い方	・「step 10」は「10進む」で語順が逆。 ・語順が逆のものは、命令だから倒置法になっていると思えば、ある程度理解できる。「進め１０」「回れ４５度」「回れ右！」
11〜20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う） turnとturnToの違い 一人称（monkey）を省ける事	・「turnTo」の前置詞「to」がとれていない。日本語では「４５度回る」「バナナに向く」と動詞が変わって助詞がつく。英語ではそれが、turn, turnToに対応と説明。 ・「to」がつくと、命令と思っても語順が理解しにくくなる。（「回れバナナ」ではバナナが回ってしまう。「バナナ向けバナナ」とでも理解するのか。）
21〜30	3	timesループ	3.times -> 　turn left 　step 5	アルゴリズムの構築（バナナを取るまでの手順の明確化）	・「3.times」の語順は日本語と同じ「3回」。
31〜50	4	変数	a = 10 step a b = distanceTo banana	変数を使う意味が理解できない 代入した値をイメージできない	・「distanceTo ○○」は「○○までの距離」と語順が逆。 ・「step distanceTo ○○」は「○○までの距離を進む」で語順が完全に逆。
51〜60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	変数と配列の違いがわからない（bananaとbananasの「s」の有無で変数か配列かが異なるということへの戸惑い） 手順を間違えやすい	・配列は英語の複数形で「s」が付いていることに注意させる。
61〜75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b	forが繰り返しであるという理解 Tab（４つスペース）の意味	・「for」という英語がピンとこない。直訳すると「bがバナナ配列の中にいる間」。または「bのためにバナナ配列の中で繰り返す」と意訳する。
76〜90	7	関数	goto = (a) -> 　turnTo a 　step distanceTo a	goto以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。 goto以下はあくまで関数であるという理解 引数の理解	・「goto ○○」も「○○へ行く」と語順が逆。
91〜105	8	untilループ	until near match 　turnTo match 　step 1	関数の戻り値をそのまま繰り返しの判定としているため、ここを曖昧にして進んだ場合に125からのbooleanで再度同じようにひっかかってしまう。 条件を満たすまでの繰り返しであることの理解	・英語の語順だと「near match」になる。日本語なら「マッチの近く」と語順が逆。 ・「until○○」も「○○になるまで」と語順が逆。

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（英語）
106～116	9	if文	if banana.frozen() 　goat.goto banana 　goat.hit()	条件の定義	・「banana.frozen()」は「バナナが凍っている」で、語順が同じ。
117～124	10	if　else文	if banana.green() 　goat.goto banana else 　goto banana	条件の定義	・「banana.green()」は「バナナが緑色」で、語順が同じ。
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　wait() until bear.sleeping() or bear.playing() 　wait()	andとorの違い	・「bear.sleeping()」は「熊が寝ている」で、語順が同じ。 ・「A and B」は「AもBも」、「A or B」は「AかBか」。日本語の「AかBか」は「男か女か」のようにどちらか一方の意味で、コンピュータの意味と違うのに注意する。 ・「hasGlass()」「hasBowTie()」の「has」は日本語で「メガネをかける/する」「蝶ネクタイをつける/する」と動詞が違う
146～151	12	not	if not banana.rotten() 　goto banana	notを書く位置（文法的な）	・英語の「not」の位置がピンときにくいので、漢字熟語を例にして説明する。「不合格、不参加、非常識」などの「不○○」とか「非○○」のイメージ。
152～160	13	<，==	if health()<70 　goto healthZone 　until health()==100 　　wait()	大なりと小なりの違い 以上と以下と違い、該当の数字は含まない事	・「A>B」は小学校で「A大なりB」と教わるが、これも本来は語順が違う。「AがBより大きい」と「大きい」の部分が後ろに来るのが日本語の語順。 ・「healthZone」は「回復場所」で日本語の語順と同じ。直訳すると「健康場所」でちょっと変。
161～175	14	戻り値	return no return yes return not crow.watching()	値を返すという意味自体がわからない	・「return ○○」も、語順が逆で「○○を返す」。「戻り値は○○」と読み替えたほうがしっくりくる。 ・「crow.watching()」は「カラスが見てる」で語順が同じ。
176～189	15	onKey	onKey = (key) -> 　if key == 'w' 　　step 1	有利に進めるキーの設定（キーをどれだけ増やしても星３とれる）	・「onKey」も「キーが押されたら」と語順が逆。「押された」の部分は英語にはない。
190～195	16	onMouseMove	onMouseMove = (pos) -> 　bat.setX pos.x 　bat.setY pos.y	マウスの動きに関する設定である事	・「onMouseMove」も「マウスが動いたら」と語順が逆。 ・「setX」も「Xにセット」と語順が逆。
196～210	17	onClick	hippo.onClick = () -> 　monkey.turnTo hippo 　monkey.toss()	対象物をクリックした場合の命令であること	・「onClick」も「クリックされたら」と語順が逆。

表１　英語と日本語の相違に注目した予約語と識別子の主たる問題点（続き）

さらに、コードの冒険内で利用する予約語と識別子について、悉皆的に調べたものを付録１に示す。付録１は授業において、簡易な辞書（リファレンス）として利用できるよう、「英日」と「日英」のバージョンを作成してある。

（２）算数・数学の理解

学生の算数などにおける基礎学力の不足に起因する問題である。例えば、定規や分度器の使い方、座標軸と回転の関係、座標の概念、ベン図の概念の有無、大小関係（不等式）である。またこの周辺のスキルとして、物事を丁寧に図化して整理したり、途中のプロセスを段階を追って見える化するスキルがある。これらの該当箇所を分析した。但し、算数的な能力の不足については、他の補完方法（基礎学力対策）による量的な補強が必要であると考えられる。そのため、本年度は悉皆的な分析はしていない。

表２　算数・数学の学力不足に注目した主たる問題点

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（数学）
0〜10	1	step，turn	step 10 turn 45 turn right	定規の正しい使い方	・角度において「−」が右回りか左回りか。ノートに絵を書いて整理させる。 ・猿基準の長さ・角度しか出ない事への対処。定規・角度がピンと来ていない場合、「トードーの算数」をやらせる。
11〜20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う） turnとturnToの違い 一人称（monkey）を省ける事	
21〜30	3	timesループ	3.times -> 　turn left 　step 5	アルゴリズムの構築（バナナを取るまでの手順の明確化）	（手順を書き出すという習慣の有無）
31〜50	4	変数	a = 10 step a b = distanceTo banana	変数を使う意味が理解できない 代入した値をイメージできない	（数学の文字式とコンピュータの変数の概念の違い） （数学の等式とコンピュータの代入の概念の違い）
51〜60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	変数と配列の違いがわからない（bananaとbananasの「s」の有無で変数か配列かが異なるということへの戸惑い） 手順を間違えやすい	（手順を書き出すという習慣の有無）
61〜75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b	forが繰り返しであるという理解 Tab（４つスペース）の意味	
76〜90	7	関数	goto = (a) -> 　turnTo a 　step distanceTo a	gotoを以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。 gotoを以下はあくまで関数であるという理解 引数の理解	（数学の関数の概念とコンピュータの関数の概念の違い）
91〜105	8	untilループ	until near match 　turnTo match 　step 1	問題点（まとめ）	
106〜116	9	if文	if banana.frozen() 　goat.goto banana 　goat.hit()	条件の定義	
117〜124	10	if　else文	if banana.green() 　goat.goto banana else 　goto banana	条件の定義	
125〜145	11	and，or	until bear.sleeping() and tiger.sleeping() 　wait() until bear.sleeping() or bear.playing() 　wait()	andとorの違い	（and, or の関係がわからない可能性）
146〜151	12	not	if not banana.rotten() 　goto banana	notを書く位置（文法的な）	

表２　算数・数学の学力不足に注目した主たる問題点（続き）

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（数学）
152～160	13	<，==	if health()<70 goto healthZone until health()==100 wait()	大なりと小なりの違い 以上と以下と違い、該当の数字は含まない事	（大小関係の不等号がわからない可能性）
161～175	14	戻り値	return no return yes return not crow.watching()	値を返すという意味自体がわからない	
176～189	15	onKey	onKey = (key) -> if key == 'w' step 1	有利に進めるキーの設定（キーをどれだけ増やしても星３とれる）	
190～195	16	onMouseMove	onMouseMove = (pos) -> bat.setX pos.x bat.setY pos.y	マウスの動きに関する設定である事	（座標がわからない可能性）
196～210	17	onClick	hippo.onClick = () -> monkey.turnTo hippo monkey.toss()	対象物をクリックした場合の命令であること	

（３）CodeMonkey のレッスンの構成・ステップ

画面上で主人公の猿のモンタを動かしてアイテムをとってゆくというコンテンツのコンセプトに関わる問題と、学習項目同士の組み合わせが不適切であったり、十分な練習量が与えられていない、という問題であると推測される。

前者に関しては、CodeMonkey 外で補足する方針でゆく。後者に関しては、学習項目の組み合わせに注目して分析する。以下、後者に関する分析結果を示す。分析結果は「メインモード」についてはほぼ悉皆的であるが、「スキルモード」については分析対象としていない。次年度必要に応じて、メインモードとスキルモードの全コードを対象に、悉皆的な分析を行う。

表３　学習項目の組み合わせ

チャレンジ	レッスン	名称	主な処理	組み合わせ
0～10	1	step，turn	step 10 turn 45 turn right	・メソッド 数値 ・メソッド 定数（rightなど） ・「（オブジェクト.)メソッド」の「オブジェクト.」部分が省略されている
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	・レッスン1では「monkey.」が省略されている。このレッスンの直前にプレステージP2を入れる。 ・メソッド オブジェクト ・オブジェクト.メソッド 引数 ・オブジェクト.メソッド オブジェクト
21～30	3	timesループ	3.times -> turn left step 5	・ループ関数(整数.times）とメソッド ・変数.times ・ループ関数とメソッド、オブジェクト.メソッド
31～50	4	変数	a = 10 step a b = distanceTo banana	・変数、変数への代入 ・変数の自己代入 ・distanceTo オブジェクト ・ループとメソッド、オブジェクト.メソッド ・追加チャレンジ（46-50）にて、ループと自己代入

表３　学習項目の組み合わせ（続き）

チャレンジ	レッスン	名称	主な処理	組み合わせ
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	・メソッド 配列の要素 ・オブジェクト.メソッド 配列の要素 ・配列の要素.メソッド 引数 ・配列とループの組み合わせはなし ・55から57は応用できる
61～75	6	forループ	for b in bananas 　　turnTo b 　　step distanceTo b	・forループと変数 ・forループと「メソッド 変数」 ・forループと「メソッド メソッド 変数」 ・forループと「変数.メソッド オブジェクト」 ・forループの入れ子 ・入れ子のforループの演習が不足（75のみ）
76～90	7	関数	goto = (a) -> 　　turnTo a 　　step distanceTo a	・関数()　引数なしの関数呼び出し ・関数 定数 で関数呼び出し ・関数の定義　（但し、goto関数のみ） ・関数の入れ子（関数内で「メソッド 引数」「メソッド メソッド 引数」） ・関数 配列 で関数呼び出し ・関数内でforループ ・関数 変数 で関数呼び出し ・88から応用できる ・２変数の関数定義、２変数の関数呼び出しも可能
91～105	8	untilループ	until near match 　　turnTo match 　　step 1	・最初から until near オブジェクトで始まる ・「until メソッド オブジェクト」 ・untilループ内で、「メソッド 定数」 ・untilループ内で、「メソッド メソッド 定数」 ・until 数式、until 関数がない ・「until オブジェクト.メソッド オブジェクト」 ・関数内でuntilループ ・untilループ内で、「メソッド 引数」 ・forループ内で関数呼び出ししその中でuntilループ ・何もしない wait()という関数 ・wait()との組み合わせで何もしないuntilループ ・関数の入れ子 ・forループ内でuntilループと関数の入れ子呼び出し ・何もしないuntilループと何かするuntilループの組み合わせ ・99から複雑
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	・オブジェクト.メソッド() ・オブジェクト.メソッド オブジェクト ・オブジェクト.メソッド 配列の要素 ・if オブジェクト.メソッド() ・if節の中で、オブジェクト.メソッド() ・if節の中で、オブジェクト.メソッド オブジェクト ・forループの中で、if 変数.メソッド() ・forループの中で、if節の中のオブジェクト.メソッド()、ブジェクト.メソッド オブジェクト

表３　学習項目の組み合わせ（続き）

チャレンジ	レッスン	名称	主な処理	組み合わせ
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	・until とandの組み合わせ ・forループとuntilループとandの組み合わせ ・forループとuntilループとandとif/else文の組み合わせ ・関数内でuntilループとandの組み合わせ ・until とorの組み合わせ ・forループとuntilループとorの組み合わせ ・forループとuntilループとorとif/else文の組み合わせ ・関数内でuntilループとorの組み合わせ ・for in [オブジェクト, オブジェクト] ・for文の入れ子 ・for文の入れ子の中で、for in [オブジェクト, オブジェクト] ・for文の入れ子の中からuntilを含む関数を呼ぶ ・チャレンジ140は応用できる ・チャレンジ141-145は追加チャレンジ ・for文の中のif文（if 引数.メソッド） ・for文の中のif文の中の引数.メソッド 定数 ・for文の中のif文で and ・for文の中のif/else文で and / or ・for文の入れ子の中でif文の中に or
146～151	12	not	if not banana.rotten() 　　goto banana	・if文の中でnot ・配列の思い出し ・for節の中でif/else文 ・for節の中のif文でnot
152～160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	・if文の中で メソッド() == 定数 ・until文の中でメソッド() == 定数 ・for文の中のuntil文の中でメソッド() == 定数 ・if/else文の中で メソッド() < 定数 ・for文中でif メソッド() < 定数 ・for文中のif文 の中で、until メソッド() == 定数 ・whileと組み合わせたほうがわかりやすいものもある
161～175	14	戻り値	return no return yes return not crow.watching()	・関数の中でreturn ・関数の中のreturnでnot ・if/else文の中で関数() ・161のステップが大きい（が、みんな問題ないようだ） ・for文の中でif 関数 引数 ・return文の中で メソッド()==定数 ・return文の中で メソッド() < 定数 ・for文中のif文 の中でuntil文 ・return文の中で or ・return文の中で 引数.メソッド() ・until文の中で関数 オブジェクト and 関数 オブジェクト ・return yes/no ・returnのあとは実行されない ・if/else文の中で 関数() ・return not の戻り値とuntilの組み合わせ ・return not とorと配列の組み合わせ ・return not とorと配列の組み合わせとifとforの組み合わせ ・for文とreturnとnotの組み合わせをuntilと組み合わせる ・for文とif文とreturnの組み合わせをuntilと組み合わせる ・whileと組み合わせたほうがわかりやすいものもある

表３　学習項目の組み合わせ（続き）

チャレンジ	レッスン	名称	主な処理	組み合わせ
176～189	15	onKey	onKey = (key) -> 　if key == 'w' 　　step 1	・イベントドリブン（onKey） ・onKeyとif文の組み合わせ
190～195	16	onMouseMove	onMouseMove = (pos) -> 　bat.setX pos.x 　bat.setY pos.y	・イベントドリブン（onMouseMove） ・onKeyとonMouseMoveの組み合わせ ・setX、setY ・onKeyとif、onMouseMoveとsetX, setYの組み合わせ
196～210	17	onClick	hippo.onClick = () -> 　monkey.turnTo hippo 　monkey.toss()	・イベントドリブン（オブジェクト.onClick） ・forループと変数.onClick ・forループと変数.onClickと変数.メソッド, メソッド 変数 ・moverという変数 ・onClickとonKeyの組み合わせ ・forループと変数.onClickとmoverとonKeyとifの組み合わせ

（４）コンピュータ言語一般の問題およびモダンコンピュータ言語の問題

CodeMonkey の学習範囲から言えば、変数、代入、配列、関数、関数の引数、関数の戻り値、変数のスコープが主要なつまずき箇所であるので、そこに注目して分析する。

