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 3 日本語版

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）　P3 ～ 7

レッスンガイド　P8 ～

Unit 7 Function：関数

Lesson 1　P9

Lesson 2　P12

Lesson 3　P15

Lesson 4　P19

Lesson 5　P22

Unit 8 Compound Conditionals and Logic Operators: 複合条件と論理演算子

Lesson 1　P25

Lesson 2　P27

Lesson 3　P31

Lesson 4　P35

Lesson 5　P37

Unit 9 Data and Mathfunction: データと数学関数

Lesson 1　P40

Lesson 2　P43

Lesson 3　P47

Lesson 4　P52

Lesson 5　P58

Lesson 6　P61

Unit 10 Lists: リスト

Lesson 1　P64

Lesson 2　P67

Lesson 3　P73

Lesson 4　P80

Lesson 5　P86

生徒用ワークシート　P89 ～

SAMPLE

Lesson Objectives：レッスンの目的

Part I

Unit I Hardware & Software ユニットI 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 I 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 I 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）	☐ カ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）	☐ カ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 7 Impacting the Environment with Functions
ユニット 7 関数

Unit Introduction: ユニット紹介

このユニットでは、環境への影響を調査することを通して、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究します。生徒たちは、プログラムで関数を作成し使用する方法と、関数を使用する利点を調べます。

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な順序でデザインされています：

- ☐ プログラム内で関数を作成し、呼び出す
- ☐ 関数にパラメータを設定することを探究する
- ☐ プログラムで関数を使用する最適なタイミングを検討する
- ☐ コード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
- ☐ 問題の定義と分解 ユニット学習の目標

ユニット学習の期待

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

- ☐ 関数を使用すると、どのように効率的なプログラムを作成できますか。
- ☐ 関数が呼び出されたときに、どのように可変性を持たせることができますか？

SAMPLE

ユニット 7 レッスン 1
カメのトラブル

カメのトラブル

生徒がプログラムの中で関数をどのように作り、どのように使うかを調べます。

主な目的

生徒は次のことを行います：

- ☐ カメのヒレを動かすプログラムを書く。
- ☐ 状況に応じて異なる反応やコード行が実行されるように、プログラムを修正する。

ボキャブラリー

関数

探究への質問

- ☐ コードの行を何度も使うにはどうしたらよいでしょうか。

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

- ☐ 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

私たちが身の回りの環境にどのような影響を与えているかについて、生徒を引きつけます。
ビニール袋やその他のゴミの中に閉じ込められたウミガメのビデオを見せながら、話し合いを進めます。続けて、人間が環境に与えるその他の影響について話し合います。

2. 探究する

生徒は、動物が生息地の汚染にどのように反応するかを調べます。
生徒を SPIKE アプリの組み立て図セクションに誘導します。ここで生徒は、ホッパーモデルの組み立て説明書にアクセスできます。生徒にモデルを作らせます。作り方の説明書は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。このレッスンでは、ホッパーをウミガメに見立てます。
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。

やってみる

ウミガメのヒレを動かして泳いだり、陸を歩いたりするようにプログラミングする方法を考えさせます。ウミガメはゆっくりと動くことを忘れないでください。
生徒に、ウミガメのヒレを動かすプログラムを書くように指示します。

サンプルプログラム

```
from hub import port
import runloop
import motor_pair

async def main():

```

SAMPLE

```
    #ポートEとFのモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    #速度500で1440度分まっすぐに進む
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=500)

runloop.run(main())
```

生徒にカメをプログラムする時間を与えます。

ゴミの中を移動する

カメがどのように動くかを話し合います。カメは 1 つのスピードで泳いだり歩いたりしません。カメは怯えているときは速く泳ぎます。ビニール袋や海で拾ったゴミから出ようと必死になっているとき、カメはヒレを前後に動かすかもしれません。

生徒たちに、状況に応じて異なる反応やコード行が実行されるようにプログラムを修正するよう求めます。これを行う一つの方法は、関数を使うことです。関数とは、プログラムの中でいつでも呼び出すことができる、再利用可能なコードのブロックです。
関数をセットアップする (定義する) には、コードの def 行から始めて、関数の名前、括弧、コロンで終わるように生徒に説明します。(生徒には、関数の動作に意味のある名前をつけるように注意してください)。
例 :

```
async def frightened() :
        await motor_pair.move_for_degrees (motor_pair.PAIR_1, 1440, 0, velocity=1000)
```

サンプルプログラムを生徒に説明し、関数がどのように定義され、使用するプログラムで呼び出されるかを確認します。

サンプルプログラム

```
from hub import port
import runloop
import motor_pair

# カメを早く動かす関数
async def fritened():
   # 速度 1000 度 / 秒で、まっすぐに 1440 度分移動する
   await motor_pair.move_for_degrees (motor_pair.PAIR_1, 1440, 0, velocity=1000)

# メイン関数
async def main():
   # ポート E と F のモーターをペアにする
   motor_pair.pair (motor_pair.PAIR_1, port. E, port.F)
   # はやく動く関数を呼び出す
   await fritened()

runloop.run(main())
```

SAMPLE

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：

☐ あなたの関数と、関数がどのように機能するか説明してください。

☐ プログラムで関数を使う利点は何ですか？

☐ 関数を定義することと、関数を呼び出すことの違いは何ですか？

関数は、プログラムの中でコードのブロックを何度も繰り返す必要があるときに使われます。私たちのプログラムでは、関数を作成し、それを 1 回呼び出します。この関数は、プログラムの中で何度でも呼び出すことができます。関数を作ることで、コードの行を何度も使うことなく、プログラムを短縮することができます。

4. 詳しく説明する

カメが通常の速度で泳いだり歩いたりするように、またビニール袋に閉じ込められたときにもがくように、さらに 2 つの関数を追加します。プログラムの中で必要なときに各関数を呼び出すように生徒に指示します。

```
from hub import port
import runloop
import motor_pair

async def frightened():
    #速度1000で1440度直進する
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)
async def normal():
    #速度250で1440度直進する
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=250)
async def struggle():
    #速度1000で1440度直進し、その後、後進を行う。
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1440, 0, velocity=1000)
    await motor_pair.move_for_degrees (motor_pair.PAIR_1, -1440, 0, velocity=1000)
async def main():
    #E,Fモーターをペア設定
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)
    await frightened()
    await struggle()
    await normal()

runloop.run(main())
```

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。

☐ 関数を呼び出したとき、何が起こりましたか？

☐ 関数を定義する順番は重要ですか?なぜですか？

☐ 関数を使う利点は何ですか？

自己評価

ノートに次のように答えさせます：

☐ 関数の使用について今日学んだことは何ですか。

COPY

ユニット 7 レッスン 2

複数の関数でクリーンアップ

複数の関数でクリーンアップ

生徒は、グラバーを使って物を拾うことを調べます。

主な目的

生徒は次のことを行います

- ☐ アイテムを拾うグラバーを組み立て、プログラムする。
- ☐ 複数の機能を含むようにプログラムを修正する。

ボキャブラリー

関数

探究への質問

- ☐ 複数の機能を使ってゴミを拾うように、グラバーをどのようにプログラムできるでしょうか。

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
- ☐ グラバーで拾うための様々なアイテム（ゴミをシミュレートできるアイテムが理想的です）

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

ゴミについて、またゴミが私たちの環境にどのような影響を与えるかについて、生徒を引きつけます。

海、川、公園など、さまざまな場所にあるゴミの画像やビデオを見せます。生徒たちに、ゴミがこれらの環境に住むものにどのような影響を与えるか考えさせます。

生徒に、ゴミを掃除する方法を考えさせます。

2. 探究する

生徒は、プログラム内で複数の関数を使用する方法を調べます。

生徒を SPIKE アプリの組み立て図セクションに誘導します。ここで生徒は、グラバーの組み立て図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。作成手順は、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。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生徒に指示します。生徒はハブを接続します。

ゴミを拾うための道具をどのようにプログラムするか、生徒に考えさせます。

サンプルプログラム

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

def is_pressed():
    #フォースセンサーが押されているか?
    return force_sensor.pressed(port.E)

```

SAMPLE

```
def is_released():
    #フォースセンサーが離されているか?
    return not force_sensor.pressed (port.E)

async def main():
    while True:
        #フォースセンサーが押されるまで待つ
        await runloop.until(is_pressed)
        #グラバーを閉じる
        await motor.run_to_absolute_position(port. A, 345, 500,
            direction=motor.COUNTERCLOCKWISE)
        #フォースセンサーが離されるまで待つ
        await runloop.until(is_released)
        #グラバーを開く
        await motor.run_to_absolute_position(port.A, 25, 500,
            direction=motor.SHORTEST_PATH)

runloop.run(main())
```

生徒に、両方のグラバーでさまざまな材料をつまむ実験をして、プログラムをテストさせます。それぞれのグラバーがどの材料を拾うのに最も適しているかをグループで話し合います。

複数の関数の使用

より強く握るための関数と、より軽く握るための 2 番目の関数を含むようにプログラムを修正するよう生徒に促します。
どのグラバーアタッチメントを使用するかに応じて呼び出すことができる関数を組み込む方法について生徒と話し合います。生徒は、どちらのアタッチメントが使用されているかによって、グラバーが必要に応じて開閉するようにしたいと思うでしょう。そのため、生徒はそれぞれ別の関数を作成することができます。

サンプルプログラム

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

def is_pressed():
    #フォースセンサーが押されているか?
    return force_sensor.pressed(port.E)

def is_released():
    #フォースセンサーが離されているか?
    return not force_sensor.pressed(port.E)

async def grab_and_release(grip):
    #フォースセンサーが押されるまで待つ
    await runloop.until(is_pressed)
    #グラバーを閉じる
    await motor.run_to_absolute_position(port.A, 345, grip, direction=motor.COUNTERCLOCKWISE)
    #フォースセンサーが離されるまで待つ
    await runloop.until(is_released)
    #グラバーを開く
    await motor.run_to_absolute_position(port.A, 25, grip, direction=motor.SHORTEST_PATH)

async def main():

    while True:
        grab_and_release(1000)

runloop.run(main())
```

注意：生徒は、その時に使用しているグラバーに応じてグラバーのアームを変更する必要があります。
生徒が両方のグラバーを使ってプログラムをテストし、関数が正しく動作しているかどうかを確認する時間を与えます。適切な関数を適切なタイミングで使用するために、プログラムで使用する関数の呼び出し方の重要性を生徒に思い出させます。

APPLE

3. 説明する

それぞれのプログラムがどのように動作するか、生徒と話し合います。

生徒に次のような質問をします：

☐ 2 つの関数を比較してどうですか？

☐ なぜプログラムに関数を追加したがるのか？

☐ プログラムで複数の関数を使用する場合、何を考慮する必要がありますか？

4. 詳しく説明する

各グラバーで異なるオブジェクトを拾えるように、プログラムを修正することに生徒を挑戦させます。

生徒は、各グラバーに固有の関数を含むようにプログラムを修正する必要があります。こうすることで、生徒は、グラバーアームごとに同じプログラムを使用するのではなく、どのグラバーアームを使用するかに応じて関数を呼び出すことができるようになります。

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：

☐ なぜグラバーアームごとに別の関数が必要なのですか？

☐ 各グラバーアーム用の関数を、そのグラバーアームの動作に特化させるために、何を変更しましたか？

☐ プログラムで複数の関数を定義し、必要に応じて後で呼び出すことができるようにすると便利なのはなぜですか？

自己評価

生徒にノートで次のことに答えさせます：

☐ プログラムの中で複数の関数を使うことについて、今日は何を学びましたか？

SAMPLE

ユニット 7 レッスン 3
クリーンインディケーター

クリーンインディケーター

生徒は関数におけるパラメータの役割を調べます。

主な目的

生徒は次のことを行います

- ☐ パラメータを使用する関数を作成する。
- ☐ デバッグ関数とパラメータを使用する関数について調べる。

ボキャブラリー

関数
パラメータ

探究への質問

- ☐ 関数内でパラメータをどのように使用できますか？

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

グラフの作成や、情報やデータを視覚化する方法について、生徒を話し合いに引きつけます。
生徒たちに立ち上がって、足を動かして投票の準備をするように指示します。生徒が選択できるいくつかの選択肢がある質問を生徒に投げかけます。生徒は、自分の選択肢の場所に移動して答えを示します。これは、部屋の隅でも、指定した場所でもかまいません。
生徒が投票したら、一列に並んで移動し、人間棒グラフを作成します。
質問の例をいくつか挙げます：

- ☐ 好きなアイスクリームの味は何ですか？（ 好きなフレーバーは？4 つの中から選ぶ）
- ☐ 好きな本のジャンルは？（ 3-4 点のリストから選ぶ）
- ☐ どの登場人物に一番似ていますか？（ 本や映画から選ぶ）

2. 探究する

生徒は、関数でパラメータを使用することを調べます。
生徒を SPIKE アプリの組み立て図セクションに誘導します。ここで生徒は、風速計モデルの組み立て図にアクセスできます。生徒にモデルを構築するように指示します。組み立て図は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。

生徒に、モデルの設計を変更するように指示します。生徒に、黄色と青色のブロックを 2 個ずつ積み重ねた側を、一番下に赤、次に黄色、次に青、そして一番上を緑とするよう指示します。
また、風速計の動きを妨げない場所にカラーセンサーを追加する必要があります。
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生徒に指示します。
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。
環境がきれいであるかどうか（ 緑色レベル）、またはそうでないか（ 赤色レベル）を示すインジケーターに関数を使用する方法を考えるよう、生徒に促します。インジケーターは、スタック内の適切な色のブロックに移動してレベルを示す必要があります。

SAMPLE

生徒たちに、その関数がプログラムの中でどのように見えるか、擬似コードプログラムを書いてもらいます。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：

☐ 関数の定義

☐ カラーセンサーが汚れのレベルを検出

☐ 色を読み取った位置にモーターを動かす

☐ コンソールに色を表示する

擬似コードを議論します。

パラメータの追加

生徒がプログラムにパラメータを追加し、環境を「 スキャン」してクリーンレベルを示すことに挑戦します。

生徒には、スキャンする環境として、テーブルの上に散らばる赤、紫、黄、緑の 2x4 ブロックが必要です。さらに、カラーセンサーをハブに接続してモデルに追加するよう生徒に指示します。

生徒に、カラーセンサーを使用して環境（ 余分なブロック）を「 スキャン」するためのパラメータを関数に設定するように、プログラムを修正するように指示します。パラメータは変数のようなものですが、関数の中でしか使えません。パラメータはユーザーから入力されるデータです。関数が呼び出されたときに追加される新しい情報の保存場所として機能します。

サンプルプログラム

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

#クリーンレベルをレポートする関数の定義
def report(height, color):
    #モーターをhightで指定された角度へ移動する
    runloop.run(motor.run_to_absolute_position(port.A,height,500,
        direction=motor.SHORTEST_PATH))
    #colorで指定された値をコンソールに表示する
    print('Clean Level', color)

async def main():
    while True:
        await runloop.until(lambda:button.pressed(button.LEFT))
        #検出した色に合わせたhightを設定
        if color_sensor.color(port.E) is color.RED:
            report(70, 'Red')
        elif color_sensor.color(port.E) is color.YELLOW:
            report(60, 'Yellow')
        elif color_sensor.color(port. E) is color.BLUE:
            report(50, 'Blue')
        elif color_sensor.color(port.E) is color.GREEN:
            report(25, 'Green')
        #色を検知しなかった、あるいは上記以外の色の場合0に戻る
        else:
            await motor.run_to_absolute_position(port. A, 0, 1000,
                direction=motor.SHORTEST_PATH)
            print("No Reading")

runloop.run(main())
```

生徒がプログラムを探究し、環境を複数回読み取る時間を与えます。

3. 説明する

それぞれのプログラムがどのように動作するか、生徒と話し合います。

生徒に次のような質問をします：

☐ なぜ関数にパラメータを追加したのか？

☐ パラメータと変数の違いは何ですか？

☐ プログラムで関数を使用することで、プログラムがどのように効率的になりましたか？

4. 詳しく説明する

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を生徒に議論させます。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。

デバッグアクティビティ 1:

```
1  from hub import port, button
2  import runloop
3  import motor
4  import color_sensor, color
5
6  #クリーンレベルをレポートする関数の定義
7  def report():
8      #モーターをhightで指定された角度へ移動する
9      runloop.run(motor.run_to_absolute_position(port.A, hight, 500, direction=motor.SHORTEST_PATH))
10     #colorで指定された値をコンソールに表示する
11     print('Clean Level', color)
12
13 async def main():
14
15     while True:
16         await runloop.until(lambda: button.pressed(button.LEFT))
17         #検出した色に合わせたhightを設定
18         if color_sensor.color(port.E) is color.RED:
19             report(70, 'Red')
20         elif color_sensor.color(port.E) is color.YELLOW:
21             report(60, 'Yellow')
22         elif color_sensor.color(port.E) is color.BLUE:
23             report(50, 'Blue')
24         elif color_sensor.color(port.E) is color.GREEN:
25             report(25, 'Green')
26         #色を検知しなかった、あるいは上記以外の色の場合0に戻る
27         else:
28             await motor.run_to_absolute_position(port.A, 0, 1000, direction=motor.SHORTEST_PATH)
29             print('No Reading')
30
31 runloop.run(main())
32
```

```
Traceback (most recent call last):
File "Project 6", line 31, in <module>
File "Project 6", line 19, in main
TypeError: function takes 0 positional arguments but 2 were given
```

「line 19」とあるので、19行目に注目してみるとエラーがないように見えます。

実際のエラーは、関数が定義されている 7 行目で発生しています。関数を定義する際には、関数が呼び出されたときにパラメータの値が得られるようにパラメータを設定する必要があります。現在与えられているパラメータの値は、与えられているパラメータを参照することができません。

デバッグアクティビティ 2:

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

#クリーンレベルをレポートする関数の定義
def report(hight, color):
    #モーターをhightで指定された角度へ移動する
    runloop.run(motor.run_to_absolute_position(port.A, hight, 500, direction=motor.SHORTEST_PATH))
    #colorで指定された値をコンソールに表示する
    print('Clean Level', color)

async def main():

    while True:
        await runloop.until(lambda: button.pressed(button.LEFT))
        #検出した色に合わせたhightを設定
        if color_sensor.color(port.E) is color.RED:
            report(70)
        elif color_sensor.color(port.E) is color.YELLOW:
            report(60, 'Yellow')
        elif color_sensor.color(port.E) is color.BLUE:
            report(50, 'Blue')
        elif color_sensor.color(port.E) is color.GREEN:
            report(25, 'Green')
        #色を検知しなかった、あるいは上記以外の色の場合0に戻る
        else:
            await motor.run_to_absolute_position(port.A, 0, 1000, direction=motor.SHORTEST_PATH)
            print('No Reading')

runloop.run(main())
```

Traceback (most recent call last):
File " プロジェクト 7", line 32, in <module>
File " プロジェクト 7", line 19, in main
TypeError: function takes 2 positional arguments but 1 were given

注意：このエラーメッセージの行参照は、どの色をスキャンするかによって異なります。
このエラーは生徒に 19 行目をチェックするように示すはずです。生徒は、関数がパラメータへの参照で呼び出されていることを認識するはずです。引数が 2 つあるはずなのに、1 つしか与えられていないことがエラーの原因となっています。70 は最初の引数ですが、さらに色が必要です。
生徒に、この種のプログラムで発生する可能性のある他のコメント・エラーと、エラーが発生した場合のデバッグのアイデアを考えさせます。

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：

☐ 関数やパラメータを使ったプログラミングでは、どのようなエラーに注意する必要がありますか？

☐ 関数内でパラメータを使用する方法にはどのようなものがありますか？

自己評価

SAMPLE

ユニット 7 レッスン 4
クリーンアップの自動化

クリーンアップの自動化

生徒は、分別ロボットを作成し、自動清掃プロセスの機能を使用するようにプログラム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

- ☐ 材料がリサイクル可能か不可能かを識別するために、分別ロボットをプログラムする。
- ☐ プログラムをより効率的にするために、プログラムに 2 番目の関数を組み込む。

探究への質問

☐ プログラミングで関数をどのように使うと効率的でしょうか。

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

☐ 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

清掃活動を自動化する方法について、生徒と話し合います。
環境を清潔に保つために必要な活動について生徒と話し合います。清潔さを保つためにロボット工学を使って自動化できる方法を考えましょう。ゴミになる物質もあれば、リサイクルできる物質も多いことを話し合います。生徒たちに、ゴミとみなされるさまざまな材料とリサイクルできるものを見つけさせます。
生徒が材料をゴミとリサイクル可能な材料に分別する方法を考えるように促します。

2. 探究する

生徒は、清掃を自動化し、ゴミとリサイクルできる材料を分類する方法を調査します。
生徒に組み立て図セクションを示し、品質検査ロボットのモデルを作るように指示します。組み立て方は、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。
材料がゴミかリサイクル可能かを識別するために、分別ロボットをどのようにプログラムするか生徒に考えさせます。生徒は、材料の色をチェックし、リサイクル可能な材料をロボット選別機の前の青い領域に分類し、ゴミを捨てる関数を作成します。
ワークシートを活用することを推奨します。
注意:生徒は、ゴミに色をを割り当てる必要があります。組み立ての説明で使用されている色は、黄色と紫色です。生徒が望むなら、他の色に変更させることを検討します。

