Introduction to Python Programming

Using LEGO Education SPIKE Prime Set

Python プログラミング
レッスンガイド Part 2 日本語版

Learning Systems™

SAMPLE

LEGO, the LEGO logo, the minifigure and the SPIKE logo are trademarks and /or copyrights the LEGO Group.
©2024 The LEGO Group. All rights reserved.

目　次

Lesson Objective: レッスンの目的（ レッスン一覧）　P2 ～ 7
レッスンガイド　P8 ～
Unit 4 Doing Reps with Loop: ループを使った繰り返し処理
Lesson 1　P9
Lesson 2　P12
Lesson 3　P16
Lesson 4　P21
Lesson 5　P24
Lesson 6　P27

Unit 5 Playing Games Simple Condition：
簡単な条件でゲームをする
Lesson 1　P30
Lesson 2　P35
Lesson 3　P38
Lesson 4　P42
Lesson 5　P46
Lesson 6　P51
Lesson 7　P55

Unit 6 Testing for Bugs:
Trounledhooting Hardware and Software：
バグのテスト：ハードウェア＆ソフトウェアのトラブルシュート
Lesson 1　P58
Lesson 2　P60
Lesson 3　P63
Lesson 4　P66
Lesson 5　P70

生徒用ワークシート　P73 ～

Lesson Objectives：レッスンの目的

Part 1

Unit 1 Hardware & Software ユニット1ハードウェア&ソフトウェア	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Importing Libraries ライブラリのインポート （授業時間：45 分）	• Python にライブラリをインポートする必要がある理由を学ぶ • ライブラリのインポート • プログラムを実行します
Lesson 2 Communicating with Light 光でコミュニケーション （45 分）	• ハードウェアとソフトウェアの機能の説明 • ライトマトリックスをプログラムし、簡単なプログラムをデバッグする方法を学びます
Lesson 3 Pair Programming 共同プログラミング （45 分）	• ペアで共同してプログラミングの練習 • プログラムを修正します
Lesson 3 Communicating with Sounds 音でコミュニケーション （45 分）	• ハードウェアとソフトウェアの機能の説明 • プログラムの音やビープ音を鳴らし、簡単なプログラムをデバッグする方法を学びます • サウンドパターンを作成します
Lesson 4 Digital Sign デジタル・サイン （45 ～ 90 分）	• ハードウェアとソフトウェアの機能について説明します • 光と音をプログラムしてメッセージを伝えます

Unit 2 Motors ユニット 2 モーター制御	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Making Moves with Motor モーターで動かす （授業時間：45 分）	• 時間と速度のパラメータを使用してモーターが個別に回転するようにプログラムします • ロボットダンスパーティーを作成します
Lesson 2 New Moves with Motors モーターによる新たな動き （45 分）	• 最短経路を使用して位置に移動するようにモーターをプログラムします • モーターをプログラムして特定の位置に移動します • 定義された角度だけ動かすようにモーターをプログラムします
Lesson 3 Automating Action アクションの自動化 （45 分）	• タスクを自動化するモデルを構築してプログラムします
Lesson 4 Hopper Run ホッパーラン （45 分）	• 2 つのモーターを同時に動かすようにプログラム • 前進する車輪のないロボットを構築してプログラムします
Lesson 5 Race Day ホッパーレース （45 ～ 90 分）	• 一連のステップとターンを移動するプログラムを作成します • モーターペアを複数の方法で活用

SAMPLE

Unit 3 Sensor Control ユニット 3 センサー制御	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Start Sensing センシング開始 （授業時間：45 分）	• 力センサーをプログラムします • 条件文を作成します
Lesson 2 Charging Rhino 突進するサイ （45 分）	• 力センサーについて調べる • 力が動きに及ぼす影響を理解します
Lesson 3 Cart Control カートコントロール （45 分）	• 距離センサーをプログラムします • 距離のある動きを探る • 超音波を理解します
Lesson 4 Safety Delivery 安全な配達 （45 分）	• センサーを使用して安全に移動するモデルをプログラムします • センサー使用時のモーター出力の影響を調べる
Lesson 5 Grasshopper Trouble バッタのトラブル （45 ～ 90 分）	• 適切なハードウェアの選定を行う • センサーを追加するためにモデルを再設計します

Part 2

Unit 4 Loops & Variables ユニット 4 ループ（繰り返し）と変数	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Warm Up Loop with Leo レオとのウォームアップループ （授業時間：45 分）	• ループを使ってプログラムします • 腹筋マシンを組み立て、プログラムします
Lesson 2 Counting Reps with Leo Leo と一緒に繰り返し数を数える （45 分）	• 腹筋マシンをプログラムして、繰り返し回数を数え、カウントダウンを完了する
Lesson 3 Dance Loop with Coach コーチとのダンスループ （45 分）	• for ループを使用してモデルをプログラムします • 4 つのプログラムをデバッグしてヒントとコツを学ぶ
Lesson 4 Setting Conditions for Yogas ヨガの条件設定 （45 ～ 90 分）	• センサーを使用して安全に移動するモデルをプログラムします • センサー使用時のモーター出力の影響を調べる
Lesson 5 Infinite Moves 無限の動き （45 ～ 90 分）	• 無限ループをプログラムします • ロボットが動くための条件となる力センサーを含むモデルを作成します。
Lesson 6 Leading the Team with Loops ループでチームをリードする （90 ～ 120 分）	• 繰り返しのためのモデルを設計します • ループを使用してモデルを動かすようにプログラムします

SAMPLE

Unit 5 Conditions for Games ユニット 5 ゲームのための条件	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Controlling Motion with Tilt 傾きで動きをコントロール （ 授業時間：45 分）	• モーションセンサーをプログラムします • 条件分を作成します
Lesson 2 Claw Machine クレーンゲーム （ 45 分）	• 基本的なループを作成します • 設定された条件に基づいてグラバー モデルをプログラムします
Lesson 3 Change Game Decisions ゲームの決定を変える （ 45 ~90分）	• 計画におけるフローチャートの使用方法を理解します • フローチャートを作成し、それに従ってプログラムを作成します
Lesson 4 Guess Which Color 色当てゲーム （ 45 分）	• 条件付きコードを使用してカラー センサーをプログラムします • ゲームを作ります
Lesson 5 Guessing Game 予想ゲーム （ 45 分）	• if/elif/else を使用して、複数の条件文を使用するコードを作成します • コードにループを追加します • 構文が正しくない / 欠落している、コードが欠落している、またはインデントが正しくないコーディングをデバッグします
Lesson 6 Score ! 得点! （ 45 分）	• 動きとライトマトリックスをプログラムします • 条件文の知識を応用します
Lesson 7 Game Time ゲームの時間です （ 90 分）	• ゲーム形式で満たさなければならない条件を含むコードを記述します • ロボットの応答を必要とする一連のイベントを必要とするゲームを作成します

Unit 6 Troubleshooting and Debugging ユニット 6 トラブルシュート&デバッグ	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Testing Prottypes プロトタイプのテスト （ 授業時間：45 分）	• アイデアをブレインストーミングし、問題の解決策を開発します • モデルをプログラムします
Lesson 2 Break Dancer Break Down ブレイクダンサー ブレイクダウン （ 45 分）	• 問題を特定して、プログラムをデバッグします
Lesson 3 Dance to the Beat ビートに合わせてダンス （ 45 分）	• 問題を特定して、プログラムをデバッグします
Lesson 4 Testing for troublr トラブルのテスト （ 90 分）	• モデルデザイン内のハードウェアの問題を特定して修復します
Lesson 5 Debug-inator デバッグネーター （ 45 分）	• ソフトウェアの問題をデバッグします

SAMPLE

Part 3

Unit 7 Functions ユニット 7 関数	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Turtle Trouble カメのトラブル （授業時間：45 分）	• カメの足ひれを動かすプログラムを書きます • プログラムを変更して、状況に応じて異なる反応やコード行を実行できるようにします
Lesson 2 Clean up with multiple functions 複数の関数でクリーンアップ （45 分）	• アイテムを拾うグラバーを組み立ててプログラムします • プログラムを変更して複数の関数を組み込みます
Lesson 3 Clean Indicator クリーンインディケーター （45 分）	• パラメーターを使う関数を作ります • デバッグ関数とパラメータを使用する関数を調査します
Lesson 4 Automate the Clean Up クリーンアップの自動化 （90 分）	• 資材がリサイクル可能かリサイクル不可能かを識別する仕分けロボットをプログラムします • 2 番目の関数をプログラムに組み込んでプログラムをより効率的にします
Lesson 5 Taking care of My Enviroment デバッグネーター （90 分）	• 地域を世話する環境ヘルパーを設計、構築、プログラムします

Unit 8 Compound Conditionals and Logic Operators ユニット 8 複合条件と論理演算子	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Passwoed Protection パスワードの保護 （授業時間：45 分）	• パスワードの設定を通じてサイバーセキュリティを調べます • 物理的なセキュリティ対策を検討します
Lesson 2 Make it Physicaly Safe 物理的に安全にする （45 分）	• ネストされた条件文を調べます • 物理的なセキュリティ対策を検討します
Lesson 3 Make a Safer Safe より安全な金庫を作る （45 ～ 90 分）	• 論理演算子を使用して条件を結合することを調べます • 物理的なセキュリティ対策を検討します
Lesson 4 Security Operating with Logic ロジックで運用するセキュリティ （45 ～ 90 分）	• センサーを使ったセキュリティを調べます • 2 段階のセキュリティプログラムを作成します
Lesson 5 Escape Room 脱出しろ! （90 分）	• 脱獄 ゲームをシミュレートするセキュリティ デバイスを作成します • 与えられた制約を満たすデバイスを設計します

SAMPLE

Unit 9 Data and Math Functions ユニット 9 データと数学関数	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Get Moving to Get Data データを取得するために動き出す （授業時間：45 分）	• センサーからデータを取り込む方法を検討します • カセンサーからのデータを利用する新しいプログラムを作成します
Lesson 2 Bike Riding for data データを求めて自転車に乗る （45 分）	• 一定の速度で前進するように自転車モデルをプログラムします • 数学関数を使用して自転車モデルの速度を増減するプログラムを作成します
Lesson 3 Counting Your Steps 歩数計 （45 分）	• 変数を使用して数学的計算をプログラムに統合します • 歩数を計測するプログラムを作成します
Lesson 4 Make it Move 動かしてみよう （45 分）	• 前進したり、方向を変えるようにドライビングベースをプログラムします • 数学関数を使用したプログラムを作成します
Lesson 5 Parking Lot 駐車場 （90 分）	• センサーとモーターを使用したプログラムで条件文を使用します • センサーを現実の問題に適用します
Lesson 6 My Transportation 私の交通手段 （90 分）	• 子どもたちを学校に運ぶための交通手段を設計、組み立て、プログラムします

Unit 10 List ユニット 10 リスト	
Lesson レッスン名	Objectives レッスンの目的
Lesson 1 Listing Letter リストレター （授業時間：45 分）	• リストの作成と活用 • リストを使用した複合条件を含むコード
Lesson 2 Stretch Your Muscles and Lists 筋肉とリストをストレッチする （90 分）	• モーション センサーの値をリスト内の変数で使用するプログラムを作成します • リストを使用してヨガのルーチンを作成します
Lesson 3 Mind Games マインドゲーム （90 分）	• 一つのプログラムの中で2つのリストを作ります • プログラムの中で2つのリストを比較します
Lesson 4 Jumping for List リストへジャンプ （45 ～ 90 分）	• カセンサーと距離センサーからデータを作成してリストで使用します • ジャンプのトライから収集したデータ（ジャンプの高さなど）に基づいてリストをプログラムします
Lesson 5 Word Game With Lists リストを使った単語ゲーム （90 分）	• ワードゲームを完成させるために複数のリストをプログラム内で作ります • 単語ゲームに合わせて色感知モデルをプログラムします

SAMPLE

Unit 4 Doing Reps with Loop

ユニット 4 ループを使った繰り返し処理

Unit Introduction: ユニット紹介

このユニットでは、生徒が体を動かしたりストレッチしたりする方法を調べながら、テキストベースのコーディング言語である Python の基本的なコンピュータサイエンスの原理とプログラミングの概念を探究します。生徒たちは、ストレッチやエクササイズの動きを模倣した繰り返しパターンでモデルを動かすことができるように、さまざまなタイプのループをプログラムする方法を調べます。

レッスンは、生徒が以下の分野のスキルと知識を上達させることができるようにデザインされています：

- for ループ、while ループ、無限ループのプログラミング
- ループにおける変数と条件の設定を探究する
- ループを使用したときのロボットの動きを調べる
- 既存のコードを使用して修正し、新しいアイデアを探求する
- アルゴリズムの作成をサポートする擬似コードを作成する
- コードのコメント機能を利用して、プログラムの一部を文書化する
- 問題を定義して分解する

ユニット学習の目標

このユニットでは、ループのプログラミング方法を探究し、モデルを動かしてストレッチさせるために各タイプのループをいつ使用するかについて理解を深めます。また、ループ内の変数や条件の設定など、ループを効果的に活用する方法を体験します。アルゴリズムの作成をサポートする擬似コードや、プログラムを文書化するためのコードコメントを活用します。

探求への質問：

- 繰り返しパターンはプログラミングでどのように使えるか？
- プログラム作成におけるループのさまざまな使い方とは？

SAMPLE

ユニット 4 レッスン 1
レオとウォームアップループ

レオとウォームアップループ

生徒はループについて学び、ループを使ってプログラミングを行います。

主な目的

生徒は次のことを行います：

- ループを使ってプログラムを作成する。
- 腹筋マシンを組み立てプログラムする。

探求のための質問

効率的なプログラムを作成しながら、いかにコードの行を繰り返すことができるか？

必要な機材

* SPIKE Prime セット
* SPIKE アプリがインストールされたデバイス
* ノート

準備する

- SPIKE Prime ハブが充電されていることを確認してください (特に Bluetooth 経由で接続している場合)

1. 引きつける

生徒を立たせ、椅子、机、または部屋の周りを正方形に歩かせます。歩きながら、個々の動きについて生徒に説明させます (一歩前へ、一歩前へ、右 / 左へ曲がるなど)。

自分の動きをプログラムにすることについて、生徒と話し合います。この段階的なプログラムを作成するために必要なコードの行数を生徒に考えさせます。疑似コードとは、プログラムに何をさせたいかを言葉で書くことです。

生徒たちに、これだけの行数のコードを入力することにわくわくするかどうかを尋ねます。プログラムをより効率的にするための代替案つまり、ある動作やタスクを実行するために使用するコード行数がより少ない、より短いプログラム）について話し合います。

必要に応じて生徒に、ステップの中で繰り返されるパターンを認識するように促します。

生徒にループを紹介します。ループを使うと、コードの行を繰り返すことができ、プログラムが短くなります。これはプログラマーの時間を節約し、コンピュータがプログラムを実行しやすいコードにします。

2. 探究する

生徒は、トレーナーモデルであるレオを使用して、for ループを探求します。

SPIKE アプリの「 組み立て図」セクションに生徒を誘導します。ここで生徒は、トレーナーであるレオの構築手順にアクセスできます。生徒にモデルを構築させます。

https://education.lego.com/ja-us/support/spike-prime/building-instructions にも作り方の説明があります。

Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒に、すでにプログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

SAMPLE

活動を開始

Leo が腹筋を 1 回完了するようにプログラミングするよう生徒に指示します。

生徒に、レオが腹筋を 1 回完了するようにプログラムするよう求めます。メモやナレッジベースを参考にして、生徒が自分でプログラムを作成するように促します。生徒が苦戦している場合は、必要に応じてサンプルプログラムを提供します。
サンプルプログラム

```
from hub import port
import runloop
import motor_pair

async def main():
    #ポートB、ポートFのモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    #モーターを一方向に動かした後、逆方向に動かす
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500)
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=-500)

runloop.run(main())
```

生徒とプログラムについて話し合います。

レオにとって腹筋は 1 回で十分な運動量だと思うかどうかを生徒に尋ねます。少なくとも 5 回の腹筋に挑戦するよう生徒に促します。同じ動作を数回繰り返すようにプログラムを変更する方法について話し合います。

生徒たちは、コードを 5 回コピーして貼り付けることを提案するかもしれません。これは一つの方法ですが、あまり効率的ではありません。必要に応じて効率について生徒と話し合います。

生徒に、どうすれば 1 行のコードを追加するだけでその動作を行うことができるかを考えさせます。このコードを生徒と共有し、生徒が 1 行の新しいコードを特定できるかどうかを確認します。生徒にこのコード行を追加してプログラムを実行するように指示します。

```
from hub import port
import runloop
import motor_pair

async def main():
    #ポートB、ポートFのモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(500)
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=-500)
        await runloop.sleep_ms(500)

runloop.run(main())
```

生徒とプログラムについて話し合います。

3. 説明する

プログラムがどのように動作したか、生徒と話し合います。
生徒に次のような質問をします：
・新しいプログラムはどのように動きましたか？
・for index in range“ 行はプログラムに何を教えていますか？
・効率とは何か？コードをコピー・アンド・ペーストする代わりに、for ループを使うことは、どのように効率的ですか？

生徒たちに、これは「 for ループ」または繰り返し文の例であり、コードの一部を決められた回数繰り返すことができることを説明します。この場合、括弧で示したようにコードを 5 回繰り返しています。for ループに続くコード行がインデントされていることに注目してください。for ループの後にインデントされたコード行はすべてループされます。

4. さらに詳しく説明する

生徒に、レオが腹筋をした後（ ループの後）にすることをプログラムに追加するよう挑戦させます。

生徒に、画像や音を入れたり、コンソールに何かを表示したりするコードを追加するように指示します。レオは少なくとも 5 回の腹筋をしなければなりませんが、その後に何か新しいことをしなければなりません。ループの中で繰り返されたくないものは、インデントされないことを生徒に思い出させます。

注意：生徒はモーターのパワーレベルを変更することができます。パワーレベルを変えると、レオの動き方が変わります。時間の許す限り、このことを一緒に調べることを検討してください。

生徒が最終的なプログラムを共有できるようにし、どのようにコードを追加できたかを話し合います。特に、生徒がインデントを使うところと使わないところに注意します。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。
生徒に次のような質問をします：
・プログラムをより効率的にするために、for ループをどのように使うことができますか？（ コードを短くする）
・ループが終了したことをプログラムでどのように示しますか？

自己評価

生徒にノートで次のことに答えさせます：
・プログラムの中で for ループを使うことについて、今日何を学びましたか？
・プログラムで for ループを使用することについて、今日学んだことは何ですか？

SAMPLE

ユニット 4 レッスン 2

Leo と一緒に繰り返し数を数える

主な目標
生徒は次のことを行います：
・腹筋マシンをプログラムして、回数を数え、カウントダウンを完了する。

ボキャブラリー
カウンタ変数

Leo と一緒に繰り返し数を数える

生徒は、for ループを適用し、count 変数の使い方を学びます。

探求のための質問

- ロボットが関数を実行する回数を数えるにはどうすればよいですか？

必要な教材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備

・SPIKE Prime ハブが充電されていることを確認してください（特に Bluetooth で接続する場合）。
・レオを使ったウォームアップループのレッスンで使用したトレーナーモデル、レオを生徒が作ったことを確認します。

1. 引きつける

生徒に 1 分間軽い運動をさせます。ジャンピングジャックやつま先タッチなど、1 分間に何度も動作を繰り返すことができるものであれば何でもよい。何回やったかを数えさせることはしない。

1 分間が終了したら、その運動を何回できたかを生徒に報告させる。数えるように伝えていないので、ほとんどの生徒は分からないでしょう。
各自がどの程度のエクササイズを行ったか分かれば、どのような情報を共有できるかについて、短い話し合いをします。例えば、繰り返した回数の合計、男子と女子の比較、グラフの作成、最高から最低までの範囲などが考えられます。

2. 探求する

生徒は、for ループでカウント変数を学習します。

生徒と、レオがウォームアップトレーニングを行うときに繰り返し回数をカウントする方法を話し合います。生徒には、「レオとのウォームアップループ」レッスンのサンプルプログラムを示して、話し合いをサポートします。

SAMPLE

サンプルプログラム

```
1  from hub import port
2  import runloop
3  import motor_pair
4
5  async def main():
6      #ポートB、ポートFのモーターをペアにする
7      motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)
8
9      for index in range(5):
10         await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500)
11         await runloop.sleep_ms(500)
12         await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=-500)
13         await runloop.sleep_ms(500)
14
15 runloop.run(main())
16
```

生徒は、「for index in range」に含まれる「5」という数字が、レオが完了する腹筋の合計回数を示していることに気づくかもしれません。生徒に、どのように数を数えることができるか、また、どこで数が表示されるかについて考えさせます。

「for count in range」行の「index」という単語が**変数**であることを生徒に説明します。生徒たちは、どんな単語でも、あるいはただの文字でも、変数に割り当てることができることを理解します。この変数は、コードがループする回数を数えます。最初のループは 0、次に 1、という具合に。ループは括弧内の数字に達すると止まります。つまり、インデックスは 0,1,2,3,4 と数えることになります。これでループは 5 回になることを説明します。同じ名前を一貫して使う必要があります。

index という変数に対する print 関数を含むこの新しいプログラムを生徒に見せます。この新しいコードを追加することで、生徒は Leo と一緒にカウントを追うことができ、腹筋が何回完了したかを知ることができます。

```
1  from hub import port
2  import runloop
3  import motor_pair
4
5  async def main():
6      #ポートB、ポートFのモーターをペアにする
7      motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)
8
9      for index in range(5):
10         await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500)
11         await runloop.sleep_ms(500)
12
13         await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=-500)
14         await runloop.sleep_ms(500)
15
16         #0から始まるindexに+1づつ加算してコンソールに表示
17         print(index+1)
18
19     print('Whew! That was tough.')
20
21 runloop.run(main())
22
```

SAMPLE

レオと一緒にウォームアップループのプログラムを保存している場合は、これらの新しいコードをプログラムに追加するように生徒に指示します。そうでない場合は、生徒がこのコードを入力する必要があります。生徒がコンソールを開いてカウントを見るようにします。

3. 説明する

生徒とプログラムについて話し合います。レオが完了した腹筋の回数を表示するために、print 関数がどのように動作しているかを確認します。
生徒に次のような質問をします：
・プログラムはどのように動作しましたか？
・カウンター変数とは何ですか？
・print(count+1) という行の目的は何ですか？
・最後の行「Whew ！ That was tough!」がコンソールに 5 回表示されないのはなぜですか？

生徒がカウンタ変数に慣れ、どんな量でも変数に代入できることを理解できるようにします。カウンタ変数は、プログラムのループ回数を代入したり、値を表示するために print 関数で使用したりするなど、コードの中でさまざまな方法で使用できます。

4. もっと詳しく

レオのためにカウントダウンを作成することに生徒を挑戦させます。

運動するとき、カウントアップする代わりに 1 までカウントダウンすることがあることを生徒に説明します。生徒には、5 から 1 まで数えるようにコードを変更し、最後に楽しい音を出して終了を示すようにします。

サンプルプログラム

```
from hub import port
import runloop
import motor_pair

async def main():
    #ポートB、ポートFのモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    for index in range(5):
        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(500)

        await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=-500)
        await runloop.sleep_ms(500)

        #0から始まるindexに+1づつ加算してコンソールに表示
        print(index+1)

    print('Whew! That was tough.')

runloop.run(main())
```

生徒がプログラムでカウンタ変数を使用する方法を探求する時間を追加します。繰り返しの後にやる気を起こさせるようなことを加えたり、5 つずつ数えたりする方法について、生徒に考えを促します。
生徒に最終的なプログラムを発表させます。カウンタ変数のさまざまな使い方について話し合います。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。
生徒に次のような質問をします：
・カウンタ変数を使うには、どのような方法がありますか？
・ループはどのようにプログラムに組み込むことができますか？
・ループが完了した後、プログラムはどのように継続できますか？
・プログラムの中で for ループを使うには、どのような方法がありますか？

自己評価

ノートに次のように答えさせる：
・カウンタ変数の使用について、今日何を学びましたか？

SAMPLE

ユニット 4 レッスン 3
コーチとのダンスループ

コーチとのダンスループ

ループのプログラミングを練習します。

探求のための質問

- プログラムをより効率的にする方法は?