「オブジェクト.メソッド」という語順が従来型のコンピュータ言語の語順に比べて日本人にはわかりやすいという利点がある。本年度は、引数にオブジェクトが渡されるという点に注目することにし、それ以外の点については本年度は分析しない。

必要に応じて、その対策も記述した。

表４　コンピュータ言語固有の問題

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（コンピュータ）
0～10	1	step，turn	step 10 turn 45 turn right	定規の正しい使い方	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。 ・語順が逆のものは、命令だから倒置法になっていると思えば、ある程度理解できる。「進め１０」「回れ４５度」「回れ右！」
	P2				・レッスン１で省略された部分を加えて「オブジェクト.メソッド」の練習 ・（紙上で、）レッスン１の題材のいくつかについて「monkey.」をつけて書かせる
11～20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う） turnとturnToの違い 一人称（monkey）を省ける事	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。駅から学校、玄関から教室までを例に説明する。単純化した地図をつくっておく。 ・レッスン１では「monkey.」が省略されている。この章の直前にプレステージP2を入れる。
21～30	3	timesループ	3.times -> 　turn left 　step 5	アルゴリズムの構築（バナナを取るまでの手順の明確化）	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。３階まで階段を登るなどを例に（繰り返しで表現できる）。
31～50	4	変数	a = 10 step a b = distanceTo banana	変数を使う意味が理解できない 代入した値をイメージできない	・変数の機能をイメージできていない。コンピュータでの文字・変数は「箱」なので、「箱」をつかって説明する。自己代入のi=i+1を「箱」で実感させる。

表４　コンピュータ言語固有の問題

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（コンピュータ）
	P4				・i=i+〇の練習量が不足 ・（紙上で、）５章の題材のいくつかについて変数と自己代入のi=i+〇を使って書かせる。 ★55-57をつかって、定数部分を変数に置き換えて変数を配列を組み合わせる練習、さらにi=i+1などと組み合わせる練習
51～60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	問題点（まとめ）	・手順をノートに書かせる。 ・配列と変数の違いを説明する。どちらも「箱」なのは同じ。グループとメンバーの関係。
61～75	6	forループ	for b in bananas 　　turnTo b 　　step distanceTo b	問題点（まとめ）	・「for」を使うメリットを説明する。Forを使わずに５回・１０回と回数が増えるとミスが多くなる。
	P6				★forループの入れ子の例題をいくつかやる必要あり
76～90	7	関数	goto = (a) -> 　　turnTo a 　　step distanceTo a	goto以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。 goto以下はあくまで関数であるという理解 引数の理解	★goto 配列（goto bananas）に対応するには？ ★チャレンジ88から90を利用して様々な組み合わせに挑戦させる ★チャレンジ88から90を利用してどういう目的でどういう分割がわかりやすいのか考えさせる（89の模範解答はモジュール分割がおかしい。【別解】） ・goto以外の関数のイメージをふくらませるためにレッスン追加
	P7				・関数のイメージの脹らませ。 ・紙上で、関数づくりの練習をさせる。いろんな関数、いろんな引数 ・コンビニへ行く ・駅で乗り換えの例（乗り換えTo 阪和線） ★引数が２つ以上のイメージの脹らませ。 ★コピー機の例（コピー（100%,5枚）） ★さらに引数として変数を渡した時の挙動（変数のスコープ） ★引数も日本語の語順が逆になる。「copy (100%, 5枚)」なら「100%で、５枚、コピーする」。
91～105	8	untilループ	until near match 　　turnTo match 　　step 1	関数の戻り値をそのまま繰り返しの判定としているため、ここを曖昧にして進んだ場合に125からのbooleanで再度同じようにひっかかってしまう。 条件を満たすまでの繰り返しであることの理解	★変数と関数の引数のスコープについて書き出して納得する（特にチャンレンジ９９以降）
	P8				・untilの直後に数式が来るパターンを練習する ・関数の戻り値を変数に入れて、その変数をuntilで評価する練習。わかりにくい場合はn == trueまで書かせる。 ・until i == 10, until i >=10 などの練習 ・until n, until n == trueなどの練習
106～116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	条件の定義	
117～124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	条件の定義	

表４　コンピュータ言語固有の問題

チャレンジ	レッスン	名称	主な処理	問題点（まとめ）	分析（コンピュータ）
	P10				★forと配列とgoto以外の関数（引数は変数）の組み合わせの練習 ★チャレンジ124の解き方をいろいろやらせる【別解】 関数使わずに書いてみる、 先に判定してからどっちが行くか決める、 先にバナナに行ってから判定する、とか ★returnを先にここで入れたほうがわかりやすいかも 凍ってるかの判定関数 青いバナナかの判定関数
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() wait() until bear.sleeping() or bear.playing() wait()		・「A and B」は「AもBも」、「A or B」は「AかBか」。日本語の「AかBか」は「男か女か」のようにどちらか一方の意味で、コンピュータの意味と違うのに注意する。
	P11				★この辺でも、forと配列とgoto以外の関数（引数は配列）の組み合わせ ★チャレンジ124の解き方をいろいろやらせる stepperでループする考えでやるとbananasを引数にすることになる ★チャレンジ140のやり方をいろいろやらせる【別解】 配列のtigersを引数にした関数 配列のbananasを引数にした関数
146～151	12	not	if not banana.rotten() goto banana		
152～160	13	<，==	if health()<70 goto healthZone until health()==100 wait()		
	P14				・紙上で、関数づくりの練習で、戻り値ありの場合をさせる。いろんな関数、いろんな引数で。 ・関数のイメージの脹らませを応用して説明。成功したか失敗したかを返す。 ・駅で乗り換えの例（乗り換えTo 阪和線　成功・失敗） ・コピー機の例（コピー（100%,5枚）成功・失敗） ・代金の例（代金（100円、５個）） ・消費税の例（消費税（8%､100円））
161～175	14	戻り値	return no return yes return not crow.watching()		（値を返すという例を別途用意して実感させることが必要）
176～189	15	onKey	onKey = (key) -> if key == 'w' step 1		（イベントドリブンな記述の仕方）
190～195	16	onMouseMove	onMouseMove = (pos) -> bat.setX pos.x bat.setY pos.y		（イベントドリブンな記述の仕方）
196～210	17	onClick	hippo.onClick = () -> monkey.turnTo hippo monkey.toss()		（onClickはオブジェクトに設定する。それがオブジェクトを対象とした文で宣言するということと、その実行はonClickイベントで発動されるということの発想の切り分けが難しい。他の部分は順次処理であるが、イベントドリブンな部分は割り込みになってその中では順次処理ということ。）

（4）流れ図およびC言語など他の言語との接点

流れ図については、CodeMonkey の学習範囲から言えば、ループ処理と判断処理のところが主である。言語仕様の相違については、CodeMonkey にあって他の言語にない言語仕様の部分が問題になる。とりわけ、ループとループカウンタの部分が大きく相違している。

a．文末の「；」などの記号の有無

CoffeeScript は、文末の「；」や「：」、節を表す「{}」、引数を表す「()」が不要であるなど、記述が簡略でよいという特徴がある。このことは、他の言語との相違であるが、初期の教育において、この簡略さは有利である。文末の「；」などは、他の言語を学ぶ際に追加して学べば良い。そのため、CodeMonkey 内では対策は考えない。逆に、他の言語の授業において、文末の記号などが必要であると教育することで対処する。

b．一定回数のループ（for ループ）

コンピュータ言語では一定回数のループを実現するために、ループカウンタをおいてそのカウンタをループごとに一定値で加減する必要がある。ループカウンタは、変数への自己代入が必須であり、加えて終了条件の判断が必要になる。「コードの冒険」では、この処理が煩雑であるため、「整数.times」「for 変数 in コレクション」の2つの命令を用いることで、ループカウンタの利用を回避している。

CodeMonkey 内ではこれで問題はない。ループのアルゴリズムの習得そのものでもこれで問題はない。しかしながら、C言語などの他の言語では、ループカウンタを意識する必要がある。またアルゴリズムの理解においても、ループカウンタを意識できるほうがよい。そこで、他の言語との接続を意識した追加の演習を用意することで対処する。

①変数への自己代入。自己代入がピンとこない学生を想定するなら、i++など加算演算子を用いた記述を教える。

②for (i=1; i<=10; i++)に対応した処理の仕方を教えておく。CoffeeScript では、for i in [1..10]、for i in [1..コレクション.length]の2つの表記が可能である。これにより、終了条件の判断を回避してループカウンタの利用のみが可能である。終了条件の判断を明示的に入れる場合は、until ループの利用 (i=1, until i=10, 処理, i++) が必要である。（注意： 配列の添字が「0」から始まる問題への対処も必要）

③学習の進捗に応じて、これらの記述の仕方を教える。既存のレッスンを利用し、これらの学習に必要な学習項目の学習が終わった段階で、活用できる既存のレッスンに戻ってこれらの学習項目を習得させる。

c．終了条件つきのループ（until/while ループ）

CoffeeScript では、while ループと、until ループの2種類が利用できる。しかし、「コードの冒険」で取り上げられるのは、until ループだけである。C言語、Java、Python、PHP で

はいずれも while ループしかない。そのため、until ループを学んだあと、not を学んでから while ループを導入し、活用できる既存のレッスンに戻って while を習得させる。また、return を学んでから、活用できる既存のレッスンに戻って while を復習して定着させる。

d．まとめ

以上のことをレッスンごとに整理した。

表５　他の言語との接続

チャレンジ	レッスン	名称	主な処理	他の言語との接続	アルゴリズム	C言語比較	Java比較	Python比較	PHP比較
0〜10	1	step，turn	step 10 turn 45 turn right						
11〜20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10						
21〜30	3	timesループ	3.times ->　turn left　step 5		ループ	for (初期化式; 条件式; 変化式)	for (変数：コレクション)	for 変数 in range(整数)	foreach (コレクション as $変数)
31〜50	4	変数	a = 10 step a b = distanceTo banana	★i=i+1は、i++と書けることを教えておく。		+=, ++	+=, ++	+=, ++	+=, ++
51〜60	5	配列	beavers[0].step 10 step distanceTo bananas[1]						
	P5			★55-57をつかって、定数部分を変数に置き換えて変数を配列を組み合わせる練習、さらにi=i+1などと組み合わせる練習					
61〜75	6	forループ	for b in bananas　turnTo b　step distanceTo b		ループ	for (変数：コレクション)	for (変数：コレクション)	for 変数 in コレクション	foreach (コレクション as $変数)
	P6			★（紙上で、）３章と６章の題材のいくつかを使って、for のカウンタ変数を使った処理を学ぶ。 ★for (i=1, i<=10, i++)に対応した処理の仕方を教える。（for i in [1..10]、for i in [1..bananas.length]） （または、i= 1 10.times 処理 i++、10.times文はループの中で何回目かわからない） ★配列の数を数える方法は、「配列.length」（bananas.length、バナナの長さ。日本語と語順が同じ）	ループと判断の関係	for (初期化式; 条件式; 変化式)	for (初期化式; 条件式; 変化式)	for 変数 in シーケンス	for (初期化式; 条件式; 変化式)
76〜90	7	関数	goto = (a) ->　turnTo a　step distanceTo a		サブルーチン				
91〜105	8	untilループ	until near match　turnTo match　step 1		判断	untilなし	untilなし	untilなし	untilなし
106〜116	9	if文	if banana.frozen()　goat.goto banana　goat.hit()		判断				
117〜124	10	if　else文	if banana.green()　goat.goto banana else　goto banana		判断				
125〜145	11	and，or	until bear.sleeping() and tiger.sleeping()　wait() until bear.sleeping() or bear.playing()　wait()		ベン図				
146〜151	12	not	if not banana.rotten()　goto banana		ベン図とnot, and, or				

表５　他の言語との接続（続き）

チャレンジ	レッスン	名称	主な処理	他の言語との接続	アルゴリズム	C言語比較	Java比較	Python比較	PHP比較
	P12			★このへんで、whileを入れたほうがいい。 ★３，６，７章を使うか、９から１２章を使うか。 ★until ugeuge と while not ugeuge が同じという理解をさせて、whileのループへ導く。（untilは○○になるまで、whileはＸＸの間） ★until bear.sleeping(), while not bear.sleeping(), while bear.起きている() のように誘導する。単にnotをつけるだけでなく関数の名前を変えたほうがよいことを理解させる。 ★while型がわかりやすい時とuntil型がわかりやすい時を理解させる。 ★「起きている関数(not sleeping())、生の関数(not rotten())、黄色関数(not green())」などを作つくって not 部分をラッピングすることが可能か？	判断	untilがないため	untilがないため	untilがないため	untilがないため
152～160	13	<，==	if health()<70 　goto healthZone 　until health()==100 　　wait()		判断				
161～175	14	戻り値	return no return yes return not crow.watching()		関数				
				★untilに代えてwhileとreturnの組み合わせをやらせる。 ★14章をつかって練習 ★単に条件や戻り値にnot をつけるだけでなく、関数の名前を適切なものに変えさせる。【別解】	判断	untilがないため	untilがないため	untilがないため	untilがないため
176～189	15	onKey	onKey = (key) -> 　if key == 'w' 　　step 1						
190～195	16	onMouseMove	onMouseMove = (pos) -> 　bat.setX pos.x 　bat.setY pos.y						
196～210	17	onClick	hippo.onClick = () -> 　monkey.turnTo hippo 　monkey.toss()						

４．３　CodeMonkey のつまずき対策

以上の分析から、対策を整理した。

（１）対策の整理の視点

対策の整理にあたっては、

・「コードの冒険」の習得の歩留まりを向上する（基礎対策）

・「コードの冒険」を出発として複雑な処理ができるように能力を上げる（応用対策）

の２つの視点で整理した。

基礎対策の対象者は、

・「コードの冒険」で、時々つまずく

・「コードの冒険」で、コーディングに時間がかかる

ような学生である。プログラミングに関する最終的な目標は、

・簡単なマクロが使える（Office や Adobe 製品）

・Web で使われる簡単な JavaScript が理解できる

・組み込みソフトにおける簡単なプログラムが理解できる

くらいである。

応用対策の対象者は、

・「コードの冒険」で、つまずきが少ない

・「コードの冒険」では力が余る

ような学生である。プログラミングに関する最終的な目標は、

・C 言語など他の言語も習得する

・アプリやゲームのプログラミングができるようになる

・プログラマとして就職できる

くらいである。

（2）具体的な対策

基礎対策については「・」で、応用対策については「★」で対策を区別して記述した。なお、付録２にこれまでのつまづきのポイントから対策までを一覧できる表を示した。ここでは、対策部分のみを示す。