サンプルプログラム

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

SAMPLE

```
import distance_sensor
from app import sound

async def main():
    await get_up()
    await runloop.until(lambda:color_sensor.color(port.D) != color.UNKNOWN)
    await check_color()

async def get_up():
    pixels = [100] * 4
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    distance_sensor.show(port.C, pixels)

async def accept():
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    sound.play('Triumph')
    for x in range(5):
        await motor.run_to_absolute_position(port.A,15,250,direction=motor.SHORTEST_PATH)
        await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)

async def refuse():
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    sound.play('Oops')
    await motor.run_to_absolute_position(port.F,50,1000,direction=motor.SHORTEST_PATH)
    for x in range(3):
          await motor.run_for_degrees (port.F,-100, 1000)
          await motor.run_for_degrees (port.F,100, 1000)

    await motor.run_to_absolute_position(port.F,0,1000,direction=motor.SHORTEST_PATH)

async def check_color():
    await motor.run_to_absolute_position(port.F, 270,250, direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.A, 60,250, direction=motor.SHORTEST_PATH)
    await runloop.sleep_ms(500)

    if color_sensor.color(port.D) is color.MAGENTA:
        await accept()
    else:
        await refuse()

runloop.run(main())
```

生徒にプログラムを作成し、実行する時間を与えます。プログラミング中に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させます。生徒は、プログラムをテストしながらデバッグする必要があるかもしれません。

SAMPLE

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：

☐ なぜこのプログラムで関数を使ったのか？

☐ プログラムで関数を使う他の方法は何か？

☐ 生徒がパラメータを含む場合、どのようなパラメータを使用したか、またその理由は何か。

ここで使用したサンプルプログラムを生徒に見せ、プログラムを準備しやすくするために含めるべきコードコメントを（ # を使用して）記述するように求めます。

4. 詳しく説明する

ロボットをより便利にするアイデアについて生徒と話し合います。プログラムの冒頭で変数や関数などを定義する時間を取ることで、プログラム本体をより短く、実行しやすくすることができます。

ロボットをより便利にする方法を話し合います。生徒は改善点を明文化し、多角的なアプローチを取れるよう促します。

生徒には、コードにコメントを入れるように注意するように促します。

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：

☐ プログラムに関数を追加したらどうなりましたか？

☐ 大きなプログラムを作るのに、なぜ効率が重要なのですか？

☐ プログラムで関数を使用するには、どのような方法がありますか?他の生徒がこの課題にどのように取り組んだかから、何を学びましたか？

自己評価

生徒にノートで次のことに答えさせます：

☐ プログラムの効率化について、今日学んだことは何ですか？

SAMPLE

ユニット 7 レッスン 5
環境を大切にする

環境を大切にする

主な目的

生徒は次のことを行います

☐ 地域の世話をする環境ヘルパーを設計、製作、プログラムする。

生徒は、地域の環境問題と、その解決に役立つ可能性のある方法を調べます。

探究への質問

☐ ロボットはどのように環境保護に役立てることができるでしょうか。

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

☐ 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

地元の環境について生徒を引きつけます。

公園や校庭、湖や川などの水辺など、地元の環境について生徒に考えさせます。このような地域とそこに生息する動物について話し合いを促します。生徒が環境を保全するロボットを作ることで、それらの生き物を助ける方法にはどのようなものがあるかを考えさせます。

グループとして、生徒のアイデアについて話し合います。その後、ディスカッションを行います。

ブレインストーミング

小グループに分かれ、以前の授業で使用したアイデアを参考にしながら、環境とそこに住む生物に役立つ新しいタイプのロボット・ヘルパーを作るためのアイデアをいくつか出し合います。生徒たちは、清潔度を示したり、ゴミやリサイクルを分別したり、物を拾ったりする機能を追加することを考えるかもしれません。また、生徒たちは、何か新しいものをデザインして、ゴミをある場所に押し込んで、その跡に種を撒くようなヘルパーを作ろうとするかもしれません。各グループは、それぞれのニーズをサポートするユニークなアイデアを出すべきです。

2. 探究する

生徒たちは、環境をきれいにするロボット・ヘルパーを設計、製作、プログラミングします。

設計と正しいアイデアの選択

生徒は、自分たちの環境に適したロボットヘルパーを設計、製作、プログラミングします。制約条件は以下のとおりです：

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. プログラムに少なくとも 1 つの関数を含むこと。
5. 関数と連動するパラメータを含むこと。

SAMPLE

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。

テストと反復

生徒が自分のアイデアをテストして分析し、必要に応じて改善する時間を設けます。生徒は、ソリューションを作り始める際に、設定された設計基準とフローチャートに照らして、設計をテストし、評価する必要があります。

生徒がモデルのスケッチや写真を使用して、設計の過程を記録するようにします。

時間の許す限り、生徒が自分の設計についてフィードバックを受けられるようにします。このフィードバックは、他のグループまたは教師から受けることができます。

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：

- ☐ 少なくとも 1 つの関数を使用して環境ヘルパーを作成するモデルを、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のコメントを話してもらい、説明してもらいます。
- ☐ 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?
- ☐ どのようにパラメータを組み込みましたか?
- ☐ 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の後、生徒が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、設計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。

生徒が最終的な設計を発表し、それがどのように基準を満たしたかを説明できるようにします。

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?
- ☐ どのような関数やパラメータを含めましたか?

自己評価

生徒にノートで次のことに答えさせます：

- ☐ 与えられた基準に基づいて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 8 Compound Conditionals and Logic Operators
ユニット 8 複合条件と論理演算子

Unit Introduction: ユニット紹介

このユニットでは、サイバーセキュリティとデジタルセーフティの側面を探究しながら、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学びます。論理演算子や入れ子の条件文を活用しながら、セキュリティ対策を施したモデルを設計、構築、プログラミングすることで、システムを安全に保つ方法を探究します。生徒は、物理的およびデジタル的なセキュリティ対策を活用することで、安全の重要性を認識します。

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な順序でデザインされています：

- ☐ パスワードの設定によるサイバーセキュリティの調査
- ☐ 物理的なセキュリティ対策を探究する
- ☐ 入れ子になった条件文について調べる
- ☐ 論理演算子を使って条件を組み合わせる
- ☐ セキュリティのためのセンサーの使用について調べる
- ☐ 2 段階のセキュリティプログラムを作成する
- ☐ 物理的安全機能とデジタル安全機能の両方を備えたセキュリティ装置をシミュレートする

ユニット学習の期待

このユニットでは、条件文と論理演算子のプログラミング方法を探究します。物理的な安全対策とデジタル的な安全対策の両方を利用したさまざまなプロトタイプを作成しながら、条件文と論理演算子を使用するタイミングについての理解を深めます。生徒は、新しい知識とスキルをサイバーセキュリティ・シミュレーションに取り入れます。

探求への質問：

- ☐ 条件文と論理演算子を安全対策としてどのように使用できるでしょうか。
- ☐ デジタル・ウェルネスにとってサイバーセキュリティが重要なのはなぜでしょうか。
- ☐ 物理的安全対策とデジタル安全対策をどのように組み合わせることができるでしょうか。

SAMPLE

ユニット 8 レッスン 1
パスワードの保護

パスワードの保護

主な目的

生徒は次のことを行います

- ☐ サイバーセキュリティについて調べる。
- ☐ パスワードを設定する。
- ☐ 物理的なセキュリティ対策を探ってみる。

探究への質問

☐ 生徒はブロックを使ってパスワードを作成し、セキュリティ対策を探究します。

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）。
- ☐ 各生徒は SPIKE プライムセットから 5 色の 2x4 ブロックが必要です

1. 引きつける

鍵に関するディスカッションを始めます。鍵はどこで使うのか?なぜ鍵を使うのか。

何かを安全に保管したいときに鍵を使うことがあることを生徒に説明します。生徒に次のような質問をします：

- ☐ パスワードとは何ですか?
- ☐ パスワードはどこで使われているのですか?
- ☐ 強力なパスワードは、どのようにして私たちの情報を安全に保つことができるのですか?
- ☐ なぜパスワードを共有しないのですか?

生徒にパスワードの例を共有させ、実際のパスワードを共有しないように注意させます。

2. 探究する

生徒は、より安全なパスワードの作成について調べます。

生徒に、セットから 5 色の 2x4 ブロックを選ぶように指示します。

2 人 1 組になり、各チームにパートナー A とパートナー B を割り当てます。パートナー A は、パートナー B に何を書いているか見られないように、自分のノートに 3 つの数字を書きます。パートナー B は、最初に書かれた数字を当てます。パートナー A は、何回目で当てられたかを、数字の下に回数を書きます。2 番目と 3 番目の数字も同じことを繰り返します。役割を交代します。パートナー B はノートに数字を 3 つ書き、パートナー A が数字を推測し、パートナー B が推測に要した回数を書きます。

クラス全員で結果を話し合います。それぞれの数字について、いくつの答えが考えられるかを生徒に尋ねます。考えられる答えは 10 個（ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9）です。3 桁のパスコードにはいくつの答えが考えられるでしょうか。（ 10 x 10 x 10）。[数字 000 から 999]。

パスワードの安全性をさらに高めるにはどうしたらよいか、一緒に話し合います。生徒は、さらに数字を追加して 4 桁のパスワードを作れば、より安全であることを認識します。

ブロックのパスワード

ブロックを使って 5 段階のパスワードを作成します。

各パートナーは、まずパスワードの最初のステップまたは最初の色を積み重ねの一番下に、最後のステップが積み重ねの一番上になるようにブロックの積み重ねを作ってパスワードを作成します。生徒は好きな順番で積み重ねることができます。

SAMPLE

3. 説明する

パスワードがどのように機能する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：

☐ 3 桁の数字と 5 つのブロックのパスワードでは、どちらが推測が難しかったですか？

☐ 5 つのブロックのパスワードはどうですか？

☐ ブロックのパスワードの最後の数色を推測するのが簡単だったのはなぜですか？

☐ パスワードをもっと強くするにはどうしたらいいでしょうか？

4. 詳しく説明する

生徒に、より安全なパスワードを考えさせます。

生徒に、SPIKE プライムセットからハブを取り出すように指示します。生徒がパスワードを作成し、そのパスワードが正しいかどうかを、画像、言葉、音を使ってハブに表示することに挑戦します。

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生徒に指示します。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。

ハブを使用してパスワードを作成するプログラムについて、アイデアを出し合います。モーションセンサーの向きを特定の位置に設定する（ 上、前、左など）、モーションセンサーのジェスチャーを使用する（ シェイクやタップなど）、ボタンを押すなどのアイデアが考えられます。生徒たちは、パスワードが正しければスマイリーフェイスを表示したり、ハブに「 グッジョブ」とプリントしたりすることを考えるかもしれません。

生徒に、パスワードの一部としてハブをどのように使用する予定かを擬似コードで書くように指示します。その後、生徒は擬似コードに従ってプログラムを作成します。生徒には、何が起こるべきかを示すために、プログラムの中でコードコメントを使用するように注意してください。また、コンソールにエラーメッセージが表示されないか見ながらプログラムを作成します。

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：

☐ パスワードを難しくしたらどうなりましたか？

☐ なぜパスワードを使うのですか？

☐ 物理的なパスワードを作るにはどうしたらいいですか？

自己評価

生徒にノートで次のことに答えさせます：

☐ パスワードの使い方について、今日学んだことは何ですか？

SAMPLE

ユニット 8 レッスン 2
物理的に安全にする

物理的に安全にする

主な目的

生徒は次のことを行います

- ☐ サイバーセキュリティについて調べる。
- ☐ パスワードを設定する。
- ☐ 物理的なセキュリティ対策を探ってみる

探究への質問

☐ 生徒は条件文を使って、物理的に保護された情報の保存場所を作ります。

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

鍵に関するディスカッションを始めます。鍵はどこで使うのか？なぜ鍵を使うのか。
何かを安全に保管したいときに鍵を使うことがあることを生徒に説明します。生徒に次のような質問をします：

☐ パスワードとは何ですか？
☐ パスワードはどこで使われているのですか？
☐ 強力なパスワードは、どのようにして私たちの情報を安全に保つことができるのですか？
☐ なぜパスワードを共有しないのですか？

生徒にパスワードの例を共有させ、実際のパスワードを共有しないように注意させます。

2. 探究する

生徒を SPIKE アプリの組み立て図セクションに誘導します。ここで生徒は、金庫のモデルの作り方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。作り方の説明書は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。
注意：このモデルは、次のレッスン「 より安全な金庫を作る」でも使用します。付属のモーターはこのレッスンでは使用しませんが、次のレッスンでは使用します。
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。
生徒に、金庫がどのようにロックされ、適切な操作が行われるとロックが解除されるかを調べさせます。生徒に、サンプルプログラムをプログラミング・キャンバスに入力し、実行するように指示します。

サンプルプログラム

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

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

```

SAMPLE

```
async def main():
    await sound.beep(262, 200)
    await sound.beep(523, 200)

    #金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position(port.B, 0)
    await motor.run_to_absolute_position(port.C, 270, 500, direction=motor.CLOCKWISE)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    #金庫を解錠
    await runloop.until(pressed)
    await sound.beep(523, 200)
    await motor.run_for_time(port.C, 1000, 500)
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(5000)

runloop.run(main())
```

現在、金庫がどのようにロックされているかに注意しながら、グループでこのプログラムについて話し合います。生徒に、金庫の中の情報が現在どの程度保護されているかを考えさせます。金庫を使った保護は 1 段階しかないことに注意します。

アップグレードの時間

生徒たちは、より安全な金庫を作るために入れ子ループを探究します。

入れ子になった条件文を含むようにコードを変更することに生徒を挑戦させます。生徒には、ある部分を別の部分に入れると入れ子になることを説明します。ループと条件文を互いに入れ子にして、より複雑なプログラムを作ることができます。生徒に、プログラムがどのように動作するかを示すフローチャートを作成することから始めるように指示します。フローチャートを完成させ、グループで話し合う時間を生徒に与えます。

生徒に、入れ子ループを含む新しいプログラムを作成する時間を与えます。プログラムの中でコードコメントを使用して、プログラムが何を行うべきかを説明するように生徒に思い出させます。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。

サンプルプログラム

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

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

async def main():
    await sound.beep(262, 200)
    await sound.beep(523, 200)

    #金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position(port.B, 0)
    await motor.run_to_absolute_position(port.C, 270, 500, direction=motor.CLOCKWISE)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    #金庫を解錠
    while True:
        print('Open Me')
        if motor.relative_position(port.B) > 180:
            await runloop.until(pressed)
            await sound.beep(523, 200)
            await motor.run_for_time(port.C, 1000, 500)
            light_matrix.show_image(light_matrix.IMAGE_YES)
            await runloop.sleep_ms(5000)
            break
```

SAMPLE

```
28          else:
29              print('Still Locked!')
30              await runloop.sleep_ms(1000)
31
32          print('You may enter!')
33
34  runloop.run(main())
```

生徒に最終プログラムを発表する時間を与えます。

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：

- ☐ 金庫はどのように機能し、安全を提供していますか？
- ☐ 金庫はどの程度安全だと思いますか？
- ☐ 入れ子になった条件文はどのように機能しますか？
- ☐ この課題で難しかったことは何ですか?どこでプログラミングの問題にぶつかりましたか？
- ☐ break のコード行は何をしているのですか？

break 文がループを終了させることを生徒に説明します。

4. 詳しく説明する

生徒に、2 段階の保護の順序を変更し、プログラムのエラーを特定することに挑戦させます。
金庫を開けるステップの順序をどのように変更できるかを生徒に考えさせます。生徒が調査できるように、サンプルプログラムを共有します。このプログラムにはエラーはありません。しかし、このプログラムは金庫を開けることができません。生徒たちに、どこに問題があるかを特定し、プログラムをデバッグするように指示します。

サンプルプログラム

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

def pressed():
    return button.pressed(button.LEFT) and button.pressed(button.RIGHT)

async def main():
    await sound.beep(262, 208)
    await sound.beep(523, 200)

    # 金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position (port.B,0)
    await motor.run_to_absolute_position(port.C, 270, 500,direction=motor.SHORTEST_PATH)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を開錠
    while True:
        print("Open Me")
        await runloop.until(pressed)
```

SAMPLE

```
    if motor.relative_position (port.B) > 180:
        await sound.beep (523, 288)
        await motor.run_for_time(port.C, 1000, 500)
        light_matrix.show_image(light_matrix.IMAGE_YES)
        await runloop.sleep_ms(5000)
        break
    else:
        print('Still Locked!')
        await runloop.sleep_ms(1000)

print('You may enter!')

runloop.run(main())
```

生徒たちに、このプログラムはどう違うか聞いてみましょう。
注意：金庫を開けるには、両方のボタンを同時に押す必要があります。

生徒にプログラムを作り直す時間を与えます。最終的なプログラムをグループで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：

☐ 条件文を入れ子にするとどうなりますか？
☐ 条件文を入れ子にすることで、プログラムはより複雑になり、安全性はどうなりますか？
☐ 複合条件文をプログラミングするときに注意すべきエラーは何ですか？

自己評価

生徒にノートで次のことに答えさせます：

☐ 入れ子になった条件文の使用について、今日何を学びましたか？

SAMPLE

ユニット 8 レッスン 3
より安全な金庫を作ろう

物理的に安全にする

生徒は、論理演算子を持つ複合条件文を使って、物理的に保護された情報の保存場所を作ることを調べます。

主な目的

生徒は次のことを行います

- ☐ 論理演算子を使用して条件を組み合わせることを調べる。
- ☐ 物理的なセキュリティ対策を探究する。

探究への質問

☐ 生徒は、論理演算子を持つ複合条件文を使って、物理的に保護された情報の保存場所を作ることを調べます。

必要な機材

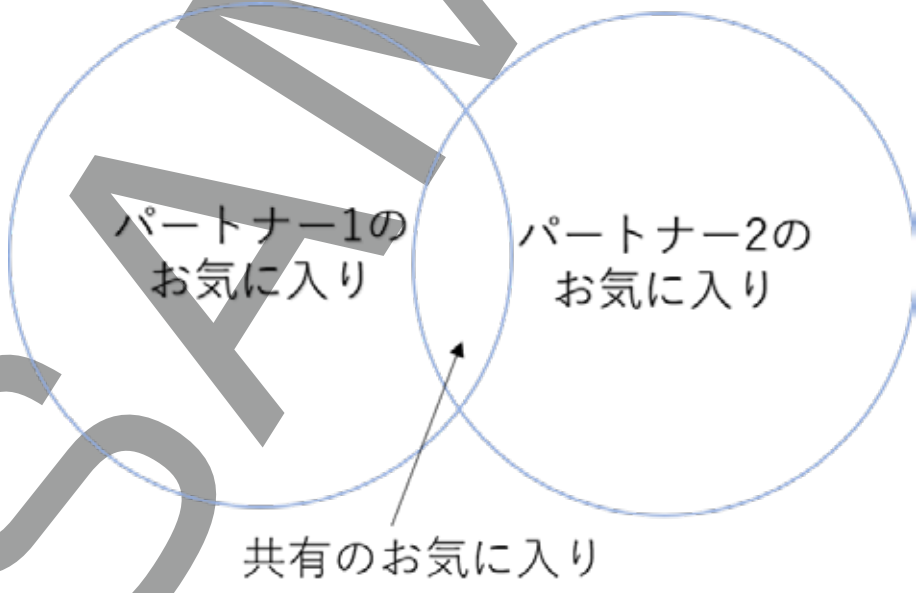
☐ SPIKE プライムセット
☐ SPIKE アプリがインストールされたデバイス
☐ ノート

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、「物理的に安全にする」のレッスンで作成した金庫のモデルを持っていることを確認します。

1. 引きつける

複数のアイデアを 1 つにまとめる方法について生徒に考えさせます。
生徒と好きな食べ物について話し合います。生徒が自分の好きなものについて 1 分間考えた後、2 人 1 組になり、それぞれのリストを比較させます。生徒に、自分の好きな食べ物を示すベン図を作成するように指示します。



ベン図について一緒に話し合います。2 人の生徒の共通点は何かを考えます。生徒たちに、もし一緒に昼食を食べ、分け合わなければならないとしたら、何を食べたいかを考えさせます。それぞれのリストにあるものを食べたいでしょうか、それとも 2 人とも好きなものを食べたいでしょうか。
生徒の考えを一緒に話し合います。生徒たちに、and、or、not という言葉を使いながら、自分なら何を食べるか、食べないかを考えさせます。例えば、相手のリストではなく、自分のリストか共有のリストのものを食べることができます。

SAMPLE

2. 探究する

生徒は、論理演算子を使用して、2 つの条件文を組み合わせて複合条件文を作成することを調べます。
生徒を SPIKE アプリの「 組み立て図」セクションに誘導します。ここで生徒は、安全な金庫の組み立て図にアクセスできます。生徒にモデルを作成させます。「 物理的に安全にする」レッスンで作成した金庫のモデルが生徒の手元にある場合、
生徒はアームを組み立て、取り付けるだけでよいです。組み立て方は、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。

条件文を組み合わせる

生徒に金庫がどのように機能するかを調べさせます。

サンプルプログラム

```
from hub import port, sound, light_matrix, button
import runloop
import motor
import app
import time

async def main():
    await sound.beep (262, 200)
    await sound.beep (523, 200)

    # 金庫をロック
    await motor.run_for_time(port.C, 1000, -500)
    await motor.run_to_absolute_position(port.B, 0, 500, direction=motor.SHORTEST_PATH, stop=motor.COAST)
    motor.reset_relative_position(port.B,0)
    await motor.run_to_absolute_position(port.E, 0, 500, direction=motor.SHORTEST_PATH)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を解錠
    await unlock()

async def unlock():
    start_time = time.ticks_ms()
    while not button.pressed(button.LEFT) and motor.relative_position (port.B) < 180:
        await sound.beep(262, 200)
        await motor.run_for_degrees (port. E, 15, 500)
        await runloop.sleep_ms(800)
        if time.ticks_diff(time.ticks_ms(), start_time) > 5000:
            await app.sound.play('Bonk')
            return
        light_matrix.show_image(light_matrix.IMAGE_NO)
```

SAMPLE