必要な教材

- SPIKE プライム セット
- SPIKE アプリがインストールされているデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認してください (特に Bluetooth で接続する場合)。

主な目的

生徒は次のことを行います：

- for ループを使用してモデルをプログラミングする。
- 4 つのプログラムをデバッグして、ヒントとコツを学ぶ。

1. 引きつける

生徒に足を揃えて立ってもらいます。右足を上げ、体を少し右に動かし、右足を下げます。
次に左足を上げ、体を少し左に動かし、左足を下ろします。クラス全員で一緒になって、それぞれの方向に 5 回ずつ動きます。
先ほどの動作を、for ループを使って擬似コードを書かせます。

2. 探究する

生徒は、for ループを使って動くようにコーチモデルをプログラミングすることを探求します。
https://education.lego.com/en-us/support/spike-prime/building-instructions に記載されている作り方を使用して、コーチモデルを作るように生徒に指示します。
Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒に、すでにプログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

やってみよう

生徒に、コーチモデルがどのように作られているかを調べさせ、このモデルがどのようにプログラムされ、私たちのためにワークアウトやダンス (前後に動く) をリードできるかを話し合います。
生徒にこのサンプルプログラムを渡して、コーチモデルを動かしてエクササイズをリードさせる方法を話し合います。生徒にプログラムを説明させ、プログラムをより効率的にするための提案をさせます。

SAMPLE

サンプルプログラム

```
from hub import port
import runloop
import motor

async def main():
    #速度300で1/4秒（250ミリ秒）動かす
    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, -300)
    await motor.run_for_time(port.F, 250, -300)

    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, -300)
    await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

数行の繰り返しからループを含むコードに変更するよう、必要に応じて生徒に促します。生徒は、ループを何回繰り返すかを決める必要があります。

このワークアウトでは、コーチは片方の足（モーター）を動かし、次にもう片方のモーターを動かすことを望んでいることを生徒に伝えます。生徒には、２つのモーターを区別するために、それぞれのモーターに具体的な名前をつけるよう、配慮する必要があることを思い出させてください。変数に前もって名前をつけておくことは、プログラムを通してその情報を簡単に再利用するための重要なステップです。サンプルプログラムにあるように、それぞれのモーターが接続されているポートを使うなど、モーターの変数に名前を付けるアイデアを話し合ってください。

コーチが動けるようにプログラムを修正し、テストする時間を生徒に設けます。手助けが必要な場合は、ナレッジベースとノートを活用してください。また、エラーメッセージが表示されないかコンソールを見るように注意してください。

必要であれば、このサンプルプログラムを生徒に提供してください。

```
from hub import port
import runloop
import motor

async def main():

    for index in range(10):
        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

生徒にプログラムを探求させ、さまざまな値を変えて、コーチモデルがどのように動くかを確認させます。

SAMPLE

3. 説明する

プログラムがどのように機能したか、生徒と話し合います。

生徒に次のような質問をします：

- どのようにしてプログラムをより効率的にしましたか？
- コードの「 in range」の部分は、モデルに何をするように言っていますか？
- プログラムがカウントを終了すると、何が起こりますか？

4. もっと詳しく探究する

生徒にループプログラムのバグを認識させる。
各プログラムとエラーメッセージを生徒に見せる。バグを修正するために各コード行のどこを変更する必要があるかを生徒に話し合わせる。クラスで変更を行い、各変更後にプログラムが正しく実行されることを確認する。

デバッグ活動 1:

ループコードのバグを認識する。

```
1  from hub import port
2  import runloop
3  import motor
4
5  async def main():
6
7      for index in range()
8          await motor.run_for_time(port.B, 250, 300)
9          await motor.run_for_time(port.F, 250, 300)
10
11         await motor.run_for_time(port.B, 250, -300)
12         await motor.run_for_time(port.F, 250, -300)
13
14 runloop.run(main())
15
```

Traceback (most recent call last):
File“ dance loop with coach” , line 14, in <module>
File“ dance loop with coach” , line 7, in main
TypeError: function missing 1 required positional arguments

生徒は、エラーが 7 行目にあることに気づくでしょう。ループ関数には、ループを実行させる回数である引数という必須項目が欠けています。
生徒に、含まれるべき数を特定するように指示します。どの数字でもかまいませんが、生徒はコメントをヒントとして使うことができます。

SAMPLE

デバッグ活動 2:

モーター変数のバグを認識する。

```
1 from hub import port
2 import runloop
3 import motor
4
5 async def main():
6
7     for index in range(10)
8         await motor.run_for_time( 250, 300)
9         await motor.run_for_time(port.F, 250, 300)
10
11        await motor.run_for_time(port.B, 250, -300)
12        await motor.run_for_time(port.F, 250, -300)
13
14 runloop.run(main())
15
```

Traceback (most recent call last):

File“ dance loop with coach” , line 14, in <module>

File“ dance loop with coach” , line 8, in main

TypeError: function missing 1 required positional arguments

生徒は、エラーが８行目を指しており、引数が１つ足りないと言っていることに気づくでしょう。モーターポートが括弧の中にないことに。

デバッグ活動 3：

プログラムの構造上のバグを認識する。

```
1 from hub import port
2 import runloop
3 import motor
4
5 async def main():
6
7     for index in range(10)
8
9     await motor.run_for_time( 250, 300)
10    await motor.run_for_time(port.F, 250, 300)
11
12        await motor.run_for_time(port.B, 250, -300)
13        await motor.run_for_time(port.F, 250, -300)
14
15 runloop.run(main())
```

Traceback (most recent call last):

File“ Project 2” , line 8

SyntaxError: invalid syntax

生徒は９行目にエラーがあることを気づくでしょう。無効な構文は、プログラムが変換のために与えられたコマンドを認識できないことを意味します。for ループのコード行を与えたが、次のコード行をインデントしなかったため、ループに何をすべきか指示していないことを生徒に認識させます。ループに含めたいすべての行をインデントする必要があります。

生徒は、10 行目も同様にインデントする必要があることを認識しなければなりません。なぜ 10 行目にエラーメッセージが出なかったのか、生徒に考えさせましょう。エラーが出るのは、問題が発生したコードの最初の行だけです。これは、別のエラーを修正している間に、１つのエラーを見逃してしまうことを意味することがあることを生徒に説明します。

SAMPLE

デバッグ活動 4：

起こりうるもう一つの問題は、プログラムがエラーと認識しないエラーを発生させることです。

```
1  from hub import port
2  import runloop
3  import motor
4
5  async def main():
6
7      for index in range(10)
8
9          await motor.run_for_time(port.B 250, 300)
10     await motor.run_for_time(port.F, 250, 300)
11
12         await motor.run_for_time(port.B, 250, -300)
13         await motor.run_for_time(port.F, 250, -300)
14
15 runloop.run(main())
```

デバッグ活動 3 のコードで、9 行目をインデントしたもの、つまり前のエラーの修正箇所を生徒に見せてください。しかし、10 行目をインデントしないとどうなるでしょう。生徒に、新たに発生するエラーを予測させましょう。

Traceback (most recent call last):
File“ dance loop with coach” , line 11
IndentationError: unexpected indent

エラーメッセージを生徒に見せましょう。生徒たちは 12 行目のインデントをしなかったことによるエラーを予想したかもしれませんが、実際のエラーは 9 行目で発生しました。生徒に、プログラムが何をしていたかを考えさせます。プログラムは 7 行目のループを読み、それが 9 行目のループへのアクションにつながった（インデントで示されたつながり）ことを説明します。10 行目がインデントされていないとき、プログラムはループが終了したことを読み取りました。しかし、12 行目にインデントがあると、プログラムは混乱します。インデントを必要とするものは何もなかったので、プログラムはエラーを読み取りました。

さまざまなエラーメッセージとトラブルシューティングの方法について、クラス全員で話し合う。

5. 評価する

教師の観察

生徒とプログラムについて話し合う。
生徒に次のような質問をする：

- プログラムをより効率的にする方法にはどのようなものがあるか？
- ループはどのように使えますか？
- エラーメッセージを受け取ったとき、どのような行動をとるべきか。
- なぜエンジニアはロボットに反復運動をさせるのか？

自己評価

生徒にノートで次のことを書かせる：

- エラーのないループを使ったプログラムの作成について、今日何を学びましたか。
- 今日、どのようなデバッグのコツを学びましたか。

SAMPLE

ユニット 4 レッスン 4
ヨガの条件設定

ヨガの条件設定

生徒は while ループの使い方について調べます。

主な目的

生徒は以下のことを行います：

- while 文について調べる。
- while ループを使用してモデルをプログラミングする。

探究のための質問

- ループはどのように特定の条件が満たされたときだけ使えるか？
- ループは特定の条件が満たされたときだけ使用できるのか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（ 特に Bluetooth で接続する場合）
- 生徒がコーチとのダンスループのレッスンで使用したコーチモデルを組み立てていることを確認する。

1. 引きつける

条件文がどのように機能するかを生徒に考えさせます。
文がどのように機能するかを生徒に考えさせます。

生徒と教師の発言ゲームを行います。
ゲームのルール

- 生徒は先生の指示する動作に従わなければならない。
- 例えば、「 先生がジャンプと言う」と言ったら、 生徒はジャンプする。
- もし、 あなたがただ「 ジャンプ」と言っただけなら、 生徒はジャンプしないでしょう。

生徒と一緒に発言ゲームを数ラウンド行います。

いつその行為を行うかを決める条件について、生徒と話し合う。ゲームでは、アクションをする条件は「先生が言う」でした。生徒たちは、その条件が成立している間だけ、その行為を行うことになります。例えば、 条件法を少し複雑にするには、 次のようにします。

- 靴を履いていたらジャンプしなさい。
- 先生が靴を履いていたらジャンプしなさいと言う。
- 先生が T シャツを着ているなら、 腕を上げなさいと言う。

設定された条件に基づいて何かをする場合について、 他の例を話し合いをする（ 例としては、 天候によって服装が決まったり、 夕食後のデザートしか食べられないと親に言われたり、など）。その条件が成立している間だけ、どのような行動をとるかを話し合う。例えば、雨が降っているときだけ傘をさす。 雨がやんだら、 普通は傘をしまう。 条件が成立しなくなったら、 その行動を止める。

SAMPLE

2. 探究する

生徒は、while ループを使用して動くコーチモデルのプログラミングをします。
Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。生徒に、すでにプログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

指示に従う

コーチが指示したときだけ動くように、私たちが指示に従うようにしたいことを生徒と話し合います。コーチがヨガのトレーニングに設定した条件は、5 より小さい数です。その条件が真のときだけモデルが動くようにすることを生徒に説明します。

生徒に、モーターを動かしてヨガのポーズを作るサンプルプログラムを教えます。生徒に、このプログラムをプログラミングキャンバスに入力するように指示し、プログラムを実行させます。

```
1  from hub import port
2  import runloop
3  import motor
4
5  async def main():
6
7      count = 2
8      #変数countが5より小さいときモーターを動かす
9      while count < 5:
10         await motor.run_for_time(port.B 250, 300)
11         await motor.run_for_time(port.F, 250, 300)
12
13         await motor.run_for_time(port.B, 250, -300)
14         await motor.run_for_time(port.F, 250, -300)
15
16 runloop.run(main())
```

生徒にプログラムを実行し、探求する時間を与えます。サンプルプログラムでは、例として count を 2 としています。count 変数の目的は数値を代入することなので、どんな数値でも含めることができます。生徒に、count 変数に新しい数字や値を入れるように促します（ count=10、count=2 など）。
count はどんな値でも設定できる変数であることを生徒に指摘する。ここでは、練習を数えるために、この変数に count という名前をつけています。しかし、後で同じ名前を使って参照するのであれば、生徒は変数にどんな名前をつけても良いでしょう。

3. 説明する

プログラムがどのように動いたか、生徒と話し合います。
生徒に次のような質問をします：

- このプログラムでは何が新しくなりましたか？
- count 変数に代入された値を変更するとどうなりますか？5 より高いと？5 より低いと？
- プログラムの中で「 while」は何をしていますか？
- プログラムはいつ停止しますか？
- これはループの例としてどうですか？

このプログラムで設定されている条件は、count 変数に 5 より小さい数値が代入されているときだけモーターが動くことであることを生徒に説明します。もし count 変数が 5 以上に変わったら、モデルは動きません。while はプログラムに条件文を作り、設定された条件が真である場合にのみ動作するように指示している。

最後の問題に答えるために、これは条件文の良い例であるが、実際にはループではないことを生徒に説明する。このプログラムでは、今すぐ動作を繰り返すように指示するものは何もありません。

SAMPLE

4. もっと詳しく探究する

このプログラムを while ループにすることに生徒を挑戦させる。

このプログラムを while ループに変更する方法について生徒と話し合います。生徒にこのサンプルプログラムを与えます。追加されたステップと、これがどのようにループを作るかを話し合う。

生徒にプログラムを実行させます。

```
1  from hub import port
2  import runloop
3  import motor
4
5  async def main():
6
7      count = 2
8      #変数countが5より小さいときモーターを動かす
9      while count < 5:
10         await motor.run_for_time(port.B 250, 300)
11         await motor.run_for_time(port.F, 250, 300)
12
13         await motor.run_for_time(port.B, 250, -300)
14         await motor.run_for_time(port.F, 250, -300)
15
16         count+=1
17
18 runloop.run(main())
```

生徒にプログラムを探求する時間を与えます。生徒は、count 変数の値を変えてみて、プログラムがどのように動作するかを確認します。話し合いのために、説明のセクションと同じ問題を再度検討します。

生徒がプログラムを変更して、while count に異なる値を設定できるようにします。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。

生徒に次のような質問をします：

- プログラムの最後に count+1 を追加したとき、何が起こりましたか？
- while ループはどのように働きますか？
- プログラムの中で while ループを使うのは、どんなときに役に立ちますか？

自己評価

ノートに次のことを書かせる：

- while ループの使い方について、今日何を学びましたか？

SAMPLE

ユニット 4 レッスン 5
無限の動き

無限の動き

生徒は while ループを使って調べます。

主な目的

生徒は次のことを行います：

- 無限ループをプログラムする。
- ロボットが動くための条件となるフォースセンサーを含むモデルを作成する。

探究のための質問

- 条件が常に真になるようなループを作るにはどうすればよいか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）
- 生徒が Coach を使ったダンスループのレッスンで使用した Coach モデルを組み立てていることを確認する。

1. 引きつける

生徒を条件設定についての会話に引きつけます。
生徒に、特定の条件を満たす必要がある場合の例を話し合いで挙げるよう促します。簡単な例としては、お腹が空いたときだけ食事をすると決めたり、誰かに言われたときだけ部屋を掃除したりすることなどがあります。生徒に、それぞれの場合の条件を挙げさせます。

次に、ある条件が常に真である場合の例を見つけるよう生徒に指示します。生徒たちは、真であることを止めない（条件に偽がない）条件を設定できるアイデアを話し合いで出し合います。例えば、毎日太陽が昇ることなどが考えられます。太陽が昇ることを条件とする文を書くように生徒に促します。例：太陽が昇れば、外は明るくなる。

太陽は毎日昇るので、これは毎日正しい。
常に真である条件を持つループは無限ループであることを生徒に説明します。無限ループとは、決して偽にならない while ループのことである。

2. 探究する

生徒は、無限ループを使ってコーチモデルを動かすプログラミングを探求します。

Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。生徒に、すでにプログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

コーチは私たちを長いハイキングに連れて行きたいと言っています。生徒と、移動の条件を設定する while ループを使用して、コーチモデルを移動させるループを含める方法について話し合います。

生徒に、ヨガの条件設定のレッスンで作ったプログラムを見直すように指示します。生徒に、while ループが常に真で、終了しないように条件を設定する方法を話し合います。

SAMPLE

ヨガの条件設定のサンプルプログラム

```
from hub import port
import runloop
import motor

async def main():

    count = 2
    #変数countが5より小さいときモーターを動かす
    while count < 5:
        await motor.run_for_time(port.B 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

条件が常に真であることを確認するために、生徒にいくつかの異なるアイデアを試す時間を与える。解決策の例としては、カウントを 1 に設定することや、while カウントを 0 より大きく設定することなどがあります。

調査している間、コンソールにエラーが表示されないか注意するよう、生徒に思い出させる。

While True

常に真である（偽にならない）ループの条件を作成するという目標を達成するための、新しいコード行を生徒に紹介します。

このコード行を生徒と共有し、今作成したプログラムのどこに位置すると思うか（何を置き換えるべきか）を尋ねます。

生徒に、プログラムを無限にループさせる while True 行を含むようにプログラムを変更するよう指示します。

サンプルプログラム

```
from hub import port
import runloop
import motor

async def main():

    while True:
        await motor.run_for_time(port.B 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

3. 説明する

プログラムがどのように機能したかを生徒と話し合います。

生徒に次のような質問をします：

- この課題の何が難しかったか？
- このループは while ループとどう違うか？
- どのような場合に無限ループを使いたくなりますか？

SAMPLE

4. もっと詳しく探究する

フォースセンサーをコーチのモデルに追加します。

生徒には、ポート D にセンサーを接続して、フォースセンサーをモデルに追加するように指示します。

センサーを取り付けた状態で、ループ内でセンサーを使用するよう、生徒たちにプログラムを更新してもらいます。ループは常に真であり続けますが、フォースセンサーの使用も含める必要があります。

```
from hub import port
import runloop
import motor
import force_sensor

def is_pressed():
    return force_sensor.pressed(port.D)

async def main():

    while True:
        #力センサーを押すと動き出す
        await runloop.until(is_pressed)

        await motor.run_for_time(port.B 250, 300)
        await motor.run_for_time(port.F, 250, 300)

        await motor.run_for_time(port.B, 250, -300)
        await motor.run_for_time(port.F, 250, -300)

runloop.run(main())
```

生徒にプログラムをテストしたり修正したりする時間を与えます。エラーメッセージが表示されないかコンソールを見るように生徒に注意を促します。

生徒がプログラムを共有できるようにします。異なるアプローチについてグループで話し合います。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。

生徒に次のような質問をします：

- 何が while ループを無限ループに変えますか？
- 無限ループを作成する際に最も重要なことは何ですか？
- ループを使用する際に、センサーをどのように含めることができますか？