表６　「コードの冒険」のつまずき対策

チャレンジ	レッスン	名称	主な処理	数学	分析（コンピュータ）	他の言語との接続	アルゴリズム
0〜10	1	step，turn	step 10 turn 45 turn right	・角度において「－」が右回りか左回りか。ノートに絵を書いて整理させる。 ・猿基準の長さ・角度しか出ない事への対処。定規・角度がピンと来ていない場合、「トードーの算数」をやらせる。	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。 ・語順が逆のものは、命令だから倒置法になっていると思えば、ある程度理解できる。「進め１０」「回れ４５度」「回れ右！」		
	P2				・レッスン１で省略された部分を加えて「オブジェクト.メソッド」の練習 ・（紙上で、）レッスン１の題材のいくつかについて「monkey.」をつけて書かせる		
11〜20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10		・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。駅から学校、玄関から教室までを例に説明する。単純化した地図をつくっておく。 ・レッスン１では「monkey.」が省略されている。この章の直前にブレステージP2を入れる。		
21〜30	3	timesループ	3.times -> 　turn left 　step 5	（手順を書き出すという習慣の有無）	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。３階まで階段を登るなどを例に（繰り返しで表現できる）。		ループ
31〜50	4	変数	a = 10 step a b = distanceTo banana	（数学の文字式とコンピュータの変数の概念の違い） （数学の等式とコンピュータの代入の概念の違い）	・変数の機能をイメージできていない。コンピュータでの文字・変数は「箱」なので、「箱」をつかって説明する。自己代入のi=i+1を「箱」で実感させる。	★i=i+1は、i++と書けることを教えておく。	
51〜60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	（手順を書き出すという習慣の有無）	・手順をノートに書かせる。 ・配列と変数の違いを説明する。どちらも「箱」なのは同じ。グループとメンバーの関係。		
	P4				・i=i+○の練習量が不足 ・（紙上で、）５章の題材のいくつかについて変数と自己代入のi=i+○を使って書かせる。 ★55-57をつかって、定数部分を変数に置き換えて変数を配列を組み合わせる練習、さらにi=i+1などと組み合わせる練習		
61〜75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b		・「for」を使うメリットを説明する。Forを使わずに５回・１０回と回数が増えるとミスが多くなる。		ループ
	P6				★forループの入れ子の例題をいくつかやる必要あり	★（紙上で、）３章と６章の題材のいくつかを使って、for のカウンタ変数を使った処理を学ぶ。 ★for (i=1, i<=10, i++)に対応した処理の仕方を教える。（for i in [1..10]、for i in [1..bananas.length]）（または、i= 1 10.times 処理 i++、10.times文はループの中で何回目かわからない） ★配列の数を数える方法は、「配列.length」（bananas.length、バナナの長さ。日本語と語順が同じ）	ループと判断の関係
76〜90	7	関数	goto = (a) -> 　turnTo a 　step distanceTo a	（数学の関数の概念とコンピュータの関数の概念の違い）	★goto 配列（goto bananas）に対応するには？ ★チャレンジ88から90を利用して様々な組み合わせに挑戦させる ★チャレンジ88から90を利用してどういう目的でどういう分割がわかりやすいのか考えさせる（89の模範解答はモジュール分割がおかしい。【別解】） ・goto以外の関数のイメージをふくらませるためにレッスン追加	0	サブルーチン

表６　「コードの冒険」のつまずき対策（続き）

チャレンジ	レッスン	名称	主な処理	数学	分析（コンピュータ）	他の言語との接続	アルゴリズム
	P7				・関数のイメージの膨らませ。 ・紙上で、関数づくりの練習をさせる。いろんな関数、いろんな引数 ・コンビニへ行く ・駅で乗り換えの例（乗り換えTo 阪和線） ★引数が２つ以上のイメージの膨らませ。 ★コピー機の例（コピー（100%,5枚）） ★さらに引数として変数を渡した時の挙動（変数のスコープ） ★引数も日本語の語順が逆になる。「copy (100%, 5枚)」なら「100%で、５枚、コピーする」。		
91～105	8	untilループ	until near match 　turnTo match 　step 1		★変数と関数の引数のスコープについて書き出して納得する（特にチャンレンジ９９以降）		判断
	P8				・untilの直後に数式が来るパターンを練習する ・関数の戻り値を変数に入れて、その変数をuntilで評価する練習。わかりにくい場合はn == trueまで書かせる。 ・until i == 10, until i >=10 などの練習 ・until n, until n == trueなどの練習		
106～116	9	if文	if banana.frozen() 　goat.goto banana 　goat.hit()				判断
117～124	10	if　else文	if banana.green() 　goat.goto banana else 　goto banana				判断
	P10				★forと配列とgoto以外の関数（引数は変数）の組み合わせの練習 ★チャレンジ124の解き方をいろいろやらせる【別解】 　関数使わずに書いてみる、 　先に判定してからどっちが行くか決める、 　先にバナナに行ってから判定する、とか ★returnを先にここで入れたほうがわかりやすいかも 　凍ってるかの判定関数 　青いバナナかの判定関数		
125～145	11	and，or	until bear.sleeping() and tiger.sleeping() 　wait() until bear.sleeping() or bear.playing() 　wait()	（and, or の関係がわからない可能性）	・「A and B」は「AもBも」、「A or B」は「AかBか」。日本語の「AかBか」は「男か女か」のようにどちらか一方の意味で、コンピュータの意味と違うのに注意する。		ベン図
	P11				★この辺でも、forと配列とgoto以外の関数（引数は配列）の組み合わせ ★チャレンジ124の解き方をいろいろやらせる 　stepperでループする考えでやるとbananasを引数にすることになる ★チャレンジ140のやり方をいろいろやらせる【別解】 　配列のtigersを引数にした関数 　配列のbananasを引数にした関数		
146～151	12	not	if not banana.rotten() 　goto banana				ベン図とnot, and, or
	P12					★このへんで、whileを入れたほうがいい。 ★３，６，７章を使うか、９から１２章を使うか。 ★until ugeuge と while not ugeuge が同じという理解をさせて、whileのループへ導く。（untilは○○になるまで、whileはＸＸの間） ★until bear.sleeping(), while not bear.sleeping(), while bear.起きている() のように誘導する。単にnotをつけるだけでなく関数の名前を変えたほうがよいことを理解させる。 ★while型がわかりやすい時とuntil型がわかりやすい時を理解させる。 ★「起きている関数(not sleeping())、生の関数(not rotten())、黄色関数(not green())」などを作つくって not 部分をラッピングすることが可能か？	判断
152～160	13	<，==	if health()<70 　goto healthZone 　until health()==100 　　wait()	（大小関係の不等号がわからない可能性）			判断
	P14				・紙上で、関数づくりの練習で、戻り値ありの場合をさせる。いろんな関数、いろんな引数で。 ・関数のイメージの膨らませを応用して説明。成功したか失敗したかを返す。 ・駅で乗り換えの例（乗り換えTo 阪和線　成功・失敗） ・コピー機の例（コピー（100%,5枚）成功・失敗） ・代金の例（代金（100円、５個）） ・消費税の例（消費税（8%､100円））		

表６　「コードの冒険」のつまずき対策（続き）

チャレンジ	レッスン	名称	主な処理	数学	分析（コンピュータ）	他の言語との接続	アルゴリズム
161～175	14	戻り値	return no return yes return not crow.watching()		（値を返すという例を別途用意して実感させることが必要）		関数
	P14				・紙上で、関数づくりの練習で、戻り値ありの場合をさせる。いろんな関数、いろんな引数で。 ・関数のイメージの膨らませを応用して説明。成功したか失敗したかを返す。 ・駅で乗り換えの例（乗り換えTo 阪和線　成功・失敗） ・コピー機の例（コピー（100%,5枚）成功・失敗） ・代金の例（代金（100円、５個）） ・消費税の例（消費税（8%,100円）））		判断
176～189	15	onKey	onKey = (key) -> 　if key == 'w' 　　step 1		（イベントドリブンな記述の仕方）		
190～195	16	onMouseMove	onMouseMove = (pos) -> 　bat.setX pos.x 　bat.setY pos.y	（座標がわからない可能性）	（イベントドリブンな記述の仕方）		
196～210	17	onClick	hippo.onClick = () -> 　monkey.turnTo hippo 　monkey.toss()		（onClickはオブジェクトに設定する。それがオブジェクトを対象とした文で宣言するということと、その実行はonClickイベントで発動されるということの発想の切り分けが難しい。他の部分は順次処理であるが、イベントドリブンな部分は割り込みになってその中では順次処理ということ。）		

（３）実施方法

ａ．基本的な考え方

基礎対策についてクラス全体にしたほうがいいものと、特定のつまずいている学生のみが対象のものがある。その区別はクラスによって異なるので、ここでは基準は示さない。クラス全体にしたほうがいいものは、授業の冒頭に全体に向けて実施する。その他のものは、個別対応である。

応用対策については、各授業の最後の１５から２０分くらいの時間に説明する。課題についての簡単なプリントを作っておき、それを配布して説明する。その際、「コードの冒険」でせいいっぱいの学生については自分の作業をつづけるように指示する。また、これらは応用的なもので、これができなくとも CodeMonkey には関係がないと安心させて話をする。応用対策の作業は授業内に完結できない場合宿題とすればよい。せいいっぱいの学生でも意欲があれば自宅で取り組んでよい。次年度この応用対策用のプリントを制作する。

ｂ．参考情報

本プロジェクトのアドバイザーである飯箸泰宏先生より、助言をいただいたので、その本文を上げるので、参考にしていただきたい。

コードモンキーをはじめとする課題学習（～実技学習）は進度も理解度も異なりますので、次のような工夫がほしいです。

① グループ学習を大幅に取り入れて、評価は個人評価半分+グループごとの進度半分とすると、進んだ子は遅れた子を必死に教えます。グループごとの進度点はクラス内に公開しておくとグループ同士の競争になります。私は個人の課題評価点（時系列がわかる状態で）もクラス内公開にしていました。

② 進度制限を一律にしない。個別に当面の課題のクリアが確認されたら次のステージを開

放するなどで、進んでいる子が飽きてしまわないようにすることが大事と思います。全てのステージを終えてしまった生徒・学生には、自由課題に取組むことを許可するとあらかじめ宣言しておくと、意欲の向上を図れます。自由課題の案のいくつかは教師側で準備しておいて、本人の希望案も適切であれば許可すればよいと思います。

③ つまづきは、全員の生徒に生ずるのではありませんから、つまづきに対して一斉講話するなどのことをすると他の生徒が飽きてしまうことにも注意が必要です。課題学習はテンポよく進みたいというのが生徒・学生の心理です。以下の補足を参考にしていただければ幸いです。

補足:

進度の違う生徒が同席しますから、一斉にアナウンスできるのは、授業の最初と最後、およびよほどの危険が生じたときくらいしかできません。私の場合は、「ヒントブック（オリジナル自家製)」を作成して、電子ファイル（多くはワード）で事前配布していました。

「教師がしゃべらなくてよい（一斉講話しない)」がモットーです。教師は一斉講話しない代わりに学生の傍に行って質問に答えるほうが効果的です。

内容はほぼマニュアルに近いです。マニュアルとの違いは、「なぜこうなったか自分で考えよう」などという記述が所々にあるくらいです。

グループ内で、「何で?」と聞き合って、ほとんどが解決します。互いの質問のきっかけになっています。それでもわからない場合は手を挙げてもらって、教師が駆け付けて説明します。

ヒントブックは、ほとんどが図解で、図に対する解説を箇条書きで書きます。長い文章はほとんど読んでくれないからです。

さらに、例えば偏差値 43 の某 KKSK 大学理工学部では、極力漢字を避け、漢字でないとむしろ分かりにくい術語については括弧書きで読みと意味を書きました。

授業の最後には「振り返りブログ」を書いて、当日の成果物は途中でもよいので必ずメール添付で提出することになっていましたので、どちらかに学生のつまづきが発覚します。つまづいた学生に「ヒントブックのここを見て」というだけで済むかというとそうはいきません。ヒントブックを読み間違いしたり理解していない場合や予期しなかったつまづきに対しては、2 つの対処がありました。

・追加ヒント・ノート（まとまると追加ヒント集）を作って配布する。

メールで届いたりブログに書かれたつまづき箇所については本人宛の解説を返信します。

二人以上がつまづいたもの（または二人以上がつまづきそうなもの）は追加ヒント・ノートにして全員に配布する原則にしました。

目安として、3 割以上の学生が必要とする追加ヒントは次の学期にはヒントブックに取り入れます。2 割未満の学生しか必要がない追加ヒントは、追加ヒントだけにし

ます。

ヒントブックの本筋から外れた説明が多いと煩わしくなって、普通に進捗する学生はヒントブックを読まなくなりますから、別ヒントにしたのです。

・お助けチームを作って活動させる。

私の授業は半期で平均 45 の課題がありましたので、グループを超えて助けた学生には助けた回数（つまづいていた学生が「分かった」と宣言した回数）に応じて他の課題を完了したのと同じだけの点数をあげることにしていました。

「困った学生は手を挙げろ、先生かお助け隊を派遣する」というルールにしていました。1 回教えると 2 点加算するくらいの感覚になります。

つまづいている学生が多いと教師だけでは手が回りませんので、その場で進捗の早い学生から順にお助け隊を命じて救援に向かう先を指定します。

救援が完了したら教師に駆け寄って「誰が、こんなところでつまづいていたので、こう教えたらうまく行きました」という口頭報告をします。

教師は素早くメモを作りつつ、救援先の学生の名前を呼んで、「××君、この問題は解決したんだね」と聞いて、良い返事だったら加点するという方式でした。

お助け隊になって救援しに行ったのに、逆に教えてもらって帰ってくる学生もいて、てんやわんやになりますが、にぎやかで楽しい授業になります。

５．CodeMonkey の代替システム

CodeMonkey に依存せず同等の結果を出せるようにするため、CodeMonkey のクローンを開発しておくことその対策のひとつである。クローンが満たすべき要件については次年度以降にまとめるが、本年度は委員会より指示のあった３つの方法と、CodeMonkey の学習コンテンツで Python を学ぶ簡易コースである「トリビアチャットボット」について簡単に情報を収集した。

３つの方法とは、

・Pyxel　・・・　Python 上のゲームエンジン。レトロゲーム用のエンジンである。作者が日本人であり、日本語で情報が取りやすい。Python 上に構築されているため、クローンを作る場合、システム内の言語を CoffeeScript ではなく Python に変更できる利点がある。

・日本語プリプロセッサ　・・・　CodeMonkey のつまづき原因の一つは、CoffeeScript の文法が英語の影響を受けていることや変数名や関数名、予約語が英語がベースになっていることである。クローンにおいて、変数名や関数名、予約語などを日本語化できれば、学習効率を向上できると予想できる。それにはクローン内で書かれたコードを日本語プリプロセッサを介してネイティブ言語に変換する必要がある。

・Moocs　・・・　クローンにおいて、学生の管理を行う場合、その機能を外部化することで開発工数を削減できる可能性がある。また、学習システムとして教師側からも学生側からも他の科目と同一のプラットフォームで管理できるメリットがある。

また、「トリビアチャットボット」とは、

・トリビアチャットボット　・・・　CodeMonkey の学習コンテンツのひとつ。「コードの冒険」と同じくブラウザ上で動作するプログラミング教材で、「Python」を使って、メッセージのやり取りを行うチャットボットの作り方を学ぶコース。サルのモンタ・チャットボットが出すトリビア（豆知識）を問う質問に、ユーザーが回答、チャットボットが正否を判定して、チャットの最後に正答数に応じたメッセージを自動的に返す、という簡単なチャットボットの作り方を 16 のエクササイズを通して学ぶ。

以下のこの４つについて、情報収集したことを整理した。

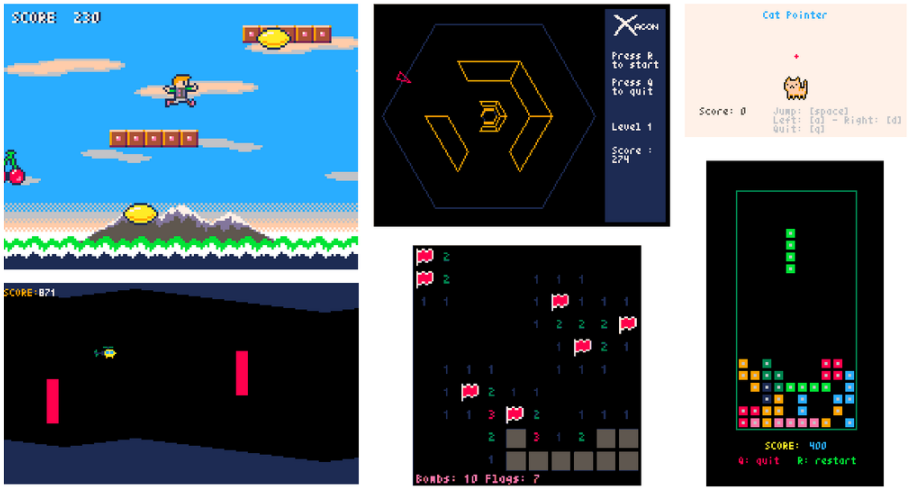
５．１　Pyxel

Python 上のゲームエンジン。レトロゲーム用のエンジンである。作者が日本人であり、日本語で情報が取りやすい。Python 上に構築されているため、クローンを作る場合、シス