```
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await motor.run_to_absolute_position(port.E, 0, 500, direction=motor.SHORTEST_PATH)
    await motor.run_for_time(port.C, 1000, 500)
    await app.sound.play('Wand')

runloop.run(main())
```

現在、金庫がどのようにロックされているかに注意しながら、グループでこのプログラムについて話し合います。
生徒に、金庫の中の情報が現在どの程度保護されているかを考えさせます。
サンプルプログラムを注意深く見直し、新しいコード行を確認します。while 文を見るように生徒に促します。

```
while not button.pressed(button.LEFT) and  motor.relative_position(port.B) < 180:
```

生徒は、この行で 2 つの条件文が扱われていることを認識します。複合条件文という言葉を生徒に紹介します。複合条件文とは、2 つのブール式を組み合わせた文であることを説明します。すべての条件が満たされない限り、条件文は真になりません。
論理演算子は条件文を組み合わせるために使用されることを生徒に説明します。この文で使われている 2 つの論理演算子、「 and」と「 not」を指摘します。「 or」は 3 番目のタイプの論理演算子です。
このプログラムでは、and 論理演算子を使用して 2 つの条件を設定しています。生徒には、and（ 小文字）という単語は、条件全体が真であるためには、条件の両方の部分が真でなければならないことを示すために使用されることを説明します。また、論理演算子では複数のブール式や値を組み合わせることができることも説明します。これらは、プログラムでは 1 つのブール値として評価されます。
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。生徒はプログラムを実行します。生徒がプログラムを実行した後、一緒に話し合います。

3. 説明する

プログラムがどのように動作したかを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、2 つの条件を 1 つにまとめることができますか？
□　論理演算子を使用すると、どのような結果が得られますか？
協力して、このプログラムを説明するフローチャートを作成します。1 つまたは両方の条件が偽の場合、条件全体が偽になります。
生徒に真理値表を紹介します。これらの表はブール演算で役立ちます。論理演算子を使用して、プログラムのすべての可能な結果を列挙する式です。

真理値表の例

And の真理値表

条件1 (true/false)	and	条件2 (true/false)	結果
True	and	True	True
True	and	False	False
True	and	True	False
True	and	False	False

SAMPLE

Or の真理値表

条件1 (true/false)	or	条件2 (true/false)	結果
True	or	True	True
True	or	False	True
True	or	True	True
True	or	False	False

Not の真理値表

条件 (true/false)	結果
True	False
False	True

サンプルプログラムの真理値表を作成します。各条件がどのように評価されるかの組み合わせに基づいて、考えられる結果を記入します。各条件の合計が真または偽と評価される理由を説明するように生徒に促します。
条件を組み合わせたプログラムを作成する際に、フローチャートや真理値表がどのように役立つかを生徒と話し合います。

4. 詳しく説明する

条件文の論理演算子をテストするために、さまざまな条件を作成します。
真理値表を参考にして、真理をテストするためのセキュリティを作成するために、さまざまな組み合わせを試すように生徒に指示します。
生徒たちは、表のすべての組み合わせで条件 1 と条件 2 を真から偽に変更したときにプログラムを数回実行し、結果がすべて真になるかどうかを確認します。

その結果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：

□ プログラムの論理演算子を変えたらどうなりましたか？

□ 論理演算子を使ってプログラムを作成する際に、真理値表はどのように使用できますか？

自己評価

生徒にノートで次のことに答えさせます：

□ 論理演算子の使用と 2 つの条件の組み合わせについて、今日何を学びましたか？

SAMPLE

ユニット 8 レッスン 4
ロジックで操作するセキュリティ

ロジックで操作するセキュリティ

生徒は論理演算子を使用して、センサーを使ったセキュリティアラームの作り方を学びます。

主な目的

生徒は次のことを行います

- ☐ セキュリティのためのセンサーの使用について調べます。
- ☐ 二段階のセキュリティプログラムを作成します。

探究への質問

☐ より複雑な条件文を作成するために、センサーと論理演算子をどのように使用できますか。

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

セキュリティについて生徒と話し合います。
生徒たちに、安全な場所（ 電話、コンピュータ、ドアなど）について考えさせます。生徒たちに、これらの場所をどのように安全にしているか話してもらいます。
物理的なセキュリティの選択肢と「 侵入」を発見する方法について話し合います。動きを検知したり、パスワードを設定したりする方法について生徒に考えさせます。

2. 探究する

生徒たちは、2 つのステップを使って、より安全な方法を調べます。
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、品質検査ロボットのモデルの組み立て図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。

生徒に、動きを警告するセキュリティデバイスを作成するように指示します。モデルに含まれている 2 つのセンサーと、これらを使用して物理的な警告装置を作成する方法について話し合います。
生徒に、プログラムがどのように動作するかを説明する擬似コードプログラムを書くように指示します。
プログラムには以下を含めます：

- ☐ 最初に距離センサーを使って動きを検出する。
- ☐ 次にカラーセンサーをパスコードとして使用する。
- ☐ 少なくとも 1 つの論理演算子を含む。

SAMPLE

擬似コードを使用してプログラムを作成するように生徒に指示します。期待される動作を示すために、プログラムの中でコードコメントを使用するように生徒に思い出させます。

必要に応じてプログラムをテストし、修正する時間を生徒に与えます。コンソールにエラーメッセージが表示されないか確認するよう、生徒に注意を促します。

3. 説明する

さまざまなプログラムの例についてクラス全員で話し合い、何が効果的かを探ります。

生徒に次のような質問をします：

- □ センサーをどのように使って、セキュリティ装置を作ることができますか？
- □ あなたのプログラムでは、論理演算子はどのように機能しましたか？
- □ どのようにして、あなたの装置で二段階のセキュリティを作ることができますか？

4. 詳しく説明する

生徒に、セキュリティデバイスに別の論理演算子を使うように指示します。

生徒に、異なる論理演算子を使用することで、プログラムの動作方法を変更する新しい擬似コードを書くように指示します。生徒は、2 つ目の論理演算子をプログラムに追加するか、現在のプログラムの論理演算子を置き換えるかを選択できます。

新しい擬似コードに従ってプログラムを修正する時間を生徒に与えます。生徒はプログラムをテストし、改良します。プログラムをテストしている間、エラーメッセージがコンソールに表示されないか注意します。

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：

- □ 論理演算子を変更したとき、何が起こりましたか？
- □ 複数の論理演算子を追加する場合、どのようなバグが考えられますか？
- □ センサーをどのように使ってセキュリティを確保しましたか？プログラムでセンサーを使用する他の方法（ 使用する他のセンサー）は何ですか？

自己評価

生徒にノートで次のことに答えさせます：

- □ セキュリティの複数のステップを使用することについて、今日何を学びましたか？

SAMPLE

ユニット 8 レッスン 5
脱出しろ!

脱出しろ!

生徒たちは自分たちの脱出ルームのセキュリティシステムを作り、プログラムします。

主な目的

生徒は次のことを行います

- ☐ 脱出ゲームをシミュレートするセキュリティ装置を作成する。
- ☐ 与えられた制約を満たす装置を設計する。

探究への質問

☐ どのように条件を設定すれば、安全な「 部屋」から脱出できるのか。

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

☐ 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

脱出ルームについて生徒と話し合います。
生徒に、脱出ゲームに参加したことがあるか、または聞いたことがあるかを尋ねます。生徒に画像やビデオを見せることも検討します。
このようなゲームでは、参加者が手がかりを使ったりパズルを解いたりして、密室から脱出する方法を考えることを説明します。

ブレインストーミング

脱出ルームで使用できるパズルについて、さまざまなアイデアを出し合います。
グループとして、脱出室で使用できる安全な装置を作るためのアイデアをいくつか出し合います。
その後、小グループに分かれ、他の生徒にセキュリティシステムや脱出防止に何が必要かを尋ねる短いアンケートを作成します。生徒たちは、使用しようと考えている例を共有し、他のアイデアを求めることができます。生徒たちは、何が楽しいか、やりがいがあるか、テーマのアイデアなど、3 つの質問を含める必要があります。生徒にアンケートを作成し、記入する時間を与えます。

2. 探究する

生徒は、脱出ルームで使用するパズルゲームを設計、製作、プログラミングします。

設計と適切なアイデアの選択

生徒は、アンケートで得た意見を考慮し、脱出部屋で使用するセキュリティ装置の設計に取りかかります。
生徒は、脱出室で使用できる装置を設計、製作、プログラミングします。制約条件は以下のとおりです：

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. プログラムに論理演算子を含むこと
5. コードを解読するための少なくとも 2 つのステップを含むこと

生徒たちは、自分の装置のアイデアのスケッチと、プログラミングのアイデアのフローチャートを作成します。

SAMPLE

テストと反復

生徒が自分のアイデアをテストし、分析する時間を設け、必要な箇所を改善します。生徒は、ソリューションを作り始める際に、設定された設計基準とフローチャートに照らして設計をテストし、評価する必要があります。

生徒がモデルのスケッチや写真を使用して、設計の過程を記録するようにします。

時間の許す限り、生徒が自分の設計についてフィードバックを受けられるようにします。他のグループや教師からのフィードバックもよいでしょう。

3. 説明する

生徒は自分のデザインを共有し、それがどのように機能するかを説明します。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：

- ☐ 少なくとも 2 つのステップを持つセキュリティ装置を作るために、あなたの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て説明するよう求めます。
- ☐ 設計を作成するときに、どのような決定をしなければなりませんでしたか？
- ☐ どのような条件文を選びましたか？
- ☐ 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の後、生徒が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、設計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、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？ どのような論理演算子を使用しましたか？

自己評価

生徒にノートで次のことに答えさせます：

- ☐ 与えられた基準に基づいて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 9 Data and Math Functions

ユニット 9 データと数学関数

Unit Introduction: ユニット紹介

このユニットでは、さまざまな種類の輸送手段を探究することで、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学びます。生徒はデータ型と数学関数について調べ、それらがプログラミングでどのように利用できるかを学びます。

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な順序で設計されています：

- ☐ 変数とデータの種類を調べる
- ☐ センサーからデータを取り込む方法を調べる
- ☐ プログラムで数学関数を使用する
- ☐ 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する
- ☐ センサーを現実の問題に応用する

ユニット学習の期待

このユニットでは、さまざまなタイプの輸送手段を調査しながら、さまざまなデータタイプを探究します。データを収集する方法と、センサーから収集したデータを使用する方法を調べます。また、さまざまな数学関数とプログラムへの組み込み方を探究します。センサーとモーターを使用したプログラムで、条件文を効果的に作成する方法を体験します。データ収集方法としてのセンサーの実際の使用方法を探究します。

探求への質問：

- ☐ 数学関数をプログラムに組み込むにはどうすればよいか。
- ☐ プログラムでデータを収集し使用する方法にはどのようなものがあるか。
- ☐ 実世界の問題を解決するためにデータをどのように利用できるか。

SAMPLE

ユニット 9 レッスン 1
データを得るために動き出す

データを得るために動き出す

生徒はデータの収集、表示、使用について調べます。

主な目的

生徒は次のことを行います

- ☐ センサーからデータを取り込む方法を調べる。
- ☐ カセンサーを使用してデータを提供する新しいプログラムを作成する。

探究への質問

☐ どのようにしてデータを集め、表示することができますか。

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

交通手段や、人々が移動する方法について生徒と話し合います。
生徒が、ある場所から別の場所に移動するさまざまな方法を共有するよう促します。普段、どのような交通手段を利用しているか考えさせます。さまざまな種類の交通機関を利用しなければならない状況について話し合います。
さまざまな場所の例を示して、それぞれの場所でどのような交通手段が使われるかを話し合います。徒歩、自転車、さまざまな種類の乗り物、船、飛行機などを含めるようにします。
生徒たちに、これらの場所に行くときに重要な情報やデータは何かを考えさせます。距離、所要時間、移動に必要な速さなどについて話し合います。

2. 探究する

生徒は、センサーからデータを取り込み、データを管理する方法を調べます。
生徒を SPIKE アプリの組み立て図セクションに誘導します。ここで生徒は、スマート・ケトルベル・モデルの組み立て図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。
組み立て図は、https://education.lego.com/en-us/support/spike-prime/building-instructions からも入手できます。

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。

情報を取り込むためにセンサーをどのように使用するか、生徒に考えさせます。生徒は、プログラミングキャンバスの左上の領域を見ます。ライブセンサデータが表示されている場所を示します。生徒に、モデルを動かしてカセンサーを押し、ライブデータストリームを見るように指示します。モデルを動かすと異なるデータが得られることを話し合います。生徒に、データを提供するために距離センサーをプログラムする方法を示させます。
生徒に 3 つのサンプルプログラムを確認させます。コンソールでデータを収集し、管理するのにどれが最適かを話し合います。必要に応じて、生徒にプログラムを調査させます。

SAMPLE

サンプルプログラム 1

```
from hub import port
import runloop
import distance_sensor

async def main():
    #センサーから距離の値を読み込む
    dist_mm = distance_sensor.distance(port.F)
    print('mm:',dist_mm)

runloop.run(main())
```

サンプルプログラム 2

```
from hub import port
import runloop
import distance_sensor

async def main():
    #センサーから距離の値を読み込む
    while True:
        dist_mm = distance_sensor.distance(port.F)
        print('mm:',dist_mm)

runloop.run(main())
```

サンプルプログラム 3

```
from hub import port
import runloop
import distance_sensor, force_sensor

async def main():
    #センサーから距離の値を読み込む
    while True:
        if force_sensor.pressed(port.B):
            dist_mm = distance_sensor.distance(port.F)
            print('mm:',dist_mm)
            await runloop.sleep_ms(1000)

runloop.run(main())
```

生徒に 3 つのプログラムを調査する時間を与えます。距離センサーはセンチメートルではなくミリメートルで値を返すことに注意してください。生徒は、このコードの最後に a/10 を加えて 10 で割るように変更することで、ミリメートルからセンチメートルに変換することができます。

```
dist_cm = distance_sensor.distance(port.F)/10
```

```
from hub import port
import runloop
import distance_sensor
import force_sensor

async def main():
    # センサーから距離を読み取る
    while True:
        if force_sensor.pressed(port.B):
            dist_cm = distance_sensor.distance(port.F)/10
            print('cm:',dist_cm)
            await runloop.sleep_ms(1000)

runloop.run(main())
```

3. 説明する

それぞれの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：

☐ コンソールに出力されたデータは、それぞれのプログラムでどのように見えたか？

☐ どのプログラムが、さまざまなデータポイントを収集するのに最適な方法ですか？

☐ コンソールの外でデータを整理する方法にはどのようなものがありますか？

データを表に表示するように生徒に促します。

整数、浮動小数点、文字列という異なるデータ型があることを生徒に思い出させます。距離センサーの下にあるナレッジベースを参照し、生徒がここで使用されているデータのタイプを判断できるようにします。自分のセンサーからどのような種類のデータが提供されたかを話し合います。

4. 詳しく説明する

力センサーを使用してデータを提供する新しいプログラムを作成するように生徒に指示します。

生徒に、力センサーからデータを取得する方法を調べさせます。力センサーを押してニュートン数を取得し、生徒が簡単に情報を集めて表に整理できるようにデータを表示するアイデアについて話し合います。コンソールまたはハブに表示するアイデアを検討します。

生徒が探究し、プログラムを開発する時間を与えます。生徒が自分たちのアイデアやデータを他のグループと共有するように促します。

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：

☐ 調査した 3 つのサンプルプログラムの違いは何ですか？何が同じですか？

☐ 力センサーを追加するときに何が起こりましたか？

☐ 参照するデータを表示したり、表を作成するためにデータを取り込んだりするには、どのような方法がありますか？

自己評価

生徒にノートで次のことに答えさせます：

☐ ロボットを使ってデータを収集することについて、今日は何を学びましたか？

SAMPLE

ユニット 9 レッスン 2

データを求めて自転車に乗る

データを求めて自転車に乗る

主な目的

生徒は次のことを行います

- ☐ 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する。
- ☐ 数学関数を使用して、自転車モデルの速度を増減させるプログラムを作成する。

生徒は、プログラムで数学関数を使用することを調べます。

探究への質問

☐ プログラムで数学関数をどのように使用できるか。

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

☐ 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

移動手段としての自転車の乗り方について生徒と話し合います。自転車での移動について具体的に話してもらい、自転車での移動の長所と短所を確認します。自転車に乗る可能性のある場所、特に徒歩や他の交通手段を利用するよりも自転車に乗る方が理にかなっていると思われる場所について話し合います。必要であれば、例を挙げて生徒を促します。

2. 探究する

生徒は、交通手段として自転車に乗ることについて調べます。
SPIKE アプリの組み立て図セクションに生徒を誘導します。ここで生徒はスマート自転車モデルの組み立て図にアクセスします。
生徒にモデルを組み立てるよう指示します。
組み立て図は、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、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。

生徒が、スマート自転車をどのようにプログラムして動かすかを考えるように促します。生徒は、自転車が一定の速度で前進するプログラムを書きます。
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     #モーターをペアにする
7     motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)
```

SAMPLE

```
    #速度250で720度分直進する
    motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=250)

runloop.run(main())
```

数学で動く

生徒が自転車を動かせるようになったら、プログラムに追加して自転車の速度を速くしたり遅くしたりすることに挑戦させます。ただし、プログラムに新しい動きを加えるだけではいけません。その代わりに、生徒たちは変数と数学関数を使って速度を変える必要があります。使用できる数学関数を以下に紹介します。これらはすべて、生徒にとって一般的でなじみのあるものでなければなりません。複雑な数学コードの行では、演算の順序に従うことを生徒に思い出させます。

関数	記号	サンプル	結果
加算	+	print(10+2)	12
減算	-	print(10-2)	8
乗算	*	print(10*2)	20
除算	/	print(10/2)	5

これらの数学関数を使用して、生徒は自転車の速度を増減させるプログラムを作成できるはずです。

サンプルプログラム

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

async def ride(bike_velocity):
    #バイクの速度をライトマトリックスに表示して直進する関数ride()を定義
    await light_matrix.write(bike_velocity)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=bike_velocity)

async def main():
    bike_velocity = 250
    #モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)
    #速度250で720度分直進する
    while True:
        #左ボタン押されたら50ずつ減速
        if button.pressed(button.LEFT):
            bike_velocity -= 50
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

        #右ボタン押されたら50ずつ加速
        if button.pressed(button.RIGHT):
            bike_velocity += 50
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

runloop.run(main())
```

生徒にプログラムを探究する時間を与え、足し算と引き算の値を変えてみるようにします。7 行目に次のコードがあることに注意してください。

```
light_matrix.write(str(bike_velocity))
```

`bike_velocity` の前の `str` は、整数を文字列にキャストしてライトマトリクスに書き込めるようにするプログラムです。キャストとは、特定の状況下でデータをある型から別の型に変換する方法です。このプログラムでは、データ型を恒久的に変更するのではなく、ライトマトリクスに書き込むためだけに変換しています。変数は整数のままです。

3. 説明する

この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：

- ☐ プログラムで数学関数をどのように使うことができますか？
- ☐ 数学関数を使うことで、プログラムの動作はどのように変わりましたか？
- ☐ 速度を変数で制御することの利点は何ですか？

関数 ride() は、同じコードをプログラムの 2 つの異なる場所で実行できるようにするために作りました。関数を作成することで、プログラムの長さを最小限に抑え、またプログラムをより効率的にすることができます。

4. 詳しく説明する

他の生徒から共有された内容に基づいて、新しい数学関数を試すために生徒がプログラムを修正し続けられるようにします。生徒に、自分のプログラムに新しい関数を追加してみて、動きがどのように変わるかを確認するよう求めます。生徒に自分のプログラムを共有するように指示します。

サンプルプログラム

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

async def ride(bike_velocity):
    #バイクの速度をライトマトリックスに表示して直進する関数ride()を定義
    await light_matrix.write(bike_velocity)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity=bike_velocity)

async def main():
    bike_velocity = 250
    #モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.E)
    #速度250で720度分直進する
    while True:
        #左ボタン押されたら1/2に減速
        if button.pressed(button.LEFT):
            bike_velocity /= 2
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

        #右ボタン押されたら2倍に加速
        if button.pressed(button.RIGHT):
            bike_velocity *= 2
            await ride(bike_velocity)
            await runloop.sleep_ms(2000)

runloop.run(main())
```

生徒がプログラムを探究し、さまざまな数学関数を使えるようにします。

SAMPLE

注：数値を分割すると、自動的に浮動小数点数に変換されます（ 18行目）。8行目の `velocity` パラメータは整数でなければなりません。このため、`bike_velocity` は整数にキャストされます。

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：

☐ プログラムに数学関数を追加したらどうなりましたか？

☐ 数学関数のプログラミングについて何を学びましたか？

☐ 割り算が使われたとき、どのような問題が発見されましたか？

☐ 自転車を動かすモーターが 1 つしかないのに、なぜモーターペアが使われたのか？

☐ 私たちのプログラムにどのような反復が可能か？

自己評価

生徒にノートで次のことに答えさせます：

☐ プログラムでの数学関数の使用について、今日は何を学びましたか？

SAMPLE

ユニット 9 レッスン 3

歩数計

歩数計

主な目的

生徒は次のことを行います

- ☐ 変数を使用して数学的計算をプログラムに組み込む。
- ☐ 足音を測定するプログラムを作成する。

生徒は、歩数を数えながら変数とデータを使って調べます。

探究への質問

- ☐ プログラム内で数学的な計算を追加するにはどうすればよいですか。

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

ウォーキングや歩数を数えることについて生徒と話し合います。何人の生徒がそのようなことができるデバイスを持っていますか?利用可能なアプリを使用している生徒は何人いますか?
生徒の小グループを横に一列に並べさせ、5 ～ 10 歩前進させる。なぜ歩いた距離が違うのか?アプリケーションはそれにどう対処するのか?一歩の歩幅はみんな同じですか?