自己評価

生徒にノートで次のことを書かせる：

- 無限ループの使用について今日学んだことは何ですか？

SAMPLE

ユニット 4 レッスン 6
ループでチームをリードする

ループでチームをリードする

生徒は、ループの知識を応用して、自分自身のモデルを作成し、プログラムします。自分のモデルを作成し、プログラミングします。

主な目的

生徒は次のことを行います：

- 反復モデルを設計する。
- ループを使用して動くようにモデルをプログラムする。

探究のための質問

プログラムで使用するループの最適なタイプを、どのように判断すればよいのでしょうか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）

1. 引きつける

トレーナーのレオとコーチは、最後のエクササイズを指導するために、チームをあなたに引き継ぐ準備ができています。チームのクールダウンに適したさまざまな種類のエクササイズ、ストレッチ、ヨガのポーズを話し合います。

クラスで表を作り、動きの種類別にアイデアを整理します。このような表を考えてみましょう：

体の部位	動きの例
腕	腕立て伏せ
脚	スクワット、戦士のポーズ
体幹・腹筋	プランク

2. 探究する

生徒は、運動を指導するための新しいトレーナーを設計し、製作します。生徒が上の表を使って、エクササイズの動きを完成させるために設計・製作する必要があるものについて考えるよう促します。

エクササイズを完了するために必要な動きについて生徒に考えさせます。生徒に、設計におけるモーターの使用について考えるために、適切な動き方を調べる時間を与えます。生徒は設計をスケッチします。

動作が確実に機能するように、設計を何度かテストするよう生徒に思い出させます。生徒は、設計の最も重要な部分に時間を費やすために、脚を動かすだけなら人全体を設計する必要があるなど、設計におけるトレードオフを考慮する必要があります。必要に応じて、インスピレーションを得るために SPIKE アプリの BUILD セクションを生徒に見せることを検討します。

この課題の要件

- モデルがエクササイズのシミュレーションを完了できること。
- モデルには、少なくとも 1 つの運動用モーターが含まれていること。

SAMPLE

- プログラムにはループの一種が含まれていること。
- コードを書く前に、生徒は設計のスケッチを示し、コードに何を求めるかの説明を書かなければなりません。
- 生徒は、Python コードのコメント機能を使って、コード行の意味を説明します。

生徒に新しいトレーナーを設計し、プログラミングする時間を与えます。生徒には、モデルが期待通りに動くことを確認するために、プログラムを何度かテストし、必要に応じて変更するよう注意してください。

3. 説明する

生徒は、自分の設計を共有し、それがどのように機能するかを説明します。生徒と最初の共有セッションを行います。
生徒に次のような質問をします：

- 選択した運動を完了するために、モデルをどのようにプログラムしましたか？
- 生徒にプログラムのコメントを共有し、説明するよう求めます。
- デザインを作成する際に、どのような決定をしなければなりませんでしたか？
- どのようなタイプのループを選びましたか？なぜこのタイプで、他のタイプではないのですか？
- デバッグやトラブルシューティングが必要だった部分はどこですか？
- この課題で難しかったことは何ですか？

4. もっと詳しく探究する

最初の共有セッションの後、生徒たちにプログラムを完成させるための追加時間を与えます。
生徒はデザインとプログラムを完成させます。共有セッションで得た新しいアイデアを取り入れるように生徒に促します。
フィードバックに関するレッスンを次に行う場合は、モデルを一緒に残しておきます。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。生徒に次のような質問をします：

- この課題の何が難しかったですか？
- この課題を解決するためにどのようなアプローチをとりましたか？
- どのようなループを入れましたか？

自己評価

生徒にノートに次のように答えさせます：

- 特定のタスクを実行するモデルの設計について、今日何を学びましたか？

SAMPLE

Unit 5 Playing Games with simple Conditions

ユニット 5 簡単な条件でゲームをする

Unit Introduction: ユニット紹介

このユニットでは、さまざまなゲームをプレイする中で、テキストベースのコーディング言語である Python の本質的なコンピュータサイエンスの原理とプログラミングの概念を探求します。生徒たちは、ある条件が満たされたときにモデルがどのように動くかを設定するために、ゲームを作成するときにどのように計画し、条件文をプログラムするかを調べます。

レッスンは、以下の分野のスキルと知識が上達するような順序でデザインされています：

- 条件文を使ってプログラムを作成する
- if/else、if/elif/else、while true 文を使用したプログラムの作成を探求する
- 条件文とセンサーの使い方を調べる
- 新しいアイデアを探求するために、既存のコードを使用し、修正する
- 条件文の作成をサポートするフローチャートを作成する
- プログラムの一部を文書化するコードコメント機能を活用する
- 問題の定義と分解

ユニット学習の目標

このユニットでは、ループのプログラミング方法を探求し、さまざまなゲームを作成しながら、それぞれのタイプのループをいつ使用するかについて理解を深めます。ループの中で変数や条件を設定することを含め、ループを効果的に利用する方法を体験します。生徒は、アルゴリズムの作成をサポートする擬似コードや、プログラムを文書化するためのコードコメントを活用します。

探求のための質問

- センサーがどのように反応するかについて、どのように条件を設定できますか？
- プログラムで複数の条件を設定するにはどうすればよいですか？

SAMPLE

ユニット 5 レッスン 1

傾きで動きを制御する

傾きで動きを制御する

生モーションセンサーを使って、Python で条件文を書くことを探求します。

探究のための質問

- センサーがどのように反応するかの条件をどのように設定できますか？
- モーションセンサーはどのように動作しますか？

必要な機材

- SPIKE プライム
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）。

主な目的

生徒は次のことを行います：

- モーションセンサーをプログラムする。
- 条件文を作成する。

ボキャブラリー

条件文

オリエンテーション

1. 引きつける

生徒を条件文の体験に引きつけます。

生徒には、セットから以下のパーツを取り出させます：

- マゼンタ色のフレーム
- 3 種類の黄色のパーツ
- 3 種類の赤のパーツ

生徒に、マゼンタ色の枠を紙の中央に置き、他のパーツは枠の外側に置くよう指示します。

生徒は、真（条件を満たす）のパーツを枠の中に、偽（条件を満たさない）のパーツを枠の外に置くことによって、どのパーツが与えられた条件を満たすかを識別します。

生徒と練習します。条件が黄色であることを生徒に伝えます。どのパーツが条件を満たすかを生徒と話し合い、それらのパーツを枠の中に入れるように指示します。それ以外のパーツは枠の外側に偽として残します。

ラウンドの間に枠を空にします。

ラウンド 1: 条件が赤である。

ラウンド 2: 条件は 2x2 のブロックより大きいこと（生徒は比較する必要があるかもしれません）。

ラウンド 3: ブロックの形以外のもの（車軸やテクニックピースなど）が条件。

グループ全体で話し合います。条件文とは何かを生徒に尋ねます。条件文は、指定された条件が満たされたとき、または真になったときにのみ実行されるコードの一部であることを説明します。条件文では、真と偽というブール値が使われます。

SAMPLE

2. 探索する

生徒は、ハブのモーションセンサーの使用方法を探求しながら、簡単な条件文の作成を探求します。
生徒にハブをセットから取り出し、Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒に、プログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

タップして動作させる

センサーはさまざまな方法で使用できます。モーションセンサーは、動かし方によってさまざまな動作をするようにプログラムできます。ハブを手に持ったときにどのように動かすことができると思うか、生徒と一緒にブレインストーミングを行います。生徒たちに、モーションセンサーをどのようにプログラムすべきだと思うか尋ねます。ハードウェアを正しく動作させるために、ソフトウェアはどのような情報を必要としますか？

ハブを叩くとライトマトリックスに画像が表示されるように、モーションセンサーを動作させるための擬似コードプログラムを書きましょう。擬似コードとは、プログラムに何をさせたいかを言葉で書くことです。

擬似コードのサンプル

- モーションセンサーをインポートする。
- ハブが叩かれたら
- 画像を表示

注：ステップで書き出すコードは、プログラムに入れるものと完全に一致する必要はありません。これは、生徒が何が必要かを考えるのを助けるためです。
モーションセンサーを使用して、ハブのジェスチャーやアクションに応じて異なる画像を表示するサンプルプログラムを生徒に提供します。生徒は、このプログラムをプログラミング・キャンバスに入力する必要があります。
生徒にプログラムを実行させます。

```
1  from hub import light_matrix, motion_sensor
2  import runloop
3
4  def if_tapped():
5      #ハブがタップされたとき、ジェスチャーTAPPEDを返す
6      return motion_sensor.gesture()==motion_sensor.TAPPED
7
8  async def main():
9
10     #ハブがタップされるまで待つ
11     await runloop.until(if_tapped)
12
13     #ハブがタップされたら「悲しい顔」を表示する
14     light_matrix.show_image(IMAGE_SAD)
15
16     #コンソールに「Stop please(お願い、やめて)」を表示する
17     print('Stop please')
```

生徒とプログラムについて話し合います。生徒に、このプログラムについて新しい点を指摘するよう求めます。生徒は、”if”と二重等号 (==) が使われていることに気づくかもしれません。
このプログラムは条件文であることを生徒に説明します。このプログラムは、条件が満たされた場合に、ハブに 1 つのアクションを指示するものです。ハブを叩く条件について話し合います。条件は「 if」文によって設定されます。条件が満たされたときだけ、ハブに

SAMPLE

悲しい顔が現れます。条件が満たされなければ、ハブには何も表示されません。
また、生徒に二重等号を示します。等号は変数やオブジェクトの値を設定するために使うので、条件文では使えません。Python は二重の等号を使って比較を指定します。
条件文は、設定された条件と入力を比較します。

生徒に時間を与え、さまざまな入力を変更してプログラムの条件設定を調べる。オプションは、motion_sensor.TAPPED、motion_sensor.DOUBLE_TAPPED、motion_sensor.SHAKEN、motion_sensor.FALLING、motion_sensor.UNKNOWN です。ジェスチャオプションは、ナレッジベースの API モジュール、ハブ、モーションセンサーからも参照できます。

傾けて動作させる

生徒は、アクションのためにハブをさまざまな方向に傾けることを探求します。
ジェスチャーに加えて、モーションセンサーを使用して、ハブの向きや傾きの条件を設定できます。モーションセンサーを使用して、ハブの向きや傾きによって異なる画像を表示するサンプルプログラムを生徒に提供します。生徒は、このプログラムをプログラミング・キャンバスに入力する必要があります。

生徒にプログラムを実行させます。

この例で、ハブの向きや傾きによって満たされる条件について話し合います。条件が満たされた場合のみ、ハブに矢印が表示されます。条件が満たされない場合、ハブには何も表示されません。

生徒がプログラムに追加することに挑戦させます。条件が満たされたら、コンソールにメッセージを表示し、音を鳴らします。生徒にプログラムを書いて実行する時間を与えます。

注意：ハブがすでにプログラムで指定された正しい向きになっている場合は、矢印が自動的に表示され、サウンドが再生されます。
サンプルプログラム

```
1  from hub import light_matrix, motion_sensor
2  from app import sound
3  import runloop
4
5  def face_up():
6      #ハブが上向き（USBポートが上）のとき、BACKをを返す
7      return motion_sensor.up_face()==motion_sensor.TOP
8
9  async def main():
10
11     #ハブが上向きになるまでまつ
12     await runloop.until(face_up)
13
14     #ハブがタップされたら「矢印（上向き）」を表示する
15     light_matrix.show_image(IMAGE_ARROW_N)
16
17     #コンソールに「I am awake(起きました)」を表示する
18     print('I am awake')
19     #サウンド「Applause 1」を再生する
20     await sound.play('Applause 1')
21
22 runloop.run(main())
23
```

生徒に時間を与え、異なる入力を変更することで、プログラムの条件設定を調査させる。オリエンテーションとは、ハブの特定の面が上を向いている状態のことです。

向きのオプションは次のとおりです：

スピーカー側は motion_sensor.FRONT です。
USB 側は motion_sensor.BACK です。
A、C、E ポート側は motion_sensor.LEFT です。
B、D、F ポート側は motion_sensor.RIGHT です。
ライトマトリックス側は motion_sensor.TOP です。
バッテリー側は motion_sensor.BOTTOM です。

方向オプションは、ナレッジベースのハブセクションの「モーションセンサー」でも参照できます。
必要に応じてナレッジベースを参照し、コード行を追加するように生徒を促してください。また、生徒はエラーメッセージが表示されないかコンソールを見てください。

3. 説明する

生徒にプログラムを共有させ、プログラムがどのように動作したか話し合います。
生徒に次のような問題を出します：

- プログラムの中の条件文とは何ですか？
- “if”は条件を設定するために何をしますか？
- “if”の後の行がインデントされているのはなぜですか？
- モーションセンサーはどのように動作しますか？
- モーションセンサーに設定できる条件は何ですか？

4. もっと詳しく探究する

生徒たちに、モーションセンサーを使って物語と対話するゲームを作ることに挑戦してもらいます。
生徒に、空白のある短い物語を作るように指示します。その空白は、モーションセンサーからのアクションによって埋められ、画像を表示したり、サウンドを再生したりできるようにします。生徒は、print() 関数を使用して、コンソールにストーリーの言葉を含めることができます。

疑似コードプログラムとして書かれた簡単な例は以下の通りです：

- Print(‘ある日、ジェーンは公園を歩いていました。彼女は上を見上げました。’)。
- モーションセンサーを上に傾ける。
- ゴースト画像が表示される。
- 音が鳴る。
- Print(‘それから彼女は目を覚まし、それが夢であることに気づいた’)。

生徒にストーリーゲームを作る時間を与える。生徒は、最初に擬似コードでストーリーを書き出します。生徒には、プログラムの各部を説明するコードコメントも含めるように指示します。
生徒のグループは、自分たちのストーリーを他のグループと共有します。モーションセンサーのさまざまな使用方法について生徒に考えさせます。

SAMPLE

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。

生徒に次のような質問をします。

- 条件文はどのように機能しますか？
- モーションセンサーはどのように動作しますか？

自己評価

生徒に日記で次のことを答えてもらいます。

- 条件文の使い方について、今日学んだことは何ですか？

SAMPLE

ユニット 5 レッスン 2
クローマシン

クローマシン

生徒は、条件文を使ってブロックを動かすロボットハンドを作成します。

主な目的

生徒は次のことを行います：

- 基本的なループを作成する。
- 設定された条件に基づいてグラバー・モデルをプログラムする。

探究のための質問

- if 文を使わずに条件文を作るにはどうしたらよいか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）

1. 引きつける

ロボットアームの動きについて生徒に考えさせます。さまざまなタイプのロボットアームの画像やビデオをいくつか共有します。例えば、溶接、製造、出荷、ピッキング、仕分けなど。
ロボットアームがどのように動き、機能するかについて生徒と話し合います。ロボットアームがどのように設計されているかを生徒に指摘させます。
話し合いの場を移し、爪を使って景品をつかむアーケードゲームをしたことがあるかどうかを生徒に尋ねます。生徒たちに、このゲームが学習したロボットアームとどのように似ているか、また異なっているかを比較させます。それぞれの考えを話し合います。

2. 探究する

生徒は、条件文を使って物をつかむロボットハンドをプログラムすることを探求します。
生徒を SPIKE アプリの BUILD セクションに誘導します。ここで生徒は、ロボットハンドの作成手順にアクセスできます。生徒にモデルを作るように指示します。作り方の説明は、https://education.lego.com/en-us/support/spike-prime/building-instructions にもあります。

Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。生徒に、プログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

ロボットハンドを使用する

ロボットハンドにクローマシンのような物体を拾わせるアイデアを話し合います。ハンドを開閉して物を拾うようにプログラムする方法を考えます。生徒は、ハンドが開いたり閉じたりするためには、何をすべきか条件を設定する必要があることに気づくはずです。
このサンプルプログラムを生徒と共有します。生徒がどのように動作するかを理解できるように、グループでプログラムを確認します。
このプログラムでは、左ボタンを長押ししてハンドを閉じる条件を設定します。離すとハンドが開きます。
生徒はこのプログラムをプログラミング・キャンバスに入力する必要があります。生徒はこのプログラムを実行します。

SAMPLE

```python
from hub import port, button
import runloop
import motor

def pressed():
    #ハブの左ボタンが押されたかどうかを返す
    return button.pressed(button.LEFT)

def released():
    #ハブの左ボタンが離されたかどうかを返す
    return not button.pressed(button.LEFT)

async def main():

    #左ボタンが押されるまでまつ
    await runloop.until(pressed)

    #モーターを回転する（ハンドを開く）
    motor.run(port.F, -750)

    #左ボタンが離されるまでまつ
    await runloop.until(released)

    #モーターを回転する（ハンドを閉じる）
    motor.run(port.F, 750)

runloop.run(main())
```

プログラムがどのように動作するかを生徒に見せた後、生徒のセットから 2x4 ブロックをすべて取り除くように指示します。ロボットハンドをクローマシーンに見立てて、ブロックを拾って動かす練習をする。なお、このプログラムは 1 回しか動作しないように設定されています。

3. 説明する

プログラムがどのように動作したかを生徒と話し合います。
生徒に次のような質問をします：

- プログラムはどのように動作しますか？
- プログラムの中で、条件文はどのように使われていますか？条件とは何ですか？
- プログラムの中で真と偽を示すものは何ですか？
- if 文を使っていなくても、これは条件文であることを生徒に指摘します。なぜこれが条件文とみなされるのか、説明しなさい。このプログラムは、ボタンが押されて放されたことに基づいてモーターを動かすという条件を使っています。

4. 詳しく説明する

リレーゲームに挑戦します。2 つのグループに協力してもらい、1 つのロボットハンドからもう 1 つのロボットハンドにブロックを渡してもらいます。グループ 1 はレンガをつかみ、グループ 2 の方を向いてレンガを渡そうとする。次に 2 番目のグループがレンガを渡し返します。ブロックの受け渡しを何回できるか挑戦します。
この挑戦について生徒と話し合います。プログラムを変更する必要があるかどうかを生徒に尋ねます。生徒は、手を 1 回以上開閉できるようにする必要があることを認識します。プログラムに while ループを含めることを生徒に考えさせます。
生徒にプログラムを修正して実行する時間を与えます。エラーメッセージが表示されないかコンソールを見るように生徒に注意を促します。

SAMPLE

サンプルプログラム

```
from hub import port, button
import runloop
import motor

def pressed():
    #ハブの左ボタンが押されたかどうかを返す
    return button.pressed(button.LEFT)

def released():
    #ハブの左ボタンが離されたかどうかを返す
    return not button.pressed(button.LEFT)

async def main():

    while True:
        #左ボタンが押されるまでまつ
        await runloop.until(pressed)
        #モーターを回転する（ハンドを開く）
        motor.run(port.F, -750)
        #左ボタンが離されるまでまつ
        await runloop.until(released)
        #モーターを回転する（ハンドを閉じる）
        motor.run(port.F, 750)

runloop.run(main())
```

生徒にブロックを前後に渡す練習をさせます。最後の課題として、クラス全員に１つのブロックまたは複数のブロックを全グループに渡すことに挑戦させることを検討します。

この経験をグループで話し合います。生徒に、条件文とは何か、条件文を作成するさまざまな方法を定義するよう求めます。生徒たちは、条件文とは、プログラム中に真または偽の条件を設定するものであることを認識します。この例では、「while True」によって条件が設定され、条件が真である限りループが継続することを示しています。もし条件が真でなければ、アクションは起こりません。

生徒にブロックを前後に渡す練習をさせます。最後の課題として、クラス全員に１つのブロックまたは複数のブロックを全グループに渡すことに挑戦させることを検討します。

この経験をグループで話し合います。生徒に、条件文とは何か、条件文を作成するさまざまな方法を定義するよう求めます。生徒たちは、条件文とは、プログラム中に真または偽の条件を設定するものであることを認識します。この例では、「while True」によって条件が設定され、条件が真である限りループが継続することを示しています。もし条件が真でなければ、アクションは起こりません。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。

生徒に次のような質問を出します：

- ロボットハンドは条件文にどう反応したか？
- 条件文の作成にはどのような方法がありますか？
- 条件が満たされているかどうかを手動で示すことができるプログラムを作成するにはどうすればよいですか？

自己評価

ノートに次のように答えさせます：

- 条件文とループの使い方について、今日学んだことは何ですか？

SAMPLE

ユニット 5 レッスン 3
ゲームの決定を図表化する

ゲームの決定を図表化する

フローチャートの作成方法を学ぶ。

探究のための質問

- フローチャートはプログラムの設計をどのように手助けできるか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（ 特に Bluetooth で接続する場合）
- 生徒が、クローゲームのレッスンで使用したロボットハンドのモデルを組み立てていることを確認する

主な目的

生徒は次のことを行います：

- 計画にフローチャートを使用する方法を理解する。
- フローチャートを作成し、それに従ったプログラムを書くことができる。

ボキャブラリー

フローチャート

1. 引きつける

毎日行う決定について生徒と話し合います。生徒にいくつかの例を挙げさせます。
生徒に例を挙げて促します。一つの決定は、毎朝朝食に何を食べるかということかもしれません。朝食に何を食べるかを決める要因は何か、生徒と話し合います。これには、空腹の度合いや時間の長さなどが含まれるかもしれません。

生徒に自分の例を選ばせ、決断を下すために自分自身に問いかける問題のリストを書き出します。
以下はその例です：

- 学校へ行く前に朝食を食べたい。
- すごくお腹が空いた？はい
- 食べる時間はありますか？はい
- 私の好きなシリアルはある？ある
- 好きなシリアルを食べることにする。

