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 日本語版

Learning Systems™

目　次

レゴ エデュケーション SPIKE プライム セットを使用した Python プログラミング入門

概要

このコースでは、学生は、LEGO Education SPIKE Prime セットを使用して、プログラミングのベストプラクティスとともに Python プログラミング言語の基礎を学びます。一連のレッスンを通じて、学生は重要なライブラリ、ハードウェアとソフトウェアの使用方法を学びます。 モーターとセンサーを制御し、条件文とループを使用してプログラムのフローを制御し、Python データ型と変数を使用してデータを保存します。彼らは独自のカスタム プログラムを定義して文書化し、スクリプトを作成し、エラーを処理します。最も重要なことは、学生が Python でのコーディング スキルを練習および上達するために、この知識すべてを実際の状況で使用する機会が複数、継続的にあることです。コースの終わりまでに、生徒は次のような力をつけます。

- ロボットまたはモデルのプロトタイプを設計、対話的に開発、プログラミングする
- 協力して作業し、フィードバックをやり取りし、提案を取り入れる
- ハードウェアとソフトウェアの両方の問題をデバッグおよびトラブルシューティングする
- アルゴリズム、データ、複合条件、センサー、ループ、ブール論理を使用しコーディングする
- フローチャートまたは疑似コードを明確にして複雑な問題に対処する
- 問題をサブ問題へ部分的に分解する
- バイアスとアクセシビリティの問題について話し合う
- モデルやプログラミングを含む問題の解決策を伝達する

レッスンの目的

このコースの期間中、学生は抽象化、アルゴリズム、プログラミング、データの分野でアイデアを分析、プロトタイピング、コミュニケーションすることで、Python プログラミングの知識を深めます。 学生は、Python プログラムを作成することで、成果物を作成し、モーター、センサー、ライト、サウンドを効果的に使用するモデルを構築します。 彼らは、疑似コードの作成、条件ステートメント、ループ、ブール論理の使用、線形および計算的思考を含むさまざまなプログラミング手法を利用して、さまざまなタスクを実行します。 学生は、Python の知識をさまざまなガイド付きのオープンエンドのプロジェクトに適用し、最終的には現実世界の問題に対する解決策を提示します。

コースデザイン

このコースは、数多くの Python プログラミング スキルと成果に取り組むために開発されました。 教師が生徒に Python プログラミングの習熟度を高める継続的な機会を提供できるように、レッスンは構造的にしっかりとしたエクスペリエンスにまとめられ、複雑さが増しました。このコースは 10 のユニットに分かれています。ユニットには 4 ～ 7 つのレッスンがあります。各 45 ～ 60 分のレッスンで、生徒は Python の習熟度を高めるための高いレベルの取り組みを経験します。

このコースには次の単元が含まれています。

- ユニット 1: ハードウェア + ソフトウェア
- ユニット 2: モーター
- ユニット 3: センサー制御
- ユニット 4: ループと変数
- ユニット 5: ゲームの条件
- ユニット 6: トラブルシューティングとデバッグ
- ユニット 7: 関数
- ユニット 8: 複合条件と論理演算子
- ユニット 9: データと数学関数
- ユニット 10: リスト

※日本語翻訳版では、ユニット1～3を Part 1、ユニット 4 ～ 6 を Part 2、ユニット 7 ～ 10 を Part 3として提供します。

レッスン全体を通じて、戦略的な質問と主要な目標が、Python プログラミングの熟練度を向上させるプロセスを通じて学生を導きます。各レッスンにリストされている主な目標は、各生徒が関連するスキルを身につけているかどうかを判断するために使用できます。コース内のプロジェクトには、以前に説明した Python の概念の全体的な理解と応用を詳しく説明するためのジャーナリングやルーブリックを介した特定のドキュメントが含まれます。

Python を使用して SPIKE アプリを始める

新しいプロジェクトを開く

生徒にソフトウェアを開いてライブラリのオプションを確認するように指示します。

1. APIKE アプリを開く
2. 新しいプロジェクトを選択します
3. Python を選択
4.「 作成」を選択します

2.

3.

4.

プログラミング キャンバス

最初のレッスンで生徒にソフトウェアの機能を数分間見てもらい、 下の図に示すように重要な機能を説明しておくことをお勧めします。

1. メイン画面の白いスペースがプログラミング エリアです。新しい Python プロジェクトを開くと、プログラミングエリアにサンプル プログラムがすでに読み込まれています。
2. 左上隅にハブアイコンが表示されています。ここでハブを接続します。接続所の状態によってアイコンを囲む色が変化します。
3. 下部にコンソールと、画面サイズの変更や操作の取り消しを可能にするいくつかの機能が表示されます。 コンソールには、print() 関数で表示したメッセージやエラー メッセージが表示されます。 コンソールが表示されていない場合は、キャンバスの中央下部にある 2 本の水平バーをクリックします。 コンソールが上に展開して表示されます。
4. 右側にナレッジベースが表示されます。 ナレッジ ベースはサポートを提供し、コードの参照として機能します。

ハブアイコンの状態

ハブアイコンはハブとの接続の状態を色で表します。

接続なしの状態

USB ケーブルで接続中

Bluetooth で接続中

更に、ハブにモーターやセンサーが接続されていると、ポートと接続中のセンサー、モーターのリアルタイムのじょうほうが表示されます。

ナレッジベースの使用

ナレッジベースでは、入門サポート(スタートガイド) と、Python を使用した SPIKE Prime のテキストコーディングへの簡単な移行を提供します。ナレッジベースは API ライブラリで、SPIKE Prime セットからハードウェアにリンクされた機能を見つける場所を提供します。さらに、プログラミングキャンバスにコピーアンドペーストできるサンプルコード、エラーメッセージの説明、各関数の型と値が提供されます。

スタートガイド

入門プロジェクトです。1. から順番に進めることで SPIKE Prime での Python テキストコードの基礎を学べます。

API モジュール

SPIKE Prime 用に用意されている Python ライブラリの情報を提供します。

ライブラリ詳細

使用したいハードウェアを選択すると関連するライブラリが表示されます。使用したいコマンドを選ぶと詳細、サンプルが表示されます。

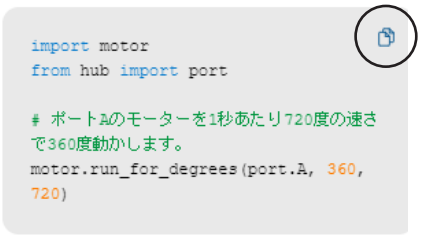
ナレッジベースからのコピー&ペースト

プログラミングを簡単にするために、ナレッジ ベースでいくつかのサンプル プログラムが提供されています。 学生はいつでもこれらをコピーしてプログラミング キャンバスに貼り付けることができます。

コピーして貼り付け :

1. ナレッジ ベース内のプログラミング ボックスの右上隅にある小さな青いアイコンをクリックします。
2. コードを貼り付けたいボックス内で右クリックします。
3. 「 貼り付ける」を選択してします。

1.



2.

3.

```
1 import motor
2 from hub import port
3
4 # ポートAのモーターを1秒あたり720度の速さで360度動かします。
5 motor.run_for_degrees(port.A, 360, 720)
```

Bluetooth 経由でハブを接続する

ハブをデバイス(ソフトウェア) に接続する方法を生徒に説明します。 ハブは USB ケーブルまたは Bluetooth 経由で接続できます。

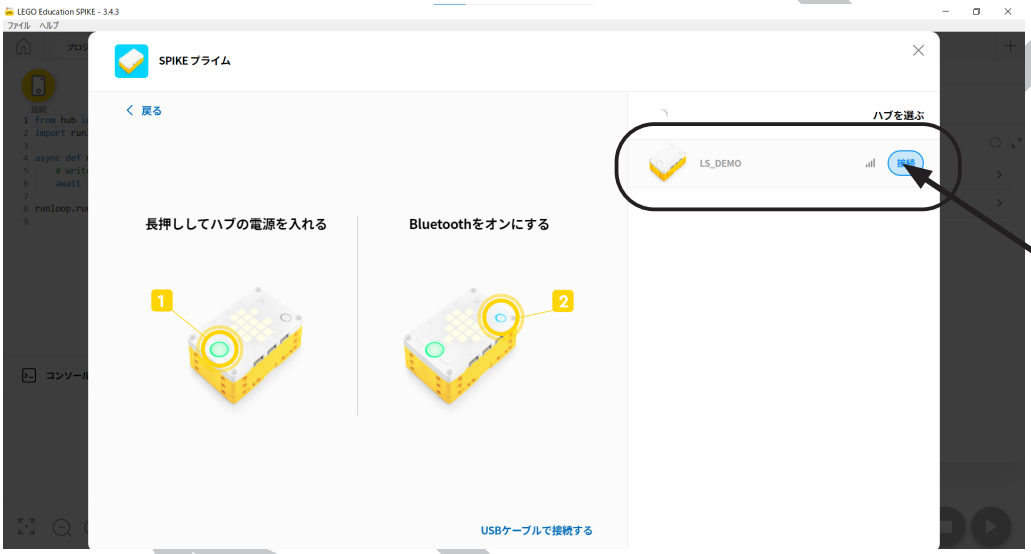
Bluetooth 経由で接続 :

1. ハブアイコンをクリックします
2. ハブの右上部分にある小さな丸いボタンを押して Bluetooth をオンにします
3. 数秒以内にハブ名が画面の右側に表示されます
4. 正しいハブ名を選択し「 接続」をクリックすると接続されます

1.

2.

3.



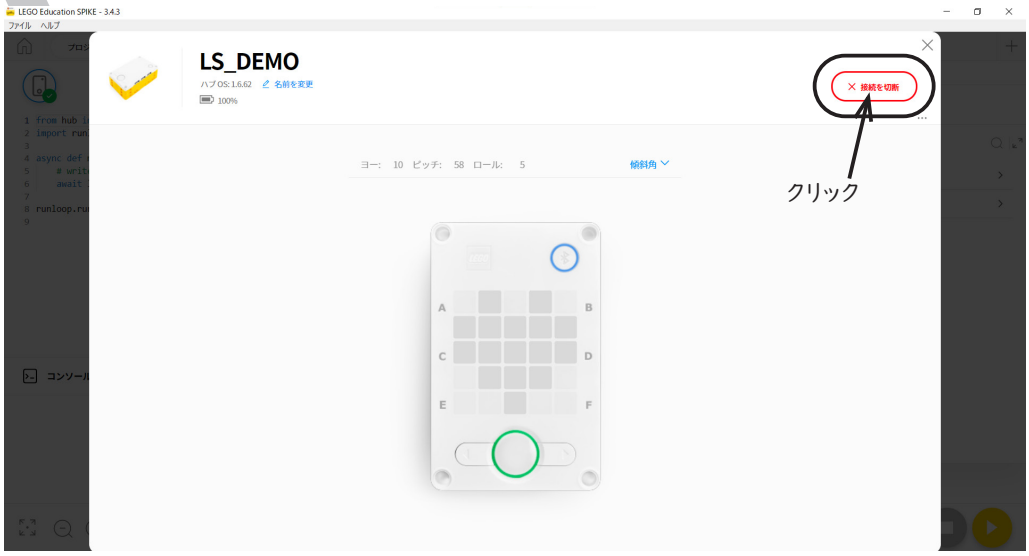
4. クリック

接続を切断する場合

1. ハブアイコン(Bluetooth 接続中) をクリック
2. 右上の「 接続を切断」をクリックすると切断します

1.

2.