2. 探究する

生徒は、移動手段として徒歩を選択する際に、スマートデバイスを使用して歩数を追跡することを調べます。
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、万歩計モデルの組み立て図にアクセスできます。生徒が作る必要があるのは万歩計だけで、スタンドは必要ありません。生徒にモデルを作るように指示します。
組み立て図は、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、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。

何歩進んだかを判断する方法について生徒に考えさせます。生徒に、歩数を判断するためにどのセンサーを使用できるかをブレインストーミングで考えさせます。生徒をモーションセンサーに誘導します。特に、ハブのロール、ピッチ、ヨーについて考えさせます。
自分の体に取り付けたときのモデルの動き方について話し合います。ハブのピッチとは、ハブの前面が前方または後方に傾くことです (伸ばした手の指先が上または下に傾くのと同じです)。 ハブのロールとは、ハブの側面が右または左に傾くことです (伸ばした手を転がすと小指が下または上に転がるのと同じです) 。ハブのヨーとは、ハブの上部が上を向いたまま右または左に回転することです (伸ばした手を回すと指が最初は前を向いていたが、右または左を指すように動くのと同じです)。
生徒はステップを測定するプログラムを作成する必要があります。

SAMPLE

サンプルプログラム

```
from hub import light_matrix,motion_sensor,button
import runloop

async def main():
    step = 0
    # ヨー角を 0 にリセットする
    motion_sensor.reset_yaw(0)
    await light_matrix.write('Walk!')

    while True:
        # 傾き角を取得
        tilt = motion_sensor.tilt_angles()
        # タプルからヨー角を取り出す
        yaw = tilt[0]

        if yaw > 75:
            step += 1
        elif yaw < -75:
            step += 1
        elif button.pressed(button.LEFT):
            await light_matrix.write(str(step))
            break

runloop.run(main())
```

生徒にプログラムを探究し、異なる値を試す時間を与える。
データを作成したり比較したりするときに使える新しい数学関数をいくつか生徒に紹介する。

関数	記号
未満	<
以下	<=
超過	>
以上	>=
等しい	==
等しくない	!=

これらの新しい関数が、あなたのステップを得るためにプログラムでどのように使われたかを議論しなさい。

プログラムの改良

歩数計はヨー角度を取得するとき、1 秒間に何度もデータを収集しました。プログラムは、角度が歩幅の一部であることを知りません。
そこで、歩幅を特定するための変数を追加して、ヨー角が 1 回だけ収集されるようにします。

SAMPLE

サンプルプログラム

```
from hub import button, light_matrix, motion_sensor
import runloop
import time

def time_since (some_time):
    return time.ticks_diff(time.ticks_ms(), some_time)

async def main():
    # 変数を定義
    step =0
    true_step = False

    await light_matrix.write('Walk!')
    while True:
        # 傾き角を取得
        tilt = motion_sensor.tilt_angles()
        # ヨー角を取り出す
        yaw = tilt[0]

        if yaw > 75 and true_step is False:
            step += 1
            true_step = True
        elif yaw < -75 and true_step is True:
            step += 1
            true_step = False
        elif button.pressed (button.LEFT):
            await light_matrix.write(str(step))
            break

runloop.run(main())
```

生徒にプログラムを探究する時間を与える。

3. 説明する

プログラムがどのように動作したか、生徒と話し合う。

生徒に次のような質問をする：

- ☐ 歩数は何を使って数えたのですか？
- ☐ 万歩計の精度はどのくらいですか？説明してください。
- ☐ 万歩計が歩数を数えすぎていたのはなぜだと思いますか？

SAMPLE

4. 詳しく説明する

速さを計算する数学的な計算をプログラムに追加するように生徒に指示します。速度は、時間に対する距離で測定されることを忘れないでください。

プログラムをテストする際には、データが正確であること、プログラムが計画どおりに実行されていることを確認するために、コンソールに情報を出力することをお勧めします。

サンプルプログラム

```
from hub import light_matrix, motion_sensor, button
import runloop
import time

def time_since(some_time):
    return time.ticks_diff(time.ticks_ms(), some_time)

async def main():
    # 変数を定義
    step = 0
    true_step = False
    stride = 3
    distance_walked = 0
    time_walked = 0
    speed = 0
    start_time=0
    elasped = 0

    await light_matrix.write('Walk!')
    # タイマースタート
    start_time = time.ticks_ms()
    while True:
        # 傾き角を取得
        tilt = motion_sensor.tilt_angles()
        # タプルからヨー角を取り出す
        yaw = tilt[0]

        if yaw > 75 and true_step is False:
            step += 1
            true_step = True
        elif yaw < 75 and true_step is True:
            step += 1
            true_step = False
        elif button.pressed (button.LEFT):
            await light_matrix.write(str(step))
            break
    # タイマーストップ
    elasped = time_since(start_time)
```

SAMPLE

```
    # 計算
    distance_walked = step * stride
    time_walked = elasped/1000
    speed = distance_walked / time_walked
    # 歩いた時間と速度を 2 桁にまとめる
    time_walked = str(round(time_walked, 2))
    speed = str(round(speed, 2))
    # 計算した角データをコンソールに表示する
    print('You took' + str(step) + 'steps')
    print('Your stride is' + str(stride) + 'feet')
    print('You walked for' + str(time_walked) + 'seconds')
    print('You walked' + str(distance_walked) + 'feet')
    print('Your speed is' + str(speed) + 'feet per second')

runloop.run(main())
```

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：

- ☐ true_step 変数を追加したとき、何が起こりましたか？
- ☐ true_step 変数は、どのようにしてモーションセンサーが読み取り値を収集しないようにしますか？
- ☐ プログラミングの計算について何を学びましたか？
- ☐ 歩数計を改善するために、どのような繰り返しができますか？

自己評価

生徒にノートで次のことに答えさせます：

- ☐ プログラムで計算を使用することについて、今日何を学びましたか？

SAMPLE

ユニット 9 レッスン 4
動かしてみよう

動かしてみよう

生徒は、数学的関数を使用して、自動車のようなモデルがどのように前進したり方向転換したりするかを調べます。

主な目的

生徒は次のことを行います

- ☐ 前進および方向転換を行う運転ベースをプログラムする。
- ☐ 数学的関数を含むプログラムを作成する。

探究への質問

☐ 数学的な関数を使って、どのように方向を変えることができるでしょうか。

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

☐ 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

ドライバーが自動車をどのように動かすかについて、生徒と話し合います。
ドライバーが自動車の動きを制御するさまざまな方法について生徒を討論に導きます。例えば、ドライバーは自動車を前進させたり後退させたりすることができます。ドライバーは自動車を停止させることができる。ドライバーは、さまざまな角度で左折や右折もできる。
生徒には、探究するドライビング・ベース・モデルがどのように自動車と同じように動くかをビデオで見せる。トレーニングキャンプ 1 のレッスンで使用した引きつけるビデオを使用することができます。ビデオは以下から入手できます。
https://education.lego.com/en-us/lessons/prime-competition-ready/training-camp-1-driving-around#ignite-a-discussion.
生徒とビデオについて話し合う。

2. 探究する

生徒は、交通手段の一種である自動車のようなモデルについて調べます。
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、ドライビング ベース モデルの組み立て図にアクセスできます。生徒に、ドライビングベースモデル 1 を作るように指示します。
組み立て図は、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、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。

生徒が、ドライビングベースが動くようにプログラムする方法について考えるように促します。生徒は、モーターをプログラムすることで、車両が前進、後退、旋回するプログラムを書く必要があります。プログラム作成のヒントが必要な場合は、「 データを求めて自転車に乗る」のレッスンで使用したサンプル プログラムを参照するよう生徒に提案します。このプログラムと同様に、生徒は速度の加算（ 速度を上げる）と減算（ 速度を下げる）の方法を含める必要があります。
車のような乗り物を運転するときは、制限速度に従わなければならないことを生徒に思い出させます。前進するだけでなく、モデルのハブに自分のドライビングベースの速度を表示し、速度計として機能させることにも挑戦させます。

SAMPLE

サンプルプログラム

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

async def main():
    driving_velocity = 250
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
    #ride 関数を定義
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)

    # 前進する条件と速度を設定する
    while True:
        if button.pressed(button.LEFT):
            driving_velocity -= 50
            await ride()
            await runloop.sleep_ms(3000)
        elif button.pressed (button.RIGHT):
            driving_velocity += 50
            await ride()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

生徒にプログラムを探究する時間を与えます。プログラムを実行しながら、右または左のボタンを押し続け、走行ベースがどのように 50 を足したり引いたりしながら動き続けるかを確認するように促します。生徒は、ハブ上で毎回新しい速度を見ることができます。

曲線走行

自動車は直線だけで走行することはあまりないので、プログラムを修正してさまざまな方向に移動することに挑戦します。生徒には、どのように曲がるかを設定するのに役立つ数学関数を使用するように促します。

サンプルプログラム

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

async def main():
    # 速度の初期設定値
    driving_velocity = 250
    # ターンの距離を設定
    turning_left = -360
    turning_right = 360
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
```

SAMPLE

```
    #ride 関数と turn 関数を定義する
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees (motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)

    async def turn_left():
        await light_matrix.write(str(turning_left/4))
        await motor.run_for_degrees (port.C, turning_left, 250)

    async def turn_right():
        await light_matrix.write(str(turning_right/4))
        await motor.run_for_degrees(port.D, turning_right, 250)

    # 前進する速度と条件を設定
    while True:
        await ride()
        if button.pressed (button.LEFT):
            driving_velocity -= 50
            turning_left -= 360
            await turn_left()
            await runloop.sleep_ms(3000)
        elif button.pressed (button.RIGHT):
            driving_velocity += 50
            turning_right += 360
            await turn_right()
            await runloop.sleep_ms(3000)

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う。生徒に次のような質問をします：

□　なぜプログラムは速度の加減算を可能にしたのですか？

□　どのようにプログラムを修正し、旋回や異なる方向への移動を含めることができましたか？

□　ボタンを押さなくてもドライブベースが動き続けたのはなぜですか？

回転を表示する際、実際の車輪の回転度をドライブベースがどれだけ回転したかに変換する数式が使用されている。

この場合、1 つのモーターが 360 度回転すると、車輪が一回転し、ドライブベースが 90 度回転する。数式を使えば、360/n=90 の答えを求めることができる。この場合、n=4 である。従って、light_matrix.write(str(turning_left/4)) と light_matrix.write(str(turning_right/4)) の行でこの数式が使われているのがわかる。

4. 詳しく説明する

生徒にプログラムのバグを認識させます。

以下の各プログラムとその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を生徒に議論させます。生徒がプログラムを入力するか、クラス全体で確認します。

クラスとして変更を加え、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ことを検討します。

SAMPLE

デバッグアクティビティ 1

プログラムのエラーを見つけます。プログラムが実行され、プログラムのその部分を読み取るアクションが取られるまで、エラーが見つからないことがあることを生徒に思い出させます。

```
 1: from hub import port, light_matrix, button
 2: import runloop
 3: import motor_pair
 4:
 5: async def main():
 6:    # 速度の初期設定値
 7:    driving_velocity = 250
 8:
 9:    # モーターをペアにする
10:    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
11:
12:    #ride 関数を定義する
13:    async def ride():
14:        await light_matrix.write(str(driving_velocity))
15:        await motor_pair.move_for_degrees (motor_pair.PAIR_1,0, velocity = driving_velocity)
16:
17:    # 前進する速度と条件を設定
18:    while True:
19:        if button.pressed (button.LEFT):
20:            driving_velocity -= 50
21:            await ride()
22:            await runloop.sleep_ms(3000)
23:
24:        elif button.pressed (button.RIGHT):
25:            driving_velocity += 50
26:            await ride()
27:            await runloop.sleep_ms(3000)
28:
29: runloop.run(main())
```

エラーメッセージ

```
Traceback (most recent call last):
File "drive base 1", line 29, in <module>
File "drive base 1", line 21, in main
File "drive base 1", line 15, in ride
TypeError: function missing 1 required positional arguments
```

エラーメッセージには、正しくない行がいくつか示されています。ただし、実際にはエラーのある行は 1 行だけです。15 行目では、ドライビングベースに前進する距離を指示する度数が削除されています。

このコード行は、

```
await motor_pair.move_for_degrees (motor_pair. PAIR_1, 720, 0, velocity driving_velocity)
```

とする必要があります。

ボタンが押されてベースが動くと、関数を参照し、前進する方法を知るために必要な回転度数を持っていません。

SAMPLE

デバッグアクティビティ 2

プログラムのエラーを見つけます。プログラムが実行され、プログラムのその部分を読み取るアクションが取られるまで、エラーが見つからないことがあることを生徒に思い出させます。

```
 1: from hub import port, light_matrix, button
 2: import runloop
 3: import motor_pair
 4:
 5: driving_velocity = 250
 6:
 7: async def main():
 8:   # モーターをペアにする
 9:   motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
10:
11:   #ride 関数を定義する
12:   async def ride():
13:       await light_matrix.write(str(driving_velocity))
14:       await motor_pair.move_for_degrees (motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)
15:
16:   # 前進する速度と条件を設定
17:   while True:
18:       if button.pressed (button.LEFT):
19:           driving_velocity -= 50
20:           await ride()
21:           await runloop.sleep_ms(3000)
22:
23:       elif button.pressed (button.RIGHT):
24:           driving_velocity += 50
25:           await ride()
26:           await runloop.sleep_ms(3000)
27:
28: runloop.run(main())
```

エラーメッセージ

```
Traceback (most recent call last):
File "drive base1", line 28, in <module>
File "drive base1", line 19, in main
NameError: local variable referenced before assignment
```

エラーメッセージは生徒がボタンを押すまで表示されません。生徒はエラーが 19 行目で発生していることを認識する必要があります。エラーメッセージは、ローカル変数が代入の前に参照されていることを示しています。これはプログラムのタイプミスで、プログラムが間違った場所に設定された変数を参照しようとしています。変数は main 関数の中で定義されるべきです（ 原因は5行目です）。5行目の一行を main 関数の中へ移動すればエラーはなくなります。

SAMPLE

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：

☐ 数学の関数を使用して、直進時や旋回時のドライビングベースの動きをどのように変えることができましたか？

☐ 複雑なプログラムを作成するときに、どのようなエラーが起こりうるか。このようなエラーをデバッグするにはどのような方法がありますか？

☐ 1つのモーターの代わりに、どのようにドライブベースを使って回転させることができますか？

自己評価

生徒にノートで次のことに答えさせます：

☐ 1つのプログラムで複数の数学関数を使用することについて、今日何を学びましたか？

SAMPLE

ユニット 9 レッスン 5

駐車場

駐車場

主な目的

生徒は次のことを行います

- ☐ 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する。
- ☐ センサーを実際の問題に応用する。

生徒は、自律的に駐車するための車両のプログラミングについて調べます。

探究への質問

- ☐ 数学の関数は自律走行車の作成をどのようにサポートできるか。

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
- ☐ 駐車場のプリントまたは床に貼ったテープ

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）。
- ☐ 動かしてみようレッスンで使用したドライビングベースモデルが生徒の手元にあることを確認する。

1. 引きつける

セルフパーキングに焦点を当てた自律走行車に生徒と話し合います。

自律駐車する車のビデオや画像を見せた後、これらの車がどのように機能し、現在どのように使用されているかについて生徒と話し合います。

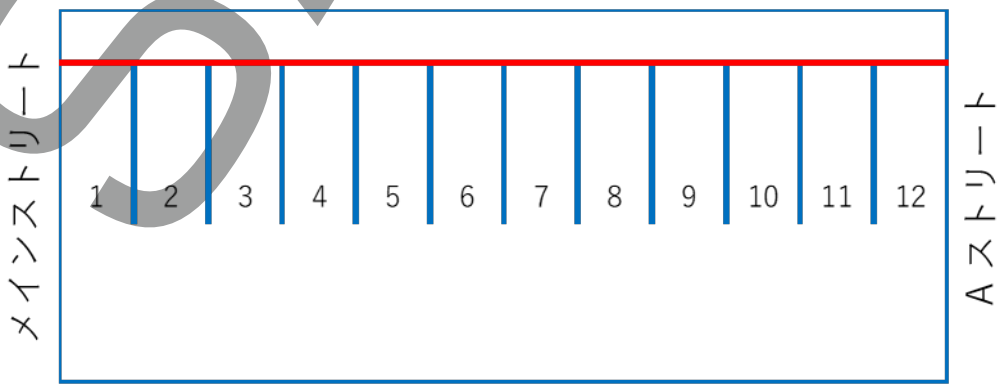
自分で駐車できる車の例に焦点を当て、自分で縦列駐車できる車種はいくつかあります。これらの車がどのように機能するのか、センサーがどのような役割を果たしているのか話し合います。

2. 探究する

生徒は、補助なしで駐車できるように、ドライビングベースを計画し、プログラムします。

各チームに、割り当てられた駐車スペース（1 ～ 12）と、駐車場に入るための道路を指定します。

生徒に駐車場の写真を見せます。この例を使用しても、床に貼ったものを使用しても良いです。クラスでの話し合いには、生徒全員が駐車場を見る必要があるため、オーバーヘッドプロジェクターで写真を映す方が簡単かもしれません。



SAMPLE

生徒と一緒にルールを確認します：

☐ 2 台の車は同時に駐車場に入らなければならない。

☐ どちらの車も駐車スペース以外では止まってはならない。

☐ 駐車場に入ったら、誰も車に触れてはならない。

☐ 車がバックできるのは、スペースから出るときだけです。

☐ 車のどの部分も線を越えて動くことはできない。

2 台の車を安全に駐車する方法をクラスで話し合います。車同士がすれ違う必要がある場合はどうなるかを考え、同時に進入する車の各ペアを確認します。

駐車の課題

生徒は、他の車両に衝突することなく、指定されたスペースに自律駐車するように車両をプログラムします。

生徒に、センサーが他の車両との衝突を避けるためにどこに付ければよいか考えさせます。生徒は、必要に応じて、アプリの組み立てセクションをヒントとして使用できます。

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。

生徒が車を駐車するようにプログラムする時間を与えます。生徒に、プログラムに含める必要のある以下の要素を教えます：

☐ 駐車場全体の距離を使用すること。

☐ 複数の数学関数を使用すること。

☐ 距離センサーを使用すること。

☐ 生徒にロボットのプログラムを作成させ、2 つのチームに分かれて自律走行車の駐車を完成させる。

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：

正しい駐車スペースへの移動をどのように処理するかを説明してもらいます。

☐ 空いている駐車場と満車の駐車場のどちらが停めやすかったですか？

☐ どのような問題を克服するのが最も困難でしたか？

☐ この挑戦で簡単だったことは何ですか？

4. 詳しく説明する

すべてのロボットが駐車場に揃ったので、そろそろ出発の時間です。

次のシナリオのいずれかを選択します。

駐車場を空にする：

A. すべてのロボットが、入場した順番に、入場した通りに出る。

B. すべてのロボットが、入ってきた順番とは逆の順番で、入ってきた場所の反対側の通りに出る。

C. すべてのロボットが同時にメイン通りに出る。

D. 奇数スペースのすべてのロボットがメインストリートへ同期して退場し、その後に偶数スペースのすべてのロボットが A ストリートへ同期して退場する。

クラスで話し合い、シナリオを選択します。そして、擬似コードを書いてロボットをプログラムします。全員の準備ができたら、駐車場を設置し、すべてのロボットを駐車場から退場させます。

SAMPLE

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：

☐ 問題を解決するための複数の方法をどのように思いつきましたか？

☐ 自律走行車の一部としてセンサーをどのように使用できましたか？

☐ プログラムに数学の機能をどのように組み込むことができましたか？

自己評価

生徒にノートで次のことに答えさせます：

☐ 1つのプログラムで複数の数学関数を使用することについて、今日何を学びましたか？

SAMPLE

ユニット 9 レッスン 6

私の交通手段

私の交通手段

生徒は、自宅から学校までの交通手段を自分で作成し、プログラムします。

主な目的

生徒は次のことを行います

☐ 学校へ行くための移動手段を設計、製作、およびプログラムする。

探究への質問

☐ プログラミングの数学的機能を念頭に置いて乗り物を設計す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

☐ 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

生徒が自宅から学校までどのように通学しているかについて、生徒と話し合います。

生徒たちに、毎日の通学路がどのようになっているかを尋ねます。生徒を場所別に小グループに分け、話し合うことを検討します。通学路を説明する際に、地図を作ったり、画像を共有したりするのも良いでしょう。

グループ全体で、生徒の通学路の様子について話し合います。生徒は水辺を渡ったり、橋を渡ったりする必要があるのか、山を越えるのか、街を通るのかを考えます。他にどんな例があるかを聞いてみましょう。

ブレインストーミング

さまざまな種類の地形に対応できる通学用の乗り物について、さまざまなアイデアを出し合います。

小グループに分かれ、必要な地形を簡単に移動できる新しいタイプの通学用乗り物のアイデアを出し合います。各グループは、それぞれの通学ニーズを満たすユニークなアイデアを考えます。

さらにアイデアを深めるために、小グループに分かれて、他の生徒に通学に必要な乗り物について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、他のアイデアを求めることができます。生徒たちは、安全に通学するためには何が効果的か、何が楽しいか、何が必要かについて、3 つの質問を含めます。

生徒にアンケートを作成し、記入する時間を与えます。生徒は、アンケートの結果からブレインストーミングのリストに追加のアイデアを加えることができます。

2. 探究する

生徒は、通学に必要な乗り物を設計、製作、プログラムします。

正しいアイデアを選択して設計する

生徒は、アンケートから得た意見を考慮し、乗り物の設計を開始します。

生徒は、自分たちを学校に連れて行くために使用できる移動手段を設計、構築し、プログラムします。制約条件は次のとおりです：

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。

SAMPLE

3. 少なくとも 1 つのセンサーを使用すること。
4. プログラムに数学関数が含まれていること。
5. 少なくとも 2 種類の地形に対応すること。

生徒は、組み立てのアイデアのスケッチとプログラミングのアイデアのフローチャートを作成します。

テストと反復

生徒が自分のアイデアをテストして分析し、必要に応じて改善する時間を設けます。生徒は、ソリューションを作り始めるときに、設定された設計基準とフローチャートに照らして設計をテストし、評価する必要があります。
生徒がモデルのスケッチや写真を使い、デザインのプロセスを記録するようにします。
時間の許す限り、生徒が自分たちのデザインについてフィードバックを受けられるようにします。これは、他のグループや教師からでも良いです。

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：

- ☐ 少なくとも 2 種類の地形で機能する乗り物を作成するために、モデルをどのようにプログラムしましたか?生徒たちに、プログラムのコメントを共有してもらい、説明してもらいます。
- ☐ 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?
- ☐ どのような数学関数を選択しましたか?
- ☐ 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の後、生徒が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、設計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。
生徒が最終的な設計を発表し、それがどのように基準を満たしたかを説明できるようにします。
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?
- ☐ この課題を解くためにどのようなアプローチをとりましたか?
- ☐ どのような数学関数を取り入れたか、またその理由は?