テム内の言語を CoffeeScript ではなく Python に変更できる利点がある。

（1）概要

Pyxel（ピクセル）は、ドット絵タイプのゲームを簡単に作れる「レトロゲームエンジン」である。



a. 作者　　日本人の tkitao さん
http://tkitao.hatenablog.com/entry/2018/11/24/185346

b. 歴史　　2018 年 7 月 30 日にリリースされた、非常に新しいゲームエンジン。
海外を中心に急速にユーザーが増えている。

c. ソース　　GitHub でオープンソースとして公開されている。
https://github.com/kitao/pyxel

d. 特徴　　使える色は 16 色のみ、同時に再生できる音は 4 音までなど、レトロゲーム機を意識したシンプルな仕様で、Python でドット絵スタイルのゲームづくりが気軽に楽しめる。

e. 仕様　　Windows、Mac、Linux 対応
Python3 によるコード記述
16 色固定パレット
256x256 サイズ、3 画像バンク
256x256 サイズ、8 タイルマップ
4 音同時再生、定義可能な 64 サウンド
任意のサウンドを組み合わせ可能な 8 ミュージック
キーボード、マウス、ゲームパッド
画像・サウンド編集ツール

（2）主な情報ソース

a. 作者による紹介記事

レトロゲームエンジン Pyxel でプログラミングを始めよう！

http://tkitao.hatenablog.com/entry/2018/11/24/185346

特長① Python でプログラムできる

特長② マルチプラットフォーム対応

特長③ 覚えることが少ない（11 種類の命令を覚えるだけで、すべての描画機能。5 種類の命令を覚えるだけで、効果音や音楽の作成と再生ができる）

特長④ 専用ツールが同梱されている（絵や音楽の作成ツール同梱）

特長⑤ 無料で使える(MIT ライセンスで公開)

特長⑥ 作者が日本人(作者が書いた日本語マニュアルもある)

b. はやぶささんによる「Python でレトロゲームを作ろう！」

https://cpp-learning.com/pyxel_tutorial/

Day1 -画像の扱い方-

Day2 -マウス操作-

Day3 -クラス化-

Day4 -ベクトル-

Day5 -ショット攻撃-

Day6 -当たり判定-

Day7 -人工知能-

※関連記事　ゲームソフト開発を題材にしたオブジェクト指向入門

https://cpp-learning.com/object-oriented/

c. Pacman 風ゲームをつくる

Pyxel でパックマンぽいゲームを作る

https://qiita.com/stmn/items/4048d4af2a9613594b60

d. パズドラ風ゲームをつくる

話題の pyxel でパズドラ風パズルゲームを作ってみた。

https://note.mu/pluswing/n/nd2a3a0499e05

e. 開発者コミュニティ「冬休みに Pyxel でレトロゲームをつくってみる会」

https://connpass.com/event/113463/

（3）簡単なまとめ

CodeMonkey クローンのうち、主人公が動くゲーム画面部分については、Pyxel 程度の能力があれば十分である。一方で、画面の解像度が 256 x 256 に限定されている。これでは、おそらく、コードの画面をゲーム画面の中に設けるのは無理があるだろう。

クローンシステムは、大きく分けて、チャレンジ内のコードの入力・評価・デバッグを行うコースの管理のエンジンと、コードに基づいてゲームを再現するゲームのエンジンの 2

つからなるシステムで、ゲームエンジン側に Pyxel を使う形になるのではないか（図１）。

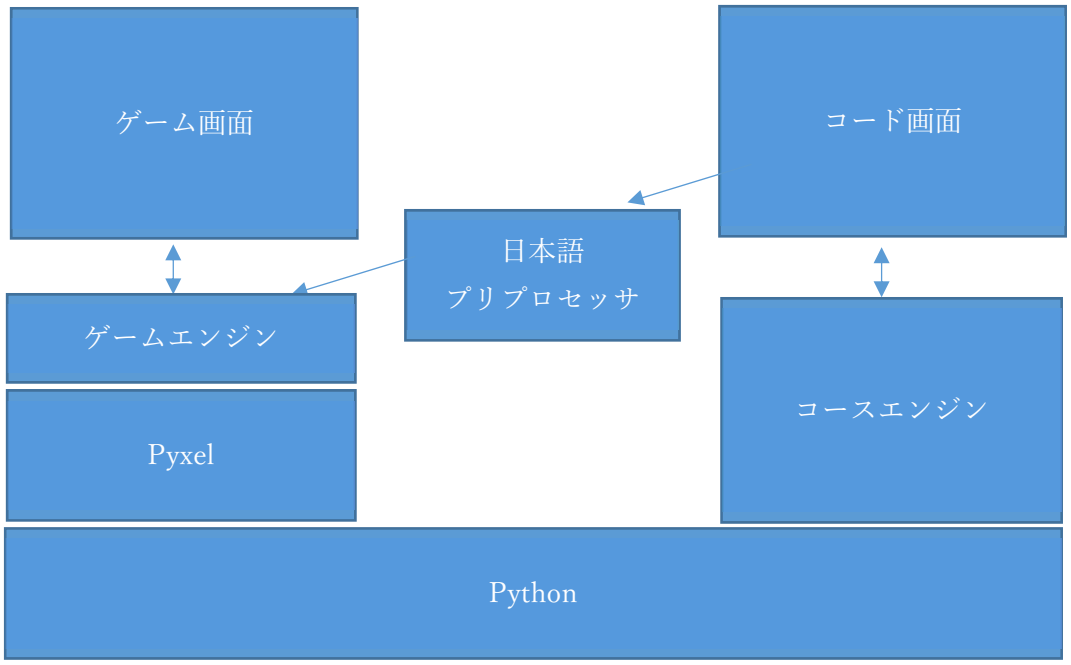


図１　クローンの基本的なシステム構成

５．２　日本語プリプロセッサ

CodeMonkey のつまづき原因の一つは、CoffeeScript の文法が英語の影響を受けていることや変数名や関数名、予約語が英語がベースになっていることである。クローンにおいて、変数名や関数名、予約語などを日本語化できれば、学習効率を向上できると予想できる。それにはクローン内で書かれたコードを日本語プリプロセッサを介してネイティブ言語に変換する必要がある。

現在の CodeMonkey では、命令や変数名などが英語となっている。またプログラミング言語の語順も英語の語順が基本となっている場合が多い。プログラミング言語の開発がアメリカ発だということもあり、英語を基本としてプログラミング言語が開発されることは圧倒的なデファクトスタンダードとなっている。

しかし、頭の中の考えとコードの言語表現が近いと、プログラムの意図が理解しやすくなる。日本人の、とりわけプログラミング初心者にとっては、この英語の壁は大きな壁である。初心者向けの教材において、何らかの日本語表現でプログラミングできるようにする方法を考えておく必要がある。

委員会では、Python に日本語プリプロセッサを経由させることで、Python 上でより自然

に日本語が使える方法を検討する方針が打ち出されている。本節では、その方法について情報を収集した。具体的には、

・日本語プリプロセッサ
・日本語プログラミング言語について
・Python での日本語の取扱について

についてである。

（1）日本語プリプロセッサ

a．日本語プリプロセッサとは

プリプロセッサとは、一般にある処理を行うソフトウェアに対して、データ入力やデータ整形などの準備的な処理を行うソフトウェアのことである。プログラミング環境に関して言えば、コンパイラまたはインタープリタがソースコードをコンパイル／中間言語翻訳する前に、一旦ソースコードに前処理を施すためのプログラムである。

日本語プリプロセッサとは、プリプロセッサの中で、ソースコード内の命令などを日本語表記からコンピュータ言語の予約語などに変換するプログラムである。

b. 参考情報　「ことだま on Squeak」

Squeak の上に「ことだま」という日本語プログラミング言語を実装したもの。

http://crew-lab.sfc.keio.ac.jp/squeak/

初心者用プログラミング環境「ことだま on Squeak」の試み

http://gakkai.univcoop.or.jp/pcc/paper/2007/pdf/177.pdf

教育用プログラミング言語としての「言霊」と「ことだま on Squeak」の試み

https://www.cmc.osaka-u.ac.jp/publication/for-2006/17-22.pdf

（2）日本語プログラミング言語

a．日本語プログラミング言語

現在、代表的な日本語プログラミング言語には３つある。

・プロデル
・なでしこ
・ドリトル

である。その特徴について簡単に整理する。

日本語の自然さは、なでしこが一番すぐれている。

b．プロデル

ゆうとが開発したインタプリタ型の日本語プログラミング言語である。

日本語プログラミング言語の「TTSneo」を前身とし、仕様を一新。アップグレードする

形で公開された言語。TTS が公開されたのは、2000 年。TTSneo の公開は 2002 年。TTS の公開から数えると、プロデルは 18 年の歴史。プロデルは「教育向け言語」として開発されたものではなく、汎用的なソフトウェア開発言語を志向している。プロデルには様々な実用的なプラグインが用意されている他、API 連携も可能。

プロデルの公式ページ
https://rdr.utopiat.net/
プロデルの公式解説ページ
https://rdr.utopiat.net/docs/onepage/first.html
Wikipedia のページ（簡単な言語仕様あり）
https://ja.wikipedia.org/wiki/%E3%83%97%E3%83%AD%E3%83%87%E3%83%AB

ｃ．なでしこ

クジラ飛行机が開発したインタプリタ方式のスクリプト型プログラミング言語である。オープンソースプロジェクトである。

クジラ飛行机氏は、なでしこの前身である日本語プログラミング言語「ひまわり」の開発者でもある。なでしこはひまわりを改良し、より日本語に近い語順でのコーディングを可能とすることを目標の 1 つとしている。

「なでしこ 3」は、HTML/JavaScript によるブラウザでの実行が可能。ローカルにも開発環境の構築が可能。情報処理機構の 2004 年未踏ソフトウェア創造事業「未踏ユース」に採択された。コミュニティの活動が活発な言語である。

なでしこの公式ページ
https://nadesi.com/top/
なでしこリファレンス
https://nadesi.com/man/

ｄ．ドリトル

大阪電気通信大学教授の兼宗進により開発されたプロトタイプベースのオブジェクト指向言語。

ドリトルは中学校や高校の教科書や副教材に採用されている、教育向けの日本語プログラミング言語。プロデルやなでしこと比べ、より教育用途に特化し、教育現場を中心に普及が進んでいるのが特色。フリーウェア。

公式ページ
https://dolittle.eplang.jp/

公式マニュアル

https://dolittle.eplang.jp/manualv3

（３）Python での日本語の取扱

日本語変数、日本語メソッド名対応の主要なプログラミング言語としては、次のものが挙げられる。

C#

PHP

Python

Ruby

Java

Perl

node.js

Objective-C(Xcode5)

Python については、3.x 系から日本語変数名、関数名に対応している

ａ．Python 3 では識別子に ASCII 文字以外が使える

Python3 では識別子（変数/関数/クラス名）に Unicode 文字が使える。大文字アルファベット、小文字アルファベット、アンダースコア、数字、以外に日本語の全角文字が使える。数字は識別子の先頭には使えない。予約語は使えない。アンダースコアで始まる識別子は、特別な意味がある。

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise		

ｂ．日本語でプログラムを書くためのモジュール

単に変数名やメソッド名を日本語にするだけでなく、組み込みのメソッドの表記も日本語化する方法もある。

日本語でプログラム書くためのおまじない#

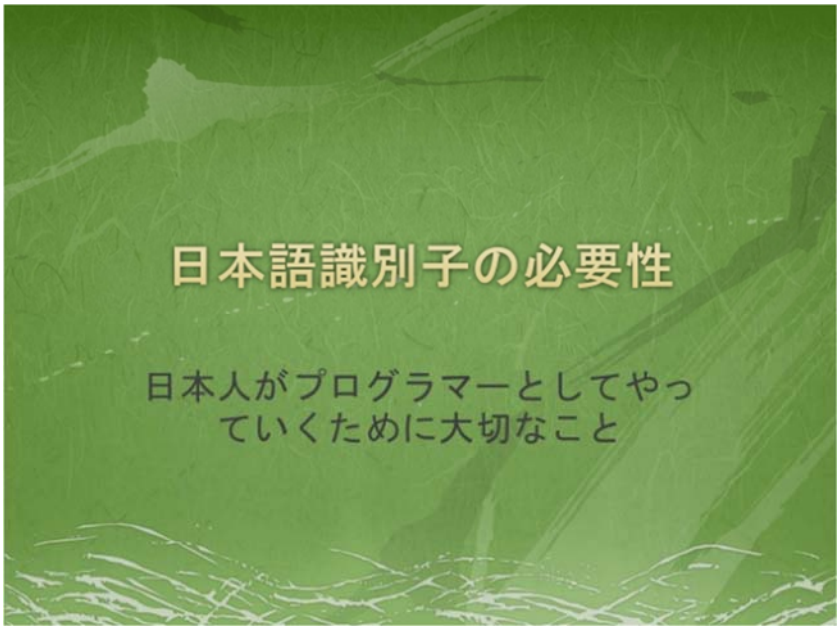
おまじない|ｪ

https://qiita.com/mutsu/items/175a2cf770c18732043b

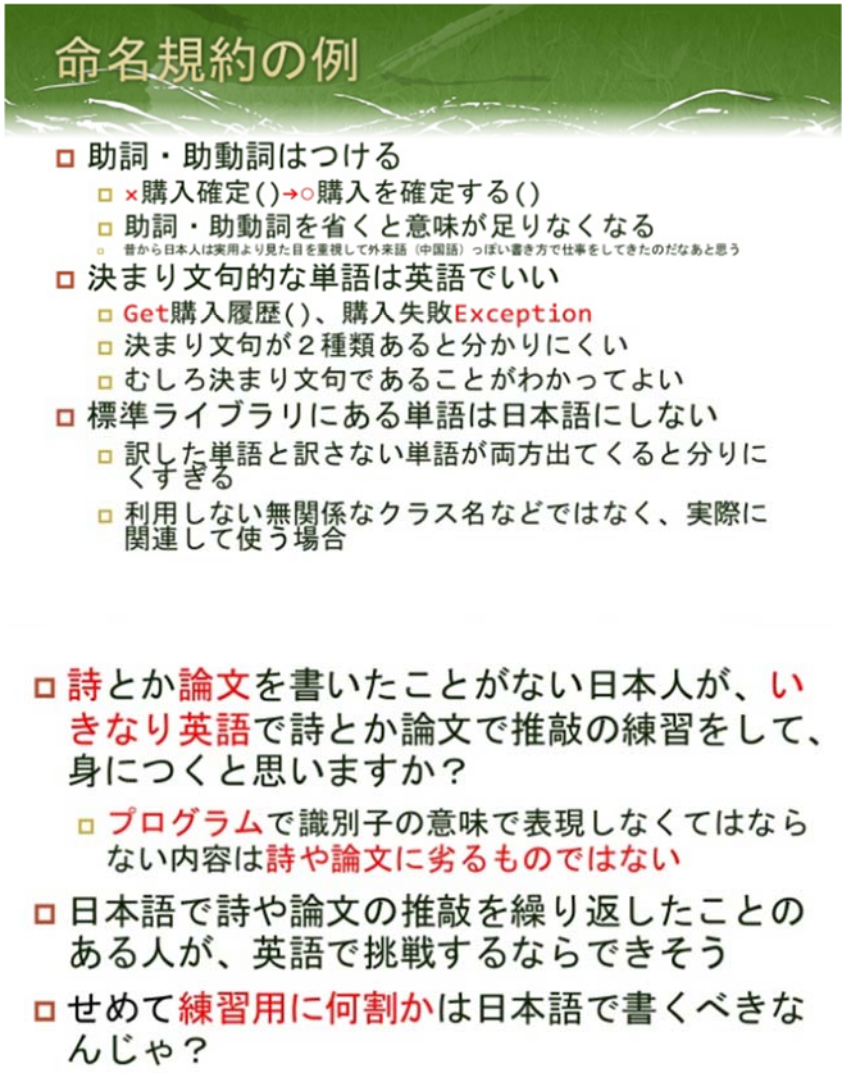
```
# -*- coding: utf-8 -*-
from おまじない import *


# 販売商品を作る
価格表 = 辞書()
価格表["りんご"] = 100
価格表["みかん"] = 50
価格表["メロン"] = 200

購入履歴 = リスト()

# 購入履歴を適当に作る
for 回数 in 連番リスト(10):
    商品名リスト = 価格表.全部の名前をゲット()
    商品名番号 = ランダムの数をゲット(0, 長さ(商品名リスト) - 1)
    購入履歴.くっつける(リスト(商品名リスト)[商品名番号])

表示("=====購入履歴=====")
for 回数, 購入品 in 数えあげる(購入履歴):
    表示("%d回目:%s" % (回数 + 1, 購入品))

# 支払合計を計算
支払合計 = 0
for 購入品 in 購入履歴:
    支払合計 += 価格表[購入品]
表示("=====支払合計=====")
表示(支払合計)
```

c．日本語の変数名、メソッド名のメリット・デメリット

①日本語識別子の必要性 potimarimo
https://www.slideshare.net/potimarimo/ss-12314191



- よいプログラムとは、人間が読みやすいプログラムです
- 人間が読みやすくするためには、識別子（単語）の選択は重要
- よいプログラマーとして成長には、識別子をじっくりと推敲することが必要です
- 日本人にはそれを理解するのが難しい環境になっています