プロジェクトの名前を変更する

プロジェクトは [マイ プロジェクト] タブに保存されます。 プロジェクトを簡単に見つけられるように、生徒は各プロジェクトに課題に関連した名前を付ける必要があります。

プロジェクトの名前を変更する :

1. プロジェクト名の右側にある 3 つの小さな縦点をクリックします。
2. ポップアップ メニューに 2 つの選択肢 (プロジェクトの名前を変更) と移動先が開きます。
3. プロジェクト名の変更を選択します。
4. 画面が変わり、現在の名前が表示されます。
5. そこにあるものを消去し、独自のタイトルを入力します。
6. 「 保存」をクリックします。

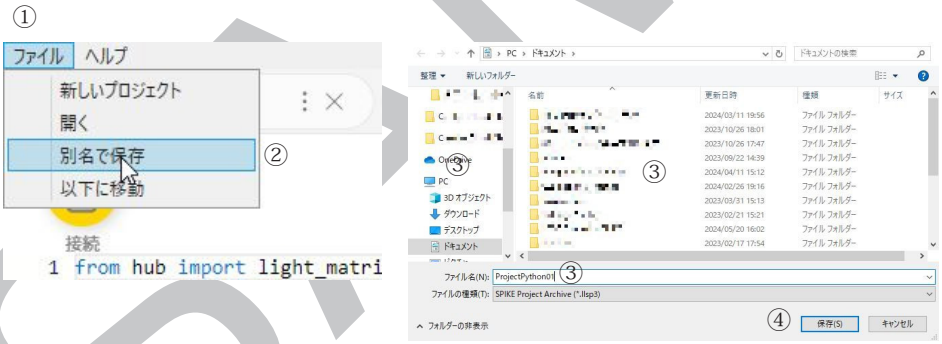
* メニューが閉じてプログラミング キャンバスに戻ります**プロジェクトの名前を変更する**

プロジェクトは [マイ プロジェクト] タブに保存されます。 プロジェクトを簡単に見つけられるように、生徒は各プロジェクトに課題に関連した名前を付ける必要があります。

別名を付けて保存

別の方法は、ウィンドウ左上の「 ファイル」メニューから「 別名で保存」を選び、名前を付けて保存する方法があります。

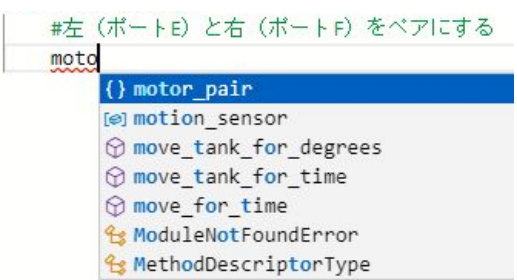
1. 「 ファイル」メニューを選択
2. 「 別名で保存」を選択
3. 保存する先のフォルダを選択し、名前を変更
4. 「 保存」を選択。ウィンドウが閉じてプログラミングキャンバスに戻ります。



その他便利な機能

入力補完機能

Python では各種コマンドやパラメーター名など決まった単語を入力する必要があり、一文字でも間違うとエラーとなってしまいます、また全てのコマンドやパラメーター名などの綴りを覚えておくこともできません。そこで、SPIKE App の Python プログラミングでは、コマンドの一部を入力すると、それを補完するように入力候補の一覧が表示される機能があります。候補を見ながら入力、または候補を選択 (マウスでクリック、なたは矢印キーで移動してエンターキーで確定) すると正確に入力できる機能です。



入力補完機能は、単語（コマンドや定数）のみならず、関数に設定すべき（設定できる）パラメーター（引数）がどのようなものかも示してくれます。

エラー警告機能

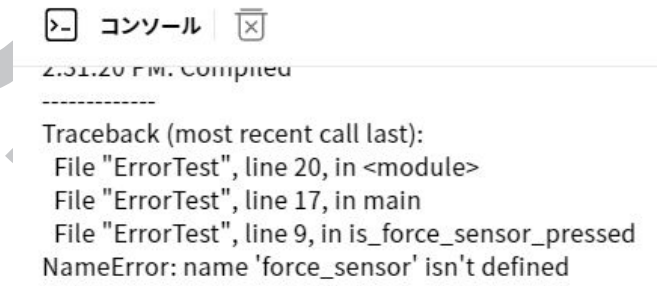
コードの中でエラーが発生部分があると、その部分に赤の波線が現れ、エラーの警告をしてくれます。
右のプログラムでは、force_sensor のところに赤波線が出ています。

```
#モーターペア、ポートをインポート
from hub import port
import runloop
import motor_pair


def is_force_sensor_pressed():
    #力センサーの入力を収集して返す
    return force_sensor.pressed(port.A)

async def main():
    #左（ポートE）と右（ポートF）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)
    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
   #力センサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)
   #停止
    motor_pair.stop(motor_pair.PAIR_1)
runloop.run(main())
```

このプログラムを実行してみると、コンソールには次のようなエラーが表示されます。



「force_sensor」が定義されていないエラーがあるとのこと。つまり、プログラムにとって「force_sensor」という単語が何かわからないということです。そこで、プログラムの上部、必要なライブラリをインポートしている部分を見ると、力センサーがインポートされていません。ここに「import force_sensor」の一文を加えると、このエラーはなくなり正常に動作することになります。

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

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カ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

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

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

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

Unit 1 Hardware and Software
ユニット 1 ハードウェアとソフトウェア

Unit Introduction: ユニット紹介

この入門ユニットでは、生徒は、さまざまな方法でコミュニケーションを取るために、重要なコンピューター サイエンスの原則と、テキストベースのコーディング言語 Python のプログラミング概念を学びます。 生徒は共同プログラミングに焦点を当てて、ハードウェアとソフトウェアの組み合わせについて探求します。

レッスンは、生徒が次の分野のスキルと知識を高めることのできる順序で設計されています。

- Python ライブラリのインポートとプログラムの実行
- ハードウェアとソフトウェアの機能の説明
- ペアプログラミングとプログラミングの役割を練習する
- 光と音をプログラムしてメッセージを伝えます
- 問題を分解し、デバッグ戦略を利用する
- 他の人に具体的なフィードバックを提供する
- フィードバックを活用してプロジェクトを改善する

ユニット学習の期待

このユニットでは、生徒は光と音を使ってアイデアを伝える方法を考えながら、ハードウェアとソフトウェアがどのように連携するかを探ります。さらに、生徒は共同プログラミングに取り組み、仲間と協力する方法や、プロジェクト全体の品質を向上させるためにフィードバックをやり取りする方法にも焦点を当てます。

探求への質問：

アイデアを他の人に伝える方法は何ですか？

ハードウェアとソフトウェアはどのように連携して問題を解決できるのでしょうか？

ユニット 1 レッスン 1
ライブラリのインポート

ライブラリのインポート

生徒は、Python でライブラリをインポートする方法と、ソフトウェアとハードウェアを接続する際にこれが重要である理由を学びます。

探求のための質問

エンジニアとコンピュータープログラマーはどのように協力してアイデアを他の人に伝える方法を作成できるでしょうか？

必要な機材

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

準備する

- 生徒をグループにします (2 人の生徒で 1 つの SPIKE Prime セットを使用します)
- SPIKE Prime ハブが充電されていることを確認してください (特に Bluetooth 経由で接続している場合)

主な目的

生徒は次のことを行います：

- Python プログラムにライブラリをインポートする必要がある理由を学びます
- ライブラリをインポートします
- プログラムを実行します

ボキャブラリ

ライブラリ
インポート

1. 引きつける

生徒とのディスカッションを活性化します。 レシピに従って料理をするときに何が必要かを考えてください。 必要な材料を購入するために店に行きます。 家に帰ったら、食事の準備に必要な道具が揃っているかどうかを確認する必要があります。 何が必要かを考えてみましょう。 コンロやオーブンが必要になる場合もありますし、ミキサーやヘラが必要になる場合もあります。
Python プログラムを作成するときは、モデルの実行を開始する前に、使用するものをすべて「 集める」必要があります。

2. 探究する

生徒はライブラリとは何か、ライブラリをインポートする方法と理由について学びます。

Python を使用してプログラミングする場合、生徒は、プログラムが使用できるハードウェアを認識するために、ライブラリ、単語、または用語をインポートする必要があります。 Python はテキストベースのコーディング言語であるため、大文字と句読点が重要です。SPIKE アプリが SPIKE Prime コンポーネントと通信するには、ハードウェアをソフトウェアにリンクする適切なライブラリが必要です。

生徒に、SPIKE Prime セットを開いてすべてのハードウェアの場所を確認するように促します。 各ハードウェアをセットから出し、テーブルの上に置きます (ハブ 1 つ、モーター 3 つ、センサー 3 つ)。

生徒に各ハードウェアを識別するよう指示します。
各ハードウェア項目を確認します。 各ハードウェアの部品を持ち上げて、同じ部品を見つけるように生徒に指示します。 中央の大きなボタンを押してハブをオンにします。

各ハードウェアをソフトウェアにどのようにインポートすると思うかを生徒に尋ねます。

SPIKE App Python キャンバスで入門プログラムを表示します。

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

ライブラリと呼ばれるハードウェア部品をインポートするためのラインを生徒に紹介します。

- インポート コードの項目に一致するハードウェアを特定するよう生徒に指示します。

    ＊ import _____

    下線部にはハードウェアの部品名が入ります

注記

- light_matrix はハブ上の 5x5 グリッドです
- button とは、ハブの前面の下部にある 3 つのボタン (左、中央、右のボタン) を指します
- light はセンターボタンを囲むライトの色です
- Force_sensor は、黒いボタンを押し込むカセンサーを指します
- motion_sensor はジャイロセンサーのことでハブ内臓されています
- speaker (スピーカー) はハブの外側にあります
- color_sensor は 1 つのライトを持つ小さな正方形のカラーセンサーです
- App はサウンドを再生する機能です
- distane_sensor は、2 つの「 目 」があるように見える長方形の超音波センサーです
- motor とは全てのサイズのモーターを指します
- motor_pair は、連携して動作する 2 つのモーターを指します

生徒にすべての部品をケースに戻し、適切な場所に正しく配置してもらいます。

3. 説明する

生徒に次のような質問をします。

- コードを書き始める前にライブラリをインポートすることが重要だと思うのはなぜですか？
- ハブの機能を使用するために何をインポートできますか？どのようにインポートされるのでしょうか？
- モーターはどのようにインポートされますか？

4. もっと詳しく探究する

生徒はライブラリのインポートを実際に行い、さまざまなタイプのライブラリのインポートがどのようなものかを確認します。
生徒に、SPIKE アプリで新しいプロジェクトを開き、プログラミングの種類として Python を選択するように指示します。 学生は自分のハブに接続する必要があります。
学生には、キャンバスにすでに書き込まれたプログラムが表示されます。

```
1 from hub import light_matrix
2 import runloop
3
4 async def main():
5     # write your code here
6     await light_matrix.write("Hi!")
7
8 runloop.run(main())
9
```

生徒に、コードで使用できるハードウェアの部分を注意深く調べてもらいます。インポートされたライブラリについて生徒と話し合います。画面の下部にある再生ボタン（三角形の付いた黄色の円）を選択して、生徒がサンプル コードを実行できるようにします。 ライトマトリックスに「Hi!」とスクロールするはずです。

トラブルシューティングのヒント：

ハブが正しく接続されていることを確認してください。 左上のプログラミングキャンバスにはハブのアイコンがあります。ハブアイコンの周りに緑色のリングがあれば、USB ケーブルで接続されています。 ハブ アイコンの周囲に青いリングがある場合、Blurtooth に接続されています。 ハブのアイコンが黄色の場合、ハブは切断されています。

インポートを開始する

ライブラリをインポートする方法の学習を開始するには、右側のパネルでナレッジ ベースを見つけるように生徒に指示します。
画面上のコードを見てください。 どのようなライブラリがインポートされましたか？ 生徒にコードの 1 行目と 2 行目を参照してもらいます。この行は何をしているのでしょうか？

生徒にコードの 6 行目を参照してもらいます。 プログラム中に何が使用されますか？

- このプログラムを実行すると何が起きるでしょうか？ パートナーと話し合ってプログラムを開始してください。

ナレッジベースの「はじめに」セクションを学生に参照してもらいます。「1. Python の概要」を読み、ライブラリのインポートに関する情報を生徒と一緒に確認します。

- 「インポートされたライブラリは .py ファイルの先頭にあり、プログラム内で 1 回だけ出現する必要があります。」という記述が何を意味するのかを尋ねます。
- 学生は .py ファイルが何であるかを知らない可能性があります。.py ファイルは、コーディングされている Python プログラムです。「.py」は Python の略です。
- .py ファイルにはキャンバス上のすべてが含まれているため、キャンバス上のどこにも .py が表示されません。
- ファイル名またはプロジェクト名は画面左上の家のアイコンの横に表示されます。 これが最初に作成されたプロジェクトの場合、プロジェクトの名前はおそらく「Project 1」になります。

生徒に、Python を使用して新しいプロジェクトを開くように指示します。 生徒にさまざまなフレーズを試してもらい、プログラムがどのように変化するかを確認します。

5. 評価する

教師の観察：

生徒に次のような質問をします。

- プログラムを実行したときにハブで何が起こりましたか？
- このプログラムを実行するためにどのライブラリを使用しましたか？
- ハブのさまざまな部品をインポートする必要があるのはなぜですか？
- エンジニアとコンピュータープログラマーはどのように協力してアイデアを他の人に伝える方法を作成できるでしょうか？

生徒たちに、教材管理の責任があることを思い出させます。 パーツをセット間で共有しないでください。 足りない部分があれば先生に聞いてください。 先生のスペアパーツには限りがあることに注意してください。何か見つからない場合は、すぐに先生に知らせてください。