自己評価

生徒にノートで次のことに答えさせます：

- ☐ 今日、与えられた基準に基づいて自分のデザインを作ることについて何を学びましたか?

SAMPLE

Unit 10 Lists
ユニット 10 リスト

Unit Introduction: ユニット紹介

このユニットでは、生徒がゲームで頭を使い、動きで筋肉を伸ばすことを通して、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究します。生徒たちは、プログラムにおけるリストの作成方法と使用方法を学び、センサーを使ってデータを収集し、そのデータをリストに整理します。リストの作り方だけでなく、リストの修正方法も習得します。

レッスンは、生徒が以下の分野のスキルと知識を向上させることができるような順序で設計されています：

- ☐ リストの作成と活用
- ☐ リストを使った複合条件式のコード
- ☐ 1 つのプログラム内で 2 つのリストを作成し、比較する
- ☐ リスト内の指定された位置またはインデックス値を見つける
- ☐ リストに含めるデータを生成する方法を探究する
- ☐ 2 つのリストを足して 1 つのリストにするようにプログラムを修正する

ユニット学習の期待

このユニットでは、生徒が頭や筋肉をストレッチやゲームを通して鍛えることができるプログラムを作成し、その中でリストを活用する方法を探究します。また、変数やリストを簡単に見分けられるように命名することの重要性に重点を置きながら、リストの効果的な利用方法、比較方法、組み合わせ方を体験します。この単元では、さまざまな種類の運動やマインドゲームを探究しながら、リストのデータ収集を支援するセンサーを活用します。

探求への質問：

- ☐ リストはプログラムでどのように使われるのか？
- ☐ リストからどのような情報を得ることができますか？
- ☐ リストで使用するために、センサーからどのように情報を生成できるか？

SAMPLE

ユニット 10 レッスン 1
文字のリストアップ

文字のリストアップ

生徒は、アンプラグド・アクティビティを通してリストの作成方法を調べ、
その知識をリストの作成とハブへの文字の表示に移します。

主な目的

生徒は次のことを行います

- ☐ リストの作成と利用
- ☐ リストを使って複合条件式をコードする。

ボキャブラリー

リスト

探究への質問

- ☐ 変数に複数の値を指定する必要があるのはなぜですか。

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

生徒とリストについて話し合います。
自分がどのようにリストを使っているか、または他の人がどのようにリストを使っているかを話し合います。生徒にリストの種類を挙げてもらいます（ 食料品リスト、ToDo リスト、選手名簿など）。
生徒に考えさせます：

- ☐ なぜ私たちはリストを使うのか。
- ☐ リストは私たちに何を与えてくれるか。
- ☐ リストの項目をどのように参照するか。
- ☐ リストはどのように整理できるか。

時間の許す限り、生徒たちにさまざまな種類のリストを作成させ、そのリストを共有して、他の生徒がリストに追加する項目や要素があるかどうかを確認します。リストの種類には、果物や食べ物の種類、色、住む場所の種類、音楽の種類などがあります。

2. 探究する

生徒は、アンプラグド活動を通して、リストの作成方法を調べます。

リストを作成する：アンプラグド

生徒たちに、セットから 2x4 のブロックをすべて取り出すように指示します。各自 5 個のブロック（ 黄、緑、紫、赤、青）を持ち、それぞれに 1 ～ 5 の数字を 1 つずつ書いた 5 枚の小さな紙または付箋を用意します。

生徒に、自分のセットの 5 個のブロックの色を挙げさせるか、名前を付けさせます。生徒はそれぞれ、黄色、緑色、紫色、赤色、青色をブロックの色として認識します。

生徒には、これが**リスト**の値であることを説明します。リストは、複数の値を格納できる変数です。Python でリストをプログラムするには、生徒は次のことを行う必要があります。

SAMPLE

リストに名前を付け、含まれる項目や**要素**を列挙し、リスト内の各要素の位置や**インデックス**の値に注意します。
生徒に、5 つのブロックを番号のついた紙か付箋に並べるように指示します。順番は特に決めません。リスト内の各ブロックの番号位置またはインデックス値を含め、選択した順序でリスト全体を書き出すように指示します。

ラウンド 1
協力し合って、各自のリストにあるさまざまなインデックスを使ってブロックを積み上げるゲームを行います。
自分のリスト(番号) を書き込んだ紙や付箋を袋やボウル、箱などに入れます。各パートナーは、その中からランダムに番号(紙や付箋) を選びます。リストのインデックスを表すその番号によって、ブロックを積み上げる順番が決まります。各生徒は、最後に 5 個のブロックを積み上げます。
パートナーがブロックを積み上げた後、色のパターンを比較させます。色のパターンが同じか違うかを生徒に尋ねます。その理由を話し合うように促します。

ラウンド 2
生徒たちに、両方 (両者) の紙または付箋を袋やボウルなどに入れるよう指示します。両方の紙を使い、5 個のブロックを積み上げます。これによって要素の重複が可能になります。生徒に 2 つのリストのうち 1 つを選択させ、積み上げゲームを繰り返します。
終了したら、新しいパターンを調べさせます。繰り返している色があるかどうかを生徒に確認させます。なぜそうなったかを話し合います。

リストのプログラミング

ハブをプログラミングして、アルファベットの中からランダムな文字を表示させます。
生徒に、SPIKE プライムセットからハブを取り出すように指示します。次に、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。
Python でリストを作成する方法について生徒と話し合います。議論をサポートするサンプルプログラムを生徒に見せます。生徒に、このコードをプログラミング・キャンバスに入力し、プログラムを実行するように指示します。

サンプルプログラム

```
from hub import light_matrix, button
import runloop

async def main():
    # リストを定義
    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'T, T, K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    # リストの 4 番目をライトマトリックスに表示する
    await light_matrix.write(letters[4])

runloop.run(main())
```

生徒は自分のライトマトリクスに "E" の文字が表示されているのを確認します。一緒にプログラムについて話し合います。[4] がインデックスであることを生徒と共有します。Python では、リストの最初の項目はインデックス [0] にあり、インデックス [4] はリストの 5 番目の項目になります。

3. 説明する

生徒と、自分たちのリストとプログラムがどのように動いたかを話し合います。

SAMPLE

生徒に次のような質問をしましょう：

☐ アンプラグドでは、どのようにリストを作成したのですか?リストを使ってどのようにスタックを作りましたか?

☐ Python でどのようにリストをプログラムできますか?

☐ ハブの上に表示される文字は何によって決まりますか?

☐ ハブの上にランダムな文字を表示させるには?

生徒と一緒にリストとその構成要素についてもう一度復習し、リストを作成するために何が必要かを理解します。

4. 詳しく説明する

生徒が作成したリストを使って、アイテムが始まるのに必要な文字をランダムに生成するブロックを使ったゲームに挑戦します。
各グループは、左ボタンを押したときにランダムな文字が生成されるようにコードを修正する必要があります。random.choice のコードを生徒に紹介し、これがどのようにランダムな文字を生成するかについて話し合います。

サンプルプログラム

```
from hub import light_matrix, button
import runloop
import random

async def main():
    # リストを定義
    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'T, T, K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    while True:
        if button.pressed(button.LEFT):
            await runloop.sleep_ms(500)
            # ランダムに選択してライトマトリックスに表示する
            await light_matrix.write(random.choice(letters))

runloop.run(main())
```

4 人 1 組でゲームを数ラウンド行います。各生徒は順番にブレインゲーム・マスターを演じ、ランダムに文字を生成するプログラムを実行します。その後、他の生徒はそれぞれ 2 分間で、自分のセットのブロックを使ってその文字で始まるアイテムを作ります。ブレインゲーム・マスターは、最後に各アイテムが何であるかを当てます。

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：

☐ ランダム文字ジェネレーターを作ったとき、何が起こったか?

☐ リストはどのように使えますか?

☐ Python でリストを作成するときに覚えておくべきことは何ですか?

自己評価

生徒にノートで次のことに答えさせます：

☐ 今日、リストの使い方について何を学びましたか?

SAMPLE

ユニット 10 レッスン 2
筋肉とリストを伸ばす

筋肉とリストを伸ばす

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の値をリストの変数として使用してプログラムを作成する。
- ☐ リストを使ってヨガのルーチンを作成する。

生徒たちは、リストを使って自分がどのような動きをするかを探求します。

探究への質問

☐ フィットネストレーナーは、どのようにして動きのリストを作成できるでしょうか？

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

☐ 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

生徒をストレッチについての会話に引きつけ、筋肉を伸ばすさまざまな方法について話し合います。ストレッチのイメージを持ちやすくするために、ストレッチをしている人の画像やビデオクリップを見せることを検討してください。
生徒が筋肉を伸ばすために立ち上がって動くよう促します。あなたが前で実演したり、部屋の中を動き回ったりすることで、生徒があなたの動きに合わせて行動するリーダーについていくタイプのゲームを行います。リーダーを務める生徒を決め、その役割を他の生徒に引き継ぐことも考えましょう。
生徒が全員体を伸ばしたところで、自分の体をどのように動かしたかを考えさせます。使った動きや筋肉をリストアップし、それらをどのように分類するかについて話し合います。

2. 探究する

生徒は、筋肉を動かしたり伸ばしたりするさまざまな方法を探求します。
SPIKE アプリの「 組み立て」セクションに生徒を誘導し、ヨガリングの組み立て図にアクセスさせます。生徒にモデルを作るように指示します。組み立て図は 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、プログラミング領域にあるコードをすべて消去させます。 生徒はハブを接続します。

生徒と協力して、モーションセンサーの値をリストの変数として使用するプログラムを作成します。ナレッジベースでモーションセンサーを探すように指示し、up_face のタブで生徒はモーションセンサーの向きに使用できる値を参照できます。生徒には、リストのさまざまなオプションを使ってプログラムを試す時間を与えます。

SAMPLE

サンプルプログラム

```
from hub import motion_sensor
import runloop

async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]

    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[3]:
            print('Good Job!')
            await runloop.sleep_ms(1000)

runloop.run(main())
```

注：スピーカー側は motion_sensor.FRONT、USB 側は motion_sensor.BACK、A, C, E ポート側が motion_sensor.LEFT、B, D, F ポート側が motion_sensor.RIGHT、ライトマトリックス側が motion_sensor.TOP、バッテリー側が motion_sensor.BOTTOM です。
生徒とプログラムについて話し合い、それぞれの位置やインデックス値がリストの中で何であるかを確認させます。例えば、motion_sensor.LEFT はインデックス値 2、motion_sensor.BACK はインデックス値 1 です。Python ではインデックス値は常に 0 から始まることを生徒に説明します。

ヨガのルーチンを作る

生徒にオプションのリストを作成させ、コンソールに選択された順番で動きを表示することで、ヨガのルーチンを作成することに挑戦させます。
生徒には、リストからいくつかの動きを含むようにプログラムを修正させ、他のチームが自分たちのヨガのルーチンを追えるようにコンソールに動きのリストを表示させます。生徒に動きを練習する時間を与えます。

サンプルプログラム

```
from hub import motion_sensor, light_matrix
import runloop
async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]

    # 次のルーティンに従います
    print('LEFT')
    print('RIGHT')
    print('BACK')
    print('FRONT')
    print('READY GO!')
```

SAMPLE

```
    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[2]:
            print('Left Side')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_W)
        elif orientation == yoga_moves[3]:
            print('Right Side')
            light_matrix.show_image(light_matrix. IMAGE_ARROW_E)
        elif orientation == yoga_moves[ 1 ]:
            print('Back')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_N)
        elif orientation == yoga_moves[0]:
            print('Front')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

生徒には、動きをミックスしたり、ルーチンにいくつかのステップを加えたりすることを勧めます。

動きをランダムにする

生徒がルーチンを練習した後、コードのランダムな選択行を追加して、予期しない動きでルーチンを変更することに挑戦します。

サンプルプログラム

```
from hub import motion_sensor, light_matrix
import random
import runloop

async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]
    hub_faces = {1:'FRONT',4:'BACK',5:'LEFT',2:'RIGHT'}

    random_move = random.choice(yoga_moves)

    # 次のルーティンに従います
    print('LEFT')
    print('RIGHT')
    print(hub_faces[random_move])
    print('BACK')
    print('FRONT')
    print(hub_faces[random_move])
    print('READY GO!')
    orientation = motion_sensor.up_face()
```

SAMPLE

```
    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[2]:
            print('Left Side')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_W)
        elif orientation == yoga_moves[3]:
            print('Right Side')
            light_matrix.show_image(light_matrix. IMAGE_ARROW_E)
        elif orientation == random_move:
            print('Random')
            light_matrix.show_image(light_matrix.IMAGE_SILLY)
        elif orientation == yoga_moves[1]:
            print('Back')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_N)
        elif orientation == yoga_moves[0]:
            print('Front')
            light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

生徒が自分のプログラムを試したり、ランダム選択のコード行を含めるさまざまな方法で遊んだりする時間を設けます。

3. 説明する

生徒にルーチンを実演してもらい、プログラムがどのように機能する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：

- ☐ ヨガのルーチンを作るのに、リストをどのように使いましたか？
- ☐ 日課にランダムな動きを加えたらどうなりましたか？
- ☐ 新しいルーチンを作るために、プログラムをどのように変えたらよいか、アイデアはありますか？

4. 詳しく説明する

生徒たちは、このリストとルーチンを発展させ、新しい動きを加えることに挑戦します。
生徒にプログラムを変更して新しい動きを加えるように指示します。まず、モーションセンサーの向きを上と下にできるように、2 つの新しいオプションをリストに追加します。ただし、生徒は現在プログラムされているリストを変更することなく、これらの新しい動きを挿入する必要があります (つまり、既存のリストの最後に 2 つの要素を追加するだけではいけません)。
生徒に list_name.insert(index_location, value) というコードの挿入行を紹介し、下のサンプルプログラムを現在のプログラムの最後に追加するように指示します。生徒はプログラムを実行して、新しく挿入された行がどのように動作するかを確認します。

サンプルプログラム

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

SAMPLE

```
async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]
    yoga_moves.insert(0,motion_sensor.TOP)
    yoga_moves.insert(1,motion_sensor.BOTTOM)

    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[0]:
            print('Top')
            light_matrix.show_image(light_matrix.IMAGE_TRIANGLE)
        elif orientation == yoga_moves[1]:
            print('Bottom')
            light_matrix.show_image(light_matrix. IMAGE_SQUARE)

runloop.run(main())
```

新しい値がリストのどこに挿入されるかを生徒に説明します。
自分たちの新しいヨガのルーチンを練習した後、グループ同士でヨガのルーチンに挑戦させます。また、del list_name[index_value] を使ってリストから要素を削除する方法も生徒に共有することを考えてみてください。
この例では、リストから「 motion_sensor.BOTTOM」を削除するには、del yoga_moves[1] のようになります。

```
yoga_moves [motion_sensor. FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]
yoga_moves.insert(0,motion_sensor.TOP)
yoga_moves.insert(1,motion_sensor.BOTTOM)

print(yoga_moves)
del yoga_moves[1]
print(yoga_moves)
```

Tips: リストとタプルと辞書オブジェクト

Python では、一般に配列と呼ばれる順序付きのコレクションをシーケンスと呼びます。
シーケンスにはリスト型、タプル型、文字列型、辞書オブジェクトなどのデータ型があります。
その中でリスト、タプル、辞書オブジェクトを紹介します。

リスト

リストは要素の削除、追加、入れ替えが自由に行えるシーケンスオブジェクトです。
要素へのアクセスはインデックスを指定します。
このユニットで主に扱う型であり、よく使われる型でもあります。
リストオブジェクトは [要素 , 要素 ,…] のように [] で要素を囲んで作成します。

```
L = [1,2,3]
```

SAMPLE

タプル

タプルはリストによく似たシーケンスオブジェクトですが、リストと違い更新不可なオブジェクトであり、
タプル作成後に要素の追加や削除は行えません。
要素へのアクセスはインデックスを指定します。
タプルオブジェクトは (要素 , 要素 ,…) のように () で要素を囲んで作成します。
SPIKE Python ではカラーセンサーの rgbi 関数の戻り値、モーションセンサーの tilt_angles の戻り値などで使われています。
例）RED = (255,0,0)

辞書オブジェクト

辞書オブジェクトは、一般に連想配列などと呼ばれるデータ構造です。
辞書には、キーと、キーに関連付けられた値の組を要素として登録し、登録した値はキーを使って参照できます。
辞書オブジェクトは { キー : 要素 , キー : 要素 ,…} のように : で区切ったキーと値を {} で囲んで作成します。
例）hub_faces = {1:'FRONT',4:'BACK',5:'LEFT',2:'RIGHT'}

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：

☐ どのようにリストに新しい要素を追加したり、リストから要素を削除したりできますか？
☐ リストを使ってどのようにヨガ計画を立てましたか？
☐ リストとセンサーをどのように一緒に使うことができましたか？

自己評価

生徒にノートで次のことに答えさせます：

☐ 今日、リストを使ってヨガの日課のような計画を立てることについて、何を学びましたか？

SAMPLE

ユニット 10 レッスン 3
マインドゲーム

マインドゲーム

生徒たちは既成概念にとらわれず、複数のリストの使い方やリストの比較方法を発見します。

主な目的

生徒は次のことを行います

- ☐ 1 つのプログラムで 2 つのリストを作成する
- ☐ プログラム内で 2 つのリストを比較する

探究への質問

☐ 2 つのリストに共通の構成要素がある場合、インデックス付けはどのように比較できますか?