②IT/Web エンジニア向け総合求人・学習サービス「paiza」のエンジニアによる評価

――新人プログラマが絶対に知っておきたい、日本語変数利用時の作法

実感したメリット

ちゃんと書けば、自分の書いたテストは、読んだときに圧倒的にわかりやすい

テストメソッド名から曖昧さがなくなる（テスト設計になる）。

適切な用語を用いると、コメント不要になり、ソースの可読性も高まり良いことづくめでした。

実感したデメリット

メソッド名に本人の癖が出やすい＋自由度が高いので、他人のテストが読めない

冗長になりがち。

プロジェクトのソースを外国人が保守する場合、日本語が読めなくてトラブルになる。

出典：

https://paiza.hatenablog.com/entry/2014/04/21/%E6%96%B0%E4%BA%BA%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%9E%E3%81%8C%E7%B5%B6%E5%AF%BE%E3%81%AB%E7%9F%A5%E3%81%A3%E3%81%A6%E3%81%8A%E3%81%8D%E3%81%9F%E3%81%84%E3%80%81%E6%97%A5%E6%9C%AC%E8%AA%9E

③日本語識別子が流行らないのは

◆ IME はそれほど負担はない

◆ 逆に使いこなせばメリットも多い

◆ どこでどう日本語を使うかのほうが使いづらい理由かも

http://var.blog.jp/archives/75345476.html

昔なら言語でアルファベット以外を使えてもその他ツールが対応してない状況だった。しかし、Unicode 対応していて当たり前の時代になり、ツールの対応面ではあまり困ることはなくなっている。

ただし、英語＝プログラム、日本語＝コメント、となっているとコメントの読みやすさが断然違う。

大きな問題は、複数人で書いている場合に、どこをどういう風に日本語にするのか、という命名規則の問題である。

例えば、「ページ」を例にすると、タイプする文字は 「pe-ji」 なので 「page」 にしてしまえばタイプ数が減る。逆に 「説明」 は 「setumei」 なので 「description」 と書くより短い。

どういうときに日本語を使うかを明確にしないと キャメルケースとスネークケースみたいな問題になる。(英語の場合の代表的な命名規則：コンスタントケース、パスカルケース、キャメルケース、スネークケース、ケバブケースなど)。変数か関数かみたいな使う場所ではなく、言葉に応じて変えたくなるので難しい。

https://qiita.com/am_nimitz3/items/7b01af53751dba5d8fb1

日本語使うなら統一して全部日本語なら単純ですが、attr とか value とか num とかアルファベットのほうが短く打ちやすいし意味もわかりやすい場合もあるのに日本語しか使えないのは不便。is***, add***, remove***, can*** みたいなプレフィックスはわかりやすいので使いたい。

④識別子に日本語を使うことについての考察 @arai-ta

https://qiita.com/arai-ta/items/08d0fdb275ff0fcf68a6

言語構造そのものを日本語化するのではなく 識別子を日本語表現にする。

```
# こういうこと
if ($経路->特定料金計算対象か()) {
    $運賃 = 運賃計算::特例42条計算($経路->出発地, $経路->到着地);
}

function 特例42条計算($出発地, $到着地, $条件 = null) {
    :
}

# こうではない
$経路 が 特定料金計算対象 ならば $運賃 は ...
```

言語構造に日本語を導入するということは、それ用の言語を採用することであり、以下の面でリスクが大きい。

- 言語の習得コスト
- 開発スキルのある人員の採用
- ライブラリ、エコシステムの利用可能性

日本語を母国語とする開発メンバーが多数を占めると確信できる状況でないとメリットを享受するのは難しい。

向いている用途

テストケース名

ドメイン知識が複雑なとき

⑤プログラム日本語で書けばいい気がするけど

https://anond.hatelabo.jp/20170421230333

日本語で書ける言語使うんじゃなくて、変数名や関数名が Unicode 対応で日本語書けるものを使えばよい。

業務システムの固有名詞とか日本語特有なものとか無理に英語化してよくわからないことになっていたり、見づらくなるくらいなら日本語使えばいいんじゃないか。

年金額を取得する関数で「年金額を取得する」「年金額を取得」「年金額を取り出す」とかの表記を迷うんじゃなくて「get 年金額」でいいと思う。

（4）まとめ

・業界の慣習として、プロは英語の識別子で書くことが多い

・英語のプログラミング言語と英語の識別子に慣れる必要がある

・一方で、プログラムを推敲する段階では断然日本語のほうがわかりやすい

・プログラムの全体を日本語化しなくとも、識別子または識別子の一部を日本語化することは、初学者にはメリットがある

・Python であれば、識別子の日本語化には障害はない
・日本語プリプロセッサをつくらなくとも目的は達せそう
・日本語プログラミング言語を採用しなくとも目的は達せそう
・もし必要な場合でも、「おまじない|」などの工夫でかなりの目的は達せられる
・日本語識別子を使う場合は、命名規則を制定しておく必要がある

補足：
本プロジェクトのアドバイザーである飯箸泰弘先生より助言があったので、掲げておく。

日本語プログラミングは教育用と未熟練工用に限定してよいように思います（なでしこ＝酒徳君作が成功している理由）。言語は、どんなにいいモノでも、結局は普及しているものに負けてしまいます。

馬場君も酒徳君も語順にこだわるようになったのは、私が日本語プログラミングサミットで、「日本語の語順」にこだわるように主張してやまなかったためです。そのため、彼らは、それまでの言語を捨てて、二人とも根本からのバージョンアップ、実は新言語に移行してしまいました。

これはこれで悪くないですが、これまでの言語にない人の心をわしづかみにする何かが不足しています。

日本語プログラミング言語にとって「日本語の語順」はこだわるべき課題の一つではありますが、唯一無二ではなく、他にもこだわるべきものがあり、優先順位が一番というわけではなかったような気がします。悪いことを言ってしまったかもしれません。

まずは、学習用・未熟練工向けにそこそこ普及してから、これになれた人が実務でも使用するようになるという自然の流れを作らなければいけないですね。優先すべきは熟練工向き仕様ではなく、学習用・未熟練工向け仕様だと思います。

５．３　Moocs

クローンにおいて、学生の管理を行う場合、その機能を外部化することで開発工数を削減できる可能性がある。また、学習システムとして教師側からも学生側からも他の科目と同一のプラットフォームで管理できるメリットがある。

クローンにおいて、学生の管理を行うプラットフォームとして、Moocs が使いうるかどうか判断するため、Moocs の基本的な情報を整理した。

（１）Moocs のトレンド

Massive Open Online Course（MOOC、ムーク）または Massive Open Online Courses（MOOCs、ムークス）は、インターネット上で誰もが無料で受講できる大規模な開かれた講義のことである。代表的なプラットフォームとしては「Coursera」「edX」や、日本版とし

ては JMOOC が提供する「gacco」「OUJ MOOC」があり、条件を満たせば修了証が交付される。（Wikipedia）

a. MOOC とは何か ポスト MOOC を見据えた次世代プラットフォームの課題
山田 恒夫　情報管理 / 57 巻（2014）6 号
https://www.jstage.jst.go.jp/article/johokanri/57/6/57_367/_html/-char/ja

b. MOOC と学習解析 - 情報処理学会電子図書館 - 国立情報学研究所
山田恒夫　情報処理学会論文誌　教育とコンピュータ　Vol.1 No.4 1-11 (Dec. 2015)

https://ipsj.ixsq.nii.ac.jp/ej/index.php?action=pages_view_main&active_action=repository_action_common_download&item_id=146537&item_no=1&attribute_id=1&file_no=1&page_id=13&block_id=8

c. MOOC、有償化の動き
2012 年に世界のエリート大学を席巻した MOOC だが、有効なビジネスモデルが見つからないまま 6 年がたち、有償化の色彩を更に強める流れが出てきている。
https://rcos.nii.ac.jp/miho/2018/07/20180702/

d. OpenMOOClms
オープンムーク lms は大学、教育機関で主に導入されているオープンソースの LMS（授業、学習支援システム：Moodle）をベースに、e ポートフォリオ・達成度評価管理、ストリーミング動画配信機能を連携させたシステム（MOOC 構築システム）。
https://lms.resonantstyle.com/

（２）Moocs の広がり
JMOOC で「プログラミング」で検索すると、ヒットするのは 8 件である。そして、2019 年 3 月で開講中なのは次の 1 件のみであった。
a. 誰でも教えられる！ プログラミング授業　応用編
一般社団法人　ファーストスタープロジェクツ
講師：斎藤 正武（中央大学）
https://www.fisdom.org/F00000066/

（３）Udemy での広がり
有償の教育プラットフォームであり、MOOC のひとつでもある Udemy で同様に「プログラミング」で検索すると、ヒット数は 1157 件。様々なプログラミング言語、様々な切り

口、様々な価格帯で提供されている。中でも「プログラミング 入門」で調べると次のようなものがヒットした。

a. こどもプログラミング教室直伝！小学生プログラミング教育の概要解説と明日の授業でつかえる、入門・応用教材まるごとパック

Scratch を使って 90 分授業 2 本できる、プログラミングスクールが現場のノウハウを活かして作った教材で安心！教えたい人必見ネタを公開。「小学生を対象にした本格スクラッチ教材。入門編と応用編」明日から授業に使える付録付（スライド・テキスト）

￥19,800

https://www.udemy.com/tinkers-tryal/

b. Scratch 3.0 入門　親子で楽しむプログラミング！　スクラッチでゼロから学ぶプログラミング的思考の第一歩

2020 年の小学校でのプログラミングの必修化に向けて、子供と楽しみながら何か取り組んで見たいとお考えなら、世界で最も親しまれている Scratch がオススメです。プログラミングは、ビジネスにも役立つ論理的思考や問題解決能力の育成に役立ちます

￥10,800

https://www.udemy.com/scratch_programming/

c. Excel VBA[第 3 弾](実用マクロ入門編)受注データを 5 秒で入力できるユーザーフォーム作成 VBA プログラミング

シリーズ 2500 名が学んだ Excel マクロ VBA 講座の人気シリーズから、リクエスト多数につき[第 3 弾]が登場！今度のマクロは、受注データを 5 秒で入力できるユーザーフォーム編。 1 つのアプリを制作するかのように工程を楽しみながらスキルを磨こう！

￥4,200

https://www.udemy.com/excel23vba3/

（4）Moodle のプラグイン

a. 役立ちそうな Moodle プラグイン情報

http://mahakala.lesc.uec.ac.jp/mahoodle/moodle/plugins/

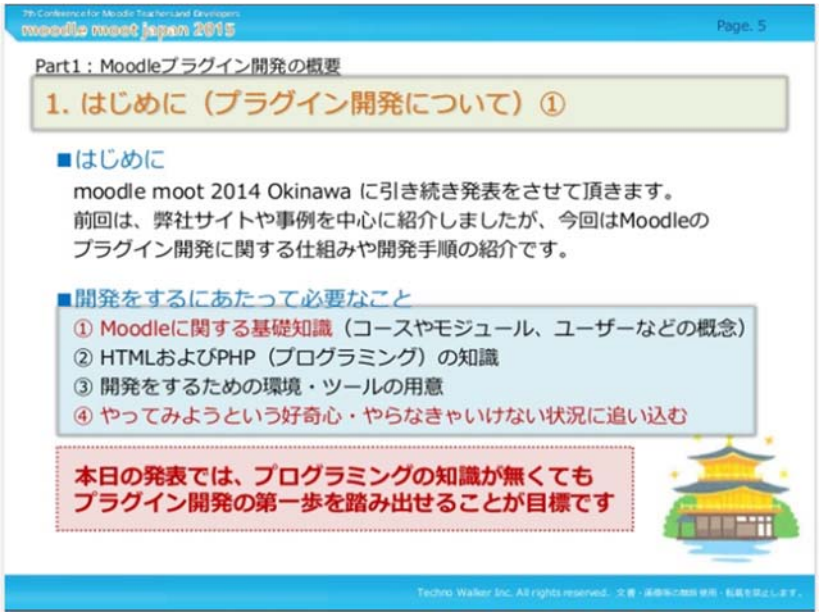
コンピテンシー関連プラグイン（※学習プラン関連プラグイン）

教材作成・エディタ関連プラグイン（※数式，化学式・化学構造エディタ，分子構造，プログラミング，問題作成，EJS）

モバイル関連プラグイン（※モバイル

モニタリング関連プラグイン（※GISMO，チェックリスト）
ポートフォリオ関連プラグイン（※Mahara 課題提出，マミライン，Exabis E-Portfolio）
アンケート関連プラグイン（※アンケート用活動モジュール）
レポート関連プラグイン
ルーブリック関連プラグイン（※Exabis Competencies，Exabis Student Review，LA e-Rubric）
テーマ関連プラグイン（※Essential，Bootstrap）
支援・管理ツール関連プラグイン
ビューア（Viewer)関連プラグイン

b. 開発初心者のための Moodle プラグインの開発と利用 for Moodle Moot 2015
https://www.slideshare.net/s-yamaoka/moodle-20150221
https://www.slideshare.net/s-yamaoka/moodle-for-moodle-moot-2015



（5）まとめ

・Mooc は大学中心に広がったシステムで、テキストと動画を手段としている。

・Mooc の LMS は Moodle ベースとなっている。

・Mooc と CodeMonkey クローンを接続する場合、Mooc の LMS と接続することになると予想される。

・CodeMonkey と同程度の管理機能を実現する場合、Moodle のプラグインを新規につくることになるのではないか。

・Udemy の LMS は公開されていないため、カスタマイズは不可能。

・クローンを何らかの Mooc 上につくるのか、独自のプラットフォーム上にするのか判断

が必要。

５．４　トリビアチャットボット

トリビアチャットボットは、CodeMonkey の学習コンテンツのひとつである。「コードの冒険」と同じくブラウザ上で動作するプログラミング教材で、「Python」を使って、メッセージのやり取りを行うチャットボットの作り方を学ぶコースだ。サルのモンタ・チャットボットが出すトリビア（豆知識）を問う質問に、ユーザーが回答、チャットボットが正否を判定して、チャットの最後に正答数に応じたメッセージを自動的に返す、という簡単なチャットボットの作り方を 16 のエクササイズを通して学ぶ。