生徒にノートに次のことを答えてもらいます。

- Python プログラムの開始時にライブラリをインポートする必要があるのはなぜですか？

ユニット 1 レッスン 2
光でコミュニケーション

光でコミュニケーション

生徒はライトマトリックスを制御して画像を表示したり、単語を書いたりする方法を学びます。

探求のための質問

- テクノロジーは問題解決にどのように役立ちますか？
- 問題の解決策としてハードウェアの使用をどのように計画できますか？

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
- ハサミ
- 色紙

準備する

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

主な目的

生徒は次のことを行います：

＊ ハードウェアとソフトウェアの機能を説明します。

＊ 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。

ボキャブラリー

デバッグ

ライトマトリックス

1. 引きつける

5x5 グリッドではどのような形状を作成できますか？ 長方形を作ってもらえますか？ 三角形？ あなたはどんな顔を作ることができますか？

コードを書く前になぜ紙で作成するのでしょうか？

生徒はハブ上のライトのプログラミングを探求します。

- ノートに 5x5 のマトリックスを描かせます。
- 生徒に、マトリックスの中に収まるように 25 枚の小さな正方形の色紙を切るように指示します。
- 簡単な画像を描くか、ハブ上のライトマトリックスに作成できる画像を生徒に見せます。 例としては、四角形やハートなどが挙げられます。
- ハブ上に画像を作成する方法を生徒に考えてもらいます。どの正方形が点灯するかを表すために、マトリックスに正方形を配置して画像または絵を作成するように指示します。 25 枚すべての正方形の紙を同時に使用する必要はありません。

2. 探究する

学生はライト マトリックス上に画像を作成する方法を学びます。

このタスクをコンピュータで実行できるステップに分割する方法を生徒に考えてもらいます。 生徒に手順を疑似コードでノートに書くように指示します。 必要な手順を考えるために、これまでのコーディング経験を思い出してもらうよう促します。 疑似コードは、何が起こるべきかについての段階的な指示を示す、言葉で書かれた一連の指示です。疑似コードはコードの作成を支援するために使用されます。

ヒント： どのライブラリをインポートする必要がありますか？

生徒が自分のアイデアを共有できるようにします。
Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。 すでにプログラミング領域にあるコードを消去するように生徒に指示します。 ハブを接続します。

以下のサンプルコードを生徒に提示します。 生徒にコードをプログラミング キャンバスに入力して実行してもらいます。

```
#ライトマトリックスのインポート
from hub import light_matrix
import runloop

async def main():
    #ライトマトリクスにハッピーフェイスを表示
    light_matrix.show_image(light_matrix.IMAGE_HAPPY)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()

runloop.run(main())
```

何が起こるか生徒たちと話し合います。

プログラムがまだハブで実行されているかどうかを生徒に尋ねます。 笑顔がまだハブに表示されている場合は、プログラムはまだ実行中です。 画面の右下にある白い四角の付いた赤い円をクリックするか、ハブ上の大きな円のボタンを押して、プログラムを停止するように生徒に指示します。

生徒と一緒にコード行を確認します。 プログラムを実行すると何が起こっているのか、コードの各行がハブに何を指示しているのかについて話し合います。

新しいイメージを作成する

生徒にライト マトリックスに表示される画像を変更してもらいます。
生徒たちに、表示を「 HAPPY」から「 HEART」に変更するように指示します。これにより、ライト マトリックスの表示が変わります。
生徒に、笑顔の代わりにハートを表示するには、コードのどこを変更する必要があるかを示すように指示します。 生徒たちと話し合う。

生徒にコードと緑色のコメントを変更してもらいます。 プログラムを実行します。

```
    #ライトマトリクスにハートを表示
    light_matrix.show_image(light_matrix.IMAGE_HEART)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()
```

プログラムについて生徒と話し合ってください。 プログラム内で「 #」が何を示すのかを生徒に尋ねます。「 #」は、コンピュータに対するコマンドではなく、コードのセクションが何を行うべきかをプログラマーが覚えておくのに役立つコメントであることを示します。

生徒たちと一緒に、ハートの画像がどれくらいの時間点灯し続けるかを確認します。 await runloop.sleep_ms(5000) コマンドの直後に括弧内に「 5000」が表示されているため、画像は 5 秒間点灯しました。 括弧内の値はミリ秒数です。

3. 説明する

学生たちとプログラムについて話し合います。 生徒に次のような質問をします。

- これらのプログラムを実行するにはどのライブラリをインポートする必要がありますか？
- コードの runloop.sleep_ms 部分の目的は何ですか？
- 表示される画像を変更するには、毎回コード全体を書き直す必要がありますか？
- プログラムがまだ実行中かどうかをどのように認識しますか？ プログラムを停止するにはどのような方法がありますか？

4. もっと詳しく探究する

ハブのプログラミングワードを探求します。

2 つの画像を表示する

生徒に、ライトマトリックスに表示する 2 つの画像を作成させます。
#Light Up a heart の下のコード行をコピーし、コードの最後に貼り付けるように生徒に指示します。 生徒は 2 つの画像をコーディングすることになります。

2 番目のセクションに HAPPY を挿入してもらいます。 コードの両方のセクションでコードを実行すると何が起こるか生徒たちと話し合います。 グループとしてコード行を確認します。 コードでは、ハートが 5 秒間表示され、その後スマイリーフェイスが 5 秒間表示される必要があります。

```
#ライトマトリックスのインポート
from hub import light_matrix
import runloop

async def main():
    #ライトマトリクスにハートを表示
    light_matrix.show_image(light_matrix.IMAGE_HEART)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()

    #ライトマトリクスにハッピーフェイスを表示
    light_matrix.show_image(light_matrix.IMAGE_HAPPY)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()

runloop.run(main())
```

学生たちとプログラムについて話し合います。

デバッグ

エラーを読み取る方法を調査します。 生徒に、プログラミング キャンバスの下部中央にある 2 本の水平線をクリックするように指示します。 これにより、コンソールが開くようになります。 コンソールには、情報やエラー メッセージが出力されます。

生徒にプログラムを再度実行してもらいます。

```
#ライトマトリックスのインポート
from hub import light_matrix
import runloop

async def main():
    #ライトマトリクスにハートを表示
    light_matrix.show_image(light_matrix.IMAGE_Heart)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()

    #ライトマトリクスにハッピーフェイスを表示
    light_matrix.show_image(light_matrix.IMAGE_HAPPY)

    #5秒間待機
    await run.sleep_ms(5000)

    #ライトマトリックスをクリア
    light_matrix.clear()


runloop.run(main())
```

生徒たちに何が起こったのか説明してもらいます。 情報が正しく入力されていない場合に発生する可能性のあるエラーを示します。

コンソールのエラー メッセージが 6 行目を示していることに注意してください。6 行目を見ると、Heart という単語がすべて大文字ではないことがわかります。 Heart を HEART に変更して、プログラムを再度実行します。プログラムは正しく動作しますが、コンソールのエラー メッセージは変化しないことに注意してください。 さらにエラーを犯すと、コンソール内の次のメッセージが前のメッセージのすぐ下に表示されます。

追加の探究

生徒がハブに画像を表示するためのすべてのオプションを探索できるようにします。 学生がハブ ライト マトリックスに表示できる追加の画像にアクセスするには、ナレッジ ベースの Light Matrix.() を参照してください。 時間の許す限り、生徒に画像の変更を検討してもらう時間を与えてください。

生徒に、これまで取り組んできたコードを見て、このコードを変更して、単に画像を表示するのではなくハブに単語を書き出す方法を考えてもらいます。

画像を表示する代わりにテキストを書き込むには、ライト マトリックスをコーディングする必要があります。 生徒は最初、単語をイメージとして示す必要があると考えるかもしれません。 単語を絵のように見せるのではなく、単語を書き出すことについて生徒に考えてもらいます。

生徒たちに、ライト マトリックスに「 Hello」という言葉を書いてもらいます。

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

生徒たちに、画像を表示するために使用された以前のコードとこのコードの違いについて考えてもらいます。 このコードにより、生徒はライト マトリックス上に一度に 1 文字ずつ、右から左にスクロールしながらテキストを書くことができます。 これは文字列、または文字の配列です。 たとえば、「 Hello」という単語は、「 H – l – l – o」という文字列で構成されます。この文字列には 5 つの文字が含まれています。

このコードを使用して、生徒にハブに自分の名前を書くよう促します。 生徒がコード内で変更する必要がある唯一のものは何ですか？

オプション： 生徒がさまざまな単語を書くことを検討し、単語を書いてから画像を見せるように挑戦させます。

追加の参照として、Light Matrix Action write() の下にあるナレッジ ベースを生徒に示します。

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- await runloop.sleep_ms の目的は何ですか？
- プログラムから await runloop.sleep_ms を省略した場合はどうなりますか？
- どのようなトラブルシューティングまたはデバッグが発生しましたか？
- コード内の「#」記号で始まる緑色の単語の機能は何ですか？
- 「show_image」を使用したコーディングと「write」を使用したコーディングの違いは何ですか？

生徒にノートに次のことを答えてもらいます。

- 今日、ライト マトリックスのプログラミングについて何を学びましたか？

ユニット 1 レッスン 3
ペアプログラミング

ペアプログラミング

生徒は、一緒にプログラミングするときに役割を持つことがなぜ重要なのかを学びます。

探求のための質問

* 最終コードを作成するときにプログラマが共同作業することが多いのはなぜですか？

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ているデバイス
- ノート
- 付箋

準備する

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

主な目的

学生は次のことを行います：

- ペアプログラミングの練習
- プログラムを修正する

語彙

ドライバー
ナビゲーター
ペアプログラミング

1. 引きつける

生徒とのディスカッションを活性化します。 生徒たちに、4 人で車に乗っているところを想像してもらいます。 車に乗っている各人の役割が何になるかを話し合います。 生徒たちは、後部座席に座っている人が果たすいくつかの楽しい役割を特定するかもしれませんが、できれば、前部座席に座っている少なくとも 1 人が運転手であることを特定することができます。 生徒が誰かがナビゲートする必要があることを認識していない場合は（ 地図アプリに従っているだけであっても）、その役割を特定するための質問を促します。
ドライバーとナビゲーターの役割に注目してください。 ドライバーとナビゲーターにはどのような役割があるのかを生徒に尋ねます。

2. 探究する

学生と一緒にプログラミングに取り組む際のドライバーとナビゲーターの役割を学びます。

ペア プログラミングは、同じプロジェクトに取り組む 2 人以上の作業であることについて話し合います。 ペアプログラミングでは、各パートナーが果たすべき役割を持ちます。 これらの役割をグループで一緒に確認します。
*「 ドライバー」は、必要なプログラミングを考え、コンピューターを使用してコードを入力します。
*「 ナビゲーター」はドライバーの指示に役立ち、すべてのコードが正しく記述されていること、つまりポートが正しいこと、演算子が正確であることなどを確認します。

チームの各メンバーに役割を割り当てて、協力して作業する練習をしてもらいます。
生徒にハブ上にイメージを作成するように指示します。 学生はナレッジベースのハブ – ライトマトリックスを表示して、プリロードされたすべての画像を確認できます。

- パートナー 1 は「 ドライバー」として働きます。
- パートナー 2 は「 ナビゲーター」となります。 お気に入りの画像を選択してください。
- パートナー 1 は、パートナー 2 が選択した画像を表示するプログラムを入力する必要があります。
- パートナー 2 は、プログラムのテスト前に、入力されたプログラミングを検証します。

生徒にハブを持たせて、クラスで画像を共有してもらいます。
生徒に役割を交代して同じアクティビティをしてもらいます。

各生徒に付箋を 1 枚ずつ配ります。
各自が一番気に入った画像の名前を付箋に書いてもらいます。
生徒たちに、画像の人気を示す棒グラフを壁に作成してもらいます。
同じ画像の付箋は使用回数を表すために縦に配置されます。 縦の棒グラフが各画像名ごとに数列作られることになります。

3. 説明する

生徒と一緒に棒グラフを確認し、お気に入りの画像について話し合います。

生徒たちが自分の役割にどのように取り組んだかを話し合います。
生徒に次のような質問をします。

- プログラミングする際に各人が役割を持つことが重要だと思うのはなぜですか？
- ドライバーの役割について重要なことは何ですか？
- ナビゲーターの役割について重要なことは何ですか？
- パートナーはどれくらいの頻度で役割を交代すべきだと思いますか？