いくつかのグループに例を発表させる。生徒に、条件文になりそうな箇所を挙げてもらう。例えば、それぞれ何を食べるかを決めるための条件にはどのようなものがありますか？また、もし何か違う問題に答えていたら、出力や結末がどのように変わるかを考えさせる。

2. 探究する

生徒は、フローチャートを作成してプログラムを文書化することを探求し、次にスタッキングゲームのプログラムを作成します。

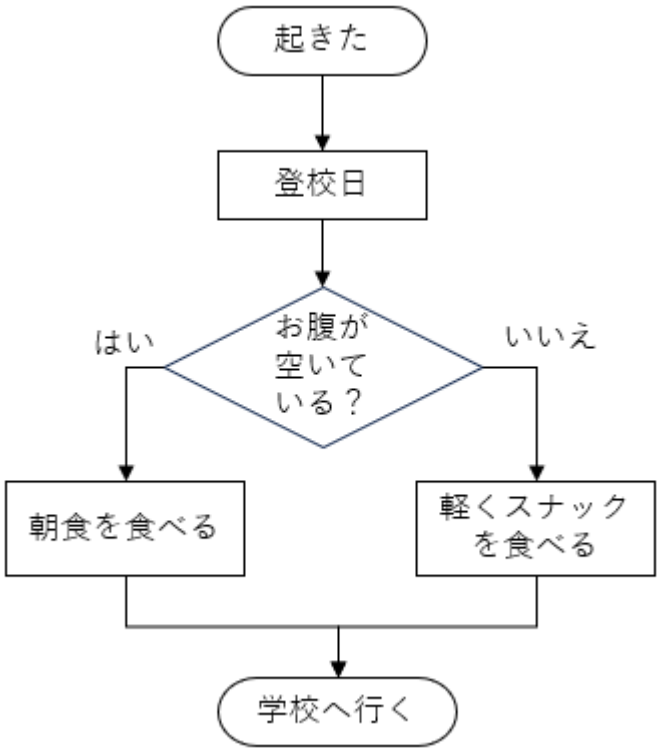
フローチャートの導入

フローチャートは、プログラムを整理し計画するためのツールとして使用することを生徒に説明します。フローチャートは、複雑な問題を分解するためのアルゴリズムを整理し、順序立てるために使用されます。フローチャートは、条件が満たされたり満たされなかったりした場合にプログラムが取り得るさまざまな経路をマップするのに役立つので、条件文を含むプログラムを書くときに特に役立ちます。
フローチャートの一般的な記号を紹介し、生徒が自分の作業で使えるようにします。生徒に表のコピーを渡すか、教室のどこかに掲示することを検討してください。

SAMPLE

記号	意味
	始まり/終わり プログラムの始まりと終わりを示す
	入力/出力 入力処理または出力処理を示す
	プロセス・手順 手順を示す
	分岐 条件による分岐処理を示す
	フロー・流れ プログラムの流れを示す

引きつける活動を基に、協力してフローチャートを作成します。このフローチャートを生徒と共有し、どのような結果が起こりうるか話し合います。



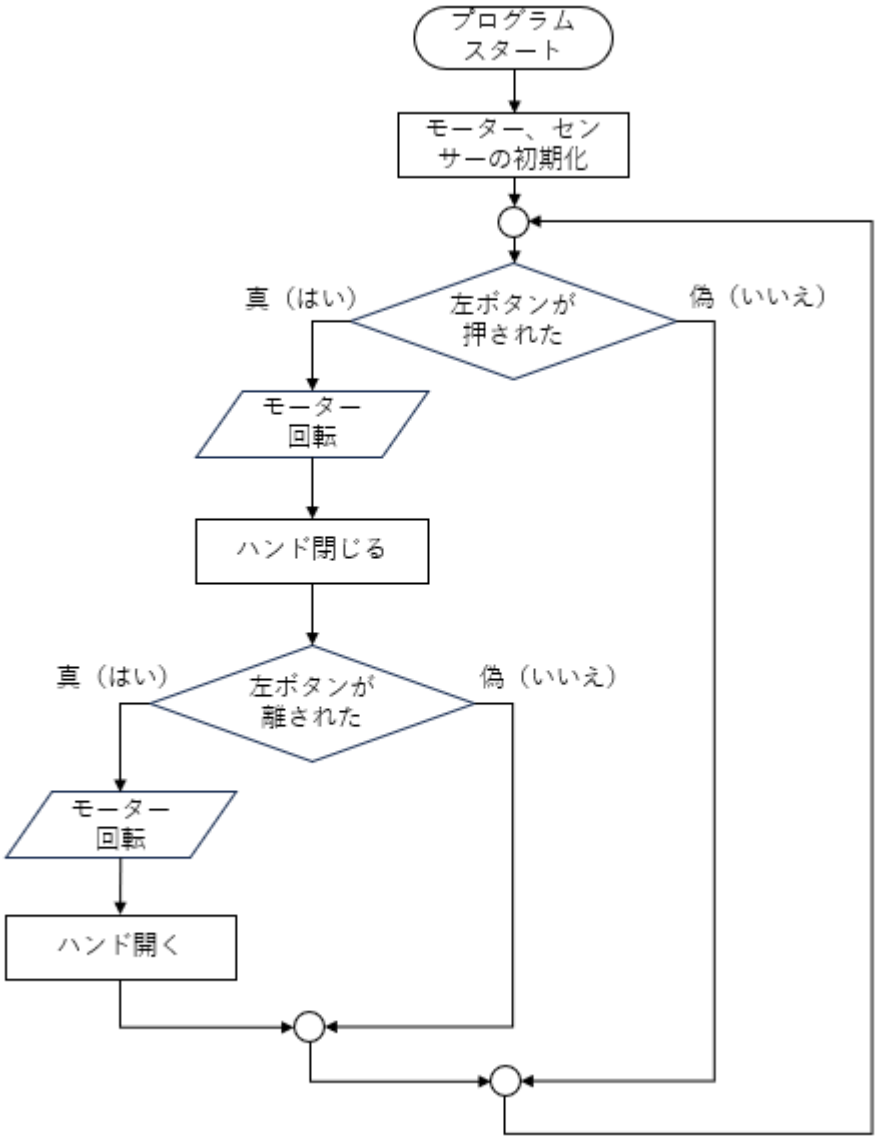
グループでフローチャートを話し合います。結果は１つではないことに注意してください。この問題は、ある条件を設定しています。お腹が空いていて条件が真になるか、お腹が空いていなくて条件が偽になるかのどちらかです。

生徒に時間を与え、引きつける活動にある例題をもとに、用意された記号を使って練習しながら、自分のフローチャートを完成させます。グループでフローチャートを話し合います。

積み上げゲーム

Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒は、クローゲームのレッスンで作成したプログラムを開きます。生徒はハブを接続します。
生徒がロボットハンドを使って、より正確な動きをするように挑戦します。生徒は、3 分間にできるだけ多くのブロックを積み上げることに挑戦します。
開始する前に、この課題を完了するためのフローチャートを作成する必要があることを生徒に説明します。生徒は、開始（すべてのブロックを分離）から終了（すべてのブロックを積み重ねる）までに必要な手順を書き出します。

生徒は、何が必要かを考えるために、クローゲームプログラムを参照することができます。
フローチャートの例



フローチャートの例を生徒と話し合います。条件文がどこにあるか、while ループが含まれていることを指摘します。必要に応じてフローチャートの例を示します。

グループ内の各生徒に、順番にブロックをいくつ重ねることができるか見てもらいます。生徒は、クローゲームのプログラムを再利用しても、新しいプログラムを作成しても構いません。
生徒に課題を完了する時間を与えます。

3. 説明する

生徒と課題について話し合います。

生徒に次のような問題を出します：

- フローチャートはプログラム作成にどのように役立ちますか？
- フローチャートの作成で難しいことは何ですか？
- どのようにブロックを積み重ねることができましたか？
- プログラムのどの部分が条件文ですか？
- この課題の何が難しかったですか？

4. 詳しく説明する

生徒に競争に挑戦させます。

もう１回、積み上げゲームをして時間を計るように生徒に促します。

今度は、生徒がすべてのブロックを積み上げ、次に積み上げます。生徒のグループ同士で競争させます。

生徒に課題を完了する時間を与えます。課題について話し合います。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。

生徒に次のような質問をする：

- フローチャートの目的は何ですか？
- フローチャートの各セクションの形が違うのはなぜですか？
- ブロックを積もうとしたらどうなりましたか？これは簡単な仕事ですか？

自己評価

ノートに次のように答えさせます：

- フローチャートの使い方について今日学んだことは何ですか？

SAMPLE

ユニット 5 レッスン 4
色当てゲーム

色当てゲーム

生徒は、ゲームを作成するための色センサーのプログラミングを学びます。

主な目的

生徒は次のことを行います：

- 条件コードを使用してカラーセンサーをプログラムする。
- ゲームを作成する。

探求のための質問

- 色センサーは、反応を引き起こすための情報を提供するためにどのように使用できますか？

必要な材料

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）

1. 引きつける

カラーセンサーの仕組みについて生徒に考えさせます。
4 つの異なるカラーブロックを使用して、ゲームに挑戦します。生徒には、カラーブロックを掲げて、その色に割り当てられたアクションを行うよう伝えます。生徒が見ることができる場所に、以下の動作を書いておくことを検討します。

青 ＝ 拍手
黄色 ＝ ジャンプ
緑 ＝ ストンプ
赤 ＝ くるくる回る
他の色 ＝ 何もしない

生徒に立ち上がってもらい、異なる色を掲げ、生徒の反応を見ながら数ラウンド行います。最初はゆっくりと、最後のラウンドでは生徒の動きが非常に速くなるように、スピードを上げることを考慮します。
このゲームについてクラスで話し合います。生徒たちに、ある行動をするように指示したものは何かを尋ねます。これはカラーセンサーに似ていると話し合いましょう。センサーは「見た」色に基づいて情報（入力）を受け取り、プログラムによって指定された行動を取ります（または、その色がコードにない場合は何もしません）。

2. 探究する

生徒はカラーセンサーと条件文の操作を探求します。
SPIKE アプリの BUILD セクションに生徒を誘導します。ここで、生徒は犬のキキの作成手順にアクセスできます。生徒はモデルを作成します。
https://education.lego.com/en-us/support/spike-prime/building-instructions にも組み立て手順があります。
生徒に Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒にプログラミング領域にすでにあるコードをすべて消去するように指示します。生徒はハブを接続します。

SAMPLE

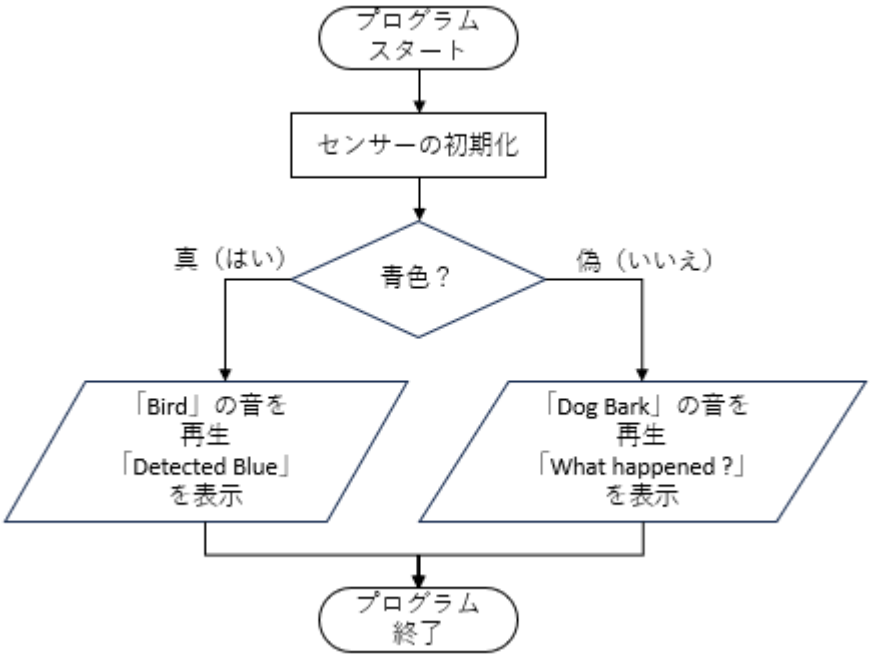
If/Else 文

生徒は、カラーセンサーの使用方法を調べ、if/else 文を作成します。

生徒に、カラーセンサーの「 目」を使ってさまざまな色を見ることができるキキのモデルを紹介します。生徒に、情報を提供するためにカラーセンサーを使用する方法を考えるよう促します。生徒に、自分たちが作ったカラーセンサーをモデルの中央に置くように指示します。青色が検出された場合にキキが 1 つの音を鳴らし、そうでない場合に別の音を鳴らす条件文の作成方法について生徒と話し合います。

このプログラムのフローチャートを作成します。

サンプルフローチャート



カラーセンサーを使用するためのサンプルプログラムを生徒に提供します。生徒は、このプログラムをプログラミングキャンバスに入力する必要があります。生徒にプログラムを実行させます。

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():

    #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
    if color_sensor.color(port.A)==color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
    #青を検知しなかったら（青以外だったら）、犬の鳴き声1を再生する
    else:
        await sound.play('Dog Bark 1')
        print('What happened')

runloop.run(main())
```

生徒に、検出された色を変えるなど、プログラムを調べる時間を与えます。グループでプログラムについて話し合います。

3. 説明する

プログラムがどのように機能したかを生徒と話し合います。

生徒に次のような質問をします：

- このプログラムはどのように動作しますか？
- if/else 文はどのように動作しますか（ true/false 言語を使って）？
- このプログラムでは、なぜモデルは 3 色すべてに異なる反応を示さないのか？
- else 文はどのように動作しますか？

4. 詳しく説明する

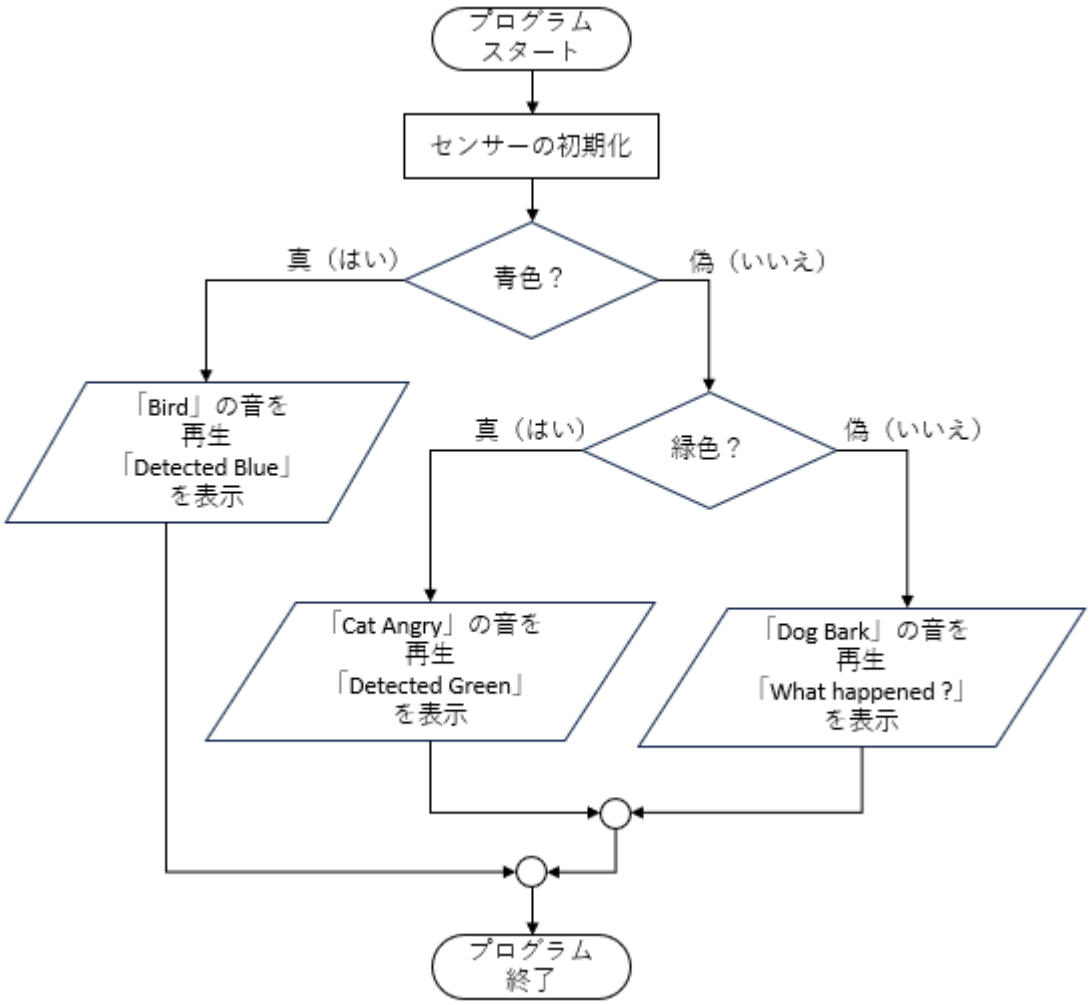
生徒は elif コードを使用して、条件文を 1 つの文にまとめる方法を調べます。

それぞれの色に対して異なる音を鳴らすことができるように、プログラムをどのように追加したらよいかを生徒に考えさせます。青が検出されたら 1 つの音を鳴らし、緑が検出されたら別の音を鳴らし、どちらも検出されなかったら 3 つ目の音を鳴らすように生徒に指示します。

生徒に elif 文を紹介します。elif は else と if を組み合わせたもので、if 文の後にのみ書くことができます。elif 文は、最初の条件または if 文が偽と読み取られた場合、プログラムに新しい条件を設定し、プログラムの次のステップを提供します。

サンプルフローチャートを生徒と共有し、elif 文を含むプログラムの経路について話し合います。

サンプルフローチャート



生徒に、elif 文を含むようにプログラムを修正するように指示します。生徒は、elif は最初の条件の if と elif によって設定された条件の else の間にあるべきであることを認識する必要があります。

サンプルプログラム

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():

    #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
    if color_sensor.color(port.A)==color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
    #または緑を検知したら、猫の起こり声（Cat Angry）を再生する
    elif color_sensor.color(port.A)==color.GREEN:
        await sound.play('Cat Angry')
        print('Green Detected')
    #青または緑以外だったら）、犬の鳴き声1を再生する
    else:
        await sound.play('Dog Bark 1')
        print('What happened')

runloop.run(main())
```

生徒にプログラムを探求し、音を変更する時間を与えます。生徒にプログラムを共有するように指示します。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。

生徒に次のような質問をします：

- if/elif/else プログラムはどのように動作しましたか？
- 条件文にループを追加したらどうなりましたか？
- なぜ if/else や elif 文をプログラムに入れたくなるのですか？

自己評価

ノートに次のように答えさせます：

- 条件文とループの併用について、今日何を学びましたか？

SAMPLE

ユニット 5 レッスン 5

推理ゲーム

推理ゲーム

ループを使った条件文の使い方を調べます。

探究のための質問

- なぜプログラマーは if/elif/else 条件文とループを組み合わせるのでしょうか。

必要な教材

- SPIKE プライムセット
- SPIKE アプリがインストールされている端末
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）。
- 生徒が、色当てレッスンで使用した犬のキキのモデルを組み立てていることを確認します。

主な目的

生徒は次のことを行います：

- if/elif/else プログラミングを使用して、複数の条件文を使用するコードを書く。
- コードにループを追加する。
- 不正確な構文、コードの欠落、または誤ったインデントがあるコードをデバッグする。

1. 引きつける

複数の選択肢から選ぶことを生徒に考えさせます。

生徒に、複数の選択肢が提示されたときにどのように選択するかを話し合うよう促します。生徒に、朝食ビュッフェにいるところを想像してもらいます。どのように食べるものを決めるのでしょうか。生徒たちに、自分ならどれを食べるかを 3 つ選んでもらいます。

生徒たちに自分の選択を発表させます。どのような影響（気分、時間、好きなものなど）を受けて選択したのかを話し合います。ビュッフェで料理の皿を作ることが、ある品目を選び（皿に加える）、他の品目を選ばない（列に並ばない）ことができるコードを作ることに似ていることを説明します。

2. 探究する

生徒は、ループで設定された条件に従って色に反応するようにキキモデルをプログラミングすることを探究します。

生徒に対して、色当てレッスンのために保存したプログラムを開くように指示します。生徒はハブを接続します。生徒にプログラムを 1 回実行させて、その仕組みを覚えさせることを検討します。

if/elif/else 文によるループの使用

生徒に、色当てレッスンで使用したプログラムをどのように修正して、プログラムが複数の色を識別できるように条件を繰り返すことができるか考えさせます。生徒は、while ループによって、カラーセンサーが設定された条件に従って読み取った色に反応し続けることを理解します。

生徒に、色当てゲームのレッスンで作成したフローチャートを更新するか、新しいフローチャートを作成して、プログラムがどのように動作するかを示すように指示します。

生徒にプログラムを修正して試す時間を与えます。コンソールにエラーメッセージが表示されないか注意するように生徒に思い出させます。元のプログラムを失わないように、新しいプログラムとして保存するように生徒に勧めます。