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

異なるリストがどのように類似しているかを考えるために、生徒と会話をします。
生徒に、セット内の 2x4 色ブロック 10 個をすべて取り出すよう指示します。グループ内の各生徒に、赤、マゼンタ、黄、青、緑を含む各色を 1 つずつ取ってもらいます。
パートナー 1 は、パートナー 2 に見せないように、自分の 5 つのブロックを好きな順番で積み重ねます。パートナー 1 に、同じ積み木を作るためにパートナー 2 に積み木の作り方を説明するよう求めます (積み木の作り方を見せる、見せずに順番を言うなど)。ペアが同じ積み重ねだと思ったら、完成です。
パートナーに交代してもらい、もう一度練習します。このとき、パートナーはスタックについてコミュニケーションする新しい方法を考え出さなければなりません (別の方法で説明する、単語を使わないなど、最初のアプローチとは異なる方法です)。
各グループがいくつかのラウンドを終えたら、このゲームがどのように行われるかをクラスで議論し、このゲームの擬似コードプログラムを書くように生徒に指示します

2. 探究する

生徒は、複数のリストを含むプログラムと、リスト内の指定された位置またはインデックス値を見つける方法を調べます。
SPIKE アプリの組み立てセクションに生徒を誘導し、ここで生徒はゲームマスターモデルの組み立て図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。
組み立て図は 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、生徒はハブを接続します。
生徒はゲームマスターモデルを使って、キャンディースティックブロックの色を読み取ります。生徒に以下のサンプルプログラムを提供し、一緒にプログラムを確認して何を行うかを話し合います。生徒にプログラムを実行させ、キャンディースティックを 1 本モデルに投入させます。

SAMPLE

サンプルプログラム

```
from hub import light_matrix, button, port
import runloop
import motor
import color_sensor
import color
from app import sound

def feed_candy_1():
    return button.pressed(button.LEFT)

def feed_candy_2():
    return button.pressed(button.RIGHT)

async def main():
    # リストを作成する
    candy1 = []
    candy2 =[]
    # ゲームマスターが棒付きキャンディー 1 を食べる
    await runloop.until(feed_candy_1)
    light_matrix.clear()
    candy1.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')
    # キャンディー 1 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy1.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)
    # ゲームマスターが棒付きキャンディー 2 を食べる
    await runloop.until(feed_candy_2)
    light_matrix.clear()
    candy2.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')
    # キャンディー 2 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy2.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees (port.A, 95, 500)
    # 両方のキャンディスティックの同じ位置に赤いブロックがある場合、赤いブロックの位置を照らす。
    candy1_red_index = -1
    if color.RED in candy1:
        candy1_red_index = candy1.index(color.RED)
    candy2_red_index = -1
```

SAMPLE

```
    if color. RED in candy2:
        candy2_red_index = candy2.index(color.RED)
    for x in range(5):
        print(candy1 [x])
    if candy1 _red_index == candy2_red_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1 _red_index, 100)
            await sound.play('Win')
    else:
        light_matrix.show_image(light_matrix.IMAGE_NO)
    await sound.play('Oops')

runloop.run(main())
```

生徒にプログラムを調査する時間を与えます。

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：

- ☐ プログラムはどのように機能したのか？
- ☐ このプログラムはどのようにリストを使っているのか？
- ☐ このプログラムは 2 つのリストにどのようにインデックスを付け、比較していますか？

4. 詳しく説明する

すべての色をインデックス付けして比較し、同じブロックスタックがあるかどうかをチェックする新しいプログラムを作成するよう、生徒に挑戦させます。
他の 4 色をチェックするために、プログラムをどのように修正するか、生徒と話し合います：

1. 他の色にインデックスを付ける必要があります。
2. 他の色のリストを比較する必要があります。
3. 他の色のピクセルがスタック上の同じ位置にある場合、他の色のピクセルをセットする必要があります。

サンプルプログラム

```
1.  from hub import light_matrix, button, port
2.  import runloop
3.  import motor
4.  import color_sensor
5.  import color
6.  from app import sound
7.
8.  def feed_candy_1 ():
9.      return button.pressed (button.LEFT)
10.
11. def feed_candy_2():
12.     return button.pressed (button.RIGHT)
```

SAMPLE

```
13.
14. async def main():
15.     # リストを作成する
16.     candy1 = []
17.     candy2 = []
18.     # ゲームマスターが棒付きキャンディー 1 を食べる
19.     await runloop.until(feed_candy_1)
20.     light_matrix.clear()
21.     candy1.clear()
22.     await motor.run_for_time(port.A, 2000, -500)
23.     await sound.play('Bite')
24.     await sound.play('Bite')
25.
26.     # キャンディー 1 と呼ばれるリストに色のシーケンスを読み込んで記録する。
27.     for x in range(5):
28.         candy1.append(color_sensor.color(port.B))
29.         await runloop.sleep_ms(1000)
30.         await motor.run_for_degrees(port.A, 95, 500)
31.
32.     # ゲームマスターが棒付きキャンディー 2 を食べる
33.     await runloop.until(feed_candy_2)
34.     light_matrix.clear()
35.     candy2.clear()
36.     await motor.run_for_time(port.A, 2000, -500)
37.     await sound.play('Bite')
38.     await sound.play('Bite')
39.
40.     # キャンディー 2 と呼ばれるリストに色のシーケンスを読み込んで記録する。
41.     for x in range(5):
42.         candy2.append(color_sensor.color(port.B))
43.         await runloop.sleep_ms(1000)
44.         await motor.run_for_degrees(port.A, 95, 500)
45.
46.     candy1_red_index = candy1.index(color.RED) if color.RED in candy1 else -1
47.     candy2_red_index = candy2.index(color.RED) if color.RED in candy2 else -1
48.     candy1_yellow_index = candy1.index(color.YELLOW) if color.YELLOW in candy1 else -1
49.     candy2_yellow_index = candy2.index(color.YELLOW) if color.YELLOW in candy2 else -1
50.     candy1_green_index = candy1.index(color.GREEN) if color.GREEN in candy1 else -1
51.     candy2_green_index = candy2.index(color.GREEN) if color.GREEN in candy2 else -1
52.     candy1_magenta_index = candy1.index(color.MAGENTA) if color.MAGENTA in candy1 else -1
53.     candy2_magenta_index = candy2.index(color.MAGENTA) if color.MAGENTA in candy2 else -1
54.     candy1_blue_index = candy1.index(color.BLUE) if color.BLUE in candy1 else -1
55.     candy2_blue_index = candy2.index(color.BLUE) if color.BLUE in candy2 else -1
56.     # 数字は色に対応しています
57.     for x in range(5):
58.         print(candy1[x],candy2[x])
59.     # 色ブロックが両方のキャンディスティックで同じ位置にある場合、色ブロックの位置を点灯させる。
```

SAMPLE

```
60.  if candy1_red_index == candy2_red_index:
61.      for x in range(5):
62.          light_matrix.set_pixel(x, candy1_red_index, 100)
63.   await sound.play('Win')
64.
65.   if candy1_yellow_index == candy2_yellow_index:
66.      for x in range(5):
67.          light_matrix.set_pixel(x, candy1_yellow_index, 100)
68.      await sound.play('Win')
69.
70.   if candy1_green_index == candy2_green_index:
71.      for x in range(5):
72.          light_matrix.set_pixel(x, candy1_green_index, 100)
73.      await sound.play('Win')
74.
75.   if candy1_magenta_index == candy2_magenta_index:
76.      for x in range(5):
77.          light_matrix.set_pixel(x,candy1_magenta_index, 100)
78.      await sound.play('Win')
79.
80.   if candy1_blue_index == candy2_blue_index:
81.      for x in range(5):
82.          light_matrix.set_pixel(x, candy1_blue_index, 100)
83.      await sound.play('Win')
84.
85.   if candy1_red_index == candy2_red_index and candy1_yellow_index == candy2_yellow_index and candy1_
      green_index == candy2_green_index and candy1_magenta_index == candy2_magenta_index and candy1_
      blue_index == candy2_blue_index:
86.      await sound.play('Triumph')
87.
88. runloop.run(main())
```

生徒とプログラムの内容について話し合います。このプログラムには効率的ではない点がいくつか見られるため、リストや for 文を活用して、より効率的なプログラムへと改善する活動を行うことも可能です。

例

上記プログラムの 46 行目以降を変更しています。

```
# candy1_red_index = candy1.index(color.RED) if color.RED in candy1 else -1
# candy2_red_index = candy2.index(color.RED) if color.RED in candy2 else -1
# candy1_yellow_index = candy1.index(color.YELLOW) if color.YELLOW in candy1 else -1
# candy2_yellow_index = candy2.index(color.YELLOW) if color.YELLOW in candy2 else -1
# candy1_green_index = candy1.index(color.GREEN) if color.GREEN in candy1 else -1
# candy2_green_index = candy2.index(color.GREEN) if color.GREEN in candy2 else -1
# candy1_magenta_index = candy1.index(color.MAGENTA) if color.MAGENTA in candy1 else -1
# candy2_magenta_index = candy2.index(color.MAGENTA) if color.MAGENTA in candy2 else -1
# candy1_blue_index = candy1.index(color.BLUE) if color.BLUE in candy1 else -1
# candy2_blue_index = candy2.index(color.BLUE) if color.BLUE in candy2 else -1
```

SAMPLE

```
# インデックスと色定数を保存するリストを追加で作成
candy1_indexs = []
candy2_indexs = []
color_constants = [color.RED,color.YELLOW,color.GREEN,color.MAGENTA,color.BLUE]

for i in range(5):
    candy1_indexs.append(candy1.index(color_constants[i]) if color_constants[i] in candy1 else -1)
    candy2_indexs.append(candy2.index(color_constants[i]) if color_constants[i] in candy2 else -1)

# 数字は色に対応しています
for x in range(5):
    print(candy1[x],candy2[x])

# if candy1_red_index == candy2_red_index:
#     for x in range(5):
#         light_matrix.set_pixel(x, candy1_red_index, 100)
# await sound.play('Win')

# if candy1_yellow_index == candy2_yellow_index:
#     for x in range(5):
#         light_matrix.set_pixel(x, candy1_yellow_index, 100)
#     await sound.play('Win')

# if candy1_green_index == candy2_green_index:
#     for x in range(5):
#         light_matrix.set_pixel(x, candy1_green_index, 100)
#     await sound.play('Win')

# if candy1_magenta_index == candy2_magenta_index:
#     for x in range(5):
#         light_matrix.set_pixel(x,candy1_magenta_index, 100)
#     await sound.play('Win')

# if candy1_blue_index == candy2_blue_index:
#     for x in range(5):
#         light_matrix.set_pixel(x, candy1_blue_index, 100)
#     await sound.play('Win')

for i in range(5):
    if candy1_indexs[i] == candy2_indexs[i]:
        for j in range(5):
            light_matrix.set_pixel(j, candy1_indexs[i], 100)
        await sound.play('Win')

 # if candy1_red_index == candy2_red_index and candy1_yellow_index == candy2_yellow_index and candy1_
green_index == candy2_green_index and candy1_magenta_index == candy2_magenta_index and candy1_blue_
```

SAMPLE

```
index == candy2_blue_index:
        # await sound.play('Triumph')
    for i in range(5):
        if not candy1_indexs[i] == candy2_indexs[i]:
            break
        if i == 4:
            await sound.play('Triumph')

runloop.run(main())
```

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：

☐ 複数のリストを使う方法にはどのようなものがありますか？

☐ リストがプログラムにおいてどのような利点がありますか？

☐ リストから得られる情報にはどのようなものがありますか？

自己評価

生徒にノートで次のことに答えさせます：

☐ 複数のリス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 10 レッスン 4
リストへのジャンプ

リストへのジャンプ

生徒はリストを使ってプログラムを作成し、身体的パフォーマンスのデータを取得します。

主な目的

生徒は次のことを行います

- ☐ カセンサーと距離センサーから、リストで使用するデータを作成する。
- ☐ ジャンプの試行から収集したデータに基づいて、リストをプログラムする (例 : ジャンプの高さ)

探究への質問

パフォーマンスに関する個人的な統計を表すリストを作成するにはどうしたらよいでしょうか。

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

- ☐ 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

スクワットをするときの体の動きについて生徒と話し合いをする。(例えば、足が曲がる、胴体が地面に近づくなど) 生徒に立ち上がってスクワットをしてもらいます。

次に、生徒にその場でまっすぐジャンプさせます。ジャンプ中にどのような体の動きをしたかを生徒に尋ねます。スクワット中に体がどれだけ低くなるか、またはジャンプの頂点で体がどれだけ高くなるかを測定する方法はありますか ?

2. 探究する

生徒は、リストに含めるデータを生成する方法を調べます。

生徒を SPIKE アプリの組み立てセクションに誘導します。ここで生徒は、スマートケトルベルモデルの組み立て図にアクセスできます。生徒にモデルを組み立てるように指示します。組み立て図は、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。

パート 1 : リストのデータを作成します。

生徒は、ジャンプの高さをいくつか提供するようにケトルベルモデルをプログラムする必要があります。生徒は、この情報を使用して、後で使用するリストを作成します。

生徒と一緒にサンプルプログラムを見直します。このプログラムでのカセンサーと距離センサーの使用方法について話し合います。生徒は、カセンサーを使用してジャンプが起こるタイミングを示し、距離センサーを使用してジャンプの高さを測定します。生徒にプログラムを実行させ、高さにどのような値が得られるか確認させます。

SAMPLE

サンプルプログラム

```
from hub import port, light_matrix
import runloop
import distance_sensor
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    runloop.until(is_force_sensor_pressed)
    await light_matrix.write('JUMP!')
    runloop.until(is_force_sensor_released)
    dist_cm = distance_sensor.distance(port.F)/10
    print(str(dist_cm) + 'cm')

runloop.run(main())
```

注意：生徒はカセンサーを押し、押したままジャンプし、ジャンプの高さに達したらセンサーを離すこと。カーペットの床を見るとき、距離センサーは正しく読み取れないかもしれない。ジャンプが -0.1cm を示したら、別の場所を試してください。

生徒がプログラムを探究した後、チームの各人に 3 回のジャンプを行うように指示します。生徒は、各ジャンプの値を表に記録します。

表の例

	一人目	二人目
1 回目		
2 回目		
3 回目		

パート 2：リストの作成

生徒は、パート 1 で作成したデータを使用して、新しいプログラムで 2 つのリストを作成します。

各生徒のリストと、各生徒のジャンプの試行からのデータをリストの値として含む新しいプログラムの作成方法について、生徒と話し合います。プログラム中に変数とリストを簡単に区別できるように名前を付けることの重要性を生徒に思い出させます。ここで生徒は、個々のリストに自分の名前を使うことができます。

生徒に、両方のリストを含むプログラムを作成し、それぞれのリストの最大値を示し、それぞれの値をハブで共有するように指示します。

リストの最大値を示す max(name1) 行のコードを生徒に紹介します。

サンプルプログラム（ 生徒は自分の名前と表のデータを使用する）：

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

SAMPLE

```
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたらライトマトリクスに最大値を表示
    await runloop.until(is_force_sensor_pressed)
    await light_matrix.write(str(max(name1)))
    await runloop.sleep_ms(2000)
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(1000)
    await light_matrix.write(str(max (name2)))
    await runloop.sleep_ms(2000)
    light_matrix.show_image(light_matrix.IMAGE_YES)

runloop.run(main())
```

生徒に、各リストの出力や最大値を表示するさまざまな方法を試して、プログラムを調査する時間を与えます。

パート 3：リストをまとめる

生徒は 2 つのリストを足して 1 つのリストを作るようにプログラムを修正します。
生徒には、リストの連結により、＋記号を使って 2 つのリストを 1 つにまとめることができることを説明します。リストを連結する方法として、list3 = list1 + list2 のコード例を生徒と共有します。生徒たちは自分のリストを連結し、両方のリストから新しい最大数または最高数を求めることに挑戦します。

サンプルプログラム

```
from hub import port, light_matrix
import runloop
import force_sensor
```

SAMPLE

```
# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたら 2 つのリストを結合
    await runloop.until(is_force_sensor_pressed)
    combo_list = name1 + name2
    # 結合したリストの内容をコンソールに表示
    print(combo_list)
    # 結合したリストの数値の合計をライトマトリックスに表示
    await light_matrix.write(str(sum(combo_list)))
    light_matrix.show_image(light_matrix.IMAGE_YES)

runloop.run(main())
```

生徒には、結合したリストの合計値の出力するさまざまな方法を試して、プログラムを調査する時間を与えます。また、結合したリストの合計を表示したように、それぞれのリストの合計を表示することもできます。

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：

- ☐ どのようにして、リストに含める独自のデータを生成できましたか？
- ☐ カセンサーを、アクションを起こすために押すボタンとしてどのように組み込むことができましたか？
- ☐ リストに関するさまざまな情報（合計や最大値など）をどのように見つけることができましたか?他にどのような値を見つけることができると思いますか？
- ☐ リストはどのように組み合わせることができますか?どのような場合に便利ですか？

4. 詳しく説明する

生徒に、ある数字がリストに含まれているかどうかを示すよう求めます。

生徒と、リスト内の情報の探し方について話し合います。生徒はプログラムのリストを簡単に見ることができますが、リストに何が含まれているかを調べる必要がある場合もあります。このようなことが起こりうる方法と理由をグループで話し合います。

SAMPLE

生徒に、コンボ・リストがあるプログラムを修正して、X という数字がコンボリストに含まれているかどうかを確認するよう求めます。
X という数字がコンボリストに含まれているかどうかを確認するために、コンボリストがあるプログラムを修正するよう生徒に指示します。
注：X の数字は生徒のデータに従って、適切な数を選択します。
サンプルプログラムを生徒と共有し、プログラムがリストに数字が含まれているかどうかをどのように チェックしているかを生徒に確認させます。

サンプルプログラム

```
from hub import port, light_matrix
import runloop
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたら 2 つのリストを結合
    await runloop.until(is_force_sensor_pressed)
    combo_list = name1 + name2
    # 結合したリストの内容をコンソールに表示
    print(combo_list)
    # 結合したリストの数値の合計をライトマトリックスに表示
    await light_matrix.write(str(sum(combo_list)))
    print(sum(combo_list))
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(2000)
    light_matrix.clear()

    #x: 表の中のデータから 1 つ選択
    if x in combo_list:
        print('The number x is in the list')
        light_matrix.show_image(light_matrix.IMAGE_YES)
    else:
```

SAMPLE

```
        print('The number x is not in the list')
        light_matrix.show_image(light_matrix.IMAGE_NO)

runloop.run(main())
```

生徒にサンプルプログラムと同様にプログラムを修正し、さまざまな値を検索して、プログラムに何が含まれるか、含まれないかを確認させます。生徒は、プログラムに含まれるか含まれないかを示すために、さらにプログラムを修正することができます。

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：

- ☐ 複数のリストを使う方法にはどのようなものがありますか？
- ☐ リストがプログラムにおいてどのような利点がありますか？
- ☐ リストから得られる情報にはどのようなものがありますか？

自己評価

生徒にノートで次のことに答えさせます：

- ☐ 複数のリス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 10 レッスン 5
単語ゲームとリスト

単語ゲームとリスト

主な目的

生徒は次のことを行います

- ☐ 単語ゲームを完成させるためにプログラム内で複数のリストを作成する
- ☐ 単語ゲームに合わせたカラーセンサー付きモデルをプログラムする

生徒は、単語ゲームに基づいて物語を完成させるためにリストを作成します。

探究への質問

複数のリストを同時に使用し、アクセスする方法は何でしょうか？

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

- ☐ 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

生徒を単語ゲームに参加させ、後で物語を完成させるために特定の種類の単語を挙げさせます。
以下の例に対して、生徒に例となる単語を挙げてもらいます：

- ☐ 場所
- ☐ 形容詞
- ☐ 道具または機械
- ☐ 色
- ☐ 動物の種類

サンプルストーリーを読み聞かせます。生徒が選んだ単語を空欄に入れて楽しみましょう。

「 ケイトとカイルは、(場所) に座っているときに釣りに行くことに決めました。ケイトは本当に (形容詞) な魚を釣りたいと思っていました。まず、魚を釣るための (道具または機械) を見つける必要がありました。幸運なことに、カイルは隣人が (色) の釣り網を持っていることを知っています。彼らは網を借りることを頼み、すぐに池に向かいましたが、魚の代わりに (動物) が見えました。」

楽しむために、異なる単語を使ってストーリーを何度か読み聞かせることを考えてみてください。

生徒に、自分のストーリーを作成するよう挑戦させましょう。そのストーリーには、ゲーム中に追加する必要がある単語が少なくとも 5 つ含まれている必要があります。ストーリーの長さや特定のトピック、または空欄にする単語の種類（ 名詞、動詞、形容詞など）についての設定を明確に伝えましょう。

2. 探究する

生徒は、センサーを使ったリストを利用して、自分の単語ゲームを作成することを探求します。
生徒に SPIKE アプリの組み立て図セクションに移動させ、いくつかのモデルの組み立て図からインスピレーションを得て、自分のカラーセンサーをデザインして作成するように指示します。生徒に自分のモデルを設計して組み立てるように指示します。

SAMPLE

ストーリーのプログラミング

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。プログラミングエリアに既にあるコードは削除するように指示し、生徒はハブを接続します。
生徒に、名詞、動詞、形容詞などの単語の種類のリストを使って、ストーリーの空欄に置き換えるためのプログラムを作成するように指示します。リストは各自のストーリーで使用する単語の種類に合ったものでなければならないことを思い出させましょう。そのため、皆が同じリストを作成するわけではありません。

さらに、生徒は単語ゲームに合わせてカラーセンサー付きモデルをプログラムする必要があります。5 つの異なる色を使用して、感知された各色がストーリーの単語を提示する条件文を作成するように生徒に指示します（ 例：青は名詞、赤は動詞など）。

テストと改善

生徒が自分のアイデアをテストし、分析する時間を設け、必要に応じて改善を行えるようにします。生徒は、自分の解決策を作成し始める際に、設定されたデザイン基準やフローチャートに対して、自分の設計をテストし評価すべきです。
生徒は、デザインの過程を記録するために、モデルのスケッチや写真を使用するようにします。
時間が許す限り、他のグループや教師からのフィードバックを受ける機会を与えましょう。

3. 説明する

プログラムの動作について生徒と話し合います。
生徒に以下のような質問をします：

- ☐ あなたのカラーセンサー付きモデルはどのように機能しますか？
- ☐ あなたのモデルはストーリーとどのように連携していますか？
- ☐ ストーリーをプログラミングする際にどのような決定をしましたか？
- ☐ 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

初回の共有セッション後に生徒が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最終的に確定させなければなりません。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励ましましょう。

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います。
生徒に次のような質問をします：

- ☐ この課題で難しかったことは何ですか？
- ☐ この課題を解決するためのアプローチは何でしたか？
- ☐ センサーとリストはどのように連携しますか？

自己評価

生徒にノートで次のことに答えさせます：

- ☐ リストの作成やセンサーとの連携について、今日学んだことは？

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 7 Impacting the Environment with Functions:
関数

Lesson 1 P91-92
Lesson 2 P93-94
Lesson 3 P95-98
Lesson 4 P99-100
Lesson 5 P101-102

Unit 8 Compound Conditionals and Logic Operators:
複合条件と論理演算子

Lesson 1 P103-104
Lesson 2 P105-108
Lesson 3 P109-110
Lesson 4 P111-112
Lesson 5 P113-114

Unit 9 Data and Math Functions:
データと数学関数

Lesson 1 P115-116
Lesson 2 P117-118
Lesson 3 P119-117
Lesson 4 P121-124
Lesson 5 P125-126
Lesson 6 P127-128

Unit 10 Lists:
リスト

Lesson 1 P129-130
Lesson 2 P131-134
Lesson 3 P135-138
Lesson 4 P139-142
Lesson 5 P143-144

SAMPLE

ユニット 7　レッスン 1
カメのトラブル

ホッパーモデルをウミガメに見立てて、ヒレを動かすプログラムを作成しましょう。

例：下線部を埋めて正しく動くようにしましょう。

```
from hub import port
import runloop
import motor_pair

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

  # 速度 500 で 1440 度分まっすぐに進む
  await motor_pair.move_for_degrees(_____, ____, 0, velocity=___)

runloop.run(main())
```

関数を定義する

関数を定義するには def から始めて、関数の名前、括弧、コロンで終わります。

インデントされた次行以降を関数の内容として扱います。

```
async def 関数名 ():
_____実行内容 1
_____実行内容 2
```

例：

```
async def frightened() :
        await motor_pair.move_for_degrees (motor_pair.PAIR_1, 1440, 0, velocity=1000)
```