4. もっと詳しく探究する

生徒たちは協力してデバッグ ゲームをプレイします。

各パートナーが順番に「 ドライバー」と「 ナビゲーター」を務め、これらの役割を練習することを生徒に説明します。

- ドライバーは、単語または画像、あるいはその両方の組み合わせを挿入する新しいプログラムを作成します。
- ドライバーは意図的にミスをします。
- ナビゲーターの役割は、エラーを見つけて修正方法を説明することです。

生徒は複数のラウンドを完了して、それぞれが両方の役割を確実に練習できるようにする必要があります。

クラス全体で、生徒たちがそれぞれの役割でどのように成功することができたかについて話し合います。

5. 評価する

教師の観察

プログラムについて生徒たちと話し合います。
生徒に次のような質問をします。

- ライト マトリックスで使用できる画像のリストはどこで見つけましたか？
- チームとしてどのように協力しましたか？
- チームメイトを助けるために、それぞれの役割で何をしましたか？

生徒にノートに次のことを答えてもらいます。
* 今日学んだことで、パートナーと協力してプログラミングする際に役立つことは何ですか？
* チームメイトを助けるために、それぞれの役割で何をしましたか？

ユニット 1 レッスン 4

音でコミュニケーション

サウンドをコントロール

学生はさまざまなタイプのサウンドをプログラムする方法を学びます。

探究のための質問

音はアイデアを伝えるのにどのように役立ちますか？

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

主な目的

学生は次のことを行います：

- ハードウェアとソフトウェアの機能の説明
- プログラムはビープ音を鳴らし、簡単なプログラムのデバッグ方法を学習します。
- サウンドパターンを作成する

ボキャブラリ

デバッグ

通信する

準備する

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

1. 引きつける

お互いに通信する方法はたくさんあります。
生徒が頻繁にコミュニケーションをとる方法をいくつか共有できるようにします。

話したり、書いたり、テキストメッセージを送信したりせずにコミュニケーションをとるという課題に生徒を参加させます。

2. 探究する

生徒はサウンドをプログラムする方法を学びます。

チームの各パートナーに、異なる色の 3 つのブロックの一致するセットをケースから出させます。
ペアの各生徒が同じ 3 色のブロックを持っていることを確認します。
パートナー A は 3 つのブロックを好きな順番で積み上げていきます。 たとえば、生徒のブロックが緑、赤、青の場合、一番下が緑、中央が赤、一番上が青というように。
パートナー A は、会話せずにブロックの積み方をパートナー B に伝える必要があります。一番簡単な方法は、パートナー A が組み立てたブロックを見せることです。

次に、交代してパートナーBがブロックを積み重ねててパートナーAに伝えるように生徒に指示します。ただし、今回は学生同士でスタックを見せ合うことはできません。

- 音を使ってお互いにコミュニケーションをとる方法を考え出すようチームに指示します。
- 例えば、緑色のレンガは拍手で、赤色のレンガはスナップで、青色のレンガは足を踏み鳴らして表されるなどと言って、アイデアを実証します。
- ブロックを積み上げながらそれぞれの音をだします。
- 生徒は各ブロックに対して独自の音を割り当て、協力してブロックを積み上げるように努めてください。

音を、会話以外のコミュニケーションにどのように使用できるかを話し合います。 例としては、誰かの注意を引くために口笛を吹いたりクラクションを鳴らしたり、承認を示すために手を叩いたり、何かに対して非常に怖いことを意味する叫び声などが挙げられます。

音を出すことはコミュニケーションをとるための素晴らしい方法です。

ピー、ピー

ハブからビープ音が出ていることを調べます。

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。 すでにプログラミング領域にあるコードを消去するように指示します。 生徒はハブを接続します。
このサンプル プログラムを使用して、ハブを使用して音がどのように生成されるかを生徒に調べてもらいます。
以下のサンプルコードを生徒に提示します。 生徒にコードをプログラミング キャンバスに入力してもらいます。

```
#サウンドのインポート
from hub import sound
import runloop

async def main():
    #ビープ音を１秒間再生
    await sound.beep(400, 1000, 100

runloop.run(main())
```

生徒と一緒にコードを見て、コードのどのセクションがソフトウェアに何をすべきかを指示しているか（スピーカーのインポート）、コードのどの部分がハブに何をすべきかを指示しているか（ビープ音）、そしてどの部分がプログラマへの単なるメモであるかを確認します (# 緑色の部分)。

生徒に、別の音を出すために変更できる領域を特定してもらいます。
生徒たちに、別の数字を試すよう促します。

生徒が数字を変更して作成した音を共有できるようにします。
生徒に、コード内で 400、1000、100 という数字が何を表しているかを特定してもらいます。
数字 400 は周波数を表し、数字 1000 は時間の長さをミリ秒単位で表し、数字 100 は音量を表すことを説明します。

デバッグ

生徒たちに、400（頻度）という数字を 100 より小さい数字に置き換えてもらいます。何が起こるか話し合います。
注：学生はすでに自分でこれを発見している可能性があります。 関連性が生じたら話し合ってください。

```
#サウンドのインポート
from hub import sound
import runloop

async def main():
    #ビープ音を１秒間再生
    await sound.beep(100, 1000, 100)

runloop.run(main())
```

生徒は、エラーが表示されていないにもかかわらず、ビープ音が聞こえないことに遭遇するでしょう。
生徒たちに何が起こったのか説明してもらいます。

曲を再生する

生徒たちに、調査で学んだことを活かしてコード行を追加して曲を作成するように指示します。 生徒は、さまざまなサウンドを含むコードを数行追加して曲を作成する必要があります。

生徒に、ナレッジベースの「 スピーカー」セクションでサウンドオプションを参照し、beep() を選択するように指導します。 このガイドでは、さまざまなサウンドを生成するためにコードで使用する数値の範囲と、エラーに関するガイダンスを提供します。
時間が許す限り、生徒が探究できるようにします。

ここでは、苦戦する可能性のある生徒向けのサンプルソングを紹介します。

```
#サウンドのインポート
from hub import sound
import runloop

async def main():
    #ビープ音で曲を奏でる
    await sound.beep(400, 1000, 100)
    await sound.beep(450, 500, 100)
    await sound.beep(500, 1000, 100)

runloop.run(main())
```

3. 説明する

生徒たちに最終的なプログラムを互いに共有し、コードが何を示しているかを説明してもらいます。

次のような質問をしてください。

- どのライブラリをインポートする必要がありますか ?
- コードにどのように行を追加しましたか ?
- プログラム内のコードの各行は何を意味しますか ?
- このタスクの何が難しかったですか ?
- プログラミングのどこでエラーが発生しましたか ? どのように修正またはデバッグしましたか ?

ビープ音コードに含まれる数値は整数または小数を含む可能性があるため、float 型であることを生徒に説明します。 たとえば、より細かく秒を 1.5 秒 などに設定できます。

4. もっと詳しく探究する

音 (サウンド) を鳴らす

事前に録音されたサウンドの再生方法を調べます。

ビープ音の再生に加えて、猫の鳴き声や犬の鳴き声などの事前設定された音を再生するようにプログラムすることもできることを生徒に説明します。 生徒に、これまで取り組んできたコードを見て、このコードを変更して単なるビープ音ではなくプリセットのサウンドを再生する方法を考えてもらいます。

生徒がサウンド モジュールをインポートする必要があることを理解できるようにします。 生徒は、どのようなプリセットサウンドが利用可能であるかを知る必要があります。生徒は、ワードブロックモードで音再生ブロックを使って利用可能なすべてのサウンドを見つけることができます。 学生は後でこれらにアクセスできます。

以下のサンプル プログラムを使用して、どのように音が鳴るかを調べるためのコードを作成するよう生徒に指示します。 このコードをプログラミングキャンバスに既存のプロジェクトを変更して入力するよう生徒に促します。

```
#サウンドファイルのインポート
from app import sound
import runloop

async def main():
    #アプリに用意されたサウンドファイルを再生する
    await sound.play('Cat Meow 1')

runloop.run(main())
```

トラブルシューティングのヒント：

生徒が課題をプログラムするために適切なライブラリをインポートしていることを確認します。 追加のデバッグ情報については、ナレッジベースを参照してください。

注記：

サウンドはハブではなくデバイスから発信されるため、デバイスのスピーカーがオンになっていることを確認してください。

このコードがビープ音を鳴らすコードと異なる点は何なのかを生徒たちと話し合います。 生徒は、音が数字ではなく名前に基づいていることを理解する必要があります。

生徒は、音がハブからではなく自分のデバイス（アプリ）から出ていることを認識する必要があります。 生徒たちにその理由を考えてもらいます。 スピーカーをインポートすると、ビープ音が聞こえるようになります。 サウンドをインポートすると、デバイスを通じてサウンドが聞こえるようになります。

「Cat Meow」（猫の鳴き声）という言葉の後に「1」がある理由を生徒に尋ねます。 この場合、1は、他の類似した音である Cat Meow 2 と Cat Meow 3 を区別するためのものです。数値は、その変数の文字列または名前の一部です。 生徒に文字列とは何かを覚えているかどうか尋ねます。

コードを「Cat Meow 1」から「Triumph」または「Doorbell1」に変更して、生徒が他の音を試すことができるようにします。 利用可能な音を見つけるためにワードブロックのレッスンを参照するように生徒に促します。

パターンを作成するために提供されているオプションのライブラリからさまざまなサウンドを試すように生徒に挑戦させます。 生徒が自分の音を他のグループと共有できるようにします。

音を使ってさまざまな方法でコミュニケーションを図る方法について生徒と話し合います。

5. 評価する

教師の観察：

学生たちとプログラムについて話し合います。
生徒に次のような質問をします。

- ビープ音のプログラミング音とサウンドのプログラミングとどのように異なりますか？
- ビープ音が鳴るとき、音はどこで聞こえますか？
- サウンドが再生されると、どこで音が聞こえますか？

生徒にノートに次のことを答えてもらいます。

- ビープ音やサウンドを使用して通信するにはどうすればよいですか？

ユニット 1 レッスン 5
デジタルサイン

デジタルサイン

ライト マトリックスの使用とサウンドのプログラミングに関する知識を応用して、広告用のデジタル サインを設計、構築、プログラムします。

探究のための質問

アイデアを伝えるために光と音をどのように組み合わせ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
- 付箋
- ハサミ
- 色紙

準備する

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

主な目的

生徒は次のことを行います：

- ハードウェアとソフトウェアの機能の説明
- 光と音でメッセージを伝えるプログラムです。

ボキャブラリー

宣伝

1. 引きつける

ケイトとカイルはポップコーンスタンドを作りましたが、誰もポップコーンを買ってくれないようです。ケイトは人々の注目を集める看板を作るというアイデアを持っています。 まず、紙の看板を作ることを考えます。 次に、フェアでいつも見るクールな点滅する看板について考えます。 あなたのチームは、ケイトとカイルがポップコーン スタンド用のデジタル サインを作成するのを手伝うことができますか？

2. 探究する

生徒は、ライトマトリックスを使用した看板を設計および組み立て、ポップコーンやその他の商品を宣伝するメッセージを表示するようにプログラムします。 どの商品を販売し、宣伝する必要があるかを生徒に選択させることを検討してください。

生徒たちに、ハブのライトマトリックスを使用してデジタルサインを作成するさまざまな方法を考えるように促します。 デジタルサインの画像またはビデオを表示します。 生徒に、これまでに見たシジタルサインの例を挙げてもらいます。

生徒は、ライトマトリックスに表示するメッセージを書き出すなど、サインのデザインをスケッチする必要があります。 生徒は、自分たちのメッセージを表示できるようにハブを保持できるように、看板のデザインを検討する必要があります。ハブだけを置いて看板とすることはできません、ハブを持ち上げ固定できるようにデザインするよう説明します。

ヒント：

ハブの向きを考慮してください。 生徒は、看板に言葉や画像を含めるかどうかを検討する必要があります。

メッセージが明確であることを確認するためにプログラムを数回テストするよう生徒に伝えます。 生徒は、広告が明確であること（人々がメッセージを理解すること）を確保しながら、メッセージの長さ（読みやすさ）のデザインにおけるトレードオフを考慮する必要があります。

このチャレンジの必要条件：

- ライトマトリックス（ ハブ）は自立しません。 生徒はそのためのフレームを設計する必要があります。
- デジタルサインは、ポップコーンまたは選択した商品を宣伝するものでなければなりません。
- コードを書く前に、生徒はスケッチまたはデザインを見せて、コードに書きたい内容の説明を書かなければなりません。
- 生徒は、Python コードのコメント機能を使用して、コード行の意図を説明する必要があります。