Python ベースの CodeMonkey クローンができた場合、どのような学習体験になるのか、トリビアチャットボットである程度体験可能なのではないか。トリビアチャットボットの使用感について整理した。

（１）トリビアチャットボットとコードの冒険の違い

a. 指示出しの画面

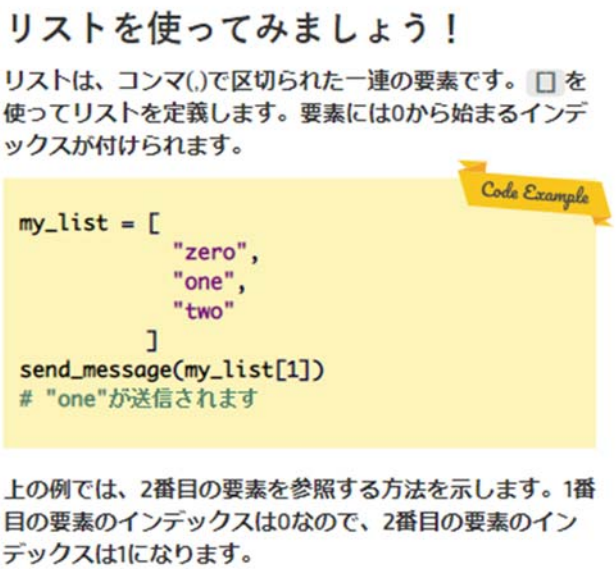
①コードの冒険

簡単なインストラクションが出る程度。

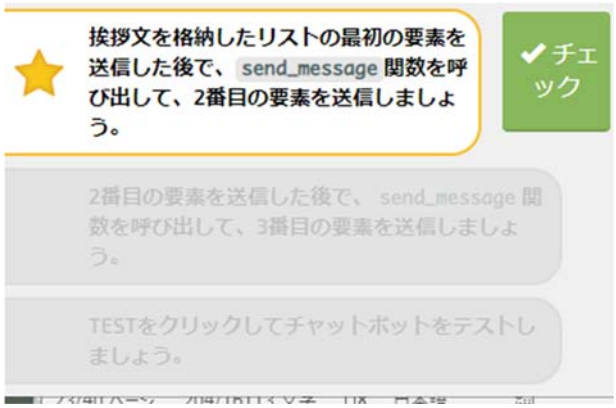
コード画面とゲーム画面にオーバーラップするダイアログして表示される。

②トリビアチャットボット

解説部分



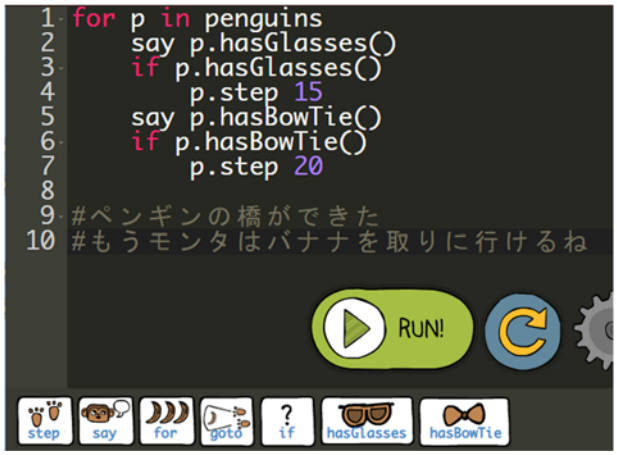
段階的に正解に導くチェック



この２つの部分は画面の左側にある。

b. コード画面

①コードの冒険



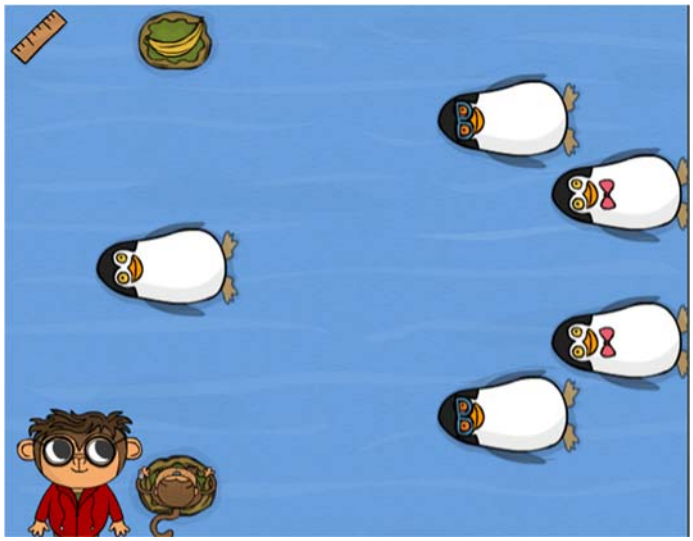
画面の右側にある。

②トリビアチャットボット

画面の真ん中にある。

c. ゲーム画面

①コードの冒険



画面の左側にある。

②トリビアチャットボット

画面の右側にある。

d. まとめ

コードの冒険では、説明は最小限である。トリビアチャットボットでは、1ペインが説明に当てられている。さらに、プログラムを段階的にチェックして、プログラムを完成させるようになっている。

コードの冒険では、ゲーム画面を見てコードを考えるという形であった。チャットボット

では、ゲーム画面を見て考えるというより、説明やコードのコメントを読んでプログラミングする形である。直感的な理解がしにくい。チャットボットの性格上、説明を主体にしないとできないのかもしれない。

（２）言語仕様の違い

ａ．プログラミング言語の違い

①コードの冒険

CoffeeScript を独自に拡張したもの。CoffeeScript の言語仕様に加えて、CodeMonkey 用のクラスやメソッドが追加されている。ループを単純化するための、「整数.times」は CodeMonkey の独自拡張である。

②トリビアチャットボット

Python を独自に拡張したもの。Python の言語仕様に加えて、CodeMonkey 用のクラスやメソッドが追加されている。CoffeeScript の「Until」ループは、Python には存在しない。「整数.times」も存在しない。トリビアチャットボットの仕様に含まれているかは未確認。

ｂ．言語の違いによる教育上の効果

教育向けプログラミング言語としての CoffeeScript

CoffeeScript は、JavaScript の優れた部分を簡単な方法で記述するために開発された言語である。教育を目的に作られたものではない。CodeMonkey の教育経験者が、Qiita にまとめた記事によると、「小学１年生から６年生までの児童に教えてみた経験を踏まえ、教育用として優れていると感じた点」は次の５点である。

この５点について、CoffeeScript と Python を比べてみた。

	CoffeeScript	Python
・文末セミコロン不要	○	△
・引数の括弧不要	○	X
・ブロック記述のための括弧不要	○	○
・モダンなプログラミング言語	○	○
・モダンブラウザで実行可能	○	○

文末セミコロンは Python でも不要であるが、Python では、if 文の最後、for 文・while 文の最後、関数定義の def 文の最後などに「:」が必要である。

引数の括弧は CoffeeScript では省略可能である。但し、引数なしの関数では最後に（）が必要である。Python では引数の括弧は必須である。

小さな違いであるが、初期の教育においては、教育上のつまづきを生みやすいポイントである。クローンを開発する場合、この２点について、プリプロセッサなどで処理することは有効であるかもしれない。

（3）その他の違い

a. コーディング規約の違い

①コードの冒険

キャメルケース（最初の単語以外の文字の先頭を大文字）

例.　turnTo a

②トリビアチャットボット

スネークケース（文字の単語間にアンダーバ）

例.　send_message(greetings[0])

b. 背景の音の有無

①コードの冒険

あり。鳥の声など。主人公の移動の効果音もあり。あったほうが気分が変わる。

②トリビアチャットボット

なし。無音。チャットの効果音はあり。無音で説明文を読むことになり、勉強の雰囲気である。

c. 日本語の利用

①コードの冒険

②トリビアチャットボット

いずれにおいても、コード中に日本語があるとうまく動かない。

チャットボットでは、メッセージを出すことが目的のプログラムであり、メッセージを英文字でしか出せないため、学習効果を決定的にそいでいる。CodeMonkey のサーバー側で、Unicode がうまく扱えない開発環境になっていると推測される。

（4）まとめ

・チャットボットでは、説明が多く、段階的な理解が必要である。説明が長いため、あまり読まれないのではないかと感じる。

・コードの冒険に比べて、やらされ感がある。

・背景音がない影響か、チャットボットではコードの冒険に比べて、没入感が低い。

・現在の CodeMonkey では、識別子を日本語化することができない。

・Python では、文末の「：」や関数の（）などの文法がうるさい。

・クローンの言語を Python ベースにする場合、文末の「：」や関数の（）などを簡略化する処理があったほうがいいかもしれない。

・識別子を日本語化する場合、何らかのコード規約が必要である。

付録１　コードの冒険に登場する予約語と識別子を英語と日本語の違いに注目して分析した資料

bridge	物/O	橋（はし）	はし			
bush	物/O	しげみ	しげみ			
cat	物/O	ネコ	ねこ			
chase()	関数	おいかける	おいかける	chase(xxx)	xxxを追いかける	xxxの位置
coconut	物/O	ココナツ	ここなつ			
collect()	関数	あつめる	あつめる	collect(xxx)	xxxを集める	xxxの位置
comparison	用語	コンパリソン、比較（ひかく）	こんぱりそん			
comparison	用語	コンパリソン、比較（ひかく）	ひかく			
crocodile	物/O	ワニ	わに			
crow	物/O	カラス	からす			
definition	用語	定義	ていぎ			
degree	用語	角度、度	かくど			角度と度の両方の意味がある
distanceTo()	関数	〜までの距離（きょり）	までのきょり	distanceTo(xxx)	xxxまでの距離	助詞がついている、xxxの位置
drop()	関数	おとす	おとす	drop()	引数なし	引数なし
else	命令	さもなければ	さもなくば	else xxx	さもなければxxxする	「さもなければ」は日常で使わない
else	命令	エルス	えるす			
for	命令	くりかえす、の間、のために	くりかえす	for a in aaa	ａをaaaの中で（繰り返す）、aがaaaの中である間（繰り返す）、aのためにaaaの中で（繰り返す）	繰り返すという動詞がない、日本語では「aaaの中にaがある/いる間」と表現したい。語順が逆。
frozen()	関数	こおっているかどうか	こおっている	xxx.frozen()	引数なし。xxxが凍っている。	引数なし。語順は同じ。
function	用語	関数（かんすう）	かんすう			
function	用語	ファンクション	ふぁんくしょん			
function	用語	関数	かんすう			
gate	物/O	門、扉（とびら）	もん			
gate	物/O	扉	とびら			
goat	物/O	ヤギ	やぎ			
gorilla	物/O	ゴリラ	ごりら			
goto()	関数	〜へ行く、〜に行く	へいく	goto(xxx)	xxxへ行く。	助詞が付いている。xxxの位置。
goto()	関数	〜へ行く、〜に行く	にいく	goto(xxx)	xxxに行く。	助詞が付いている。xxxの位置。

コードモンキー　用語集 / 辞書

Ver. 1.0　　2019/3/10 清風情報工科学院

英語	種類	意味	読み	英語語順	日本語語順	差
-	演算	マイナス、ひく	まいなす	-A	マイナスA	
-	演算	マイナス、ひく	ひく	A - B	AからBを引く	引くの位置
+	演算	プラス、たす	ぷらす	+A	プラスA	
+	演算	プラス、たす	たす	A + B	AにBを加える	加えるの位置
<	演算	～より小さい	よりちいさい	A < B	AはBより小さい	小さいの位置
<	演算	～より小さい	ちいさい	A < B	AはBより小さい	小さいの位置
=	演算	代入（だいにゅう）、いれる	だいにゅう	A = B	AにBを代入する	代入するの位置
=	演算	代入（だいにゅう）、いれる	いれる	A = B	AにBを入れる	入れるの位置
==	演算	～と等（ひと）しい、～と同じ	とひとしい	A == B	AはBに等しい、AとBは等しい	等しいの位置
==	演算	～と等（ひと）しい、～と同じ	とおなじ	A == B	AはBに同じ、AとBは同じ	同じの位置
>	演算	～より大きい	よりおおきい	A > B	AはBより大きい	大きいの位置
>	演算	～より大きい	おおきい	A > B	AはBより大きい	大きいの位置
A and B	命令	A かつ B	かつ	A and B	AかつB、AとB	「かつ」は日常で使わない
A or B	命令	AまたはB	または	A or B	AまたはB、AかB	AもBもの意味が日本語にはない
all	英語	全部（ぜんぶ）、すべて	ぜんぶ	all xxx	xxxの全部、全xxx	語順が逆、漢字熟語の「全○○」なら同じ
all	英語	全部（ぜんぶ）、すべて	すべて	all xxx	xxxのすべて、すべてのxxx	語順が逆、「すべての○○」は違和感あり
and	演算	アンド、かつ	あんど	A and B	AかつB、AとB	「かつ」は日常で使わない
and	演算	アンド、かつ	かつ	A and B	AかつB、AとB	「かつ」は日常で使わない
array	用語	配列	はいれつ			
array	用語	アレイ	あれい			
banana	物/O	バナナ	ばなな			
bat	物/O	コウモリ	こうもり			
bear	物/O	くま	くま			
beaver	物/O	ビーバー	びーばー			
boolean	用語	ブーリアン	ぶーりあん			
boolean	用語	論理式	ろんりしき			

no	定数	論理値（否定）、ろんりち（ひてい）	ひてい			
not	演算	ノット	のっと	not A	Aでない	語順が逆。漢字熟語の「不可能」「非常識」のような感覚。
not A	命令	Aの反対（はんたい）、Aでない	のはんたい	not A	Aでない	語順が逆。漢字熟語の「不可能」「非常識」のような感覚。
not A	命令	Aの反対（はんたい）、Aでない	でない	not A	Aでない	語順が逆。漢字熟語の「不可能」「非常識」のような感覚。
object	用語	オブジェクト	おぶじぇくと			
onClick()	関数	クリックされた時	くりっく	onClick()	イベントハンドラ	語順が逆。「クリックされた時」または「クリックされたら」。
onKey()	関数	キーボードおされた時	きーぼーど	onKey()	イベントハンドラ	語順が逆。「キー入力された時」または「キー入力されたら」。
onKey()	関数	キーボードおされた時	おされた	onKey()	イベントハンドラ	語順が逆。「キー入力された時」または「キー入力されたら」。
onMouseMove()	関数	マウスが動いた時	まうす	onMouseMove()	イベントハンドラ	語順が逆。「マウスが動いた時」または「マウスが動いたら」。
onMouseMove()	関数	マウスが動いた時	うごいたら	onMouseMove()	イベントハンドラ	語順が逆。「マウスが動いた時」または「マウスが動いたら」。
or	演算	オア、または	または			
or	演算	オア、または	おあ			
password	用語	パスワード	ぱすわーど			
pile	物/O	たば	たば			
plus	用語	プラス、たす	ぷらす			
plus	用語	プラス、たす	たす			
portion	物/O	薬（くすり）	くすり			
portion	物/O	ポーション	ぽーしょん			
pos	変数/V	positionの省略、位置（いち）	いち			