SAMPLE

サンプルプログラム

```
1 from hub import port
2 import runloop
3 import color_sensor
4 import color
5 from app import sound
6
7 async def main():
8
9     #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
10     if color_sensor.color(port.A)==color.BLUE:
11         await sound.play('Bird')
12         print('Blue Detected')
13     #青を検知しなかったら（青以外だったら）、犬の鳴き声1を再生する
14     else:
15         await sound.play('Dog Bark 1')
16         print('What happened')
17
18 runloop.run(main())
19
```

生徒と一緒にプログラムを見直します。プログラムにループがあることで、何が変わったかを話し合います。生徒たちは、ループによって、最初の色が検出された後もプログラムを実行し続けることができること、つまり、新しい色が提示されたときにプログラムが反応し続けることがわかるはずです。

推理ゲームをする

生徒に、セットから他の 2 色の 2x4 ブロックを取り出してもらいます。生徒に、5 色すべてが入るようにモデルを並べ替えさせます。生徒は、各色を読み取るための新しい条件を含むようにプログラムを作り直します。複数の elif 文を含めることができることを生徒に促します。生徒が協力して、それぞれの色のブロックで異なる音を出すようにプログラムします。

サンプルプログラム

```
1 from hub import port
2 import runloop
3 import color_sensor
4 import color
5 from app import sound
6 async def main():
7     #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
8     if color_sensor.color(port.A)==color.BLUE:
9         await sound.play('Bird')
10         print('Blue Detected')
11     #または緑を検知したら、猫の起こり声（Cat Angry）を再生する
12     elif color_sensor.color(port.A)==color.GREEN:
13         await sound.play('Cat Angry')
14         print('Green Detected')
15     #または赤を検知したら、いびき（snoring）を再生する
16     elif color_sensor.color(port.A)==color.RED:
17         await sound.play('Snoring')
18         print('Red Detected')
19     #または紫（マゼンタ）を検知したら、牛の鳴き声（Moo）を再生する
20     elif color_sensor.color(port.A)==color.MAGENTA:
21         await sound.play('Moo')
22         print('Magenta Detected')
```

SAMPLE

```
    #または黄色を検知したら、Wobbleを再生する
    elif color_sensor.color(port.A)==color.YELLOW:
        await sound.play('Wobble')
        print('Yellow Detected')
    #青または緑以外だったら）、犬の鳴き声1を再生する
    else:
        await sound.play('Dog Bark 1')
        print('What happened')
runloop.run(main())
```

それぞれのパートナーが順番にゲームマスターになり、もう１人のパートナーが推理ゲームを行います。パートナー２はモデルを見ずに、パートナー１がプログラムをプレイし、モデルに選ばれた 2x4 ブロックの各色をスキャンさせます。どの色をスキャンするかはパートナー１が決めます。パートナー２は、色の組み合わせを推測するために再生される音を聞くことができます。パートナー２は、色の順番を書き留めるか、セットの他のピースを使って色の順番を示すことができます。

ゲーム終了時に推測を確認させ、パートナーを交代させます。

3. 説明する

プログラムがどのように機能したかを生徒と話し合います。
生徒に次のような質問をします：

- ループと条件文は、１つのプログラムの中でどのように連動できますか？
- このプログラムでは、複数の条件文をどのように使っていますか？
- while ループは、実際には条件文も設定していることを生徒に思い出させます。コードのその部分が真である間、それは繰り返され続けます。
- １つのプログラムにいくつの条件を設定できますか？
- プログラムはいつ終了しますか？

4. もっと詳しく探究する

色を検出するのではなく、バグをチェックする新しいゲームに挑戦します。
各プログラムとエラーメッセージを生徒に見せます。バグを修正したり、不足しているコードを追加したりするために、それぞれのコードのどこを変更する必要があるか、生徒に話し合わせます。クラスで変更を行い、各変更後にプログラムが正しく実行されることを確認します。

デバッグ活動１：

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
    #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
    if color_sensor.color(port.A)==color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
    #または緑を検知したら、犬の鳴き声1を再生する
    elif color_sensor.color(port.A)==color.GREEN:
        await ssound.play('Dog Bark 1')
        print('Green Detected')
    #青または緑以外だったら）、猫の起こり声（Cat Angry）を再生する
```

SAMPLE

```
17     else
18         await sound.play('Cat Angry')
19         print('What happened?')
20 runloop.run(main())
21
```

Traceback (most recent call last):

File“ Project 2” , line 17

SyntaxError: invalid syntax

エラーメッセージから、エラーが 17 行目にあることがわかるはずです。無効な構文とは、コード内の何かを正しくフォーマットしていないことを意味します。17 行目を見て、生徒は else 文の最後にコロン（ :）があるべきで、else として何をするかという文がインデント付きで次の行に書かれていることに気づくはずです。

デバッグ活動 2：

```
1 from hub import port
2 import runloop
3 import color_sensor
4 import color
5 from app import sound
6
7 async def main():
8     #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
9     if color_sensor.color(port.A)==color.BLUE:
10    await sound.play('Bird')
11    print('Blue Detected')
12    #または緑を検知したら、犬の鳴き声1を再生する
13    elif color_sensor.color(port.A)==color.GREEN:
14        await ssound.play('Dog Bark 1')
15        print('Green Detected')
16    #青または緑以外だったら）、猫の起こり声（Cat Angry）を再生する
17    else:
18        await sound.play('Cat Angry')
19        print('What happened?')
20 runloop.run(main())
21
```

Traceback (most recent call last):

File“ Color 2” , line 10

SyntaxError: invalid syntax

生徒は、エラーメッセージに示されているように、エラーが 10 行目にあることを認識するはずです。if で作成された条件文は、条件のコード行がインデントされていないため、条件を読み取っていません。プログラムを修正するには、条件文に必要なすべてのコード行（ 10 行目と 11 行目）をインデントする必要があります。

生徒に 10 行目だけをインデントするように指示します。プログラムをもう一度実行します。生徒には新しいエラーメッセージが表示されます。

SyntaxError: invalid syntax

Traceback (most recent call last): File“ Color 2” , line 12

SyntaxError: invalid syntax

生徒は、コードにまだエラーがあることを認識しなければなりません。10 行目をインデントすることで、条件文が完成しました。11 行目がインデントされていないため、条件は 11 行目を読んで終了します。

SAMPLE

コードの欠落 アクティビティ

```
1  from hub import port
2  import runloop
3  import color_sensor
4  import color
5  from app import sound
6
7  async def main():
8      #もしカラーセンサーがBlue(青)を検出したら、鳥の音を再生する
9      if color_sensor.color(port.A)==color.BLUE:
10         await sound.play('Bird')
11         print('Blue Detected')
12     #または緑を検知したら、犬の鳴き声1を再生する
13     elif
14
15
16     #青または緑以外だったら）、猫の起こり声（Cat Angry）を再生する
17     else
18
19
20 runloop.run(main())
```

このプログラムを生徒と話し合います。生徒たちに、このプログラムを完全なものにするには何が足りないかを尋ねましょう。デバッグと同様に、プログラムを実行してエラーメッセージを受け取ることができます。

Traceback (most recent call last):

File“ Project 2”, line 13

SyntaxError: invalid syntax

この場合、elif 文が定義されていないため、プログラムは 13 行目のエラーを送信します。生徒は、元の条件が偽と読み取られた場合に新しい条件を設定する何かを追加する必要があります。

elif 文にコードの一部を追加して、プログラムをもう一度試してみてください。生徒は、else 文にも追加する必要性を示したかもしれません。生徒が新しいプログラムを実行すると、else 文にもコードを追加するようにという新しいエラーメッセージが表示されます。このエラーメッセージは、プログラムが最初のエラーメッセージを読んでから停止するので、以前は表示されなかったことを生徒に伝えてください。つまり、追加のメッセージは最初は呼び出しません。

5. 評価する

教師の観察：

生徒とプログラムについて話し合います。

生徒に次のような質問をします：

- ループと条件文はどのように機能しますか？
- 2 つの真の応答を可能にする条件文をどのように作成できますか？
- ある入力を別の入力に置き換えるにはどうすればよいですか？

自己評価

ノートに次のように答えさせます：

- 1 つのプログラムに複数の条件を追加することについて、今日は何を学びましたか？

SAMPLE

ユニット 5 レッスン 6

スコア！

スコア！

条件文の使い方をゲームの得点に応用します。

主な目的

生徒は次のことを行います：

- 動きと光の行列をプログラムする。
- 条件文の知識を応用する。

探究のための質問

- プログラマーは、どのように条件を使ってアクションや状態を追跡できるでしょうか。

必要な教材

- SPIKE プライムセット
- SPIKE アプリがインストールされている端末
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）。

1. 引きつける

https://education.lego.com/en-us/lessons/prime-extra-resources/goal#lesson-plan にあるゴールレッスンの引きつけビデオを見ます。生徒に、プレーヤーの腕の動き方を研究するように指示します。生徒は、腕が円全体を回るように動くことを確認します。できるだけ多くのゴールを決めるために、どのようにプレーヤーをプログラムするかを生徒と話し合います。

2. 探究する

生徒は、ゴールとプレーヤーのモデルを作成して、モーターを使用したさまざまな動き方を調べます。
生徒を SPIKE アプリの BUILD セクションに誘導します。ここで、生徒はテーブルトップゲームの組み立て手順にアクセスできます。
生徒に、ゴールやアクセサリーとプレーヤーのモデルを作るように指示します。組み立て説明書は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。
生徒に、Python プログラミングキャンバスで新しいプロジェクトを開くように指示します。生徒に、プログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

ゴールを決める

生徒は、プレーヤーがゴールを決めるようにプログラミングする必要があります。モーターが動くようにプログラムする最良の方法について生徒と話し合います。どのようなコードが効率的か、複数の方法を試すように生徒に促します。
プログラムの目的を文書化するために、フローチャートを作成するように生徒に指示します。生徒が助けを必要とする場合は、ナレッジベースやノートを活用します。生徒にプログラムを書いて実行させます。

SAMPLE

```
from hub import port
import runloop
import motor

async def main():
    #ポートAのモーターを720度/秒で、180度の位置へ動かす
    #これは、アームをスイングする準備で振り上げる動作
    motor.run_to_absolute_position(port.A, 180, 720)

    #ポートAのモーターを720度/秒で、360度動かす
    motor.run_for_degrees(port.A, 360, 720)

runloop.run(main())
```

生徒に、調査して最適なプログラムを選ぶ時間を与えます。また、コンソールにエラーメッセージが表示されないか注意します。

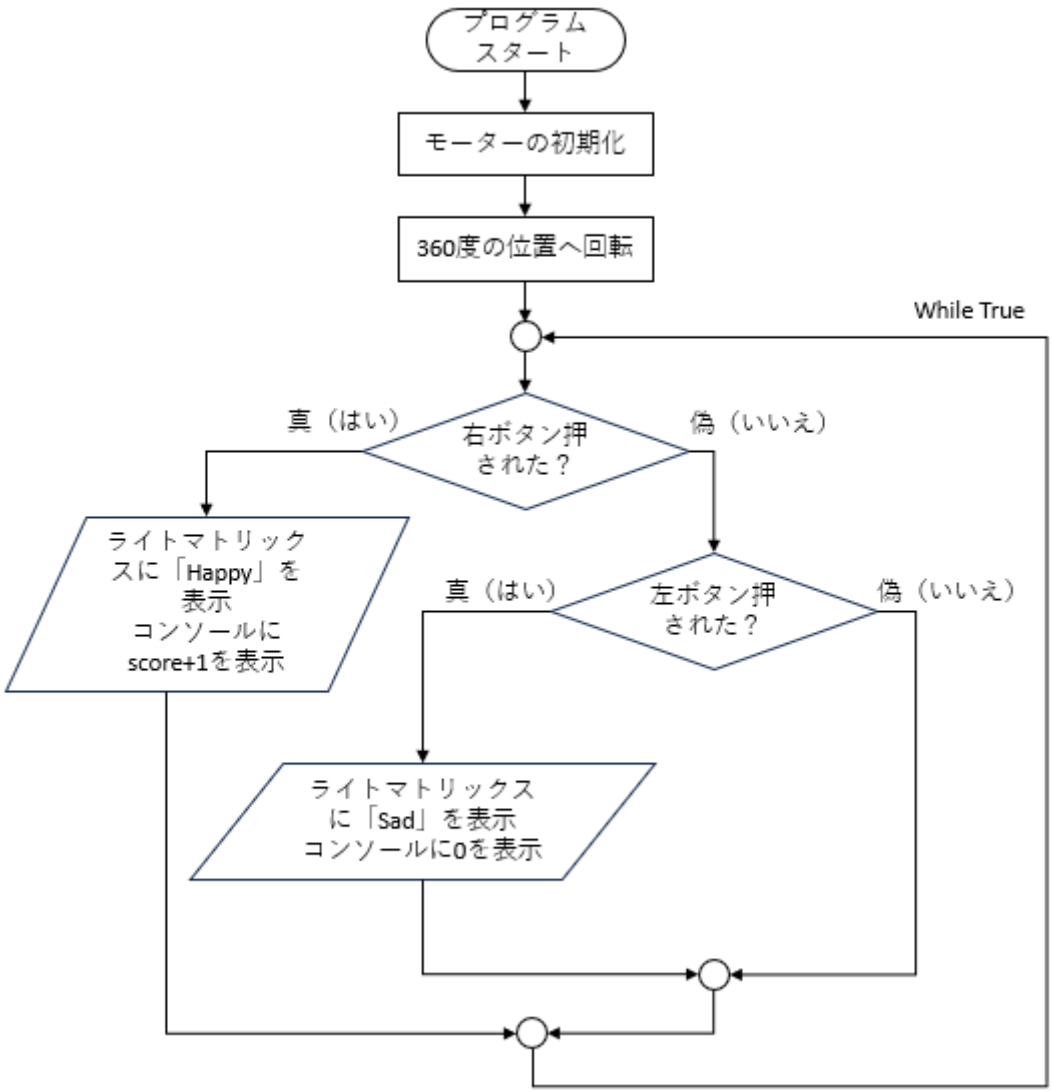
得点の管理

生徒は、自分のプレーヤーがゴールを決めるたびに得点することを示したいと考えます。

前回のコードに加え、スコアを管理するための条件文を追加します。生徒は次のことを行う必要があります：

- ゴールが決まったかどうかを示すために、画像やメッセージ、音を使ってハブをプログラムします。ゴールが決まった場合は１つの画像、決まらなかった場合は別の画像や音を選択します。
- コンソールに得点を表示し、ゴールが決まれば１を加え、決まらなければ０を表示します。

フローチャートを通じて、生徒と新しいコード行を話し合います。生徒にプログラムに新しいステップを含めるようにフローチャートを更新するよう指示します。



フローチャートについて話し合い、条件文が真と偽の両方の条件に対してどのように行動するかを確認します。

生徒にプログラムを書いて実行させます。