注記：

生徒がデジタルサインを作成するためのライトマトリックスのプログラミングに苦労している場合は、ナレッジ ベースを使用するように思い出させます。 Hub の下に情報があり、Light Matrix を選択して show_image を探して、アイデアのインスピレーションやプログラミングのサポートを得ることができます。

3. 説明する

生徒たちは自分のサインを共有し、それがどのように機能するかを説明する必要があります。
生徒たちに次のように尋ねます。

- デジタルサイン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説明してもらいます。
- サインを作成する際にどのような決定を下さなければなりませんでしたか？
- デバッグまたはトラブルシューティングが必要な領域は何でしたか？
- この課題で難しかったことは何ですか？

4. もっと詳しく探究する

生徒は、サインに音を追加することで、ポップコーン スタンドや自分の選んだアイテムに注意を引くこともできます。ハブでライトが使用されている間のみ再生されるビープ音を使用して、サインにメロディーを追加するように生徒に指示します（つまり、メッセージがスクロールしている間ビープ音がなるようにします）。

光と音を使った最終デザインをクラスで共有できるようにします。

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- 電子署名プログラム用にどのライブラリがインポートされましたか？
- ビープ音はどこで聞こえましたか？

生徒にノートに次のことを答えてもらいます。

- ビープ音やサウンドを使用して通信するにはどうすればよいですか？

Unit 2 Motors
ユニット 2 モーター制御

Unit Introduction: ユニット紹介

このユニットでは、生徒は自律的に動くモデルを作成しながら、重要なコンピューター サイエンスの原理とテキストベースのコーディング言語 Python のプログラミング概念を探究します。 生徒はロボットがどのように電動化され、動作の自動化に役立つかを調査します。

レッスンは、生徒が次の分野のスキルと知識を高めることのできる順序で設計されています。

- 時間、速度、角度の変数を使用して、単一および複数のモーターの実行をプログラムします。
- 最短経路を使用して特定の位置に移動するようにモーターをプログラムします。
- 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。
- 既存のコードを使用したり変更して、新しいアイデアを検討します。
- アルゴリズムの作成をサポートする疑似コードを作成します。
- コメント機能を利用して、プログラムの一部を文書化します。
- 問題を定義して分解します。

ユニット学習の期待

この単元では、学生はさまざまな方法でモーターを制御する方法、ロボットでモーターを効果的に利用する方法、タスクを自動化するためにモーターを利用する方法を学びます。 学生は擬似コードを利用してアルゴリズムの作成をサポートし、コード内コメントを使用してプログラムを文書化します。

探究への質問

ロボットはどのように動くのでしょうか？ 自動化するとどのようにしてタスクが簡単になるのでしょうか？ プログラムを文書化することがなぜ重要ですか？

ユニット 2 レッスン 1
モーターで動かす

モーターで動かす

生徒たちは個々のモーターを使ってダンス パーティーを作ります。

探究のための質問

エンジニアとプログラマーが協力して何かを動かすにはどうすればよいでしょうか？

必要な機材

- SPIKE プライムは学生向けセットです。
- SPIKE アプリがインストールされたデバイス
- ノート

準備する

特に Bluetooth 経由で接続されている場合は、SPIKE Prime ハブが充電されていることを確認してください。

主な目的

生徒は次のことを行います：

- 時間と速度のパラメーターを使用してモーターを個別に実行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する

ボキャブラリー

疑似コード
初期化する
モーター
スピード

1. 引きつける

生徒たちにモーターがどのように動くかについて考えてもらいます。 生徒たちにモーターとして立ってもらい横一列に列を作り、5 歩前に歩いてもらいます。 これを数回試してもらいます。 なぜ全員が同じ距離を移動しないのかを生徒に尋ねます。
「 上げる」と言ったら、生徒にしっかりと腕を上げるよう指示します。 数秒待ってから「 上げる」と言います。 なぜ全員が同時に動かなかったのでしょうか？ これを数回繰り返してもらいます。 なぜ全員が同じように動かないのかを生徒に尋ねます。
生徒たちに「 拍手」と言って拍手をしてもらいます。 数秒待ってから「 拍手」と言ってください。 これを数回試してもらいます。 なぜ全員が同時に拍手しないのか生徒に尋ねます。
同じコードでプログラムされたロボットのグループが同じコマンドを同時に与えられた場合にどのように反応するかについて話し合います。
ロボットがどのように動くのか、そしてなぜ与えられたコマンドの動きが毎回同じになるのかについて話し合います。 これがいつ役立つかを生徒に尋ねます。

2. 探索する

生徒はモーターの操作と動きをコード化する方法を学びます。
SPIKE アプリのナレッジベースの「 はじめに」セクションを生徒に案内します。 ここで生徒は「 4. モーターの制御」にアクセスできます。この入門レッスンでは、生徒に SPIKE Prime を使用した組み立てとコーディングの初期体験を提供します。

モーターのプログラミングをさらに詳しく調査するには、Python プログラミングキャンバスで新しいプロジェクトを開きます。 すでにプログラミング領域にあるコードを消去するように生徒に指示します。 学生はハブを接続します。
大型モーターをハブのポート A に差し込みます。

単一のモーターを回転させる

モーターを一定度回転させると、モーターがどのくらいの距離まで回転するかを設定できます。
モーターを動かすプログラムの方法について生徒たちとブレインストーミングします。 ハードウェアを正しく実行するためにソフトウェア

がどのような情報を必要とするかについて話し合います。
生徒と一緒にモーターを 2 秒間回転させる疑似コード プログラムを作成します。 疑似コードは、プログラムが何を行うかを言葉で記述します。
例としては次のようなものが考えられます。

1. モーターインポート
2. モーターをオン
3. 時計回りに 2 回転する

注記：

ステップで記述されたコードは、プログラムに組み込まれるものと正確に一致する必要はありません。 疑似コードを書くことは、生徒が何が必要かを判断するのに役立ちます。

モーターを動かすための以下のサンプルコードを生徒に提示します。 生徒にコードをプログラミングキャンバスに入力してもらいます。
(学生は、ナレッジベースの「 入門パート 4: モーターの制御」セクションからサンプル コードをコピーして貼り付けることができます。)

```
#モーター、ポートをインポート
import motor
from hub import port
import runloop

async def main():
    #ポートAに接続されたモーターを720度毎秒の速さで360度回転する
    await motor.run_for_degrees(port.A, 360, 720)

runloop.run(main())
```

トラブルシューティングのヒント：

生徒がモーターを A ポートに接続していることを確認するか、「 port.A」を正しいポートに変更してください。

コードの最後の行のかっこ内の 360 と 720 が何を表すかについて説明します。
これら 2 つの値に異なる数値を入力して、モーターの動作がどのように変化するかを調べるように生徒に指示します。
生徒が作成した新しい例を共有し、モーターをプログラムする方法について話し合うことができます。

複数のモーターを回転させる

ハブに中型モーターを追加するよう生徒に促します。 複数のモーターを接続すると、生徒は一度に複数のモーターを実行できるようになります。 生徒たちに、複数のモーターを必要とするどのタイプのロボットを作成できるかを考えてもらいます。 複数のモーターはどのような機能を果たすのでしょうか？

2 つのモーターを動作させるためにプログラムに何を追加する必要があるかについて、生徒たちとブレインストーミングを行います。 タスクは、2 つのモーター (1 つはポート A、2 つ目はポート B) を動作させることです。

モーターはどのポートにでも接続できることを生徒に思い出させてください。 ただし、プログラムは選択したポートと一致する必要があり、サンプルプログラムと一致しない可能性があります。

生徒に、以前に使用した疑似コードに追加するよう指示します。 モーターを同時に動作させるための文言を追加します。

学生に、両方のモーターを実行する新しいプログラムを作成するためのインスピレーションとして疑似コードを使用してもらいます。
(サンプルコード次項)

```python
#モーター、ポートをインポート
import motor
from hub import port
import runloop

async def main():
    #ポートAに接続されたモーターを720度毎秒の速さで360度回転する
    await motor.run_for_degrees(port.A, 360, 720)

    #ポートBに接続されたモーターを720度毎秒の速さで360度回転する
    await motor.run_for_degrees(port.B, 360, 720)

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たちと話し合います。 生徒たちはモーターが逆方向に回転していることに気づきましたか？ 生徒たちは、最初のモーターが作動し、次に 2 番目のモーターが作動したことに気づきましたか？ これは、待機可能オブジェクトに待機を追加したためです。

生徒に、各「 motor.run_for_degrees(port.X,360,720)」の前にある await を削除してコードを変更し、プログラムを再度実行するように指示します。 学生たちはモーターが同時に作動していることに気づきましたか？ これが待機可能の仕組みです。
待機可能アイテムの詳細については、ナレッジ ベースの「 はじめに」の「 4. モーターの制御」を参照してください。

生徒は度数と速度のどのような組み合わせを選択しましたか？

生徒に次のような質問をします。

- 1 つのモーターのみのライブラリのインポートと 2 つのモーターのライブラリのインポートの違いを説明します。
- モーターをさまざまな方法で動かすために、モーターのどのパラメータを変更できますか？
- 2 つのモーターはどのように動いたのでしょうか？ 同時に動きましたか? なぜ、あるいはなぜそうではないのでしょうか？
- モーターのプログラミング中に発生するエラーをトラブルシューティングするにはどうすればよいですか？

速度に指定された数値が整数型の入力であることを生徒に説明します。 整数型の入力には、-1000 ～ 1000 の範囲の正または負の整数を指定できます。

4. もっと詳しく探究する

ダンスパーティー

2 つの中型モーターを回転させてダンス パーティーを作成する短いプログラムを作成するよう生徒たちに挑戦させます。 音楽をかけて子供たちの創造力を高めましょう。 たとえば、両方のモーターを一方向に動かすことも、各モーターを反対方向に動かすこともできます。
生徒がモーターにブロックを追加して、何か新しいものを作成できるようにします。
ダンス パーティーを完成させるために、サウンドとハブ ライトをプログラムに追加するように生徒に促します。
プログラムをサポートするにはどのライブラリをインポートする必要があるか、また必要に応じてナレッジベースのヘルプをどこで見つけられるかを考えるよう伝えます。

学生は自分のダンスロボットを共有し、プログラムを説明する必要があります。
モーターの接続を外し、すべての部品を正しい保管場所に戻します。

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。

生徒に次のような質問をします。

- プログラムを実行したときにモーターはどうなりましたか？
- このプログラムを実行するためにどのライブラリを使用しましたか？
- 1つのモーターを動かすときと、2つのモーターを動かすときと何が違いますか？

生徒に日記で次のことを答えてもらいます。

- 複数のモーターがハブに接続されている場合、どのモーターを動かすようにプログラムしているかをどのように決定しますか？

ユニット 2 レッスン 2
モーターによる新たな動き

モーターによる新たな動き

生徒は、正確な位置と正確な角度で動くようにモーターをプログラムします。

探究のための質問

ロボットはどのようにして正確に動くのでしょうか?

必要な機材

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

準備する

* 特に Bluetooth 経由で接続している場合は、SPIKE Prime ハブが充電されていることを確認してください。
* ハブのポート A に M モーターを接続しておきます。

主な目的

生徒は次のことを行います:

- 最短経路を使用して所定の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くようにモーターをプログラムします。

ボキャブラリー

度
位置

1. 引きつける

SPIKE Prime セットのタイヤを使用して、タイヤが 1 回転したときにどれくらいの距離を走行できるかを測定するさまざまな方法を生徒に考えてもらいます。 生徒はこの距離を決定するためにいくつかの方法を考え出す必要があります。 生徒に車輪を円として考えるよう促します。 距離を決定するために円周を計算する方法を子供たちと話し合います。 生徒たちに、平面を使って車輪の円周を測るのが難しい場合はどうすれば測れるかを尋ねます。

生徒たちに、車輪の付いたロボットがあると想像してもらいます。 時間以外でモーターを動かすにはコードを書く必要があるのでしょうか? これはロボットをより正確に動かすのにどのように役立つでしょうか ? 1/2 回転 (0.5 回転)、2 回転 (2 回転) だけを動かす方法について生徒とアイデアを話し合います。

注記:

ホイールの直径は 5.6 cm (2.2 インチ) で、1 回転 (360 度) で 17.6 cm (6.9 インチ) 移動します。

2. 探究する

生徒は角度を利用して回転するプログラムの方法を探ります。

単一のモーターを角度で回転する

モーターを角度単位で動作させると、ロボットがどの程度回転するかを角度単位 (回転の 1/360) で決定できます。 モーターを動かすようにプログラムする方法を生徒たちとブレインストーミングします。

生徒は、ハードウェアを正しく実行するためにソフトウェアがどのような情報を必要とするかを考える必要があります。 生徒と一緒に、モーターを完全に 1 周回転させるための疑似コード プログラムを作成します。
例としては次のようなものが考えられます。

1. モーターインポート
2. モーターをオンにする
3. モーターを 360 度動かす

注記：

ステップで書かれたコードは、プログラム内の内容と正確に一致する必要はありません。 疑似コードは、プログラミングを直線的にコーディングできるように、生徒がどのような手順が必要かを考えるのに役立ちます。

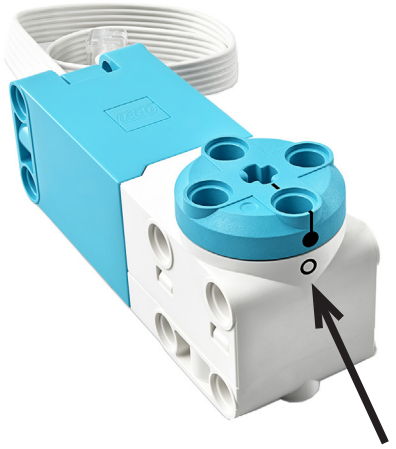
Python プログラミング キャンバスで新しいプロジェクトを開くように生徒に指示します。 すでにプログラミング領域にあるコードを消去するように生徒に指示します。 生徒はハブを接続し、モーターをハブの A ポートに差し込む必要があります。

モーターを動かすサンプルコードを生徒に提示します。 コードを書くように促します。（学生は、ナレッジベースの「入門 4: モーターの制御」セクションからこのコードをコピーしてプログラミング キャンバスに貼り付けることもできます。）学生にプログラムを実行するよう指示します。