gotoNearestHealth()	関数	一番近くの健康ゾーンへ行く	いちばんちかくの	gotoNearestHealth	引数なし	「一番近くの健康ゾーンへ行く」で、語順がぐちゃぐちゃ。英語の語順は「いく、に、近く、一番、健康」
grab()	関数	つかむ	つかむ	grab()	引数なし	引数なし
green()	関数	緑色（みどりいろ）かどうか	みどりいろ	green()	引数なし	引数なし
health()	関数	健康かどうか	けんこうか	health()	引数なし	引数なし
hasBowTie()	関数	蝶ネクタイをしてるかどうか	ちょうねくたいしてる	hasBowTie()	引数なし	引数なし
hasGlasses()	関数	メガネをしてるかどうか	めがねしてる	hasGlasses()	引数なし	引数なし
healthZone	物/O	健康ゾーン、回復ゾーン	けんこうぞーん			
healthy	変数/V	健康	けんこう			
hippo	物/O	カバ	かば			
hit()	関数	打つ	うつ	hit()	引数なし	引数なし
If	命令	もし〜ならば	もし	if xxx	もしxxxならば	「ならば」の部分がないが、インデントが「ならば」と解釈すればよい。
If	命令	もし〜ならば	ならば			
injured()	関数	ケガしているかどうか	けがしている	injured()	引数なし	引数なし
island	物/O	島（しま）	しま			
key	変数/V	キーボードのキー	きーぼーど			
key	変数/V	キーボードのキー	きー			
key bind	用語	キーバインド	きーばいんど			
left	定数	左	ひだり			
loop	用語	ループ	るーぷ			
match	物/O	マッチ	まっち			
minus	用語	マイナス、ひく	まいなす			
minus	用語	マイナス、ひく	ひく			
monkey	物/O	さる	さる			
mover	変数/V	動くもの	うごくもの			
near	関数	〜の近くかどうか	のちかくか	near xxx	xxxの近くかどうか	語順が逆。漢字熟語の「近未来」のような感覚。
no	定数	論理値（否定）、ろんりち（ひてい	ろんりち			

turtle	物/O	カメ	かめ			
until	命令	～までくりかえす	までくりかえす	until xxx	xxxになるまで。	語順が逆。プログラミング言語でよく使うwhileとは真偽の判定が逆である。
until	命令	くりかえす（～まで）	くりかえす	until xxx	xxxになるまで。	語順が逆。プログラミング言語でよく使うwhileとは真偽の判定が逆である。
until	命令	アンティル	あんてぃる	until xxx	xxxになるまで。	語順が逆。プログラミング言語でよく使うwhileとは真偽の判定が逆である。
variable	用語	変数	へんすう			
waitFor()	関数	～を待つ	をまつ	waitFor(xxx)	xxxを待つ。	語順が逆。
watching()	関数	見ているかどうか	みているかどうか	watching()	引数なし	引数なし
yes	定数	論理値（肯定）、ろんりち（こうて	ろんりち			通常のプログラミング言語では「True」を使う。
yes	定数	論理値（肯定）、ろんりち（こうて	こうてい			通常のプログラミング言語では「True」を使う。
yummy()	関数	おいしいかどうか	おいしいか	yummy(xxx)	xxxが美味しいかどうか	語順が逆。

raft	物/O	いかだ	いかだ			
return	命令	値（あたい）を返す	あたいをかえす	return xxx	xxxを返す。	語順が逆。さらに、「値を返す」という意味がわかりにくい。
return	命令	返す（値を）	かえす			
right	定数	右	みぎ			
rotten()	関数	腐（くさ）ってるかどうか	くさっている	rotten()	引数なし	引数なし
safe()	関数	安全（あんぜん）かどうか	あんぜんか	safe()	引数なし	引数なし
safeCollect()	関数	安全に集（あつ）める	あんぜんにあつめる	safeCollect(xxx)	xxxを安全に集める。	語順がばらばら。
safeFrom()	関数	～から安全かどうか	からあんぜんか	safeFrom(xxx)	xxxから安全かどうか	語順が逆。
say()	関数	言う，しゃべる	いう	say(xxx)	xxxと言う。	語順が逆。
say()	関数	言う，しゃべる	しゃべる	say(xxx)	xxxと言う。	語順が逆。
setX()	関数	Xにセットする	せっとする	setX(xxx)	xxxをXにセットする。	語順が逆。
setX()	関数	エックスにセットする	えっくす	setX(xxx)	xxxをXにセットする。	語順が逆。
setY()	関数	Yにセットする	せっとする	setY(xxx)	xxxをYにセットする。	語順が逆。
setY()	関数	ワイにセットする	わい	setY(xxx)	xxxをYにセットする。	語順が逆。
sleeping()	関数	ねている	ねている	sleeping()	引数なし	引数なし
step()	関数	すすむ	すすむ	step(xxx)	xxx進む	語順が逆。
stepper	変数/V	すすむもの	すすむもの			
tiger	物/O	トラ	とら			
times	命令	～回くりかえす	かいくりかえす	xxx.times	xxx回繰り返す	語順は同じ。「繰り返す」がない。
times	命令	くりかえす（～回）	くりかえす	xxx.times	xxx回繰り返す	語順は同じ。「繰り返す」がない。
toss()	関数	投（な）げる	なげる	toss()	引数なし	引数なし
turn()	関数	～に向（む）く	にむく	turn(xxx)	xxx度回る。	語順が逆。またプラスが右回りなのか左回りなのか、マイナスが右回りなのか左回りなのか、わかりにくい。
turnTo()	関数	～のほうに向く	のほうにむく	turnTo(xxx)	xxxのほうを向く。	語順が逆。turn xxxなら「回る」だが、turnTo xxxなら「向く」になって日本語では動詞が違う。

おとす	おとす	関数	drop()
おぶじぇくと	オブジェクト	用語	object
かいくりかえす	～回くりかえす	命令	times
かえす	返す（値を）	命令	return
かくど	角度、度	用語	degree
かつ	A かつ B	命令	A and B
かつ	アンド、かつ	演算	and
かば	カバ	物/O	hippo
かめ	カメ	物/O	turtle
からあんぜんか	～から安全かどうか	関数	safeFrom()
からす	カラス	物/O	crow
かんすう	関数（かんすう）	用語	function
かんすう	関数	用語	function
きー	キーボードのキー	変数/V	key
きーばいんど	キーバインド	用語	key bind
きーぼーど	キーボードのキー	変数/V	key
きーぼーど	キーボードおされた時	関数	onKey()
くさっている	腐（くさ）ってるかどうか	関数	rotten()
くすり	薬（くすり）	物/O	portion
くま	くま	物/O	bear
くりかえす	くりかえす、の間、のために	命令	for
くりかえす	くりかえす（～回）	命令	times
くりかえす	くりかえす（～まで）	命令	until
くりっく	クリックされた時	関数	onClick()
けがしている	ケガしているかどうか	関数	injured()
けんこう	健康	変数/V	healthy

コードモンキー　用語集 / 辞書

清風情報工科学院

読み	意味	種類	英語
あたいをかえす	値（あたい）を返す	命令	return
あつめる	あつめる	関数	collect()
あれい	アレイ	用語	array
あんぜんか	安全（あんぜん）かどうか	関数	safe()
あんぜんにあつめる	安全に集（あつ）める	関数	safeCollect()
あんてぃる	アンティル	命令	until
あんど	アンド、かつ	演算	and
いう	言う，しゃべる	関数	say()
いかだ	いかだ	物/O	raft
いち	positionの省略、位置（いち）	変数/V	pos
いちばんちかくの	一番近くの健康ゾーンへ行く	関数	gotoNearestHealth()
いれる	代入（だいにゅう）、いれる	演算	=
うごいたら	マウスが動いた時	関数	onMouseMove()
うごくもの	動くもの	変数/V	mover
うつ	打つ	関数	hit()
えっくす	エックスにセットする	関数	setX()
えるす	エルス	命令	else
おあ	オア、または	演算	or
おいかける	おいかける	関数	chase()
おいしいか	おいしいかどうか	関数	yummy()
おおきい	～より大きい	演算	>
おされた	キーボードおされた時	関数	onKey()

ていぎ	定義	用語	definition
でない	Aの反対（はんたい）、Aでない	命令	not A
とおなじ	〜と等（ひと）しい、〜と同じ	演算	==
とひとしい	〜と等（ひと）しい、〜と同じ	演算	==
とびら	扉	物/O	gate
とら	トラ	物/O	tiger
なげる	投（な）げる	関数	toss()
ならば	もし〜ならば	命令	If
にいく	〜へ行く、〜に行く	関数	goto()
にむく	〜に向（む）く	関数	turn()
ねこ	ネコ	物/O	cat
ねている	ねている	関数	sleeping()
のちかくか	〜の近くかどうか	関数	near
のっと	ノット	演算	not
のはんたい	Aの反対（はんたい）、Aでない	命令	not A
のほうにむく	〜のほうに向く	関数	turnTo()
はいれつ	配列	用語	array
はし	橋（はし）	物/O	bridge
ぱすわーど	パスワード	用語	password
ばなな	バナナ	物/O	banana
びーばー	ビーバー	物/O	beaver
ひかく	コンパリソン、比較（ひかく）	用語	comparison
ひく	マイナス、ひく	演算	-
ひく	マイナス、ひく	用語	minus
ひだり	左	定数	left
ひてい	論理値（否定）、ろんりち（ひてい）	定数	no

けんこうか	健康かどうか	関数	health()
けんこうぞーん	健康ゾーン、回復ゾーン	物/O	healthZone
こうてい	論理値（肯定）、ろんりち（こうてい）	定数	yes
こうもり	コウモリ	物/O	bat
こおっている	こおっているかどうか	関数	frozen()
ここなつ	ココナツ	物/O	coconut
ごりら	ゴリラ	物/O	gorilla
こんぱりそん	コンパリソン、比較（ひかく）	用語	comparison
さもなくば	さもなければ	命令	else
さる	さる	物/O	monkey
しげみ	しげみ	物/O	bush
しま	島（しま）	物/O	island
しゃべる	言う，しゃべる	関数	say()
すすむ	すすむ	関数	step()
すすむもの	すすむもの	変数/V	stepper
すべて	全部（ぜんぶ）、すべて	英語	all
せっとする	Xにセットする	関数	setX()
せっとする	Yにセットする	関数	setY()
ぜんぶ	全部（ぜんぶ）、すべて	英語	all
だいにゅう	代入（だいにゅう）、いれる	演算	＝
たす	プラス、たす	演算	＋
たす	プラス、たす	用語	plus
たば	たば	物/O	pile
ちいさい	～より小さい	演算	＜
ちょうねくたいしてる	蝶ネクタイをしてるかどうか	関数	hasBowTie()
つかむ	つかむ	関数	grab()

ろんりち	論理値（否定）、ろんりち（ひてい）	定数	no
ろんりち	論理値（肯定）、ろんりち（こうてい）	定数	yes
わい	ワイにセットする	関数	setY()
わに	ワニ	物/O	crocodile
をまつ	〜を待つ	関数	waitFor()

ふぁんくしょん	ファンクション	用語	function
ぶーりあん	ブーリアン	用語	boolean
ぷらす	プラス、たす	演算	+
ぷらす	プラス、たす	用語	plus
へいく	〜へ行く、〜に行く	関数	goto()
へんすう	変数	用語	variable
ぽーしょん	ポーション	物/O	portion
まいなす	マイナス、ひく	演算	-
まいなす	マイナス、ひく	用語	minus
まうす	マウスが動いた時	関数	onMouseMove()
または	AまたはB	命令	A or B
または	オア、または	演算	or
まっち	マッチ	物/O	match
までくりかえす	〜までくりかえす	命令	until
までのきょり	〜までの距離（きょり）	関数	distanceTo()
みぎ	右	定数	right
みているかどうか	見ているかどうか	関数	watching()
みどりいろ	緑色（みどりいろ）かどうか	関数	green()
めがねしてる	メガネをしてるかどうか	関数	hasGlasses()
もし	もし〜ならば	命令	If
もん	門、扉（とびら）	物/O	gate
やぎ	ヤギ	物/O	goat
よりおおきい	〜より大きい	演算	>
よりちいさい	〜より小さい	演算	<
るーぷ	ループ	用語	loop
ろんりしき	論理式	用語	boolean

付録２　コードの冒険のつまづき対策資料

91〜105	8	untilループ	until near match 　　turnTo match 　　step 1	マッチに近づくまで マッチの方に向きを変える 1歩進むを繰り返す	・最初から until near オブジェクトで始まる ・「until メソッド オブジェクト」 ・untilループ内で、「メソッド 定数」 ・untilループ内で、「メソッド メソッド 定数」 ・until 数式、until 関数がない ・「until オブジェクト.メソッド オブジェクト」 ・関数内でuntilループ ・untilループ内で、「メソッド 引数」 ・forループ内で関数呼び出ししその中でuntilループ ・何もしない wait()という関数 ・wait()との組み合わせで何もしないuntilループ ・関数の入れ子 ・forループ内でuntilループと関数の入れ子呼び出し ・何もしないuntilループと何かするuntilループの組み合わせ ・99から複雑	関数の戻り値をそのまま繰り返しの判定としているため、ここを曖昧にして進んだ場合に125からのbooleanで再度同じようにひっかかってしまう。 条件を満たすまでの繰り返しであることの理解	・英語の語順だと「near match」になる。日本語なら「マッチの近く」と語順が逆。 ・「until〇〇」も「〇〇になるまで」と語順が逆。		★変数と関数の引数のスコープについて書き出して納得する（特にチャンレンジ９９以降）		判断	untilなし	untilなし	untilなし	untilなし
	P8								・untilの直後に数式が来るパターンを練習する ・関数の戻り値を変数に入れて、その変数をuntilで評価する練習。わかりにくい場合はn == trueまで書かせる。 ・until i == 10, until i >=10 などの練習 ・until n, until n == trueなどの練習						
106〜116	9	if文	if banana.frozen() 　　goat.goto banana 　　goat.hit()	もしバナナが凍っている場合は ヤギがバナナの場所に行き ヤギが氷を砕く	・オブジェクト.メソッド() ・オブジェクト.メソッド オブジェクト ・オブジェクト.メソッド 配列の要素 ・if オブジェクト.メソッド() ・if節の中で、オブジェクト.メソッド() ・if節の中で、オブジェクト.メソッド オブジェクト ・forループの中で、if 変数.メソッド() ・forループの中で、if節の中のオブジェクト.メソッド()、ブジェクト.メソッド オブジェクト	条件の定義	・「banana.frozen()」は「バナナが凍っている」で、語順が同じ。				判断				
117〜124	10	if　else文	if banana.green() 　　goat.goto banana else 　　goto banana	もしバナナが緑色の場合は ヤギがバナナの場所に行き そうでない場合は サルがバナナの場所に行く	・if オブジェクト.メソッド() ・if節の中で、オブジェクト.メソッド オブジェクト ・else節の中で、メソッド オブジェクト ・forループの中で、if節・else節 ・forループの中で、if 変数.メソッド() ・forループの中のif節の中で、メソッド 変数、オブジェクト.メソッド 変数 ・関数の中で、if 引数.メソッド ・関数の中のif節・else節の中で、メソッド 変数、オブジェクト.メソッド 変数 ・124は応用できる	条件の定義	・「banana.green()」は「バナナが緑色」で、語順が同じ。				判断				
	P10								★forと配列とgoto以外の関数（引数は変数）の組み合わせの練習 ★チャレンジ124の解き方をいろいろやらせる【別解】 　　関数使わずに書いてみる、 　　先に判定してからどっちが行くか決める、 　　先にバナナに行ってから判定する、とか ★returnを先にここで入れたほうがわかりやすいかも 　　凍ってるかの判定関数 　　青いバナナかの判定関数								
125〜145	11	and，or	until bear.sleeping() and tiger.sleeping() 　　wait() until bear.sleeping() or bear.playing() 　　wait()	クマとトラが眠るまで 待つ クマが眠るか、遊ぶまで 待つ	・until とandの組み合わせ ・forループとuntilループとandの組み合わせ ・forループとuntilループとandとif/else文の組み合わせ ・関数内でuntilループとandの組み合わせ ・until とorの組み合わせ ・forループとuntilループとorの組み合わせ ・forループとuntilループとorとif/else文の組み合わせ ・関数内でuntilループとorの組み合わせ ・for in [オブジェクト, オブジェクト] ・for文の入れ子 ・for文の入れ子の中で、for in [オブジェクト, オブジェクト] ・for文の入れ子の中からuntilを含む関数を呼ぶ ・チャレンジ140は応用できる ・チャレンジ141-145は追加チャレンジ ・for文の中のif文（if 引数.メソッド） ・for文の中のif文の中の引数.メソッド 定数 ・for文の中のif文で and ・for文の中のif/else文で and / or ・for文の入れ子の中でif文の中に or	andとorの違い	・「bear.sleeping()」は「熊が寝ている」で、語順が同じ。 ・「A and B」は「AもBも」、「A or B」は「AかBか」。日本語の「AかBか」は「男か女か」のようにどちらか一方の意味で、コンピュータの意味と違うのに注意する。 ・「hasGlass()」「hasBowTie()」の「has」は日本語で「メガネをかける/する」「蝶ネクタイをつける/する」と動詞が違う	（and, or の関係がわからない可能性）	・「A and B」は「AもBも」、「A or B」は「AかBか」。日本語の「AかBか」は「男か女か」のようにどちらか一方の意味で、コンピュータの意味と違うのに注意する。		ベン図				
	P11								★この辺でも、forと配列とgoto以外の関数（引数は配列）の組み合わせ ★チャレンジ124の解き方をいろいろやらせる 　stepperでループする考えでやるとbananasを引数にすることになる ★チャレンジ140のやり方をいろいろやらせる【別解】 　配列のtigersを引数にした関数 　配列のbananasを引数にした関数								