```
1  from hub import port, light_matrix, button
2  import runloop
3  import motor
4
5  def right_released():
6      return not button.pressed(button.RIGHT)
7
8  def left_released():
9      return not button.pressed(button.LEFT)
10
11 async def main():
12     #ポートAのモーターを720度/秒で、180度の位置へ動かす
13     #これは、アームをスイングする準備で振り上げる動作
14     motor.run_to_absolute_position(port.A, 180, 720, direction=motor.CLOCKWISE)
15     #ポートAのモーターを720度/秒で、360度動かす
16     motor.run_for_degrees(port.A, 360, 720)
17     #得点を0にセット
18     score = 0
19     #スコアを示すループ条件文
20     while True:
21         if button.pressed(button.RIGHT):
22             await runloop.until(right_released)
23             light_matrix.show_image(light_matrix.IMAGE_HAPPY)
24             score += 1
25             print(score)
26         elif button.pressed(button.LEFT):
27             await runloop.until(left_released)
28             light_matrix.show_image(light_matrix.IMAGE_SAD)
29             print('0')
30
31 runloop.run(main())
32
```

注意：このサンプルプログラムでは、生徒は得点を保持するための変数を設定する必要があります。このサンプルでは、変数名を score とし、初期値を 0 とします。
生徒にプログラムを数回実行させ、ゴールを決めさせます。グループでコードを確認します。

3. 説明する

生徒は、プレーヤーがゴールを決めるために作成したプログラムと、ゴールが決まったことを条件文がどのように示すかを説明します。
生徒に次のような質問をします：

- このプログラムでは、どのようにしてゴールが決まったことを示せましたか？
- 条件文はどのように使われましたか？
- フローチャートは、プログラム作成においてどのように指針となりますか？
- この課題の何が難しかったですか？
- プログラムは、どのようにして if アクションと else アクションのどちらを実行するかを決定するのでしょうか？両方の条件を使うことは可能ですか？

4. 詳しく説明する

ゴールが決まった際、そのことを示すだけでなく、ゴールが決まるたびに得点を記録するようにプログラムを変更します。
生徒全員で、プログラムを変更し、プレーヤーがゴールを決めるたびに追加点が入るようにするアイデアを出し合います。どのように動作を繰り返したり、継続したりする必要があるかを考えます（ 例えば、while ループを追加することを考慮します）。

フローチャートを更新し、プログラムを修正してテストします。
生徒がプログラムをクラスで共有し、合計で何ゴール決めたかを示す時間を設けます。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。生徒に次のような質問をします：

- if/else 条件文はどのように機能しますか？
- 2 つの真の応答を可能にする条件文をどのように作成できますか？
- ある入力を別の入力に置き換えるにはどうすればよいですか？

自己評価

ノートに次のように答えさせます：

- if/else 条件文の使用について、今日は何を学びましたか？

SAMPLE

ユニット 5 レッスン 7
ゲームタイム

ゲームタイム

卓上ゲームを作成し、プログラムする方法を学びます。

探究のための質問

- 簡単な卓上ゲームを作るには、どのように条件を設定すればよいでしょうか。

必要な教材

- SPIKE プライムセット
- SPIKE アプリがインストールされている端末
- ノート

準備する

- SPIKE プライムハブが充電されていることを確認する (特に Bluetooth で接続する場合)。

主な目的

生徒は次のことを行います：

- ゲーム形式で満たすべき条件を含むコードを書く。
- ロボットに応答させる一連のイベントを必要とするゲームを作成する。

1. 引きつける

生徒を卓上ゲームに関する話題に引きつけます。
スコアのレッスンを修了している場合は、そのゲームが卓上ゲームとしてどのように機能したかを話し合います。修了していない場合は、卓上ゲームの例の画像やビデオを見せて、生徒にその利点について考えさせます。どのようなシーンで使用されるのか、何が楽しいのかを話し合います。

ブレインストーミング

卓上ゲームのアイデアを出し合い、どのようなゲームが卓上で楽しめるかを考えます。
グループで、卓上ゲームを作るためのアイデアをいくつか出し合います。楽しい卓上ゲームの要素について考えます。
次に、小グループに分かれて、卓上ゲームに求めるものを他の生徒に尋ねる簡単なアンケートを作成します。生徒は、使用しようと考えている例を共有し、他のアイデアも求めます。生徒たちは、ゲームの楽しさや挑戦、テーマについて 3 つの質問を含め、アイデアが年齢に適しているか確認します。

2. 探究する

生徒は、卓上ゲームとして使用するゲームを設計、構築、プログラムします。

設計と適切なアイデアの選択

アンケートの結果を基にして、ゲームの設計を開始します。
生徒はゲームを設計し、構築し、プログラムします。次の条件を守ります：
1. ライトマトリックスを使用すること。
2. 少なくとも 1 つのモーターを使用すること。
3. 少なくとも 1 つのセンサーを使用すること。
4. タイマー機能を含むこと。

生徒は、製作アイデアのスケッチとプログラミングアイデアのフローチャートを作成します。

SAMPLE

テストと繰り返し

生徒がアイデアをテストし、分析する時間を設け、必要に応じて改善します。生徒は、設計基準とフローチャートに照らして自分の設計をテストし、評価します。
生徒が設計の過程を記録できるように、スケッチや写真を活用します。他のグループや教師からフィードバックを受けながら設計を改善します。

3. 説明する

生徒は設計を共有し、どのように機能するかを説明します。生徒と最初の共有セッションを行います。

生徒に次のような質問をします：

- ゲームを作るために、モデルをどのようにプログラムしましたか？プログラムのコメントを共有してもらいます。
- 設計を作成する際に、どのような決定をしましたか？
- どのような条件文を使いましたか？
- デバッグやトラブルシューティングが必要だった部分はどこですか？
- この課題で難しかったことは何ですか？

4. 詳しく説明する

最初の共有セッション後、生徒がプログラムを完成させるために追加の時間を設けます。
共有セッションで得たアイデアを活かして設計を完成させます。
次のフィードバックでレッスンを終了する場合は、モデルを一緒に残しておいてください。

5. 評価する

教師の観察

生徒とプログラムについて話し合います。生徒に次のような質問をします：

- この課題で最も難しかったのは何ですか？
- 課題を解決するためにどのようなアプローチをとりましたか？
- どのようなループや条件を使用しましたか？

自己評価

ノートに次のように答えさせます：

- 自分独自の設計を行うことについて、今日学んだことは何ですか？

SAMPLE

Unit 6 Testing for Bugs:Troubleshooing Hardware and Software
ユニット 6 バグのテスト トラブルシューティング ハードウェアとソフトウェア

Unit Introduction: ユニット紹介

このユニットでは、バグの特定と修正を通じて、Python を用いたコンピュータサイエンスの基本原理とプログラミングの概念を学びます。生徒は、ロボットの設計、製作、プログラミングにおける問題の特定方法を学び、それらの問題を効果的にトラブルシューティングするスキルを身につけます。また、テストの重要性を理解し、ハードウェア（設計）に問題があるのか、ソフトウェア（プログラム）に問題があるのかを見分ける方法を習得します。

レッスンは、生徒が次の分野のスキルと知識を高められるように構成されています：

- プログラムの作成とデバッグ
- 問題が設計にあるのか、プログラムにあるのかを見極めて解決策を見つける
- 設計上の問題のトラブルシューティング
- コードコメントを使ってプログラムを文書化する
- 問題の定義と分解

ユニット学習の期待

このユニットを通じて、生徒はモデルとプログラムの設計から生じる可能性のある問題を発見し、それに対するトラブルシューティングを行うスキルを養います。ハードウェアやプログラムにどのような問題が発生しうるかを体験し、問題の原因を特定し、トラブルシューティングする方法を学びます。生徒は、アルゴリズムを考えるための擬似コードと、プログラムを文書化するためのコードコメントを用いて、デバッグを効率的に行う習慣を身につけます。

探究への質問

- ソフトウェアエンジニアは、どのようにしてプログラムのバグを特定し、修正するのか。
- エンジニアはどのようにして設計内のハードウェアとソフトウェアの問題を特定し、修復するのか。

SAMPLE

ユニット 6 レッスン 1
プロトタイプのテスト

プロトタイプのテスト

生徒はエ学設計プロセスを学び、実際に体験します。

探究のための質問

エンジニアはどのようにしてアイデアを製品にするのか。

必要な教材

SPIKE プライムセット

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート
- 未開封のペットボトル

準備

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）

主な目的

生徒は次のことを行います。

- ・アイデアを出し合う
- ・問題に対する解決策を開発する
- ・モデルをプログラミングする

ボキャブラリー

・エンジニアリング設計プロセス
- 制約条件

プロトタイプ

1. 引きつける

設計工学のプロセスについて考えます。

エンジニアは、問題の解決策を考える際に、設計工学プロセスを使用します。このプロセスを生徒に紹介し、以下のステップで進めます。

- 問題を定義する
- 問題を調査する
- 可能性のある解決策をブレインストーミングする
- 最も有望な解決策を選択する
- プロトタイプを作る
- プロトタイプのテストと評価
- 伝える
- 再設計

エンジニアがこのプロセスを使ってどのように問題を解決するか、生徒と話し合いながら具体例も共有します。

2. 探究する

生徒は、設計工学プロセスを使って橋を建設します。

問題として「橋の模型を製作する」という課題を提示し、模型の路盤（大きな黄色のテクニックプレート）はテーブルから 76mm の高さにする必要があることを説明します。

生徒に、設計工学プロセスのステップをノートに記録させ、一緒に問題を定義していきます。モデルが重量を支え、安定性を保つために必要な要素も確認します。

必要に応じて、生徒が問題を調査し、ステップ 2 を進めるための時間を与えます。

SAMPLE

次のステップでは、可能な解決策を話し合い、すべてのアイデアを記録します。以下の制約を設定します：

- 使用できるのは、黄色の大きなテクニックプレート、テクニックビーム、コネクター、軸のみ
- フレームは使用できない
- 路盤には未開封のペットボトルを 2 本入れること

生徒に最も良いアイデアを選んで試作品を作らせます。
試作品が安定しているか、ある程度の重さに耐えられるかを確認させ、ペットボトル 2 本を用いてテストします。
生徒に試作品の写真を撮らせて記録します。

3. 説明する

生徒に、自分たちの模型について説明し、最終的な設計に至るまでのプロセスを発表させます。次の質問をします：

- 使える材料があれば、どのような橋が作れますか？
- なぜ、作る前に計画を立てることが重要なのですか？
- 実際に作る中で、設計をどのように変更しましたか？
- ペットボトルの重さに耐えられるか確認する必要があったのはなぜですか？
- ペットボトルを追加したとき、どの段階が完了したのですか？

ペットボトルを追加したときにテストと反復の段階が完了することを認識させます。自分の橋がペットボトルの重さに耐えられなかった場合でも、学んだことを共有し、次にどう改善するかを話し合います。
生徒には、テストが設計が意図した通りに動作するかどうかを確認する重要なプロセスであることを理解させ、テストに基づいてどのように設計を改善できるか考えるよう促します。次の質問をします：

- モデルをテストすることが重要なのはなぜですか？
- モデルのテストから何が学べますか？
- テスト後、次のステップは何ですか？
- テストと反復の段階が完了したとわかるのはいつですか？
- 他のチームのアイデアで、自分のデザインに取り入れたいものはありましたか？

4. もっと詳しく探究する

新しい素材の追加に基づいて、設計をさらに改善する機会を与えます。
生徒に、新しい設計図が製作され、フレームや他の部品が使用できることを伝えます。顧客からの要望として、プロトタイプに 4.5kg の荷重をかけることが求められていることも伝えます。
すべてのプロセスをやり直すか、途中から進めるかは生徒に任せます。グループでどのプロセスが良いかを話し合い、新しい部品を使って橋を再設計します。4.5kg を支えることが目標であることを思い出させます。

5. 評価する

教師の観察

生徒と設計について話し合います。次の質問をします：

- 最初のプロトタイプはどのくらいうまくいきましたか？
- なぜ設計をテストすることが重要なのでしょうか？
- エンジニアはどのようにしてアイデアからプロトタイプに移行しますか？

自己評価

ノートに次のことを書かせます：

- 今日、設計工学とテストの重要性について何を学びましたか？

SAMPLE

ユニット 6 レッスン 2
ブレイクダンサー・ブレイクダウン

ブレイクダンサー・ブレイクダウン

生徒はプログラムをデバッグするための戦略を学びます。

探究への質問

- ソフトウェア・エンジニアはどのようにしてプログラムのバグを特定し、修正するのか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備

- SPIKE プライムハブが充電されていることを確認する（ 特に Bluetooth で接続する場合）。
- 生徒がブレイクダンサー・ブレイクダウンのレッスンで使用したブレイクダンサーモデルを組み立てていることを確認する。

主な目的

生徒は次のことを行います：

- 問題を特定し、プログラムをデバッグする。

ボキャブラリー

バグ
デバッグ
構文エラー
論理エラー
ランタイム・エラー

1. 引きつける

問題がハードウェア（ モデル）にあるのか、プログラムにあるのかを識別する方法について話し合います。
以前のレッスンでは、プロトタイプのテストを通じて最適な解決策を見つける方法を学びましたが、プログラミングは含まれていませんでした。今回は、モデルの設計に問題があるのか、それともプログラムに問題があるのかを特定する方法を探ります。
ブレイクダンサーのモデルがどのように動くべきかを確認するために、実際に踊っているビデオを見ましょう。
https://education.lego.com/en-us/lessons/prime-life-hacks/break-dance#ignite-a-discussion
生徒とダンサーのさまざまな動きについて話し合います。次のような質問をします：

- モデルのどの部分が動いていましたか？
- その動きを模型で再現するにはどうすればよいですか？
- 問題が発生した場合、原因がモデルなのかプログラムなのかをどうやって見分けますか？

2. 探究する

生徒はブレイクダンサーモデルを作り、バグの特定と修正について調べます。
SPIKE アプリの BUILD セクションにアクセスし、ブレイクダンサーモデルの組み立て手順を確認します。生徒にモデルを作るように指示します。組み立て説明書は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。
Python プログラミングキャンバスで新しいプロジェクトを開き、既存のコードを消去します。ハブを接続します。

モデルのテスト

生徒は、サンプルプログラムを実行してバグがあるかどうかを確認します。
生徒に、モデルがどのように動くべきかをよく観察させ、ビデオの動きと比較します。
引きつけるセクションで話し合った内容に基づき、プログラムなしで模型が正しく動くかどうかを確認します。

SAMPLE

次のような質問をしてみましょう：

- 模型はビデオと同じように動きましたか？
- どこか正しく動作していない部分はありましたか？

プログラムをテストする

Python プログラミングキャンバスでサンプルプログラムを入力し、実行してみます。

```
1  from hub import port
2  import runloop
3  import motor
4
5  async def main():
6      #脚のモーターを0度の位置へ動かす
7      await motor.run_to_absolute_position(port.F, 0, 600)
8
9      #腕のモーターを0度の位置へ動かす
10     await motor.run_to_absolute_position(port.D, 0, 400)
11
12     #腕と脚の動きを10回繰り返す
13     for x in range(10)
14         await motor.run(port.F, 600)
15         await motor.run(port.D, 400)
16
17 runloop.run(main())
```

生徒にプログラムを実行させます。エラーメッセージが表示されたことに気づくでしょうか？
エラーメッセージはプログラムにバグがあることを示していることを生徒と話し合います。生徒はコードの行を注意深くチェックして、バグを見つけて修正する必要があります。このプロセスは、プログラムのデバッグとして知られています。
生徒たちは、プログラムに何か問題があるかどうかを識別する最初の方法は、エラーメッセージを受け取ることであることを確認します。
エラーメッセージを一緒に見てみましょう：

Traceback (most recent call last):
File“ Break Dancer”, line 13
SyntaxError: invalid syntax

このエラーは、メッセージをチェックする行を生徒に示しています。生徒にこのエラーのトラブルシューティングを試みる時間を与えてください。注意：for ループに関連するコードを読むには、行末にコロン：が必要です。

3. 説明する

なぜ正しく動作していないように見えるのか、生徒とプログラムについて話し合います。
生徒に次のような質問をする：

- プログラムをデバッグするとはどういうことか？
- エラーメッセージはどのように役に立ちますか？
- プログラムを実行する前にモデルをテストすることが重要なのはなぜですか？
- “ 構文エラー”とはどういう意味ですか？

エラーの種類について
生徒と話し合います：

- 構文エラー：コードの文法エラーです。タイプミスや不適切な句読点、括弧の誤用などが原因になります。
- 論理エラー：プログラムが実行されても、期待した結果が得られない場合に発生します。コードは動作しますが、意図した通りには動きません。
- ランタイムエラー：プログラムの実行中に発生するエラーで、プログラムが一時的に動作していても、後でエラーで終了します。

SAMPLE

4. もっと詳しく探究する

エラーの種類をさらに探求する時間を追加します。
通常はバグがないようにプログラムしますが、今回はあえてエラーを探し出す課題に取り組みます。探求する段階でプログラムを変更し、3 種類のエラーすべてを再現できるか確認しましょう。

エラーの例

例 1

`await motor.run_to_absolute_position(port.F, 0, 600)` を `await motor.run_to_absolute_position(port.A, 0, 600)` に変更します。
生徒は実行時エラーメッセージを受け取ります。ポート A にモーターが接続されていないためです。

```
Traceback (most recent call last):
File“ Break Dancer” , line 16, in <module>
File“ Break Dancer” , line 8, in main
OSError: [Errno 19] ENODEV
```

例 2

`for x in range(10)` を `for x in range()` に変更してください。
これは構文エラーの一種で、関数が適切に使用されていないことを示しています。for ループには、ループ回数を示す数値が必要です。

```
Traceback (most recent call last):
File“ Break Dancer” , line 16, in <module>
File“ Break Dancer” , line 12, in main
TypeError: function missing 1 required positional arguments
```

注：予期しない結果を引き起こすために数値を変更することができますが、コンソールはそれをロジックエラーとして表示しない場合もあります。これは、意図された結果がプログラムに設定されていないからです。

5. 評価する

教師の観察

学生たちとプログラムについて話し合います。
生徒に次のような質問をします。

- テストはプログラムのバグを特定するのにどのように役立ちましたか？
- ソフトウェアではなく、ハードウェアのトラブルシューティングが必要だと判断するのはどのようなときですか？
- どのようなエラーが発生し、それをプログラムの修正にどう役立てましたか？

自己評価

生徒にノートに次のことを答えてもらいます。

- バグの特定と修正について、今日は何を学びましたか？

SAMPLE

ユニット 6 レッスン 3
ビートに合わせてダンス?

ビートに合わせてダンス?

生徒はプログラムのデバッグ方法について調べます。

探究への質問

- ソフトウェア・エンジニアはどのようにしてプログラム内の問題を特定するのか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備

- SPIKE プライムハブが充電されていることを確認する (特に Bluetooth で接続する場合)。
- 生徒がブレイクダンサー・ブレイクダウンのレッスンで使用したブレイクダンサーモデルを組み立てていることを確認する。

主な目的

生徒は次のことを行います :

- 問題を特定し、プログラムをデバッグする。

ボキャブラリー

バグ
デバッグ
構文エラー
論理エラー
ランタイム・エラー

1. 引きつける

プログラムをテストしてバグを特定し、修正する方法について話し合います。
生徒たちに、問題やトラブルに見舞われたとき、または期待通りにならなかった経験を思い出させます。
その状況を変えるために役立った手段について話し合います。
次のような質問をしてみてください :

- より良い結果を得るために、事前にどのような対策を講じることができましたか?
- 問題に気づいたとき、どのように対応しましたか?

生徒のトラブルシューティングのアイデアを話し合い、手順をリストアップし、教室に掲示します。

バグを特定するための手順

- 擬似コードまたはフローチャートを作成し、プログラムの計画を立てる。
- # を使用してプログラム内にコメントを追加し、文書化する。
- 想定されるすべての種類のデータでプログラムをテストする。
- 予想外のデータタイプや、範囲外のデータでもプログラムをテストする。

2. 探究する

生徒はブレイクダンサーモデルを使用して、バグの特定と修正を行います。
Python プログラミングキャンバスで新しいプロジェクトを開くように指示し、既存のコードをすべて消去させます。ブレイクダンサー・ブレイクダウンのレッスンを終えている場合は、そのプログラムを修正する形で始めます。ハブを接続します。

動作中のプログラムへの追加

プログラムに機能を追加する際、バグを見つけるための戦略を考えます。

SAMPLE

ブレイクダンサー・ブレイクダウンのレッスンで使ったプログラムを実行するか、それを生徒に提供して、エラーがないことを確認させます。その後、カラーセンサーが色を感知したときだけダンサーが動くようにプログラムを修正するよう指示します。

このような表を使うと便利です：

青	足だけを動かす
緑	腕だけを動かす
黄色	手足をゆっくり動かす
赤	手足を素早く動かす

生徒にプログラムを作成し、動作をテストする時間を与えます。バグがないかどうか確認するため、引きつけのセクションで話し合った手順を思い出させます。

すべてのデバッグ活動をノートに記録するよう指示し、コンソールにエラーメッセージが表示されたらコピーさせます。テスト中に見つけたエラーもノートに書き写してください。

```
1  from hub import port, light_matrix
2  import runloop
3  import motor, color_sensor, color
4
5  async def main():
6      #color_valueにカラーセンサーから色の値を取得
7      color_value = color_sensor.color(port.B)
8
9      while True:
10         await runloop.until(lambda: color_sensor.color(port.B) not in(-1, color_value))
11         color_value = color_sensor.color(port.B)
12
13         #もし青ならば腕のモーターを動かす
14         if color_value == color.BLUE:
15             light_matrix.show_image(light_matrix.IMAGE_HAPPY)
16             await motor.run_for_degrees(port.D, 360, 600)
17         #緑ならば脚を動かす
18         elif color_value == color.GREEN:
19             light_matrix.show_image(3)
20             await motor.run_for_degrees(port.F, 360, 600)
21         #黄色ならば腕と脚を同時にゆっくり動かす
22         elif color_value == color.YELLOW:
23             light_matrix.show_image(light_matrix.IMAGE_HAPPY)
24             motor.run_for_degrees(port.D, 360, 200)
25             motor.run_for_degrees(port.F, 360, 200)
26         #赤ならば腕と脚を同時に早く動かす
27         elif color_value == color.YELLOW:
28             light_matrix.show_image(light_matrix.IMAGE_HAPPY)
29             motor.run_for_degrees(port.D, 360, 600)
30             motor.run_for_degrees(port.F, 360, 800)
31         #そのほかの場合
32         else:
33             light_matrix.show_image(light_matrix.IMAGE_SAD)
34
35 runloop.run(main())
```

3. 説明する

生徒とプログラムについて話し合います。

次のような質問をします：

- プログラムでどのようなバグを見つけましたか？
- コンソールにエラーメッセージが表示されましたか？表示された場合、それはデバッグにどのように役立ちましたか？
- 期待される値と予想外の値について、どのようにテストを行いましたか？
- モデル自体に問題があった可能性はありますか？その理由は？

4. より詳しく探究する

カラーセンサーで動きを制御し、ダンサーがビートに合わせて動くようなプログラムを作成します。
生徒たちに、自分のプログラムを利用して、ダンサーがビートに合わせて踊る新しい動きを作らせます。曲を流したり、ビートをロずさんでも構いません。ダンサーはビートに合わせて速く、または遅く動くように調整します。
プログラムの調整が必要かどうかを確認する時間を与え、その後、ビートに合わせた動きを作り上げます。最終的なダンスをクラスで発表し、すべてのモデルで同時にダンスパーティを行うのも面白いでしょう。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- テストは、どのようにしてプログラムのバグ特定に役立ちましたか？
- 期待される値や予期しない値でテストを行うのはなぜですか？
- どのようなエラーが発生しましたか？そのエラーをどのように利用してプログラムを修正しましたか？

自己評価

生徒にノートに次のことを答えてもらいます。

- バグの特定と修正について、今日は何を学びましたか？

SAMPLE

ユニット 6 レッスン 4
トラブルのテスト

トラブルのテスト

生徒は、ハードウェアの問題をトラブルシューティングする方法を調べます。

主な目的

生徒は次のことを行います：

- ハードウェアの問題を特定し、修理する。

ボキャブラリー

デバッグ
擬似コード
CNC

探究への質問

エンジニアは、設計内のハードウェアとソフトウェアの問題をどのように特定し、修理するのか。

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート
- ペンまたはマーカー（チームにつき 1 本）
- ブロークン・レッスン用のプリントしたワークシート（チームにつき 1 枚）

準備

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth を使用する場合）。
- モデルのトラブルシューティングの参考資料は、https://education.lego.com/en-us/lessons/prime-invention-squad/broken#Planitem2 にあります。

1. 引きつける

どのような機械が正確な切断を繰り返し行うのか、生徒と話し合います。CNC（コンピュータ数値制御）機械は、同じ切断を繰り返すようにプログラムされています。例えば、レーザーカッターは、プラスチック、木材、段ボールを正確な形に切断します。

- 生徒たちに、CNC 機械がどのようなもので、どのように使われているかを調べさせます。
- CNC 機械で作られていると思われる物の名前を挙げさせます。

https://education.lego.com/en-us/lessons/prime-invention-squad/broken#building-tips このビデオを見て、生徒が製作する CNC マシンがどのような機能を持つべきかを理解します。
このビデオの機械のどこが悪いか、生徒と話し合います。生徒に次のような質問をします：

- 何かが正しく動作していないとき、どのようにしてわかりますか？
- 何かが壊れたときの最初の反応は？
- 問題がどこにあるのか、どうすればわかるか？

2. 探究する

生徒は、設計上の問題やハードウェアの問題のトラブルシューティングを調査するために、CNC マシンを製作します。
生徒を SPIKE アプリの BUILD セクションに誘導します。ここで生徒は、CNC マシンのモデルの組み立て手順にアクセスできます。生徒にモデルを作るように指示します。組み立て説明書は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。
Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。生徒に、すでにプログラミング領域にあるコードをすべて消去するように指示します。生徒はハブを接続します。

SAMPLE

モデルのテスト

生徒は、サンプルプログラムを試すときにハードウェアの問題があることを確認します。

このサンプルプログラムを見直して、グループ全員で CNC マシンを動かします。グループでコードを話し合い、コードの潜在的なエラーを特定します。生徒は、プログラムにエラーがないように見えることに注意します。

生徒に、このプログラムをプログラミング・キャンバスに入力するように指示します。生徒にプログラムを実行させ、エラーがないかテストさせます。

```
from hub import port
import runloop
import motor

async def main():

    while True:
        #紙を送る
        await runloop.until(right_button_pressed)
        await motor.run_for_degrees(port.C, 1000, 500)

        #正方形と長方形に切る
        #始点に移動
        await runloop.until(left_button_pressed)
        await motor.run_for_degrees(port.A, 1500, -500)

        #正方形に切る
        await motor.run_for_degrees(port.A, 400, 500)
        await motor.run_for_degrees(port.C, 575, 500)
        await motor.run_for_degrees(port.A, -400, 500)
        await motor.run_for_degrees(port.C, -575, 500)