```
#モーター、ポートをインポート
import motor
from hub import port
import runloop

async def main():
    #ポートAに接続されたモーターを720度毎秒の速さで-360度回転する
    await motor.run_for_degrees(port.A, -360, 720)

runloop.run(main())
```



モーターがどのように動いたかを生徒たちと話し合います。
位置を 0 度に設定できるモーター内のマーカーを生徒が識別できるようにします。 これにより、学生はモーターがどれだけ動くかを測定できるようになります。
生徒にマークを 0 の位置に設定するように指示します。プログラムを再度実行します。マーカーが同じ位置を返すかどうかを生徒に尋ねます。

生徒に、次数の値に別の数値を入力して、モーターがどのように動くかを確認するように促します。 大きな数字と小さな数字を試してもらいます。

生徒に、具体的に負の値を使用してモーターを動作させるようにしてください。
何が変わったのかを尋ねてください。（方向が逆です。）

```
#ポートAに接続されたモーターを720度毎秒の速さで-360度回転する
await motor.run_for_degrees(port.A, -360, 720)
```

生徒に 360 より大きい数値を入力してもらいます。どうなりますか？
生徒たちが調査後に発見したことについて話し合います。

単一のモーターを位置決めして実行する

生徒は、単一のモーターを特定の位置まで回転する方法を学びます。位置マーカーをもう一度見つけて、位置が揃っていることを確認してもらいます。 調整すると、モーターは 0 度の位置にあります。 モーターを特定の位置に移動するようにプログラムする方法を生徒と話し合います（モーターは正確な位置で停止します）。 生徒に、度の代わりに位置を使用するように疑似コードを変更するよう促します。

モーターを動かすためのサンプル コードを生徒に提示します。

```
#モーター、ポートをインポート
import motor
from hub import port
import runloop

async def main():
    #ポートAに接続されたモーターを720度毎秒の速さで0度の位置まで最短距離で移動する
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)

runloop.run(main())
```

プログラムについて生徒と話し合い、コードを実行すると何が起こるかを確認します。 生徒が 0 度の位置から開始すると、モーターは動きません。

生徒にモーターを動かして 0 度の位置から外すように指示して プログラムを実行します。 モーターは 0 度の位置に移動するはずです。 そこに着くまでにどの方向に進みましたか？

生徒たちに、毎回異なる位置でモーターを始動させてプログラムをさらに数回実行してもらいます。 何が起こるか話し合ってください。 生徒は、モーターが毎回同じように動いていないことを認識する必要があります。 モーターは 0 度の位置に戻る最短経路の方向に移動します。

コード行を一緒に確認して、モーターがこのように動作する理由を強調してください。

モーターが 0 位置に戻るまでに実行できる最長の移動はどれくらいですか？
生徒にコードを変更してモーターを別の位置で停止するように指示します。 このサンプル コードを彼らに提供して、コードをどのように変更するかについてアイデアを検討してもらうことができます。

```
#モーター、ポートをインポート
import motor
from hub import port
import runloop

async def main():
    #ポートAに接続されたモーターを720度毎秒の速さで0度の位置まで最短距離で移動する
    #その後、2秒間停止してから時計回りで90度の位置へ移動する
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
    await runloop.sleep_ms(2000)
    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)

runloop.run(main())
```

探究中に、下部コンソールにエラー メッセージが表示され、プログラムが実行されない可能性が高いことを生徒に思い出させます。

3. 説明する

「 度」と「 位置」について作成した新しいプログラム例を生徒に共有させ、モーターをプログラムする方法について話し合ってもらいます。 モーターを一定の角度で動かすように設定する方法と、これがプログラムでどのような場合に役立つかについて話し合います。

- このプログラムはどのように機能するのですか？
- 角度を使用して移動したり、位置を指定したりするようにモーターをプログラムすると、どのような場合に役立ちますか？
- 角度や位置を使用すると、「 秒」単位でプログラムするよりも正確に動作できるのはなぜですか？
- 360 以上の数値を入れた場合、モーターはどのように動きましたか?
- プログラムに必要な度数はどのように計算できますか？
- プログラムで位置に最短経路を使用する場合、0 度の位置から開始するとモーターが動かないのはなぜですか？
- モーターを逆方向に動かすようにプログラムするにはどうすればよいですか？

4. もっと詳しく探究する

協力してデバッグに取り組みます。 次の各コードを生徒に提示します。 各コードの何が問題なのかを話し合います。 生徒が各コードを実行し、コンソールに表示されるエラー メッセージを確認するようにしてください。

1. 何が欠けていますか？

```
1
2 import motor
3
4 import runloop
5
6 async def main():
7     #ポートAに接続されたモーターを720度毎秒の速さで0度の位置まで最短距離で移動する
8     #その後、2秒間停止してから時計回りで90度の位置へ移動する
9     await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
10    await runloop.sleep_ms(2000)
11    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
12
13 runloop.run(main())
```

生徒は、ポートがインポートされていないため、モーター ポートを設定できないことを認識する必要があります。 コンソールのエラーメッセージは次のとおりです

Trackback (most resent call last):
File“ Project 1”, line 8, in <mofule>
File“ Project 1”, line 6, in main
NameError:name ‘Port’ isn’t defined

生徒はインポートモーター行の後に「 from Hub import port」を追加する必要があります。 生徒にこれを追加してプログラムを再度実行して、プログラムを確認してもらいます。

2. どの数字が間違っているのか、またその理由は何ですか？

```
1 #モーター、ポートをインポート
2 import motor
3 from hub import port
4 import runloop
5
6 async def main():
7     #ポートAに接続されたモーターを720度毎秒の速さで0度の位置まで最短距離で移動する
8     #その後、2秒間停止してから時計回りで90度の位置へ移動する
9     await motor.run_to_absolute_position(port.A, 0, 7200)
10    await runloop.sleep_ms(2000)
11    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
12
13 runloop.run(main())
```

生徒は、モーター速度が 7200 に設定されており、これは許容範囲 -1000 ～ 1000 の範囲外であることを認識する必要があります。 ただし、プログラムは引き続き実行され、コンソールにエラー メッセージは表示されません。 パワーは 1000 に戻り、プログラムが実行されます。

3. 何が間違っているのでしょうか？

```
1  #モーター、ポートをインポート
2  import motor
3  from hub import port
4  import runloop
5
6  async def main():
7      #ポートAに接続されたモーターを720度毎秒の速さで0度の位置まで最短距離で移動する
8      #その後、2秒間停止してから時計回りで90度の位置へ移動する
9      await Motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
10
11
12
13 runloop.run(main())
```

生徒は、9 行目で Motor という単語が大文字になっ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、9 行目に問題があり、エラーが定義されていない値であることを示しています。

Trackback (most resent call last):
File “ Project 1”, line 13, in <module>
File “ Project 1”, line 9, in main
NameError:name ‘Motor’ isn’ t defined

5. 評価する

観察を教える：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- モーターが動くようにプログラムできるさまざまな方法には何がありますか？
- より正確な動きを可能にする方法はどれですか？
- 位置を使用して移動する方法を教えてください。

生徒にノートに次のことを答えてもらいます。

- 今日学んだことの中で、ロボットが正確に動くようにプログラミングするのに役立つかもしれないことは何ですか？

ユニット 2 レッスン 3

自動化アクション

アクションの自動化

モデルにモーターを追加して自動化します

探究のための質問

- 機構を電動化して自動化するにはどうすればよいですか？
- 自動化によりタスクが簡単になるにはどうすればよいですか？

主な目的

生徒は次のことを行います：

- タスクを自動化するモデルを構築してプログラムする

ボキャブラリー

自動化

必要な機材

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

準備する

特に Bluetooth 経由で接続されている場合は、SPIKE Prime ハブが充電されていることを確認してください。

1. 引きつける

生徒を自動化についてのディスカッションに参加させます。 生徒に自動化とは何かについてのアイデアを提供し、自動化を実現する日常品の例を提示するよう促します。 生徒が自動化がどのように行われるかを理解できるように、必要に応じて画像やビデオを検討します。

SPIKE Prime Extra Resources – アイデア、LEGO Way レッスンのビデオを生徒に見せます。 このレッスンは https://education.lego.com/en-us/lessons/prime-extra-resources/ideas-the-lego-way/#ignite-a-discussion にあります。このビデオを使用して、示されているモデルで何が自動化できるかについて議論を促します。 ビデオ内で 2 人がしている動きを自動化するためにモーターを追加するアイデアについて話し合います。

2. 探究する

キャラクターの動きを手動で動かすのではなく自動化するために、タワーアームにモーターを追加するよう生徒に課題を与えます。
学生はタワーアームに 1 つのモーターを追加するアイデアを設計します。生徒はモーターを追加するためのアイデアをスケッチし、タワーアームが動くようにプログラムする方法の疑似コードを書き出す必要があります。生徒は、動きを自動化するようにプログラムするために、タワーアームの設計を検討し、タワーアームがどのように動くかを理解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きます。生徒に、すでにプログラミング領域にあるコードを消去するように指示します。ハブを接続します。

生徒は腕の動きをプログラムします。 疑似コードを使用してプログラムを作成します。 プログラムが期待どおりに動作することを確認するために、プログラムを数回テストするよう生徒に伝えます。 # を使用してコード コメントを追加して、プログラムの手順を説明するように生徒に指示します。

3. 説明する

生徒は最終的なデザインとプログラムを共有し、プログラムがどのように動きを自動化するかを説明する必要があります。
生徒に次のような質問をします。

- 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説明してもらいます。
- モーターを追加する際にどのような決定を下す必要がありましたか？
- 動きをプログラミングする際にどのような決定を下す必要がありましたか？
- モーターを追加して動作を自動化すると、どのように作業が容易になりますか？
- デバッグまたはトラブルシューティングが必要なところは何ですか？
- この課題で難しかったことは何ですか？

4. もっと詳しく探究する

生徒を 2 つのグループに集めて、大きなグループを作ります。
グループ 1 は、モデルとプログラムをグループ 2 に発表します。 次に、グループ 2 はモデルとプログラムをグループ 1 に提示します。どちらのグループも、各プログラムで提供されるコード内コメントに注目します。グループは協力して、コード内コメントが各プログラムの概要を明確に示すようにします。
グループは協力して、自動化のためのユーザー指示を作成する必要があります。各グループは、コード内コメントを使用して、これらの指示をプログラムの先頭に追加します。

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- どのようにして機構を電動化することができましたか？
- モーターを追加するにはどのような決定を下す必要がありましたか？
- ユーザー指示を含むコード コメントを作成することが重要なのはなぜですか？

生徒に次のことをノートに答えてもらいます。

- 今日、動きに自動化を追加することについて何を学びましたか？
- 自動化が役立つのはどのような場合ですか？

ユニット 2 レッスン 4

ホッパーラン

ホッパーラン

車輪のない前進するロボットを組み立ててプログラムします。

調査すべき質問

ロボットのデザインはロボットの動き方をどのように決定するのでしょうか？

必要な材料

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

準備する

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

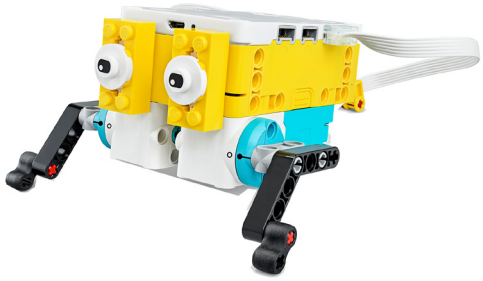
主な目的

生徒は次のことを行います：

- 2 つのモーターが同時に動くようにプログラムします。
- 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。

ボキャブラリー

モーターペア (motor_pair)



1. 引きつける

複数のモーターを使用する場合の移動方法について生徒に考えてもらいます。生徒を 4 人のグループに分けます。生徒の間に腕や定規 / 棒を持ってグループを作ります。 グループに一列に並んで一緒に動くように指示します。 何が起こるか考えてください。

2. 探究する

生徒はホッパーモデルを組み立てて、モーターを使用してさまざまな移動方法を調べます。
SPIKE アプリの「 組み立て図」セクションに案内します。 ここで生徒はホッパーの組み立て説明書にアクセスできます。 生徒たちにモデルを組み立ててもらいます。 組み立て説明書は次のサイトからも入手できます。
https://education.lego.com/ja-jp/product-resources/spike-prime/ ダウンロード / 組み立てガイド /

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。すでにプログラミング領域にあるコードを消去するように生徒に指示します。ハブを接続します。

動き出す

生徒たちに、ホッパーを 5 秒間前進させるように指示します。

同時に動くように取り付けられた 2 つのモーターを使用して、ホッパーがどのように動くかを生徒たちと話し合います。生徒に疑似コードを作成して、先に進むためにホッパーをどのようにプログラムする必要があるかを説明してもらいます。生徒が作成する疑似コードの例について話し合います。

疑似コードの例：

- モーターインポートする
- モーターを 5 秒間動かす

注：これは生徒が考えていることのサンプルであり、実際のコードを表すものではありません。

このサンプル プログラムを生徒と共有します。 生徒はこのプログラムをプログラミング キャンバスに入力する必要があります。