コードモンキーの補完ポイント

Ver. 3.0　　2019/3/10　　清風情報工科学院

チャレンジ	レッスン	名称	主な処理	意味	組み合わせ	つまずきポイント（２校）	英語	数学	分析（コンピュータ）	他の言語との接続	アルゴリズム	C言語比較	Java比較	Python比較	PHP比較
0〜10	1	step，turn	step 10 turn 45 turn right	10歩進む 45° 向きを変える 右を向く	・メソッド 数値 ・メソッド 定数（rightなど） ・「（オブジェクト.)メソッド」の「オブジェクト.」部分が省略されている	定規の正しい使い方	・「step 10」は「10進む」で語順が逆。 ・語順が逆のものは、命令だから倒置法になっていると思えば、ある程度理解できる。「進め１０」「回れ４５度」「回れ右！」	・角度において「−」が右回りか左回りか。ノートに絵を書いて整理させる。 ・猿基準の長さ・角度しか出ない事への対処。定規・角度がピンと来ていない場合、「トードーの算数」をやらせる。	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。 ・語順が逆のものは、命令だから倒置法になっていると思えば、ある程度理解できる。「進め１０」「回れ４５度」「回れ右！」						
	P2								・レッスン１で省略された部分を加えて「オブジェクト.メソッド」の練習 ・（紙上で、）レッスン１の題材のいくつかについて「monkey.」をつけて書かせる						
11〜20	2	turnTo，カメ	turnTo banana turtle.turnTo banana turtle.step 10	バナナの方に向きを変える カメがバナナの方に向きを変える カメが10歩進む	・レッスン1では「monkey.」が省略されている。このレッスンの直前にプレステージP2を入れる。 ・メソッド オブジェクト ・オブジェクト.メソッド 引数 ・オブジェクト.メソッド オブジェクト	オブジェクトに対する命令（monkey.が省略されていると理解できず、turtle.stepになぜturtle.が必要なのか戸惑う） turnとturnToの違い 一人称（monkey）を省ける事	・「turnTo」の前置詞「to」がとれていない。日本語では「４５度回る」「バナナに向く」と動詞が変わって助詞がつく。英語ではそれが、turn, turnToに対応と説明。 ・「to」がつくと、命令と思っても語順が理解しにくくなる。（「回れバナナ」ではバナナが回ってしまう。「バナナ向けバナナ」とでも理解するのか。）		・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。駅から学校、玄関から教室までを例に説明する。単純化した地図をつくっておく。 ・レッスン１では「monkey.」が省略されている。この章の直前にプレステージP2を入れる。						
21〜30	3	timesループ	3.times -> 　turn left 　step 5	左を向く、5歩進む を3回繰り返す	・ループ関数(整数.times）とメソッド ・変数.times ・ループ関数とメソッド、オブジェクト.メソッド	アルゴリズムの構築（バナナを取るまでの手順の明確化）	・「3.times」の語順は日本語と同じ「3回」。	（手順を書き出すという習慣の有無）	・アルゴリズムの身体化が必要。実際に体を動かしたり、ノートの上で書いて整理させる。３階まで階段を登るなどを例に（繰り返しで表現できる）。		ループ	for（初期化式; 条件式; 変化式）	for（変数：コレクション）	for 変数 in range(整数)	foreach（コレクション as $変数）
31〜50	4	変数	a = 10 step a b = distanceTo banana	=の値を代入する	・変数、変数への代入 ・変数の自己代入 ・distanceTo オブジェクト ・ループとメソッド、オブジェクト.メソッド ・追加チャレンジ（46-50）にて、ループと自己代入	変数を使う意味が理解できない 代入した値をイメージできない	・「distanceTo ○○」は「○○までの距離」と語順が逆。 ・「step distanceTo ○○」は「○○までの距離を進む」で語順が完全に逆。	（数学の文字式とコンピュータの変数の概念の違い） （数学の等式とコンピュータの代入の概念の違い）	・変数の機能をイメージできていない。コンピュータでの文字・変数は「箱」なので、「箱」をつかって説明する。自己代入のi=i+1を「箱」で実感させる。	★i=i+1は、i++と書けることを教えておく。			+=, ++	+=, ++	+=, ++	+=, ++
51〜60	5	配列	beavers[0].step 10 step distanceTo bananas[1]	ビーバー[0]が10歩進む バナナ[1]の距離進む	・メソッド 配列の要素 ・オブジェクト.メソッド 配列の要素 ・配列の要素.メソッド 引数 ・配列とループの組み合わせはなし ・55から57は応用できる	変数と配列の違いがわからない（bananaとbananasの「s」の有無で変数か配列かが異なるということへの戸惑い） 手順を間違えやすい	・配列は英語の複数形で「s」が付いていることに注意させる。	（手順を書き出すという習慣の有無）	・手順をノートに書かせる。 ・配列と変数の違いを説明する。どちらも「箱」なのは同じ。グループとメンバーの関係。						
	P4								・i=i+○の練習量が不足 ・（紙上で、）５章の題材のいくつかについて変数と自己代入のi=i+○を使って書かせる。 ★55-57をつかって、定数部分を変数に置き換えて変数を配列を組み合わせる練習、さらにi=i+1などと組み合わせる練習						
61〜75	6	forループ	for b in bananas 　turnTo b 　step distanceTo b	バナナの方に向きを変える バナナの距離進む をbananas[0]〜順に繰り返す	・forループと変数 ・forループと「メソッド 変数」 ・forループと「メソッド メソッド 変数」 ・forループと「変数.メソッド オブジェクト」 ・forループの入れ子 ・入れ子のforループの演習が不足（75のみ）	forが繰り返しであるという理解 Tab（４つスペース）の意味	・「for」という英語がピンとこない。直訳すると「bがバナナ配列の中にいる間」。または「bのためにバナナ配列の中で繰り返す」と意訳する。		・「for」を使うメリットを説明する。Forを使わずに５回・１０回と回数が増えるとミスが多くなる。		ループ	for（変数：コレクション）	for（変数：コレクション）	for 変数 in コレクション	foreach（コレクション as $変数）
	P6								★forループの入れ子の例題をいくつかやる必要あり	★（紙上で、）３章と６章の題材のいくつかを使って、for のカウンタ変数を使った処理を学ぶ。 ★for (i=1, i<=10, i++)に対応した処理の仕方を教える。（for i in [1..10]、for i in [1..bananas.length]）（または、i= 1 10.times 処理 i++、10.times文はループの中で何回目かわからない） ★配列の数を数える方法は、「配列.length」（bananas.length、バナナの長さ。日本語と語順が同じ）	ループと判断の関係	for（初期化式; 条件式; 変化式）	for（初期化式; 条件式; 変化式）	for 変数 in シークエンス	for（初期化式; 条件式; 変化式）
76〜90	7	関数	goto = (a) -> 　turnTo a 　step distanceTo a	関数の定義	・関数()　引数なしの関数呼び出し ・関数 定数 で関数呼び出し ・関数の定義　（但し、goto関数のみ） ・関数の入れ子（関数内で「メソッド 引数」「メソッド メソッド 引数」） ・関数 配列 で関数呼び出し ・関数内でforループ ・関数 変数 で関数呼び出し ・88から応用できる ・２変数の関数定義、２変数の関数呼び出しも可能	goto以外の関数をつくらせないので、関数＝gotoという理解をする学生がいる。 引数の説明が流されており、引数に変数、引数に配列の違いがわかりにくい。どこかで補足が必要。 goto以下はあくまで関数であるという理解 引数の理解	・「goto ○○」も「○○へ行く」と語順が逆。	（数学の関数の概念とコンピュータの関数の概念の違い）	★goto 配列（goto bananas）に対応するには？ ★チャレンジ88から90を利用して様々な組み合わせに挑戦させる ★チャレンジ88から90を利用してどういう目的でどういう分割がわかりやすいのか考えさせる（89の模範解答はモジュール分割がおかしい。【別解】） ・goto以外の関数のイメージをふくらませるためにレッスン追加		サブルーチン				
	P7								・関数のイメージの膨らませ。 ・紙上で、関数づくりの練習をさせる。いろんな関数、いろんな引数 ・コンビニへ行く ・駅で乗り換えの例（乗り換えTo 阪和線） ★引数が２つ以上のイメージの膨らませ。 ★コピー機の例（コピー（100%,5枚）） ★さらに引数として変数を渡した時の挙動（変数のスコープ） ★引数も日本語の語順が逆になる。「copy (100%, 5枚)」なら「100%で、５枚、コピーする」。						

146〜151	12	not	if not banana.rotten() 　　goto banana	もしバナナが腐っていない場合は バナナの場所に行く	・if文の中でnot ・配列の思い出し ・for節の中でif/else文 ・for節の中のif文でnot	notを書く位置（文法的な）	・英語の「not」の位置がピンときにくいので、漢字熟語を例にして説明する。「不合格、不参加、非常識」などの「不○○」とか「非○○」のイメージ。				ベン図とnot, and, or				
	P12									★このへんで、whileを入れたほうがいい。 ★３，６，７章を使うか、９から１２章を使うか。 ★until ugeuge と while not ugeuge が同じという理解をさせて、whileのループへ導く。（untilは○○になるまで、whileは X X の間） ★until bear.sleeping(), while not bear.sleeping(), while bear.起きている() のように誘導する。単にnotをつけるだけでなく関数の名前を変えたほうがよいことを理解させる。 ★while型がわかりやすい時とuntil型がわかりやすい時を理解させる。 ★「起きている関数(not sleeping())、生の関数(not rotten())、黄色関数(not green())」などを作つくって not 部分をラッピングすることが可能か？	判断	untilがないため	untilがないため	untilがないため	untilがないため
152〜160	13	<，==	if health()<70 　　goto healthZone 　　until health()==100 　　　　wait()	もし体力が70より少ない場合は 回復場所に行き 体力が100になるまで 待つ	・if文の中で メソッド() == 定数 ・until文の中でメソッド() == 定数 ・for文の中のuntil文の中でメソッド() == 定数 ・if/else文の中で メソッド() < 定数 ・for文中でif メソッド() < 定数 ・for文中のif文 の中で、until メソッド() == 定数 ・whileと組み合わせたほうがわかりやすいものもある	大なりと小なりの違い 以上と以下と違い、該当の数字は含まない事	・「A>B」は小学校で「A大なりB」と教わるが、これも本来は語順が違う。「AがBより大きい」と「大きい」の部分が後ろに来るのが日本語の語順。 ・「healthZone」は「回復場所」で日本語の語順と同じ。直訳すると「健康場所」でちょっと変。	（大小関係の不等号がわからない可能性）			判断				
	P14								・紙上で、関数づくりの練習で、戻り値ありの場合をさせる。いろんな関数、いろんな引数で。 ・関数のイメージの膨らませを応用して説明。成功したか失敗したかを返す。 ・駅で乗り換えの例（乗り換えTo 阪和線　成功・失敗） ・コピー機の例（コピー（100%,5枚）成功・失敗） ・代金の例（代金（100円、５個）） ・消費税の例（消費税（8%､100円）） ・消費税の例（消費税（8%､100円）） ・消費税の例（消費税（8%､100円））							
161〜175	14	戻り値	return no return yes return not crow.watching()	定義された値を返す	・関数の中でreturn ・関数の中のreturnでnot ・if/else文の中で関数() ・161のステップが大きい（が、みんな問題ないようだ） ・for文の中でif 関数 引数 ・return文の中で メソッド()==定数 ・return文の中で メソッド() < 定数 ・for文中のif文 の中でuntil文 ・return文の中で or ・return文の中で 引数.メソッド() ・until文の中で関数 オブジェクト and 関数 オブジェクト ・return yes/no ・returnのあとは実行されない ・if/else文の中で 関数() ・return not の戻り値とuntilの組み合わせ ・return not とorと配列の組み合わせ ・return not とorと配列の組み合わせとifとforの組み合わせ ・for文とreturnとnotの組み合わせをuntilと組み合わせる ・for文とif文とreturnの組み合わせをuntilと組み合わせる ・whileと組み合わせたほうがわかりやすいものもある	値を返すという意味自体がわからない	・「return ○○」も、語順が逆で「○○を返す」。「戻り値は○○」と読み替えたほうがしっくりくる。 ・「crow.watching()」は「カラスが見てる」で語順が同じ。		（値を返すという例を別途用意して実感させることが必要）		関数				
	P14								・紙上で、関数づくりの練習で、戻り値ありの場合をさせる。いろんな関数、いろんな引数で。 ・関数のイメージの膨らませを応用して説明。成功したか失敗したかを返す。 ・駅で乗り換えの例（乗り換えTo 阪和線　成功・失敗） ・コピー機の例（コピー（100%,5枚）成功・失敗） ・代金の例（代金（100円、５個）） ・消費税の例（消費税（8%､100円））		判断	untilがないため	untilがないため	untilがないため	untilがないため
176〜189	15	onKey	onKey = (key) -> 　　if key == 'w' 　　　　step 1	Wキーを押した場合 1歩進む	・イベントドリブン（onKey） ・onKeyとif文の組み合わせ	有利に進めるキーの設定（キーをどれだけ増やしても星３とれる）	・「onKey」も「キーが押されたら」と語順が逆。「押された」の部分は英語にはない。		（イベントドリブンな記述の仕方）						
190〜195	16	onMouseMove	onMouseMove = (pos) -> 　　bat.setX pos.x 　　bat.setY pos.y	マウスのX軸とY軸の定義	・イベントドリブン（onMouseMove） ・onKeyとonMouseMoveの組み合わせ ・setX、setY ・onKeyとif、onMouseMoveとsetX, setYの組み合わせ	マウスの動きに関する設定である事	・「onMouseMove」も「マウスが動いたら」と語順が逆。 ・「setX」も「Xにセット」と語順が逆。	（座標がわからない可能性）	（イベントドリブンな記述の仕方）						
196〜210	17	onClick	hippo.onClick = () -> 　　monkey.turnTo hippo 　　monkey.toss()	カバをクリックすると サルがカバの方を向き サルが投げる	・イベントドリブン（オブジェクト.onClick） ・forループと変数.onClick ・forループと変数.onClickと変数.メソッド, メソッド 変数 ・moverという変数 ・onClickとonKeyの組み合わせ ・forループと変数.onClickとmoverとonKeyとifの組み合わせ	対象物をクリックした場合の命令であること	・「onClick」も「クリックされたら」と語順が逆。		（onClickはオブジェクトに設定する。それがオブジェクトを対象とした文で宣言するということと、その実行はonClickイベントで発動されるということの発想の切り分けが難しい。他の部分は順次処理であるが、イベントドリブンな部分は割り込みになってその中では順次処理ということ。）						