        #長方形に切る
        await motor.run_for_degrees(port.A, -60, 500)
        await motor.run_for_degrees(port.C, -800, 500)
        await motor.run_for_degrees(port.A, 400, 500)
        await motor.run_for_degrees(port.C, 1600, 500)
        await motor.run_for_degrees(port.A, -400, 500)
        await motor.run_for_degrees(port.C, -800, 500)

runloop.run(main())
```

生徒にプログラムを実行する時間を与えます。生徒は機械が正しく動作していないことに気づきます。機械は部品を「切る」(描く)のを助けるはずです。

問題がハードウェア / モデルにあるのか、プログラムにあるのか、グループで話し合います。

生徒に次のような質問をします：

- 何が動かないのか？
- モデルはどのように動きますか？
- モデルは正しく動いていますか？
- コンソールにエラーが表示されたり、プログラムが動作していないことを示すものが表示されたりしませんでしたか？

生徒は、プログラムではなくモデルに問題があることを認識する必要があります。

設計上の問題を特定し、トラブルシューティングをする

協力して問題を特定します。マシンを見て、それがどのように動作するかを観察し、期待される動作と比べます。発見した問題をノートまたはチャート紙に書かせます。以下の手順を使用するよう生徒に促します：

- 問題を特定する。
- よく観察する。
- モデルがどのように動作するかを観察する。
- 解決策をブレインストーミングする。
- 一つ変更を加え、モデルをテストする（プログラムを実行する）。役に立ちましたか？
- 必要に応じて、他のチームに助けを求めてもよい。

必要に応じて生徒を促し、問題の特定を始める。生徒に次のような質問をします：

- 紙は思うように動いていますか？
- 鉛筆は、切り口を表現するのに必要な印をつけることができますか？
- 模型は安定していますか？

注意：特定して修正すべき問題は 4 つあります。

1. 給紙ホイールが欠けていて、Y 軸が正しく動作しない。
2. CNC マシンの上部が底部に正しく取り付けられていない。
3. 給紙ギアが逆になっており、用紙が CNC に入るのが早すぎる。
4. ペンシルキャリッジが固定されておらず、X 軸が正しく動作していない。

注：Broken レッスンは、モデルのどこが悪いかのヒントを示しています。
https://education.lego.com/en-us/lessons/prime-invention-squad/broken#Planitem0
グループで調査する問題を 1 つ選びます。紙が速く動きすぎることから始めることを考えます。問題を解決するためのアイデアをみんなで出し合います。プログラムを変更せずに、動きを遅くするにはどうしたらよいでしょうか。生徒に、歯車がどのように動いているかを見て、紙の動きを遅くするために歯車をどのように変えるかを考えるように促します。

3. 説明する

生徒とプログラムについて話し合います。
生徒に次のような質問をします：

- プログラムを実行したとき、CNC マシンについて気づいたことは何ですか？
- どうしてハードウェアの問題だとわかるのですか？
- なぜ設計のテストが重要なのですか？
- 問題がプログラムではなくモデルにあると、どのようにして判断しましたか？
- 機械のどの部分が動作していないように見えますか？
- エンジニアとして、マシンを修理するための最初のステップは何ですか？（問題を特定する。複数の問題が発見された場合は、どの問題を最初に修理するかを決める）

4. より詳しく探究する

生徒がモデルのトラブルシューティングを終えるまで、さらに時間を与えます。
生徒に、残りの問題に対する解決策をブレインストーミングで考えさせます。ハードウェアのトラブルシューティングに役立てるため、必要に応じて、生徒がプログラムを実行したモデルを再テストできるようにします。

SAMPLE

問題を特定した後、異なるグループに問題の解決に取り組むよう求めます。各グループは、CNC マシンが正常に動作するように、修正した内容を他のグループと共有します。生徒たちは、自分たちのモデルが正しく動作していることを再テストします。
モデルが正しく動作した後、すべての解決策をグループで共有します。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- テストは、トラブルシューティングが必要な問題の特定にどのように役立ちますか？
- ソフトウェアではなく、ハードウェアのトラブルシューティングの必要性をどのように判断できますか？
- ハードウェアの問題をトラブルシューティングする方法にはどのようなものがありますか？

自己評価

生徒にノートに次のことを答えてもらいます。

- ハードウェアの問題のトラブルシューティングについて、今日何を学びましたか？

SAMPLE

ユニット 6 レッスン 5

デバッグネーター

デバッグネーター

新しいモデルを設計しプログラミングすることで、ハードウェアやプログラムの問題のトラブルシューティングを練習します。

主な目的

生徒は次のことを行います：

- ソフトウェアの問題をデバッグする。

ボキャブラリー

デバッグ
擬似コード
CNC

探究への質問

- 新しいモデルを設計するとき、どのようなデバッグ技術を使用できますか？

必要な機材

- SPIKE プライムセット
- SPIKE アプリがインストールされたデバイス
- ノート

準備

- SPIKE プライムハブが充電されていることを確認する（特に Bluetooth で接続する場合）。

1. 引きつける

デバッグネーターとは何かについて、生徒同士で話し合います。
生徒に、デバッグネーターに必要なことを 3 つ以上思いつくように指示します。この話し合いの間、生徒はできるだけ多くのアイデアを集め、ノートに記録します。
必要に応じて、次のような質問で生徒を促します：

- あなたのモデルは何かを感知する必要がありますか？
- あなたのモデルはどのように動く必要がありますか？
- 何かコンソールを利用する必要がありますか？

生徒が話し合いから得たアイデアを共有できるようにします。その後、生徒たちは、自分のモデルに含める必要のある最後の 3 つの主な基準を決めます。

2. 探究する

生徒が設定した基準を満たすデバッグネーターである新しいモデルの設計、製作、プログラム、テスト、およびトラブルシューティングに挑戦します。
生徒は、引きつけるセクションで決定された 3 つの主な基準を含むように注意しながら、プロトタイプを作成します。生徒は、モデルの設計と製作中に、モデルが意図したとおりに動くかどうかをテストすることで、トラブルシューティングの方法を練習する必要があります。
生徒はモデルをプログラミングする必要があります。プログラムを作成するとき、生徒は次のことを行います。

- 最初に擬似コードプログラムを書いて、プログラムの意図する結果を示します。
- "#" のコードコメントを使用してプログラムを文書化します。
- コンソールでエラーメッセージを確認しながら、プログラムをテストします。
- 予想される結果と予想されない結果またはデータを使用して、プログラムをテストします。

生徒がモデルを設計し、構築し、プログラムする時間を与えます。生徒は、遭遇した問題と、その問題をどのように修正またはトラブルシューティングしたかを文書化します。

SAMPLE

3. 説明する

生徒が自分の作品を共有できるようにします。生徒のモデルやプログラムについて一緒に話し合います。

生徒に次のような質問をします：

- あなたのデバッグネーターは何をしますか？
- あなたのモデルには、どのような 3 つの条件がありましたか？これらの期待に応えるために、どのように何かを作りましたか？
- 自分のモデルをどのようにプログラムしましたか？生徒たちにプログラムを共有するよう求めます。
- どのような問題がありましたか？どこにバグがありましたか？どのように修正しましたか？

4. 詳しく説明する

生徒にモデルとプログラムを完成させます。

生徒がモデルやプログラムを完成させるために、さらに時間を与えます。チーム間の協力とアイデアの共有を促します。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。

生徒に次のような質問をしてください。

- デバッグネーターを作成する際に、どのような問題にぶつかりましたか？
- モデルやプログラムにエラーがないか、どのようにテストしましたか？見つかったエラーをどのようにトラブルシューティングしましたか？
- 発生した問題がモデルによるものかプログラムによるものかをどのように判断しましたか？

自己評価

ノートに次のように答えさせます：

- デバッグネーターを作成する際に、どのような問題に直面していたかを生徒に考えさせます。

SAMPLE

SAMPLE

Worksheets for Student
生徒用ワークシート

SAMPLE

生徒用ワークシート

Unit 4 Doing Reps with Loop: ループを使った繰り返し処理

Lesson 1 P75-76
Lesson 2 P77-78
Lesson 3 P79-80
Lesson 4 P81
Lesson 5 P83-84

Unit 5 Playing Games Simple Conditions:
簡単な条件でゲームをする

Lesson 1 P85-86
Lesson 2 P87-88
Lesson 3 P89-90
Lesson 4 P91-92
Lesson 5 P93-94
Lesson 6 P95-96
Lesson 7 P97-98

Unit 6 Testing for Bugs:
Troubleshooting Hardware and Software:
バグのテスト：ハードウェア&ソフトウェアのトラブルシュート

Lesson 2 P99-100
Lesson 3 P101-102
Lesson 4 P103-104

SAMPLE

UNIT4Lesson 1
Warm Up Loop with Leo

次のぷらおグラムを完成させて、LEO が一回腹筋をするようにしてみましょう。

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    # 一回腹筋をする
    await motor_pair.move_for_time(                    )
    await runloop.sleep_ms(2000)
    await motor_pair.move_for_time(                    )
    await runloop.sleep_ms(2000)
runloop.run(main())
```

LEO が 5 回腹筋するようにするためにはどうすればよいか、アイデアを書きましょう。

LEO が 5 回腹筋するように、下線部に 1 行プログラムを加えましょう。

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    #5 回腹筋をする (5 回繰り返す )

    ____________________
        await motor_pair.move_for_time(                    )
        await runloop.sleep_ms(2000)
        await motor_pair.move_for_time(                    )
        await runloop.sleep_ms(2000)
runloop.run(main())
```

SAMPLE

間違い探し

下のプログラムには間違っているところがあります、どこが間違えているでしょうか？また実行するとどうなるでしょうか？

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    #5 回腹筋をする (5 回繰り返す )
    for index in range(5):
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500 )
    await runloop.sleep_ms(2000)
    await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500 )
    await runloop.sleep_ms(2000)
runloop.run(main())
```

プログラムを改造しよう

＊ 腹筋 5 回を 1 セットとして、3 セット実行するようにプログラムしてみてください。

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.F)

    #5 回 1 セットを 3 回繰り返す

        #5 回腹筋をする (5 回繰り返す )
        for index in range(5):
            await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500 )
            await runloop.sleep_ms(2000)
            await motor_pair.move_for_time(motor_pair.PAIR_1, 500, 0, velocity=500 )
            await runloop.sleep_ms(2000)
runloop.run(main())
```

SAMPLE

UNIT4Lesson 2
Leo と回数を数える

前のレッスンのプログラムを使って、腹筋の回数を数えましょう。
プログラムの中で数を数えるためには、「変数」を使います。下のプログラムの中に「変数」が使われていますがどれでしょう？

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    #5 回腹筋をする (5 回繰り返す )
    for index in range(5):
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)
runloop.run(main())
```

次の一文をプログラムに加えて、腹筋を一回するごとにコンソールに回数をプリントするようにしましょう。
print(index+1) これをどこに入れればよいでしょう？

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    #5 回腹筋をする (5 回繰り返す )
    for index in range(5):
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)

runloop.run(main())
```

SAMPLE

次は 5 からカウントダウンするようにしましょう。
どうすればよいでしょう？

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    #5 回腹筋をする (5 回繰り返す )
    for index in range(5):
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)

runloop.run(main())
```

ライトマトリクスに変数の値を表示するには次のようにします
light_matrix.write(str(index))
この一文を適切なところに追加して、腹筋を一回するごとにライトマトリクスに回数が表示されつようにしましょう。
どこに追加すればよいでしょう？

```
from hub import port
import runloop
import motor_pair
async def main():
    # ポート B とポート F のモーターをペアにする


    #5 回腹筋をする (5 回繰り返す )
    for index in range(5):
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)
        await motor_pair.move_for_time( motor_pair.PAIR_1, 500, 0, velocity=500)
        await runloop.sleep_ms(2000)

runloop.run(main())
```

SAMPLE

ユニット 4　レッスン 3
コーチとのダンスループ

次のプログラムを実行すると、どうなるか説明しましょう。

```
from hub import port
import runloop
import motor

async def main():
    # 速度 300 で 1/4 秒 (250 ミリ秒 ) 動かす
    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, -300)

    await motor.run_for_time(port.B, 250, 300)
    await motor.run_for_time(port.F, 250, 300)

    await motor.run_for_time(port.B, 250, -300)
    await motor.run_for_time(port.F, 250, -300)
runloop.run(main())
```

このプログラムをもっと効率的に記述するにはどうすればよいでしょうか？

次のプログラムの下線部を埋めてください

```
from hub import port
import runloop
import motor

async def main():

    ____________________
    await motor.run_for_time(port.B,250,300)
    await motor.run_for_time(port.F,250,300)

    await motor.run_for_time(port.B,250,-300)
    await motor.run_for_time(port.F,250,-300)
runloop.run(main())
```

デバッグアクティビティ 1

次のプログラムの間違いを直してください。

```
from hub import port
import runloop
import motor

async def main():
    for index in range():
    await motor.run_for_time(port.B,250,300)
```

SAMPLE

```
    await motor.run_for_time(port.F,250,300)

    await motor.run_for_time(port.B, 250,-300)
    await motor.run_for_time(port.F,250,-300)

runloop.run(main())
```

デバッグアクティビティ 2

次のプログラムの間違いを直してください。

```
from hub import port
import runloop
import motor
async def main():
    for index in range():
    await motor.run_for_time(port.B,250,300)
    await motor.run_for_time(port.F,250,300)
    await motor.run_for_time(port.B, 250,-300)
    await motor.run_for_time(port.F,250,-300)
runloop.run(main())
```

デバッグアクティビティ 3

次のプログラムの間違いを直してください。

```
from hub import port
import runloop
import motor
async def main():
    for index in range(10):
        await motor.run_for_time(250,300)
        await motor.run_for_time(port.F,250,300)
        await motor.run_for_time(port.B, 250,-300)
        await motor.run_for_time(port.F,250,-300)
runloop.run(main())
```

デバッグアクティビティ 4

次のプログラムの間違いを直してください。

```
from hub import port
import runloop
import motor
async def main():
    for index in range(10):
    await motor.run_for_time(port.B,250,300)
    await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B,250,-300)
        await motor.run_for_time(port.F,250,-300)
runloop.run(main())
```

SAMPLE

ユニット 4　レッスン 4
ヨガの条件設定

次のプログラムをコーディングして動かしてみましょう。

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    # 変数 count が 5 より小さいときモーターを動かす
    while count < 5:
        await motor.run_for_time(port.B,250,300)
        await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B,250,-300)
        await motor.run_for_time(port.F,250,-300)

runloop.run(main())
```

count ＝ 2 の値を変えてみましょう。例えば、10 や 5 に変えるとどうなりますか？
次のプログラムをコーディングして動かしてみましょう。

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    # 変数 count が 5 より小さいときモーターを動かす
    while count < 5:
        await motor.run_for_time(port.B,250,300)
        await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B,250,-300)
        await motor.run_for_time(port.F,250,-300)

        count+=1

runloop.run(main())
```

どのように動いたか説明してください。

SAMPLE

SAMPLE

ユニット 4　レッスン 5
無限の動き

コーチが無限に動き続けるようにするには次のプログラムをどのように変えればよいか考えましょう。

```
from hub import port
import runloop
import motor

async def main():
    count = 2
    # 変数 count が 5 より小さいときモーターを動かす
    while count < 5:
        await motor.run_for_time(port.B,250,300)
        await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B,250,-300)
        await motor.run_for_time(port.F,250,-300)

        count+=1

runloop.run(main())
```

コーチが無限に動き続けるようにするため次のプログラムの下線部を埋めてください。

```
from hub import port
import runloop
import motor

async def main():
    while ____ :
        await motor.run_for_time(port.B,250,300)
        await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B, 250,-300)
        await motor.run_for_time(port.F,250,-300)

runloop.run(main())
```

SAMPLE

フォースセンサーを押すと動き出すようにプログラムを変更しましょう。

```
from hub import port
import runloop
import motor
import force_sensor

def is_pressed():
    return ____________________________

async def main():
    while True:
        # フォースセンサーを押すと動き出す
        ____________________________

        await motor.run_for_time(port.B, 250, 300)
        await motor.run_for_time(port.F,250,300)

        await motor.run_for_time(port.B,250,-300)
        await motor.run_for_time(port.F,250,-300)

runloop.run(main())
```

SAMPLE

ユニット 5　レッスン 1
傾きで動きを制御する

ハブをタップするとライトマトリクスに何か表示するために、まずは疑似プログラムを書きましょう

次のプログラムをプログラミングキャンパスにコーディングして動かしてみましょう。

```
from hub import light_matrix, motion_sensor
import runloop

def if_tapped():
    # ハブがタップされたとき、ジェスチャー TAPPED を返す
    return motion_sensor.gesture() == motion_sensor.TAPPED

async def main():
    # ハブがタップされるまで待つ
    await runloop.until(if_tapped)
    # ハブがタップさたら「悲しい顔」を表示する
    light_matrix.show_image(light_matrix.IMAGE_SAD)
    # コンソールに「Stop please（お願い、やめて）」を表示する
    print("Stop please")

runloop.run(main())
```

モーションセンサーのジェスチャーオプションについて motion_sensor.TAPPED 以外のオプションについてナレッジベースで調べてみましょう。

SAMPLE

次のプログラムをプログラミングキャンパスにコーディングして動かしてみましょう。
何をどうすると何が起こるか、予想してください。

```
from hub import light_matrix, motion_sensor
from app import sound
import runloop

def face_up():
    # ハブが上向き (USB ポートが上 ) のとき、True を返す
    return motion_sensor.up_face()==motion_sensor.BACK

async def main():
    # ハブが上向きになるまで待つ
    await runloop.until(face_up)

    # ハブが上向きになったら「 矢印 ( 上向き )」を表示する。
    light_matrix.show_image(light_matrix.IMAGE_ARROW_N)

    # コンソールに「 I am awake ( 起きました )」を表示する
    print("I am awake")
    # サウンド「 Applause 1」を再生する
    await sound.play('Applause 1')

runloop.run(main())
```

モーションセンサーの姿勢 (方向) について motion_sensor.BACK 以外のオプションについてナレッジベースで調べてみましょう。
それぞれの姿勢の時にその方向の矢印を表示するようにプログラムを変更してみましょう。

SAMPLE

ユニット 5　レッスン 2

クローマシン

組み立てたロボットハンドを動かすための条件、動かすためのプログラムを考えてプログラムしましょう。

例えば：

左ボタンを押したらハンドを開く、左ボタンを離したらハンドを閉じる

下のプログアムの空欄（下線部）を埋めて正しく動くようにしましょう。

```
from hub import port, button
import runloop
import motor

def pressed():
    # ハブの左ボタンが押されたかどうかを返す
    return button.pressed (__________)

def released():
    # ハブの左ボタンが離されたかどうかを返す
    return not button.pressed (__________)

async def main():
    # 左ボタンが押されるまでまつ
    await runloop.__________

    # モーターを回転する ( ハンドを開く )
    motor.run(port.F, -750)

    # 左ボタンが離されるまでまつ
    await runloop.__________

    # モーターを回転する ( ハンドを閉じる )
    motor.run(port.F, 750)

runloop.run(main())
```

SAMPLE

ブロック渡しゲームで何度も開いたり閉じたりさせるためにはどうしたらよいか考えてプログラムを作りましょう。
プログラム内の空欄（下線部）を埋めて動くようにしましょう。

```
from hub import port, button
import runloop
import motor

def pressed():
    # ハブの左ボタンが押されたかどうかを返す
    return button.pressed (button.LEFT)

def released():
    # ハブの左ボタンが離されたかどうかを返す
    return not button.pressed (button.LEFT)

async def main():
    ____________
        # 左ボタンが押されるまでまつ
        await runloop.until(pressed)
        # モーターを回転する ( ハンドを開く )
        motor.run(port.F, -750)
        # 左ボタンが離されるまでまつ
        await runloop.until (released)
        # モーターを回転する ( ハンドを閉じる )
        motor.run(port.F, 750)

runloop.run(main())
```