```
#モーター、ポートをインポート
from hub import port
import runloop
import motor_pair

async def main():
    #ポートE（左）とポートF（右）に接続されたモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    #初期設定値の速度で5秒間、前進する
    await motor_pair.move_for_time(motor_pair.PAIR_1, 5000, 0)

runloop.run(main())
```

注：コンソールのエラーに注意するよう生徒に伝えます。生徒はエラーの行を参照して、入力エラーが発生した可能性のある場所を特定できます。

ホッパーがこのコードでどのように動くかを確認します。 次に、ナレッジ ベースに移動し、「 モーター ペア」セクションを開きます。このセクションでは、2つのモーターを一緒にプログラムする方法について詳しく説明します。

move_for_time セクションを生徒と一緒に確認します。 時間、ステアリング、速度の変数を設定することに注意してください。すべて整数型入力です。時間には 0 より大きい任意の数値を指定できます。ステアリングの範囲は -100 ～ 100 です。速度の範囲は -1000 ～ 1000 です。

生徒たちに、ハブ (ライトマトリックス) に 3-2-1 のカウントダウンを表示するようにプログラムを作成して、50 cm の距離を移動するように指示します。プログラムを作成するときに、必要なライブラリと 50 cm 移動するのに必要な時間を考慮するよう生徒に伝えます。

サンプルプログラム：

```
#モーター、ポート、ライトマトリックスをインポート
from hub import port, light_matrix
import runloop
import motor_pair

async def main():
    #カウントダウン(3, 2, 1)をライトマトリクスに表示
    await light_matrix.write("3")
    await runloop.sleep_ms(1000)
    await light_matrix.write("2")
    await runloop.sleep_ms(1000)
    await light_matrix.write("1")
    await runloop.sleep_ms(1000)

    #ポートE（左）とポートF（右）に接続されたモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    #指定した速度(500)で5秒間、前進する
    await motor_pair.move_for_time(motor_pair.PAIR_1, 5000, 0, velocity=500)

runloop.run(main())
```

3. 説明する

生徒たちとどのようにしてモデルを移動できたのかを話し合い、さまざまなコードの組み合わせを一緒に確認します。
生徒に次のような質問をします。

- あなたのプログラムはどのように機能しましたか？
- 2 つのモーターのプログラミングと 1 つのモーターのプログラミングはどのように異なりましたか？
- ホッパーモデルの設計は、ホッパーモデルの動きをどのように決定するのでしょうか？
- ホッパーを動かすのに大変だったことは何ですか?
- 秒単位のプログラミングでホッパーを 50cm 移動するのに大変だったことは何ですか?

4. もっと詳しく探究する

距離と回転を使用して 2 つのモーターの移動を調べます。
生徒に 3 つの新しいコード行を紹介します。

```
#異なる量のターン
await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, -20, velocity=280)
await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, 90, velocity=280)
await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

生徒にこれらの新しいコード行を探究し、ホッパーをさまざまな方法で移動する方法を調べる時間を与えます。
どうやってモデルを動かし、さまざまなコードの組み合わせを一緒に確認することができたのかを生徒たちと話し合います。
生徒に次のような質問をします。

- コードの各行はどのように機能しましたか？
- プログラムではどのようにしてモーターを異なる方向に動かすことができましたか？
- モーターの速度を設定するにはどのような方法がありますか？
- モーターを cm で動かすように設定するとどうなりましたか？
- れらのタイプのコードはどのような場合に使用しますか？

生徒たちに、motor_pair.move_for_time 行で使用されている数値がそれぞれ時間、ステアリング、速度を表していることを指摘します。 すべての数値は整数型であり、整数のみを使用できます。
このコード行を生徒たちと特に見直して、モーターを別の方法で動かす方法について話し合います。 500 は左側のモーターの速度、1000 は右側のモーターの速度です。 3000 はホッパーが動作する時間です。

```
await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- ホッパー モデルをさまざまな方法で動かすために複数のモーターをどのようにプログラムできましたか？
- ホッパーモデルの設計は、ホッパーモデルの動きをどのように決定するのでしょうか？
- ホッパーのプログラミングで大変だったことは何ですか?

生徒にノートに次のことを答えてもらいます。

- 複数のモーターのプログラミングについて今日何を学びましたか？
- ロボットのデザインはロボットの動きをどのように決定するのでしょうか？

ユニット 2 レッスン 5

ホッパーレース!!

ホッパーレース!!

モーター制御の最適な使用を考慮してコースを移動するようにモデルをプログラムします。

主な目的

生徒は次のことを行います：

- 一連のステップとターンを通過するプログラムを作成します。
- モーターペアを多用途に活用できます。

探究のための質問

完了するタスクによって、ロボットの移動方法がどのように決まるのでしょうか？ 動きを文書化することがなぜ重要なのでしょうか？

必要な機材

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

準備する

- 特に Bluetooth 経由で接続している場合は、SPIKE Prime ハブが充電されていることを確認してください。
- 生徒がホッパーランのレッスンで使用したホッパーモデルを組み立ててあることを確認します。

1. 引きつける

車用のトラック、人用のトラック、自転車用のトラックなど、さまざまなタイプのレーストラックについて生徒とディスカッションを始めます。レースで走れるトラックにはさまざまなタイプがあります。 さまざまなタイプのレース場のビデオや画像を表示することを検討してください。

生徒たちに、直線エリアとターンを含むレース用のトラックを作成してもらいます。
各グループは、ホッパーが通過できる基本的なレースコースを設計する必要があります。 コースには直線とターンを含めて少なくとも 5 つの要素が必要です。

2. 探究する

生徒はコースを走行するためにホッパーをプログラムします。
ホッパーがコースに沿って走るためにどのようにプログラムできるかを生徒たちと話し合います。 モデルの設計を変更せずにモーターを動かすさまざまな方法を考えます。

生徒に、モデルを直進または回転させるために各モーターを動かすさまざまな方法を考えるように促します。

- たとえば、モーターペアのタンク移動を使用してモーター F がゆっくり動く一方で、モーター E は速く動く場合があります。
- たとえば、 1 つのモーターが角度を使用して移動し、もう 1 つのモーターが停止していることが考えられます。

生徒たちに、レースを完走するために必要な手順とプログラミング要素を説明する疑似コード プログラムを作成してもらいます。
Python プログラミング キャンバスで新しいプロジェクトを開きます。 すでにプログラミングエリアにあるコードを消去するように生徒に指示します。

この課題では、コース移動の各ステップのコードに # を使用したコード内コメントを含めて、モデルの動き (つまり、直進、回転、後退など) を説明するように生徒に指示します。

生徒たちにレースのステップを 1 つずつ完了するよう奨励します。この課題では、プログラムのテストと反復が重要になります。コンソールでエラー メッセージを確認し、必要に応じてサポートが必要な場合はナレッジベースを参照するよう生徒に伝えます。

3. 説明する

生徒が最終的なプログラムとホッパーをどのようにプログラミングしたかを共有できるようにします。
生徒たちに次のように尋ねます。

- レースのさまざまな部分を移動するようにホッパーをどのようにプログラムしましたか？
- デバッグでどのような問題がありましたか？プログラミング中にエラー メッセージが表示されましたか？
- この課題で難しかったことは何ですか？

生徒に、プログラムで使用されているコードのコメントを確認して、コードが適切に文書化されており、理解しやすいかどうかを判断してもらいます。 クラスでいくつかの例について話し合い、文書化のベストプラクティスについて考えます。

4. もっと詳しく探究する

生徒が他のグループによって開発された他のコースでレースに挑戦できるようにします。新しいコースを完了するには、学生は新しいプログラムを作成する必要があります。プログラムが疑似コードの期待どおりに動作することを確認するために、プログラムを数回テストするよう生徒に伝えます。# を使用してコード内コメントを追加して、プログラムの手順を説明するように生徒に指示します。

5. 評価する

教師の観察：

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- 課題の各ステップでプログラミングにどのように取り組みましたか？
- プログラムの各ステップを文書化することが重要なのはなぜですか？
- 各ステップでプログラミングをテストすることが重要なのはなぜですか？

生徒にノートに次のことを答えてもらいます。

- 長いプロセスの各ステップをチェックすることが、時間をかけてデバッグする方法の 1 つであるのはなぜですか？
- この課題で難しかったことは何ですか？

Unit 3 Sensing Trouble:Exploring Sensor
ユニット 3 センシングトラブル：センサーの探究

Unit Introduction: ユニット紹介

この単元では、生徒は、ロボットに安全な動作をさせるために、重要なコンピューターサイエンスの原理とテキストベースのコーディング言語 Python のプログラミング概念を探求します。生徒はセンサーがどのように情報を取り込むか、そしてこの情報に基づいて動作するようにロボットをプログラムする方法を学習します。レッスンは、生徒が次の分野のスキルと知識を高めることができる順序で設計されています。

- カセンサーをプログラムし、このタイプのセンサーの使用方法を検討します。
- 距離センサーをプログラムし、このタイプのセンサーの使用方法を検討します。
- ロボットの動きがセンサーの精度にどのような影響を与えるかを調査します。
- 特定の状況に使用する適切なセンサーを決定します。
- 既存のコードを使用および変更して、新しいアイデアを検討します。
- アルゴリズムの作成をサポートする疑似コードを作成します。
- コード内コメント機能を利用して、プログラムの一部を文書化します。
- 問題を定義して分解します。

ユニット学習の期待

この単元では、生徒はセンサーを制御する方法を探り、安全な動きを生み出すための情報を検出して提供するためにセンサーがどのように使用されるかを理解します。ロボットのセンサーを効果的に活用し、プログラミングする方法を体験します。生徒は擬似コードを利用してアルゴリズムの作成をサポートし、コメントをコード化してプログラムを文書化します。

探究への質問

センサーは意思決定を行うための情報をどのように提供できるのでしょうか？
センサーはどのように安全を提供できるのでしょうか？センサーはいつ使用するのが適切ですか？

このユニットで使用するセンサー

カセンサー（ Force Sensor）

距離センサー（ Distance Sensor）

ユニット 3 レッスン 1

センシング開始

センシング開始

学生は力センサーを使用した条件文とセンサーのプログラミング方法を学びます。

主な目的

生徒は次のことを行います：

- フォースセンサーをプログラムします
- 条件文を作成する

探究のための質問

センサーはどのようにしてモーターと相互作用したり、モーターを制御したりできるのでしょうか？

必要な機材

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

準備

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

1. 引きつける

センサーがどのように機能するかについて生徒たちに考えてもらいます。 生徒たちにフリーズダンスのゲームをしてもらいます。 力センサーを持ち、センサーのボタンを押している間だけ動けることを生徒に説明します。ボタンを押したり放したりするときは、それがわかるようにしてください。生徒はダンスの時間になったら好きなように動くことができますが、ボタンを放すとフリーズする必要があります。

どのように動くべきかについての情報を提供するためにセンサーがどのように使用されたかを生徒と話し合います。センサーはどのように機能するのですか？

2. 探究する

生徒は力センサーをコーディングしてセンサーの操作を学びます。
ナレッジベースの「 はじめに」セクションに案内します。 ここで 7. センサー制御にアクセスできます。 これは、SPIKE Prime で力センサーを使用する方法の情報と例を提供します。
この情報を使用して、モーターと力センサーをハブに接続する際の生徒の指導を行ってください。

Python プログラミング キャンバスで新しいプロジェクトを開きます。 すでにプログラミング領域にあるコードを消去するように生徒に指示します。 学生はハブを接続する必要があります。

プッシュ、スタート、ストップ

センサーはさまざまな方法で使用できます。 力センサーは、アクションの開始と停止をプログラムできる「ボタン」のように使用できます。

センサーが機能するようにどのようにプログラムすべきかを生徒たちとブレインストーミングします。 ソフトウェアがハードウェアを正しく実行するために必要な情報は何ですか？
力センサーを動作させて、押すとモーターを開始および停止する疑似コード プログラムを作成します。 疑似コードは、プログラムに実行させたいことを言葉で記述します。

例としては次のようなものが考えられます。

- フォーカスセンサー、モーター、ハブをインポート
- カセンサーからの入力を受信
- モーターをオン
- 時計回りに 2 秒間移動

注：ステップで作成するコード (peudocodr) は、プログラムに配置されるコードと正確に一致する必要はありません。 疑似コードは、生徒がどのようなコードが必要かを考えるのに役立ちます。

カセンサーを使用してモーターを停止および始動するためのサンプル コードを生徒に提示します。 学生はこのプログラムをプログラミング キャンバスに入力します。