やってみよう

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。

プログラム内で関数がどのように呼び出されるか、確認しましょう。

コンソールに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、バグを修正しましょう。

```
from hub import port
import runloop
import motor_pair

# カメを早く動かす関数
async def fritened():
    # 速度 1000 度 / 秒で、まっすぐに 1440 度分移動する
    await motor_pair.move_for_degrees (motor_pair.PAIR_1, 1440, 0, velocity=1000)

# メイン関数
async def main():
```

SAMPLE

```
    # ポートＥとＦのモーターをペアにする
    motor_pair.pair (motor_pair.PAIR_1, port. E, port.F)
    # はやく動く関数を呼び出す
    await fritened()

runloop.run(main())
```

関数を追加しよう

追加する関数について、どのような動きにしたいか下枠に記入しましょう。

SAMPLE

ユニット 7　レッスン 2
複数の関数でクリーンアップ

モデルをよく観察して、どのようにプログラムするか下枠に疑似コードを書きましょう。

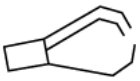
次のプログラムを入力して動かしてみましょう。

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

def is_pressed():
    # フォースセンサーが押されているか？
    return force_sensor.pressed (port.E)
def is_released():
    # フォースセンサーが離されているか？
    return not force_sensor.pressed (port.E)

async def main():

    while True:
        # フォースセンサーが押されるまで待つ
        await runloop.until(is_pressed)
        # グラバーを閉じる
        await motor.run_to_absolute_position (port.A, 345, 500, direction=motor.COUNTERCLOCKWISE)
        # フォースセンサーが離されるまで待つ
        await runloop.until(is_released)
        # グラバーを開く
        await motor.run_to_absolute_position (port.A, 25, 500, direction=motor.SHORTEST_PATH)

runloop.run(main())
```

モデルはどのように動きましたか？

SAMPLE

それぞれのグラバーがどんなゴミを拾うのに最も適しているか表を埋めましょう

	
例）大きなごみ	例）小さなごみ

グラバーアタッチメントによって握る力を変化させることができるような関数を定義して、その時のアタッチメントによって力（モーターの速度）を変えて関数を呼び出すようにプログラムを変えましょう。下記の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 motor
import force_sensor

def is_pressed():
    # フォースセンサーが押されているか？
    return force_sensor.pressed(port.E)

def is_released():
    # フォースセンサーが離されているか？
    return not force_sensor.pressed(port.E)

# アタッチメントによって握る力を変えてハンドを開閉する関数を定義
async def grab_and_release(grip):
    # フォースセンサーが押されるまで待つ
    _____
    # グラバーを閉じる
    _____
    # フォースセンサーが離されるまで待つ
    _____
    # グラバーを開く
    _____

async def main():
    while True:
        # 使っているアタッチメントによって握る力を変えて開閉する関数を呼び出す
        _____

runloop.run(main())
```

グラバーをさらに改造してみよう

さらに重いもの、大きいもの、小さいものに軽いもの様々なものを持てるようにグラバーを改造してみましょう。
プログラムも関数を複数用いて便利にしてみましょう。

SAMPLE

ユニット 7　レッスン 3
クリーンインディケーター

クリーンインディケーターで環境が綺麗であるかどうかを表示するプログラムを作成します。

まずは疑似コードを書いてみましょう。

次のプログラムを入力して動かしてみましょう。

その際、作成した関数どのような役割をしているか、確認しましょう。

```
from hub import port, button
import runloop
import motor
import color_sensor, color
# クリーンレベルをレポートする関数の定義
def report (hight, color):
    # モーターを hight で指定された角度へ移動する
    runloop.run(motor.run_to_absolute_position (port.A, hight, 500, direction=motor.SHORTEST_PATH))
    #color で指定された値をコンソールに表示する
    print('Clean Level', color)

async def main():
    while True:
        await runloop.until(lambda:button.pressed(button.LEFT))
        # 検出した色に合わせた hight を設定
        if color_sensor.color(port.E) is color.RED:
            report(70, 'Red')
        elif color_sensor.color(port.E) is color.YELLOW:
            report (60, 'Yellow')
        elif color_sensor.color(port.E) is color.BLUE:
            report(50, 'Blue')
        elif color_sensor.color(port.E) is color.GREEN:
            report (25, 'Green')
        # 色を検知しなかった、あるいは上記以外の色の場合に戻る
        else:
            await motor.run_to_absolute_position(port.A, 0, 1000, direction=motor.SHORTEST_PATH)
            print('No Reading')

runloop.run(main())
```

SAMPLE

関数にパラメーターを作ることで、プログラミングにどのようなメリットが生まれるでしょうか？

下枠に自分の考えを書いてみましょう。

デバッグアクティビティ I

次のプログラムのバグを修正してください

プログラムを入力し実行してみてからコンソールを開き、どんなエラーが出ているかを確認しましょう。

エラーメッセージを記入

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

# クリーンレベルをレポートする関数の定義
def report():
    # モーターを hight で指定された角度へ移動する
    runloop.run(motor.run_to_absolute_position (port.A, height, 500, direction=motor.SHORTEST_PATH))
   #color で指定された値をコンソールに表示する
    print('Clean Level', color)

async def main():
    while True:
        await runloop.until(lambda:button.pressed(button.LEFT))
        # 検出した色に合わせた hight を設定
        if color_sensor.color(port.E) is color.RED:
            report (70, "Red")
        elif color_sensor.color(port.E) is color.YELLOW:
            report(69, 'Yellow')
```

SAMPLE

```
22:       elif color_sensor.color(port. E) is color.BLUE:
23:         report (50, 'Blue')
24:       elif color_sensor.color(port.E) is color.GREEN:
25:          report (25, 'Green')
26:       # 色を検知しなかった、あるいは上記以外の色の場合に戻る
27:       else:
28:          await motor.run_to_absolute_position(port. A, 0, 1000, direction=motor.SHORTEST_PATH)
29:          print("No Reading")
30:
31: runloop.run(main())
```

エラーの原因は何でしたか?

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

プログラムを入力し実行してみてからコンソールを開き、どんなエラーが出ているかを確認しましょう。

エラーメッセージを記入

```
 1: from hub import port, button
 2: import runloop
 3: import motor
 4: import color_sensor
 5: import color
 6:
 7: # クリーンレベルをレポートする関数の定義
 8: def report(height, color):
 9:    # モーターを hight で指定された角度へ移動する
10:    runloop.run(motor.run_to_absolute_position (port.A, height, 500, direction=motor.SHORTEST_PATH))
11:    #color で指定された値をコンソールに表示する。
12:    print('Clean Level', color)
13:
14: async def main():
15:    while True:
16:       await runloop.until(lambda:button.pressed(button.LEFT))
17:       # 検出した色に合わせた hight を設定
18:       if color_sensor.color(port.E) is color.RED:
```

SAMPLE

```
19:         report (70)
20:     elif color_sensor.color(port.E) is color.YELLOW:
21:         report (60, 'Yellow')
22:     elif color_sensor.color(port. E) is color.BLUE:
23:         report (50, 'Blue')
24:     elif color_sensor.color(port.E) is color.GREEN:
25:         report (25, 'Green')
26:     # 色を検知しなかった、あるいは上記以外の色の場合に戻る。
27:     else:
28:         await motor.run_to_absolute_position(port. A, 0, 1000, direction=motor.SHORTEST_PATH)
29:         print("No Reading")
30:
31: runloop.run(main())
```

エラーの原因は何でしたか？

SAMPLE

ユニット 7　レッスン 4
クリーンアップの自動化

リサイクル可能な色とロボットの動きを決めましょう

リサイクル可能なとき：例）紫

例：首を縦に振る

リサイクル不可のとき：例）その他

例：首を横に振る

それぞれの動きを関数として作成しましょう

例：リサイクル可能な時の動き

```
async def accept():
    sound.play('Triumph')
    for x in range(5):
        await motor.run_to_absolute_position(port.A,15,250,direction=motor.SHORTEST_PATH)
        await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
```

色を識別する関数を作成しましょう。

上記で作成した関数も組み込みます。

例：

```
async def check_color():
    if color_sensor.color(port.D) is _____:
        関数 1
    else:
        関数 2
```

作成した関数を用いてプログラムを完成させましょう。

このロボットをさらに使いやすくするために工夫をしましょう。

具体的な案を下枠に記入してください。

SAMPLE

```
from hub import port
import runloop
import motor
import color_sensor, color
import distance_sensor
from app import sound

async def main():
    await get_up()
    await runloop.until(lambda:color_sensor.color(port.D) != color.UNKNOWN)
    await check_color()

async def get_up():
    pixels = [100] * 4
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    distance_sensor.show(port.C, pixels)

async def accept():
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    sound.play('Triumph')
    for x in range(5):
        await motor.run_to_absolute_position(port.A,15,250,direction=motor.SHORTEST_PATH)
        await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)

async def refuse():
    await motor.run_to_absolute_position(port.A,0,250,direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.F,0,250,direction=motor.SHORTEST_PATH)
    sound.play('Oops')
    await motor.run_to_absolute_position(port.F,50,1000,direction=motor.SHORTEST_PATH)
    for x in range(3):
        await motor.run_for_degrees (port.F,-100, 1000)
        await motor.run_for_degrees (port.F,100, 1000)

    await motor.run_to_absolute_position(port.F,0,1000,direction=motor.SHORTEST_PATH)

async def check_color():
    await motor.run_to_absolute_position(port.F, 270,250, direction=motor.SHORTEST_PATH)
    await motor.run_to_absolute_position(port.A, 60,250, direction=motor.SHORTEST_PATH)
    await runloop.sleep_ms(500)
    if color_sensor.color(port.D) is color.MAGENTA:
        await accept()
    else:
        await refuse()

runloop.run(main())
```

SAMPLE

ユニット 7 レッスン 5
環境を大切にする

下記5つの条件を守って、モデルとプログラムを作成しましょう。

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. プログラムに少なくとも 1 つの機能を含むこと。
5. 関数と連動するパラメータを含むこと。

モデルの設計図を書きましょう。

SAMPLE

フローチャートを書きましょう。

SAMPLE

ユニット８　レッスン１

パスワードの保護

推理ゲーム

表の上の列には自分が決めた数字を、下の列には何回目で当てられたかを書きましょう。

数字は０～９までとします。

ハブを使用してパスワードを作成しましょう。

下枠に疑似コードを記入しましょう。

SAMPLE

SAMPLE

ユニット 8　レッスン 2
物理的に安全にする

下記プログラムを入力して、実行しましょう。

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

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

async def main():
    await sound. beep (267, 200)
    await sound.beep (523, 200)

    # 金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position (port.B, 0)
    await motor.run_to_absolute_position (port.C, 270, 500, direction=motor.SHORTEST_PATH)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を開錠
    await runloop.until(pressed)
    await sound. beep (523, 200)
    await motor.run_for_time(port.C, 1000, 500)
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(5000)

runloop.run(main())
```

金庫をより安全にする

金庫に取り付けられたダイアルを用いて金庫をアップグレードします。
モーターの相対位置を取得できる機能を使ってみましょう。

motor.reset_relative_position(port: int, position: int)
パラメーターで指定した port のモーターの現在の位置を position の値にセットする
例：

```
motor.reset_relative_position(port.B,0) # モーター B の現在の相対位置を 0 に設定
```

motor.relative_position(port: int)-> int
パラメーターで指定した port のモーターが motor.reset_relative_position で設定した値から、正転方向、逆転方向にどれだけの角度、回転したかを返します
例：

```
while motor.relative_position > -180: # 相対位置が -180 度 ( 逆転方向に 180 度 ) 回転するまで待つ
    pass # 何もしない
await sound.play('Triumph')
```

SAMPLE

モーターの相対位置機能を活用して金庫をアップグレードしましょう。

モーターを特定の位置へ回転させると解錠できるようなプログラムを考えます。

下枠に疑似コードを記入してから、プログラムを作成しましょう。

疑似コードをもとにフローチャートを作成しましょう。

SAMPLE

下記2つのプログラムを比較して、開け方にどのような違いがあるでしょうか?。
より適しているものを自分のプログラムに組み込みましょう。

サンプルプログラム 1

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

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

async def main():
    await sound.beep (262, 200)
    await sound.beep (523, 200)

    # 金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position (port.B, 0)
    await motor.run_to_absolute_position (port.C, 270, 500, direction=motor.CLOCKWISE)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を解錠
    while True:
        print('Open Me')
        if motor.relative_position (port.B) > 180:
            await runloop.until(pressed)
            await sound.beep (523, 200)
            await motor.run_for_time(port.C, 1000, 500)
            light_matrix.show_image(light_matrix.IMAGE_YES)
            await runloop.sleep_ms(5000)
            break
        else:
            print('Still Locked!')
            await runloop.sleep_ms(1000)

        print('You may enter!')
runloop.run(main())
```

サンプルプログラム 2

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

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

async def main():
    await sound.beep (262, 200)
```

SAMPLE

```
    await sound.beep (523, 200)

    # 金庫をロック
    motor.stop(port.B)
    motor.reset_relative_position (port.B, 0)
    await motor.run_to_absolute_position (port.C, 270, 500, direction=motor.CLOCKWISE)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を解錠
    while True:
        print('Open Me')
        if motor.relative_position (port.B) > 180:
            await runloop.until(pressed)
            await sound.beep (523, 200)
            await motor.run_for_time(port.C, 1000, 500)
            light_matrix.show_image(light_matrix.IMAGE_YES)
            await runloop.sleep_ms(5000)
            break
        else:
            print('Still Locked!')
            await runloop.sleep_ms(1000)
        print('You may enter!')
runloop.run(main())
```

SAMPLE

ユニット 8　レッスン 3
より安全な金庫を作ろう

好きな食べ物を 10 個リストアップしましょう。

パートナーと好きな食べ物リストを比較して、「 ベン図」 を作りましょう。

下記プログラムを入力して実行してみましょう。

```
from hub import port, sound, light_matrix, button
import runloop
import motor
import app
import time

async def main():
    await sound.beep (262, 200)
    await sound.beep (523, 200)

    # 金庫をロック
    await motor.run_for_time(port.C, 1000, -500)
    await motor.run_to_absolute_position(port.B, 0, 500, direction=motor.SHORTEST_PATH, stop=motor.COAST)
    motor.reset_relative_position (port.B,0)
    await motor.run_to_absolute_position(port.E, 0, 500, direction=motor.SHORTEST_PATH)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # 金庫を解錠
    await unlock()

async def unlock():
    start_time = time.ticks_ms()
    while not button.pressed(button.LEFT) or motor.relative_position (port.B) < 180:
        await sound.beep(262, 200)
        await motor.run_for_degrees (port. E, 15, 500)
```

SAMPLE

```
        await runloop.sleep_ms(800)
        if time.ticks_diff(time.ticks_ms(),start_time) > 5000:
            await app.sound.play('Bonk')
            return
        light_matrix.show_image(light_matrix.IMAGE_NO)

    light_matrix.show_image(light_matrix.IMAGE_YES)
    await motor.run_to_absolute_position(port.E, 0, 500, direction=motor.SHORTEST_PATH)
    await motor.run_for_time(port.C, 1000, 500)
    await app.sound.play('Wand')

runloop.run(main())
```

真理値表（ロジック表）

それぞれの真理値表を完成させましょう

And の真理値表

条件1 (true/false)	and	条件2 (true/false)	結果
True	and	True	
True	and	False	
False	and	True	
False	and	False	

Or の真理値表

条件1 (true/false)	or	条件2 (true/false)	結果
True	or	True	
True	or	False	
False	or	True	
False	or	False	

Not の真理値表

条件 (true/false)	結果
True	
False	

SAMPLE

ユニット 8　レッスン 4
ロジックで操作するセキュリティ

下枠に疑似コードを記入しましょう。

論理演算子を追加した疑似コードを下枠に記入しましょう。

SAMPLE

SAMPLE

ユニット８　レッスン５

脱出しろ！

脱出ゲームで使用するパズルゲームを設計しましょう。

条件

1. ライトマトリクスを使用すること
2. モーターを必ず使用すること（ 複数可）
3. センサーを必ず使用すること（ 複数可）
4. プログラムに論理演算子を含むこと
5. コードを解読するために少なくとも2つのステップを含むこと

作成するセキュリティ装置の設計図を書きましょう。

SAMPLE

フローチャートを作成しましょう。

SAMPLE

ユニット９　レッスン１
データを得るために動き出す

下記プログラムを入力して、実行しましょう。

サンプルプログラム１

```
from hub import port
import runloop
import distance_sensor

async def main():
    # センサーから距離の値を読み込む
    dist_mm = distance_sensor.distance (port.F)
    print('mm:',dist_mm)

runloop.run(main())
```

サンプルプログラム２

```
from hub import port
import runloop
import distance_sensor

async def main():
    # センサーから距離の値を読み込む
    while True:
        dist_mm = distance_sensor.distance (port.F)
        print('mm:',dist_mm)

runloop.run(main())
```

サンプルプログラム３

```
from hub import port
import runloop
import distance_sensor, force_sensor

async def main():
    # センサーから距離の値を読み込む
    while True:
        if force_sensor.pressed (port.B):
            dist_mm = distance_sensor.distance (port.F)
            print('mm:',dist_mm)
            await runloop.sleep_ms(1000)

runloop.run(main())
```

SAMPLE

様々なデータ型

整数型(int型)	APIモジュールのほとんどの関数の戻り値やパラメーターに使用されている。 例：distance_sensor.distance()やforce_sensor.force()の戻り値
文字列型(str型)	print関数やlight_matrix.write関数などのパラメーターで使用されている。
浮動小数点型(float型)	APIモジュールでは使用されていない。変数に少数を含む値を代入した場合この型になる。 例：PI = 3.14　print(type(PI))

※ Python では float 型は単精度浮動小数点型ではなく倍制度浮動小数点型である。

フォースセンサーの値を取得し、それを整理、表示する方法を考えましょう。

SAMPLE

ユニット９　レッスン２
データを求めて自転車に乗る

まずはモデルを動かしてみましょう。

コメントを参考に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 motor_pair

async def main():
    # モーターをペアにする
    motor_pair.pair (motor_pair.PAIR_1, port.C, port.E)
    # 速度 250 で 720 度分直進する
    motor_pair._____

runloop.run(main())
```

四則演算

ハブの左右のボタンを押すことで自転車の速度を変えられるようなプログラムを作成しましょう。

四則演算に対応する算術演算子を以下に示します。

関数	記号	サンプル	結果
加算	+	print(10+2)	12
減算	-	print(10-2)	8
乗算	*	print(10*2)	20
除算	/	print(10/2)	5

サンプルプログラムを入力して、実行しましょう。

サンプルプログラム：

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

async def ride (bike_velocity):
    # バイクの速度をライトマトリックスに表示して直進する関数 ride() を定義
    await light_matrix.write(bike_velocity)
    await motor_pair.move_for_degrees (motor_pair.PAIR_1,720, 0, velocity=bike_velocity)

async def main():
    bike_velocity = 250
    # モーターをペアにする
    motor_pair.pair (motor_pair. PAIR_1, port.C, port.E)
    # 速度 250 で 720 度分直進する
    while True:
```

SAMPLE

```
    # 左ボタン押されたら 50 ずつ減速
    if button.pressed (button.LEFT):
        bike_velocity -= 50
        await ride(bike_velocity)
        await runloop.sleep_ms(2000)

    # 右ボタン押されたら 50 ずつ加速
    if button.pressed (button.RIGHT):
        bike_velocity += 50
        await ride(bike_velocity)
        await runloop.sleep_ms(2000)

runloop.run(main())
```

プログラム中の "+=" や "-=" は代入演算子といいます。
これは a = a+b や a=a-b と同義であり a+=b,a-=b と省略してプログラムを書くことができます。
これは四則演算の算術演算子すべてに対応しているため、*= や /= とプログラムに書くことができます。

プログラムを変更しよう

今のプログラムを左ボタンが押されたとき 1/2 に減速、右ボタンが押されたら 2 倍に加速するように変更しましょう。

乗算除算を行うと変数は浮動小数点型になります。
ハブの表示が小数点以下が表示されるのが気になる人は下記のようにプログラムに変更しましょう。

```
await light_matrix.write(str(bike_velocity))
```

str() はキャスト変換をする関数であり、() 内のデータを別の型に変換します。str() は () 内のデータを文字列型に変換したものを戻り値として返します。変換したデータを返すだけであり、変数の型を変更するもではないことには注意が必要です。

例：

```
a = 56 * 3.14
print(int(a),type(int(a)))
print(a,type(a))
```

SAMPLE

ユニット 9　レッスン 3

歩数計

サンプルプログラムを入力して、実行しましょう。

サンプルプログラム：

```
from hub import light_matrix,motion_sensor,button
import runloop

async def main():
    step = 0
    # ヨー角を 0 にリセットする
    motion_sensor.reset_yaw(0)
    await light_matrix.write('Walk!')

    while True:
        # 傾き角を取得
        tilt = motion_sensor.tilt_angles()
        # タプルからヨー角を取り出す
        yaw = tilt[0]

        if yaw > 75:
            step += 1
        elif yaw < -75:
            step += 1
        elif button.pressed(button.LEFT):
            await light_matrix.write(str(step))
            break

runloop.run(main())
```