SAMPLE

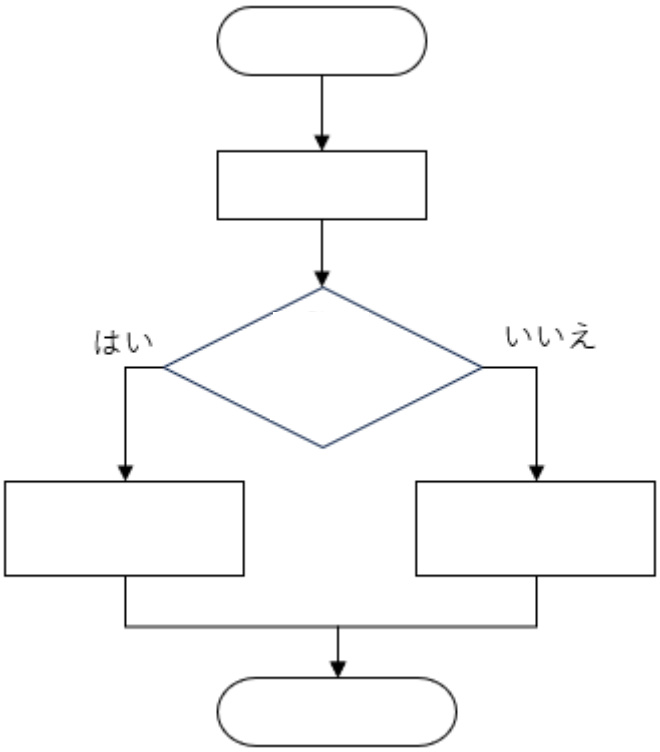
ユニット５　レッスン３
ゲームの決断を図表化する

フローチャートで使用される図形とその意味の表を完成させましょう。

記号	意味

次の行動をフローチャートとして作成してみましょう。

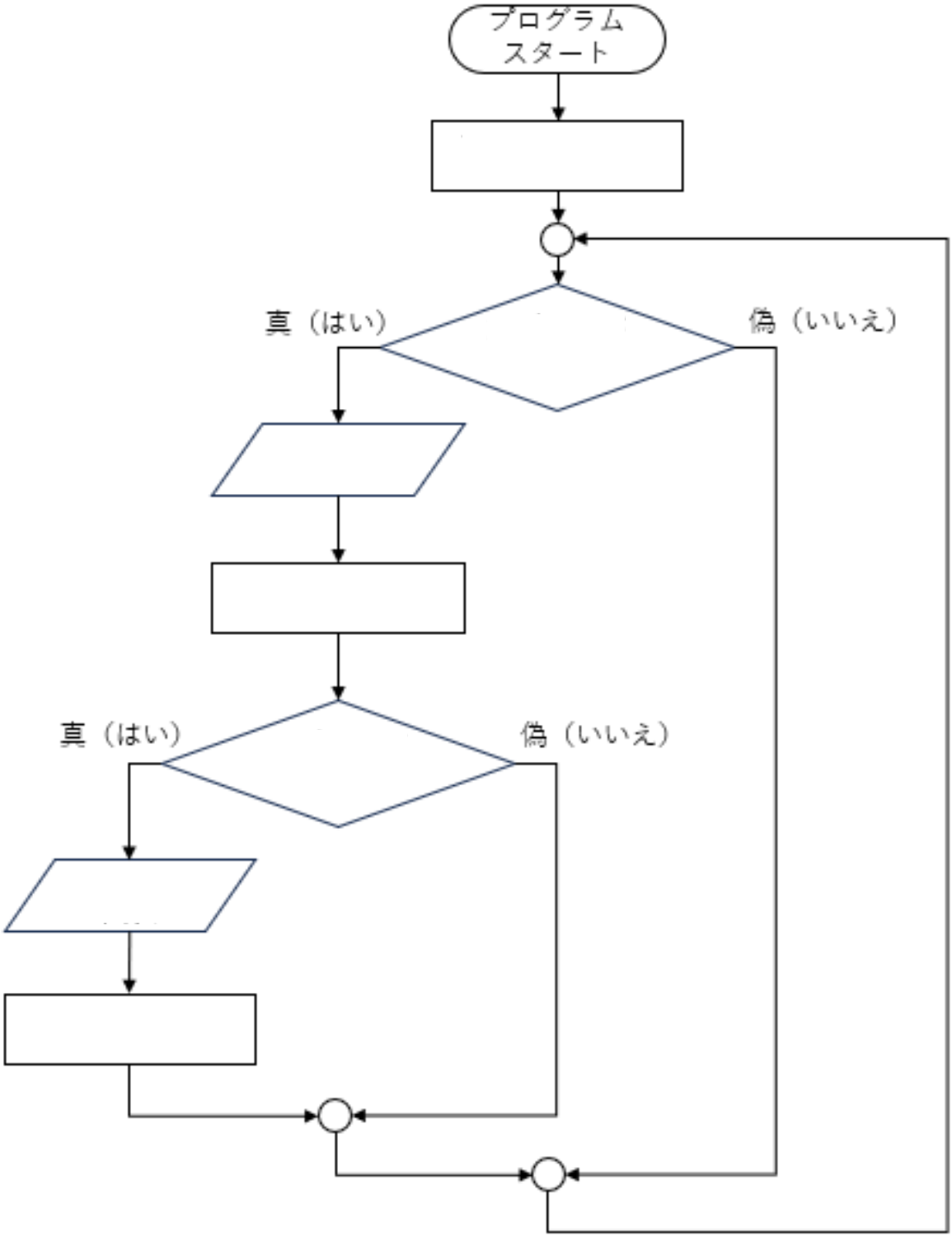
- 朝起きた
- 学校に行く日
- お中が空いている？
- はい→ちゃんと朝食を食べる
- いいえ→軽くスナックを食べる
- 学校へ行く



SAMPLE

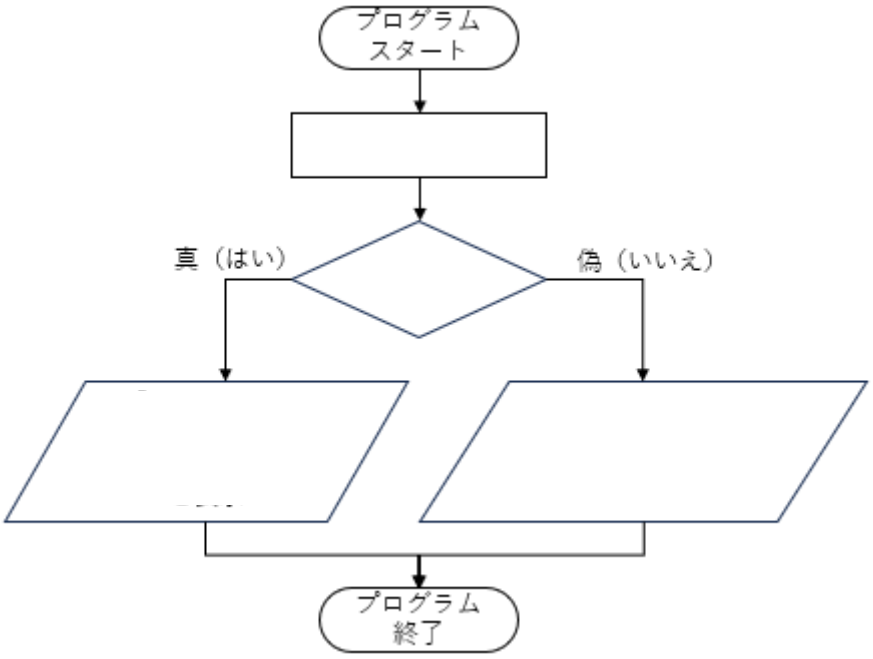
ブロック積み上げゲーム

ロボットハンドを動かすプログラムをフローチャートにしてみましょう。



ユニット 5　レッスン 4
色当てゲーム

犬の Kiki が青色のブロックを見たら鳥の音を出し、青色ではなかったら犬の鳴き声を出すプログラムを作ります。
そのために、フローチャートを作成しましょう。



フローチャートからプログラムを作ります。
次のプログラムの空欄（下線部）を埋めてプログラムを完成させましょう。

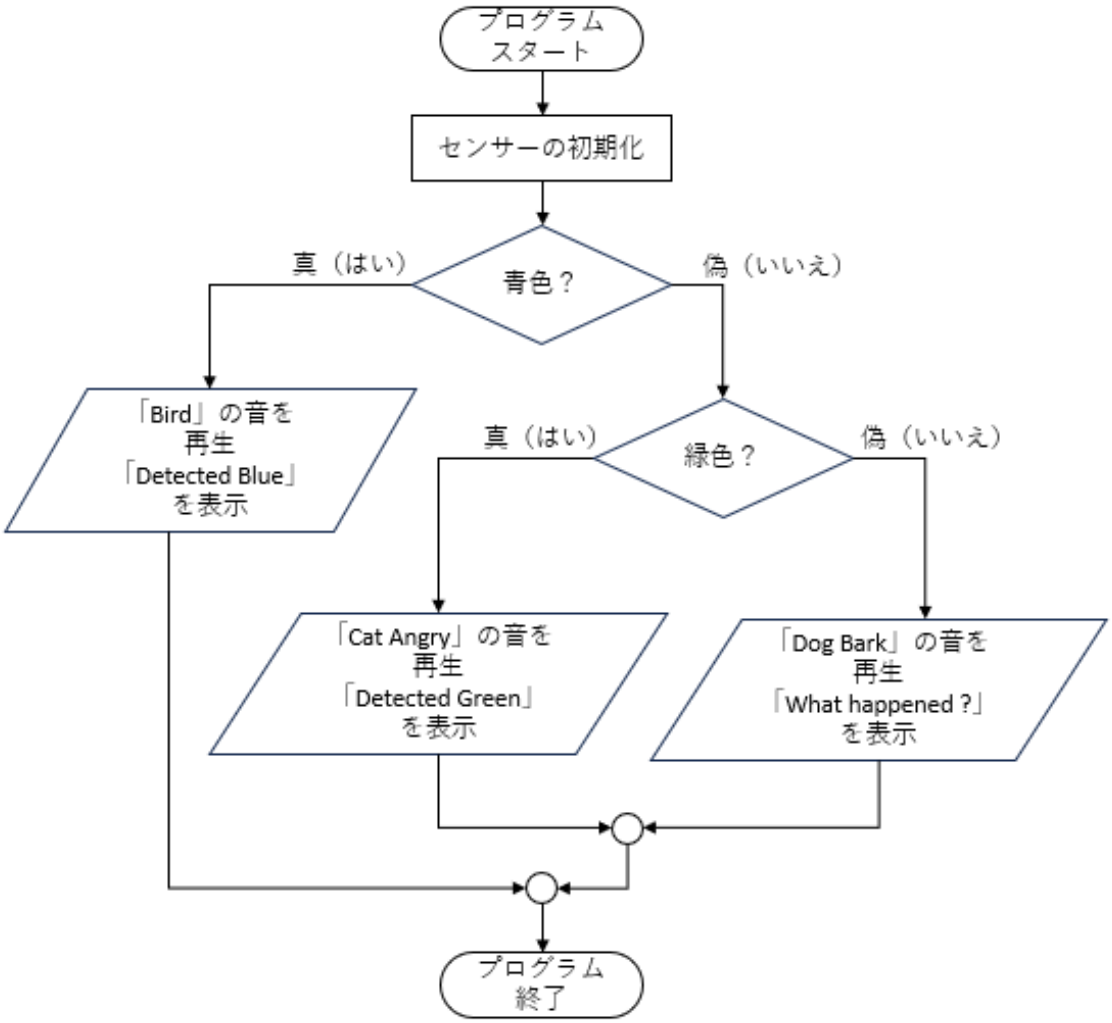
```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
    # もしカラーセンサーが Blue（青）を検出したら、鳥の音を再生する
    ____________________________________
        await sound.play('Bird')
        print('Blue Detected')
        # 青を検知しなかったら（青以外だったら）、犬の鳴き声 1 を再生する
    ____
        await sound.play('Dog Bark 1')
        print('What happened')

runloop.run(main())
```

SAMPLE

次のフローチャートを見てプログラムの空欄を埋めてプログラムを完成させましょう。




```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
    # もしカラーセンサーが Blue ( 青 ) を検出したら、鳥の音を再生する
    ____________________________
    await sound.play('Bird')
    print('Blue Detected')
    # または緑を検知したら、猫の怒り声 (Cat Angry) を再生する。
    ____________________________
    await sound.play('Cat Angry')
    print('Green Detected')
    # 青または緑以外だったら )、犬の鳴き声 I を再生する
    ____
    await sound.play('Dog Bark I')
    print('What happened')

runloop.run(main())
```

SAMPLE

ユニット５　レッスン５
推理ゲーム

色当てゲームのプログラムを、セットの中に入っている２×４ブロックの全ての色に対応させたプログラムに変更しましょう。
プログラムを作る前に、フローチャートを作成しましょう。

SAMPLE

デバッグアクティビティ１

次のプログラムのバグを修正してください

```
from hub import port
import runloop
import color_sensor
from app import sound
import color

async def main():
    # もしカラーセンサーが Blue ( 青 ) を検出したら、鳥の音を再生する
    if color_sensor.color(port.A)==color.BLUE:
        await sound.play('Bird')
        print('Blue Detected')
        # または緑を検知したら、犬の鳴き声 1 を再生する
    elif color_sensor.color(port.A)==color.GREEN:
        await sound.play('Dog Bark 1')
        print('Green Detected')
        # 青または緑以外だったら、猫の怒り声 (Cat Angry) を再生する
    else
        await sound.play('Cat Angry')
        print('What happend?')
runloop.run(main())
```

デバッグアクティビティ 2

次のプログラムのバグを修正してください

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
  # もしカラーセンサーが Blue ( 青 ) を検出したら、鳥の音を再生する
  if color_sensor.color(port.A)==color.BLUE:
  await sound.play('Bird')
  print('Blue Detected')
  # または緑を検知したら、犬の鳴き声 I を再生する
  elif color_sensor.color(port.A)==color.GREEN:
    await sound.play('Dog Bark I')
    print('Green Detected')
  # 青または緑以外だったら、猫の怒り声 (Cat Angry) を再生する
  else:
    await sound.play('Cat Angry')
    print('What happend?')

runloop.run(main())
```

消えたコード

次のプログラムを修正してください

```
from hub import port
import runloop
import color_sensor
import color
from app import sound

async def main():
  # もしカラーセンサーが Blue ( 青 ) を検出したら、鳥の音を再生する
  if color_sensor.color(port.A)==color.BLUE:
  await sound.play('Bird')
  print('Blue Detected')
  # または緑を検知したら、犬の鳴き声 I を再生する
  elif


  # 青または緑以外だったら、猫の怒り声 (Cat Angry) を再生する
  else


runloop.run(main())
```

SAMPLE

ユニット 5　レッスン 6

スコア！

ゴールを決めろ！

ゴールへ得点を決めるためにあなたのプレーヤーをどう動かしたらよいか考え、そのためのプログラムを作りましょう。
コーディングする前に、フローチャートを作ってプログラムの流れをよく考えましょう。

```
from hub import port
import runloop
import motor

async def main():
    # ポート A のモーターを 720 度 / 秒で、180 度の位置へ動かす
    # これは、アームをスイングする準備で振り上げる動作


    # ポート A のモーターを 720 度 / 秒で、360 度動かす


runloop.run(main())
```

得点を管理

以下のことができるようにコードを追加しましょう。フローチャートを作ってプログラムの流れを考えましょう。

- ゴールが決まったかどうかを示すために、画像やメッセージ、音を使ってハブをプログラムします。ゴールが決まった場合は 1 つの画像、決まらなかった場合は別の画像や音を選択します。
- コンソールに得点を表示し、ゴールが決まれば 1 を加え、決まらなければ 0 を表示します。

SAMPLE

```
from hub import port, light_matrix, button
import runloop
import motor

def right_released():
    return not button.pressed (button.RIGHT)

def left_released():
    return not button.pressed (button.LEFT)

async def main():
    # ボートAのモーターを720度/秒で、180度の位置へ動かす
    # これは、アームをスイングする準備で振り上げる動作
    motor.run_to_absolute_position (port.A, 180, 720, direction=motor.CLOCKWISE)
    # ボートAのモーターを720度/秒で、360度動かす
    motor.run_for_degrees (port.A, 360, 720)
    # 得点を0にセット
    score = 0
    # スコアを示すループ条件文
    while True:
        if button.pressed (                    ):
            await runloop.until(                  )
            light_matrix.show_image(                              )

            print(          )
        elif button.pressed (                  ):
            await runloop.until(                  )
            light_matrix.show_image(                              )
            print(     )

runloop.run(main())
```

SAMPLE

ユニット５　レッスン７
ゲームタイム

どのようなゲームを作るかアイデアを書きだしましょう。

どのような形にするか設計図を書きましょう。

どのようなプログラムにするか、フローチャートを作ってコーディングしましょう。

SAMPLE

SAMPLE

ユニット 6　レッスン 2
ブレイクダンサー・ブレイクダウン

手でモーターを回してブレイクダンサーの動きを確認しましょう。
動きにどこかおかしなところがあれば記入しましょう。

次のプログラムをプログラミングキャンバスに入力し、実行してみてください。
コンソールを開き、エラーが出ていないか確認をすること。エラーが出ている場合は、バグを修正してください。

```
from hub import port
import runloop
import motor

async def main():
    # 脚のモーターを 0 度の位置へ動かす
    await motor.run to absolute position (port.F, 0, 600)

    # 腕のモーターをの 0 度の位置へ動かす
    await motor.run to absolute position (port.D, 0, 400)

    # 腕と脚の動きを 10 回繰り返す
    for x in range(10)
        await motor.run(port.F, 600)
        await motor.run(port.D, 408)

runloop.run(main())
```

SAMPLE

次のプログラムをプログラミングキャンバスに入力し、実行してみてください。
コンソールを開き、エラーが出ていないか確認をすること。エラーが出ている場合は、バグを修正してください。
また、どのようなエラーが出ているか確認しましょう。

```
from hub import port
import runloop
import motor

async def main():
    # 脚のモーターを 0 度の位置へ動かす
    await motor.run to absolute position (port.F, 0, 600)

    # 腕のモーターをの 0 度の位置へ動かす
    await motor.run to absolute position (port.A, 0, 400)

    # 腕と脚の動きを 10 回繰り返す
    for x in range(10)
        await motor.run(port.F, 600)
        await motor.run(port.D, 408)

runloop.run(main())
```

次のプログラムをプログラミングキャンバスに入力し、実行してみてください。
コンソールを開き、エラーが出ていないか確認をすること。エラーが出ている場合は、バグを修正してください。
また、どのようなエラーが出ているか確認しましょう。

```
from hub import port
import runloop
import motor

async def main():
    # 脚のモーターを 0 度の位置へ動かす
    await motor.run to absolute position (port.F, 0, 600)

    # 腕のモーターをの 0 度の位置へ動かす
    await motor.run to absolute position (port.D, 0, 400)

    # 腕と脚の動きを 10 回繰り返す
    for x in range( )
        await motor.run(port.F, 600)
        await motor.run(port.D, 408)

runloop.run(main())
```

SAMPLE

ユニット 6　レッスン 3
ビートに合わせてダンス？

ブレイクダンサーのカラーセンサーにブロックをかざしたときに、次の表に示すように動くプログラムを作りなさい。

まず、疑似コードやフローチャートを作成してからコーディングしてから、バグを修正（デバッグ）して完成させましょう。

ブロックの色	動き
BLUE	脚だけ動く
GREEN	腕だけ動く
YELLOW	腕と脚両方がゆっくり動く
RED	腕と脚両方が早く動く

SAMPLE

```
from hub import port, light_matrix
import runloop
import motor, color_sensor, color

async def main():
    #color_value にカラーセンサーから色の値を取得
    color_value = color_sensor.color(port.B)

    while True:
        await runloop.until(lambda: color_sensor.color(port.B) not in(-1, color_value))
        color_value = color_sensor.color(port.B)
        # もし者ならば腕のモーターを動かす
        if color_value == color.BLUE:
            light_matrix.show_image(light_matrix.IMAGE_HAPPY)
            await motor.run_for_degrees (port.D, 360, 600)
        # 縁ならば脚を動かす
        elif color_value == color.GREEN:
            light_matrix.show_image(3)
            await motor.run_for_degrees (port.F, 360, 600)
        # 黄色ならば腕と脚を同時にゆっくり動かす
        elif color_value == color.YELLOW:
            light_matrix.show_image(light_matrix.IMAGE_HAPPY)
            motor.run_for_degrees (port.D, 360, 200)
            motor.run_for_degrees (port.F, 360, 200)
        # 赤ならば腕と脚を同時に早く動かす
        elif color_value == color.YELLOW:
            light_matrix.show_image(light_matrix.IMAGE_HAPPY)
            motor.run_for_degrees (port.D, 360, 600)
            motor.run_for_degrees (port.F, 360, 800)
        # そのほかの場合
        else:
            light_matrix.show_image(light_matrix.IMAGE_SAD)

runloop.run(main())
```

SAMPLE

ユニット 6 レッスン 4
トラブルのテスト

CNC 装置の組み立て図に沿って組み立て終わったら、次のプログラムをプログラミングキャンバスに入力し実行してみましょう。
コンソールにエラーが出ているか確認しましょう。エラーが出ている場合はエラー帆種類を特定してデバッグしましょう。
エラーが出ていないけれどモデルが意図通りに動かない場合は、モデルのメカニズムもチェックして修理しましょう。

```
from hub import button, port
import runloop
import motor

def right_button_pressed():
    return button.pressed (button.RIGHT)

def left_button_pressed():
    return button.pressed (button.LEFT)

async def main():
    while True:
        await runloop.until(right_button_pressed)
        await motor.run_for_degrees (port.C, 1000,500)

        await runloop.until(left_button_pressed)
        await motor.run_for_time(port.A, 1500,-500)

        await motor.run_for_degrees (port.A, 400, 500)
        await motor.run_for_degrees (port.C, 575,500)
        await motor.run_for_degrees (port.A,-400,500)
        await motor.run_for_degrees (port.C, -575,500)
        await motor.run_for_degrees (port.A,-60,500)
        await motor.run_for_degrees (port.C,-800,500)
        await motor.run_for_degrees (port. A, 400,500)
        await motor.run_for_degrees (port. C, 1600,500)
        await motor.run_for_degrees (port.A, -400,500)
        await motor.run_for_degrees (port.C,-800,500)

runloop.run(main())
```

SAMPLE

CNC 装置のメカニズムにどのような不具合がありましたか？

SAMPLE

To Be Continued in Part 3

Unit 7 Impacting the Environment with Functions

Unit 8 Compound Conditionals and Logic Operators

Unit 9 Data and Math Functions

Unit 10 Lists

SAMPLE

Introduction to Python Programing
Using LEGO Education SPIKE Prime Set
SPIKE プライム　Python プログラミングレッスンガイド　日本語版　Part 2

作　LEGO Education
翻　訳・編　集・発　行
株式会社ラーニングシステム
〒 220-0012
神奈川県横浜市西区みなとみらい 2-3-2　みなとみらい東急スクエア① 4F
TEL：045-232-4391　FAX：045-232-4392
e-Mail：custom@learningsystems.co.jp
WEB Site：https://www.learningsystems.co.jp/

LEGO, The LEGO logo, Theminifigure and the SPIKE logo are trademarks and / or copyright the LEGO Group. ©2024 The LEGO Groupe. All rights reserved.

Learning Systems™