実行したけっかどうなりましたか？

比較演算子

関数	記号
未満	<
以下	<=
超過	>
以上	>=
等しい	==
等しくない	!=

比較演算子は文字通り比較をする演算子です。

a >= b は a が b 以上であるという意味になり、比較の結果が真なら True を，偽なら False を返します。

SAMPLE

プログラムの改良

サンプルプログラムを入力して、実行しましょう。

サンプルプログラムがどのようにして、1歩だけ進んだことを判断しているか考察しましょう。

サンプルプログラム：

```
from hub import button, light_matrix, motion_sensor
import runloop
import time

# タイマーの時間差を算出する関数
def time_since (some_time):
    return time.ticks_diff(time.ticks_ms(), some_time)

async def main():
    # 変数を定義
    step =0
    true_step = False

    await light_matrix.write('Walk!')
    while True:
        # 傾き角を取得
        tilt = motion_sensor.tilt_angles()
        # ヨー角を取り出す
        yaw = tilt[0]

        if yaw > 75 and true_step is False:
            step += 1
            true_step = True
        elif yaw < -75 and true_step is True:
            step += 1
            true_step = False
        elif button.pressed (button.LEFT):
            await light_matrix.write(str(step))
            break

runloop.run(main())
```

速度を求める

歩幅をあらかじめ登録しておくことで、歩幅 * 歩数で歩行距離を算出することができます。

最後に歩行時間、歩行距離、歩行速度を出力するようにプログラムを変更してください。

歩行時間を算出するためには、タイマー機能を使用する必要があります。タイマーはミリ秒で時間を進めます。

```
    # 時間の変数定義 ( 開始の時間、停止の時間)
    start_time = 0
    elasped = 0
    # タイマースタート
    start_time = time.ticks_ms()
    # タイマーストップ
    elasped = time_since (start_time)
```

SAMPLE

ユニット 9　レッスン 4
動かしてみよう

ドライビングベースを動かすプログラムを作成します。
ユニット 9 レッスン 2 の " データを求めて自転車に乗る " のプログラムを思い出しながらプログラムを作成します。

プログラムの必要事項を示します。
☐　左ボタンで減速、右ボタンで加速
☐　ライトマトリクスに速度を表示

上記のことを念頭に置いて、下記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, light_matrix, button
import runloop
import motor_pair

async def main():
    driving_velocity = 250
    # モーターをペアにする
    motor_pair.pair(_____________________________)
    #ride 関数を定義
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 720, 0, velocity = _____________)

    # 前進する条件と速度を設定する
    while True:
        if button.pressed(button.LEFT):
            _____________ -= 50
            await ride()
            await runloop.sleep_ms(3000)
        elif button.pressed (button.RIGHT):
            _____________ += 50
            await ride()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

曲線走行

下記プログラムを入力、実行してみましょう。

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

SAMPLE

```
async def main():
    # 速度の初期設定値
    driving_velocity = 250
    # ターンの距離を設定
    turning_left = -360
    turning_right = 360
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
    #ride 関数と turn 関数を定義する
    async def ride():
        await light_matrix.write(str(driving_velocity))
        await motor_pair.move_for_degrees (motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)

    async def turn_left():
        await light_matrix.write(str(turning_left/4))
        await motor.run_for_degrees (port.C, turning_left, 250)

    async def turn_right():
        await light_matrix.write(str(turning_right/4))
        await motor.run_for_degrees(port.D, turning_right, 250)

    # 前進する速度と条件を設定
    while True:
        await ride()
        if button.pressed (button.LEFT):
            driving_velocity -= 50
            turning_left -= 360
            await turn_left()
            await runloop.sleep_ms(3000)
        elif button.pressed (button.RIGHT):
            driving_velocity += 50
            turning_right += 360
            await turn_right()
            await runloop.sleep_ms(3000)

runloop.run(main())
```

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
どんなエラーが出ているかを確認しましょう

SAMPLE

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

async def main():
    # 速度の初期設定値
    driving_velocity = 250

    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

    #ride 関数を定義する
    async def ride():
    await light_matrix.write(str(driving_velocity))
    await motor_pair.move_for_degrees (motor_pair.PAIR_1,0, velocity = driving_velocity)

    # 前進する速度と条件を設定
    while True:
        if button.pressed (button.LEFT):
            driving_velocity -= 50
            await ride()
            await runloop.sleep_ms(3000)

        elif button.pressed (button.RIGHT):
            driving_velocity += 50
            await ride()
            await runloop.sleep_ms(3000)

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

どんなエラーが出ているかを確認しましょう

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

driving_velocity = 250

async def main():
```

SAMPLE

```
 8:  # モーターをペアにする
 9:  motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
10:
11:  #ride 関数を定義する
12:  async def ride():
13:      await light_matrix.write(str(driving_velocity))
14:     await motor_pair.move_for_degrees (motor_pair.PAIR_1, 720, 0, velocity = driving_velocity)
15:
16:  # 前進する速度と条件を設定
17:  while True:
18:      if button.pressed (button.LEFT):
19:          driving_velocity -= 50
20:          await ride()
21:          await runloop.sleep_ms(3000)
22:
23:      elif button.pressed (button.RIGHT):
24:          driving_velocity += 50
25:          await ride()
26:          await runloop.sleep_ms(3000)
27:
28: runloop.run(main())
```

SAMPLE

ユニット 9　レッスン 5

駐車場

自律駐車をする車をグループで作成します。

ルール

- ☐　2 台の車は同時に駐車場に入らなければならない。
- ☐　どちらの車も駐車スペース以外では止まってはならない。
- ☐　駐車場に入ったら、誰も車に触れてはならない。
- ☐　車がバックできるのは、スペースから出るときだけです。
- ☐　車のどの部分も線を越えて動くことはできない。

グループでディスカッションを行い意見や設計に取り入れるべき要素などを下枠に記入しましょう。

SAMPLE

プログラムを作成します。

下記の要素を含むプログラムを作成しましょう

☐ 駐車場全体の距離を使用すること。

☐ 複数の数学関数を使用すること。

☐ 距離センサーを使用すること。

下枠にプログラムの疑似コードまたはフローチャートを記入しましょう。

SAMPLE

ユニット９　レッスン ６

私の交通手段

通学に必要な乗り物を設計、製作、プログラムします。

下記に制約条件を示します。

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. プログラムに数学関数が含まれていること。
5. 少なくとも２種類の地形に対応すること。

下枠に設計図を書きましょう。

SAMPLE

下枠にプログラムの疑似コードなたはフローチャートを記入しましょう。

SAMPLE

ユニット 10　レッスン 1
文字のリストアップ

下記プログラムを記入、実行しましょう。

```
from hub import light_matrix, button
import runloop

async def main():
    # リストを定義
    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'T, T, K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    # リストの4番目をライトマトリックスに表示する
    await light_matrix.write(letters[4])

runloop.run(main())
```

プログラムを実行した結果はどうなりましたか？

このことから、何がわかりますか？

4番目の D を表示させたい場合は、インデックスを何にすればよいですか？

SAMPLE

下記プログラムを記入、実行しましょう。

```
from hub import light_matrix, button
import runloop
import random

async def main():
    # リストを定義
    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'T, T, K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    while True:
        if button.pressed(button.LEFT):
            await runloop.sleep_ms(500)
            # ランダムに選択してライトマトリックスに表示する
            await light_matrix.write(random.choice(letters))

runloop.run(main())
```

リストの先頭から順番（ A から順番に Z まで）に表示するプログラムを作ってみましょう。
下記の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 light_matrix, button
import runloop
import random

async def main():
    # リストを定義
    letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'T, T, K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']


    ____________________
        # 先頭から順番にライトマトリックスに表示する
        await light_matrix.write(_____________)
        await runloop.sleep_ms(500)

runloop.run(main())
```

逆順（ Z から A へ）に表示するにはどうしたらよいでしょう？
やってみてください。

SAMPLE

ユニット 10　レッスン 2
筋肉とリストを伸ばす

下記プログラムを記入、実行しましょう。
yoga.moves[3] のインデックスを変更してみましょう。

```
from hub import motion_sensor
import runloop

async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]

    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[3]:
            print('Good Job!')
            await runloop.sleep_ms(1000)

runloop.run(main())
```

ヨガのルーチンを作る

下記プログラムを記入、実行しましょう。

```
from hub import motion_sensor, light_matrix
import runloop
async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]

    # 次のルーティンに従います
    print('LEFT')
    print('RIGHT')
    print('BACK')
    print('FRONT')
    print('READY GO!')
    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()
```

SAMPLE

```
    if orientation == yoga_moves[2]:
        print('Left Side')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_W)
    elif orientation == yoga_moves[3]:
        print('Right Side')
        light_matrix.show_image(light_matrix. IMAGE_ARROW_E)
    elif orientation == yoga_moves[ 1 ]:
        print('Back')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_N)
    elif orientation == yoga_moves[0]:
        print('Front')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

動きをミックスしたり、ルーチンにいくつかのステップを加えたりしてみましょう。

動きをランダムにする

下記プログラムを記入、実行しましょう。

```
from hub import motion_sensor, light_matrix
import random
import runloop

async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]
    hub_faces = {1:'FRONT',4:'BACK',5:'LEFT',2:'RIGHT'}

    random_move = random.choice(yoga_moves)

    # 次のルーティンに従います
    print('LEFT')
    print('RIGHT')
    print(hub_faces[random_move])
    print('BACK')
    print('FRONT')
    print(hub_faces[random_move])
    print('READY GO!')
    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[2]:
```

SAMPLE

```
        print('Left Side')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_W)
    elif orientation == yoga_moves[3]:
        print('Right Side')
        light_matrix.show_image(light_matrix. IMAGE_ARROW_E)
    elif orientation == random_move:
        print('Random')
        light_matrix.show_image(light_matrix.IMAGE_SILLY)
    elif orientation == yoga_moves[1]:
        print('Back')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_N)
    elif orientation == yoga_moves[0]:
        print('Front')
        light_matrix.show_image(light_matrix.IMAGE_ARROW_S)

runloop.run(main())
```

random_move = random.choice(yoga_moves) 行ではランダムモジュールを使い疑似乱数を生成しています。choice() 関数はリストからランダムに要素を一つ選択して返します。

リストの追加と削除

下記プログラムを記入、実行しましょう。

```
from hub import motion_sensor, light_matrix
import runloop

async def main():
    # ハブの姿勢でリストを定義
    yoga_moves = [motion_sensor.FRONT, motion_sensor.BACK, motion_sensor.LEFT, motion_sensor.RIGHT]
    # リストに要素を追加挿入
    yoga_moves.insert(0,motion_sensor.TOP)
    yoga_moves.insert(1,motion_sensor.BOTTOM)

    orientation = motion_sensor.up_face()

    while True:
        await runloop.until(lambda: motion_sensor.up_face() != orientation)
        orientation = motion_sensor.up_face()

        if orientation == yoga_moves[0]:
            print('Top')
            light_matrix.show_image(light_matrix.IMAGE_TRIANGLE)
        elif orientation == yoga_moves[1]:
            print('Bottom')
            light_matrix.show_image(light_matrix. IMAGE_SQUARE)

runloop.run(main())
```

SAMPLE

list.insert(index, object)

object を指定した位置 (index) に挿入します。
index に 0 を指定した場合はリストの先頭に挿入します。
また、index に負の値を指定すると、リストの末尾からの相対位置に挿入します。

del 文

インデックス、またはスライスを指定し要素を削除します。

```
del list[1] # 2番目の要素を削除
del list[:2] # 先頭から2番目までの要素を削除
del list[2:5] # 3番目から5番目の要素を削除
```

ほかにも
特定のインデックスではなく、最初に一致する値を削除する list.remove(object)
特定のインデックスにある項目を削除して返す list.pop(index)

などがあります。
プログラムに合わせて関数を使い分けましょう。

Tips: リストとタプルと辞書オブジェクト

Python では、一般に配列と呼ばれる順序付きのコレクションをシーケンスと呼びます。
シーケンスにはリスト型、タプル型、文字列型、辞書オブジェクトなどのデータ型があります。
その中でリスト、タプル、辞書オブジェクトを紹介します。

リスト

リストは要素の削除、追加、入れ替えが自由に行えるシーケンスオブジェクトです。
要素へのアクセスはインデックスを指定します。
このユニットで主に扱う型であり、よく使われる型でもあります。
リストオブジェクトは [要素 , 要素 ,…] のように [] で要素を囲んで作成します。

```
L = [1,2,3]
```

タプル

タプルはリストによく似たシーケンスオブジェクトですが、リストと違い更新不可なオブジェクトであり、
タプル作成後に要素の追加や削除は行えません。
要素へのアクセスはインデックスを指定します。
タプルオブジェクトは (要素 , 要素 ,…) のように () で要素を囲んで作成します。
SPIKE Python ではカラーセンサーの rgbi 関数の戻り値、モーションセンサーの tilt_angles の戻り値などで使われています。
例）RED = (255,0,0)

辞書オブジェクト

辞書オブジェクトは、一般に連想配列などと呼ばれるデータ構造です。
辞書には、キーと、キーに関連付けられた値の組を要素として登録し、登録した値はキーを使って参照できます。
辞書オブジェクトは { キー : 要素 , キー : 要素 ,…} のように : で区切ったキーと値を {} で囲んで作成します。
例）hub_faces = {1:'FRONT',4:'BACK',5:'LEFT',2:'RIGHT'}

SAMPLE

ユニット 10　レッスン 3
マインドゲーム

下記プログラムを記入、実行しましょう。

```
from hub import light_matrix, button, port
import runloop
import motor
import color_sensor
import color
from app import sound

def feed_candy_1():
    return button.pressed(button.LEFT)

def feed_candy_2():
    return button.pressed(button.RIGHT)

async def main():
    # リストを作成する
    candy1 = []
    candy2 =[]
    # ゲームマスターが棒付きキャンディー 1 を食べる
    await runloop.until(feed_candy_1)
    light_matrix.clear()
    candy1.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')
    # キャンディー 1 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy1.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)
    # ゲームマスターが棒付きキャンディー 2 を食べる
    await runloop.until(feed_candy_2)
    light_matrix.clear()
    candy2.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')
    # キャンディー 2 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy2.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees (port.A, 95, 500)
```

SAMPLE

```
    # 両方のキャンディスティックの同じ位置に赤いブロックがある場合、赤いブロックの位置を照らす。
    candy1_red_index = -1
    if color.RED in candy1:
        candy1_red_index = candy1.index(color.RED)
    candy2_red_index = -1
    if color. RED in candy2:
        candy2_red_index = candy2.index(color.RED)
    for x in range(5):
        print(candy1[x])
    if candy1_red_index == candy2_red_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_red_index, 100)
            await sound.play('Win')
    else:
        light_matrix.show_image(light_matrix.IMAGE_NO)
    await sound.play('Oops')

runloop.run(main())
```

メンバーシップ演算子 "in"

in 演算子はブール演算子で、シーケンスにオブジェクトが存在する場合 True を返します。

list.append(object)

リストの末尾に指定された値を追加します。

プログラムを説明してください。

すべての色を確認できるようにプログラムを改良しましょう。
下枠に具体的な改良案と疑似コードを記入しましょう。

SAMPLE

```
from hub import light_matrix, button, port
import runloop
import motor
import color_sensor
import color
from app import sound

def feed_candy_1():
    return button.pressed(button.LEFT)

def feed_candy_2():
    return button.pressed(button.RIGHT)

async def main():
    # リストを作成する
    candy1 = []
    candy2 = []
    # ゲームマスターが棒付きキャンディー 1 を食べる
    await runloop.until(feed_candy_1)
    light_matrix.clear()
    candy1.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')

    # キャンディー 1 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy1.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)

    # ゲームマスターが棒付きキャンディー 2 を食べる
    await runloop.until(feed_candy_2)
    light_matrix.clear()
    candy2.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')

    # キャンディー 2 と呼ばれるリストに色のシーケンスを読み込んで記録する。
    for x in range(5):
        candy2.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)

    candy1_red_index = candy1.index(color.RED) if color.RED in candy1 else -1
    candy2_red_index = candy2.index(color.RED) if color.RED in candy2 else -1
```

SAMPLE

```
    candy1_yellow_index = candy1.index(color.YELLOW) if color.YELLOW in candy1 else -1
    candy2_yellow_index = candy2.index(color.YELLOW) if color.YELLOW in candy2 else -1
    candy1_green_index = candy1.index(color.GREEN) if color.GREEN in candy1 else -1
    candy2_green_index = candy2.index(color.GREEN) if color.GREEN in candy2 else -1
    candy1_magenta_index = candy1.index(color.MAGENTA) if color.MAGENTA in candy1 else -1
    candy2_magenta_index = candy2.index(color.MAGENTA) if color.MAGENTA in candy2 else -1
    candy1_blue_index = candy1.index(color.BLUE) if color.BLUE in candy1 else -1
    candy2_blue_index = candy2.index(color.BLUE) if color.BLUE in candy2 else -1
    # 数字は色に対応しています
    for x in range(5):
        print(candy1[x],candy2[x])
    # 色ブロックが両方のキャンディスティックで同じ位置にある場合、色ブロックの位置を点灯させる。
    if candy1_red_index == candy2_red_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_red_index, 100)
    await sound.play('Win')

    if candy1_yellow_index == candy2_yellow_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_yellow_index, 100)
        await sound.play('Win')

    if candy1_green_index == candy2_green_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_green_index, 100)
        await sound.play('Win')

    if candy1_magenta_index == candy2_magenta_index:
        for x in range(5):
            light_matrix.set_pixel(x,candy1_magenta_index, 100)
        await sound.play('Win')

    if candy1_blue_index == candy2_blue_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_blue_index, 100)
        await sound.play('Win')

     if candy1_red_index == candy2_red_index and candy1_yellow_index == candy2_yellow_index and candy1_
green_index == candy2_green_index and candy1_magenta_index == candy2_magenta_index and candy1_blue_
index == candy2_blue_index:
        await sound.play('Triumph')

runloop.run(main())
```

プログラムを説明してください。

SAMPLE

ユニット 10　レッスン 4
リストへのジャンプ

下記プログラムを記入、実行しましょう。
また記録は下記表に記入しましょう。

```
from hub import port, light_matrix
import runloop
import distance_sensor
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    runloop.until(is_force_sensor_pressed)
    await light_matrix.write('JUMP!')
    runloop.until(is_force_sensor_released)
    dist_cm = distance_sensor.distance(port.F)/10
    print(str(dist_cm) + 'cm')

runloop.run(main())
```

ジャンプをして、高さを記録しましょう。
1行目にはあなたの名前とパートナーの名前を記入します。それぞれどのくらいの高さジャンプできたかを3回記録します。

	一人目	二人目
1回目		
2回目		
3回目		

最高スコアを求めよう

下記プログラムを記入します。
name1,name2 には自身とペアの名前を、"1 回目の値 , 2 回目の値 , 3 回目の値 " には表からそれぞれのスコアを入力しましょう。

```
from hub import port, light_matrix
import runloop
import force_sensor
```

SAMPLE

```
# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたらライトマトリクスに最大値を表示
    await runloop.until(is_force_sensor_pressed)
    await light_matrix.write(str(max(name1)))
    await runloop.sleep_ms(2000)
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(1000)
    await light_matrix.write(str(max (name2)))
    await runloop.sleep_ms(2000)
    light_matrix.show_image(light_matrix.IMAGE_YES)

runloop.run(main())
```

max()

与えられたシーケンスオブジェクト、または 2 つ以上の引数の中で最大のものを返します。

プログラムを説明してください。

2 人のスコアで最も高いスコアを求めましょう。
どのようにして求めればよいか、下枠に案を書きましょう。

SAMPLE

サンプルプログラム：

```
from hub import port, light_matrix
import runloop
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたら 2 つのリストを結合
    await runloop.until(is_force_sensor_pressed)
    combo_list = name1 + name2
    # 結合したリストの内容をコンソールに表示
    print(combo_list)
    # 結合したリストの中の最大の値を取り出してコンソールに表示

    # 結合したリストの数値の合計をライトマトリックスに表示
    await light_matrix.write(str(sum(combo_list)))
    light_matrix.show_image(light_matrix.IMAGE_YES)

runloop.run(main())
```

list1 + list2 と書くことでリストを結合することができます。

リストに特定の値が含まれているかをメンバーシップ演算子を用いて確認するプログラムを追記します。
メンバーシップ演算子に関してはレッスン 3 を確認しましょう。

SAMPLE

サンプルプログラム：

```
from hub import port, light_matrix
import runloop
import force_sensor

# フォースセンサーの状態を調べる関数
def is_force_sensor_pressed():
    # 押されているかどうか
    return force_sensor.pressed(port.B)

def is_force_sensor_released():
    # 解放されているかどうか
    return not force_sensor.pressed (port.B)

async def main():
    # 表のデータからそれぞれの名前のリストを作成
    #name1,name2 はそれぞれの名前
    name1 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]
    name2 = [1 回目の値 , 2 回目の値 , 3 回目の値 ]

    # それぞれのリストから最大の値を取り出してコンソールに表示
    print('Name 1:' + str(max(name1)))
    print('Name 2:' + str(max(name2)))

    # フォースセンサーが押されたら 2 つのリストを結合
    await runloop.until(is_force_sensor_pressed)
    combo_list = name1 + name2
    # 結合したリストの内容をコンソールに表示
    print(combo_list)
    # 結合したリストの数値の合計をライトマトリックスに表示
    await light_matrix.write(str(sum(combo_list)))
    print(sum(combo_list))
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(2000)
    light_matrix.clear()

    #x: 表の中のデータから 1 つ選択
    if x in combo_list:
        print('The number x is in the list')
        light_matrix.show_image(light_matrix.IMAGE_YES)
    else:
        print('The number x is not in the list')
        light_matrix.show_image(light_matrix.IMAGE_NO)

runloop.run(main())
```

SAMPLE

ユニット 10　レッスン 5
単語ゲームとリスト

単語ゲームに用いる単語を下枠に書きましょう。

下枠に疑似コードを記入しましょう。

SAMPLE

下枠にフローチャートを作りましょう

SAMPLE

Lesson Completed

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 3

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. ©2025 The LEGO Groupe. All rights reserved.

SAMPLE