```
#モーター、ポート、力センサーをインポート
from hub import port
import runloop
import motor
import force_sensor

def is_force_sensor_pressed():
    #力センサーからの入力(押されている)を収集
    return force_sensor.pressed(port.B)

def is_force_sensor_not_pressed():
    #力センサーからの入力(解放されている)を収集
    return not force_sensor.pressed(port.B)

async def main():
    #力センサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)

    #モーターを起動
    motor.run(port.A, 750)

    #力センサーが解放されるまで待つ
    await runloop.until(is_force_sensor_not_pressed)

    #モーターを止める
    motor.stop(port.A)

runloop.run(main())
```

生徒たちにプログラムを実行してもらいます。

コードを実行したときに何が起こったのかを生徒たちと話し合います。生徒は、カセンサーが押されるとモーターが毎秒 750 度で回転し始めたことを確認する必要があります。モーターはカセンサーが解放されるまで作動し続け、カセンサーが解放されてモーターが停止しました。生徒と一緒にプログラムを見直して、コードの各行を理解していることを確認します。

カセンサーの入力がどこに保存されているかを生徒に指摘します。関数などのプログラムの一部に名前を付ける方法を考慮する必要があることを説明します。何を待つべきかを伝える関数を作成しています。

このコードを変更して、プログラムがスタートするとモーターが動作を開始し、ボタンを押したときに停止する方法を生徒に考えてもらいます。 生徒にコードを変更してプログラムを実行するように指示します。

```python
#モーター、ポート、力センサーをインポート
from hub import port
import runloop
import motor
import force_sensor

def is_force_sensor_pressed():
    #力センサーからの入力を収集
    return force_sensor.pressed(port.B)

async def main():
    #モーターを起動
    motor.run(port.A, 1000)

    #力センサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)

    #モーターを止める
    motor.stop(port.A)

runloop.run(main())
```

注：学生はコードを変更して、motor.run 行を until 行の前に移動する必要があります。 これによりモーターの動作が開始され、力センサーが押されるまで待機します。 また、力センサーを押さない機能は必要ないことに注意してください。

生徒に力センサーとモーターを使って探索する時間をさらに与えます。

3. 説明する

生徒に新しいコードを共有させ、力センサーをどのように使用したかについて話し合います。
生徒に次のような質問をします。

- センサーを使用してモーターの動作を制御するにはどうすればよいですか？
- 力センサーを使用またはプログラムするさまざまな方法には何がありますか？
- モーターを始動するときに、モーターが移動する時間や距離を設定しないのはなぜですか？
- 力センサーで発生すると思われるエラーは何ですか？

4. もっと詳しく探究する

生徒たちに力センサーを使って何か新しいことに挑戦してもらいます。 生徒にコンソールを開いてもらいます。 print() 関数を使用してコンソールにメッセージを表示することができます。
新しいコード行を生徒に紹介します。 以前のコードを変更したり、コードを入力したりできます。

```python
#コンソールに「Hello!」を表示
print('Hello!')
```

コードの新しい行「 print(‘Hello!’)」が実行されると何が起こるか生徒と話し合います。

生徒は実際のロボットが動くときにメッセージを含めることができるようになりました。 生徒は、プログラム内で何が起こっているかを出力として文書化する print 関数を確認できます。
デバッグのヘルプとして、出力されるメッセージは引用符付きの括弧 ( ) で囲む必要があることを生徒に伝えます。(‘ ‘) または (“ “) とします。

1 人がボタンを押して質問に答えると、正しい答えが画面 (コンソール) に表示 される、質疑応答ゲームを作成するよう生徒に課題を与えます。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。

生徒に次のような質問をします。

- どんな方法でカセンサーが動作するようにプログラムできましたか？
- センサーとモーターが相互作用する方法にはどのようなものがありますか？
- print 関数を使用するアイデアはありますか？

生徒にのーとに次のことを答えてもらいます。

- 今日、カセンサーの使用について何を学びましたか？
- print 関数はどんな時に利用できますか?

ユニット 3 レッスン 2
突進するライノ (サイ)

主な目的

生徒は次のことを行います :

- カセンサーを探索する
- 動きの力の影響を理解する

突進するライノ (サイ)

カセンサーを使用して動きを制御する方法を学びます。

探究への質問

カセンサーを使用して動きや動作を制御するにはどうすればよいでしょうか ?

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

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

1. 引きつける

猛るサイを止める方法について生徒たちにディスカッションしてもらいます。サイが突進したり、物体にぶつかったりする様子を映したビデオを見ることを検討してください。サイがどのように動くのか、そしてそれを止めるには何が必要なのかについて話し合います。また、サイが使用する可能性のある感覚と、それらの感覚が学生がプログラムできるセンサーとどのように似ているかについて話し合うことも検討してください。

2. 探究する

学生は Rhino モデルを構築して、カセンサーを使用して移動するさまざまな方法を調査します。

生徒にアプリ内の「 組み立て図」セクションを案内します。 ここで生徒は Rhino モデルの組み立て図にアクセスできます。 生徒たちにモデルを構築してもらいます。 組み立て図は https://education.lego.com/ja-jp/product-resources/spike-prime/ ダウンロード / 組立ガイド / でも入手できます。

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。 すでにプログラミングエリアにあるコードを消去するように生徒に指示します。 ハブを接続します。

サイの走り

サイを前に走らせるよう生徒に挑戦させてください。生徒はこのプログラムから始めてください。

```
#モーターペア、ポートをインポート
from hub import port
import runloop
import motor_pair

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)

    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)

runloop.run(main())
```

生徒にプログラムを中止するよう指示します。プログラムにはサイに停止を指示するものは何もないという事実について話し合ってください。やがて、サイは何かに衝突します。

サイを止めるためのアイデアについて話し合います。1つの方法は、プログラムの停止を追加したり、特定の時間や距離などを設定したりすることです。もう1つの方法は、サイの鼻のように取り付けられた力センサーを使用することです。

力センサーを使用して、壁にぶつかったときに、猛り回るサイを止めるように生徒に促します。生徒はサイを壁または30 cm以上離れた別の丈夫な物体に向けて配置する必要があります。
このサンプルプログラムを学生と共有します。学生はこのプログラムをプログラミングキャンバスに入力します。

```
#モーターペア、ポート、力センサーをインポート
from hub import port
import runloop
import motor_pair
import force_sensor

def is_force_sensor_pressed():
    #力センサーの入力（押されている）を収集
    return force_sensor.pressed(port.E)

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)

    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)

    #力センサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)

    #モーターを止める
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

注：コンソールのエラーに注意するよう生徒に伝えます。 生徒はエラーメッセージの行を参照して、入力エラーが発生した可能性のある場所を特定できます。

生徒たちにプログラムをさらに数回実行してもらいます。 毎回、生徒はサイのモデルを壁から遠ざける必要があります。モデルをさらに前に移動してもプログラムの実行方法がどう変わらないかを生徒に調べてもらいます。

3. 説明する

Rhino モデルがどのように動いたかを生徒たちと話し合い、グループでコードを確認します。
生徒に次のような質問をします。

- プログラムはどのように機能しましたか？
- motor_pair.move コード行の 0 と 750 は何を表しますか？
- カセンサーはどのように機能しましたか？
- サイを壁から遠ざけると何が起こりましたか？異なる距離でモデルを実行すると、プログラムの動作方法は変わりましたか？
- この課題で難しかったことは何ですか？

4. もっと詳しく探究する

プログラムを変更して、サイがはやい速度で突進しているときと、ゆっくりと動いているときと止まり方を調査するように生徒に課題を与えます。

生徒たちにプログラムをあと 2 回実行してもらいます。 高い速度（Velocity = 1000）と低い速度（Velocity = 250）で 1 回すつサイを動かして止まり方に違いがあるかを調べさせます。

```
16    #はやい速度で前進
17    motor_pair.move(motor_pair.PAIR_1, 0, velocity=1000)
18
19    #低い速度で前進
20    motor_pair.move(motor_pair.PAIR_1, 0, velocity=250)
```

各プログラムの実行後に何が起こるかについて話し合います。生徒は、速度が 250 に設定されている場合、カセンサーが物体に触れている間、サイがちょうど停止していることに注意してください。しかし、速度を 1000 に設定すると、サイは物体に当たって跳ね返ります。 なぜこのようなことが起こるのかグループで話し合います。

生徒がセット内の追加のレゴパーツで小さな壁を組み立てるように指示します。生徒たちに、壁を突き抜けて突進し、適切な音を鳴らすようにサイをプログラムするよう挑戦してもらいます。生徒は、print() 関数を使用してコンソールにメッセージを入力し、壁にぶつかったときにサイが何か言っている（痛い！) 様なことも追加することができます。
生徒が最終的なプログラムを共有し、話し合うことができるようにします。

5. 評価する

教師の観察

学生たちとプログラムについて話し合います。
生徒に次のような質問をします。

- カセンサーはサイを制御するためにどのように機能しましたか？
- なぜモーターの速度がカセンサー停止時の動作に影響を与えるのでしょうか？
- カセンサーはサイに何をすべきかを伝えるための情報をプログラムにどのように提供したのでしょうか？

生徒にノートに次のことを答えてもらいます。

- 今日、カセンサーを使用してサイを制御することについて何を学びましたか？

ユニット 3 レッスン 3
カートコントロール

カートコントロール

距離センサーを探究してカートの動きを制御します。

探究への質問

- 距離センサーを使用して意思決定のための情報を提供するにはどうすればよいですか？
- より正確な動きのためにセンサー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

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

主な目的

生徒は次のことを行います：

- 距離センサーをプログラムします
- 距離のある動きを探ります
- 超音波について理解する

ボキャブラリー

Float (浮動小数点数)
整数

1. 引きつける

自動機械の使用方法について生徒たちに考えてもらいます。 これらの機械が物体への衝突を防ぐためにどのようにセンサーを使用しているかについて生徒に考えてもらいます。例として、さまざまな梱包機械や出荷機械の画像やビデオを見せることを検討してください。また、生徒に自分の例を調べて見つけるよう促すこともできます。

自動化された機械がうまく動き回る様子を生徒たちと話し合います。

2. 探究する

生徒は配送カートのモデルを組み立てて、距離センサーを使って移動するさまざまな方法を調べます。
SPIKE アプリの「組み立て図」セクションに案内します。 ここで配送カート モデルの組み立て説明書にアクセスできます。 生徒たちにモデルを組み立ててもらいます。 組み立て説明書は次のサイトからも入手できます。
https://education.lego.com/ja-jp/product-resources/spike-prime/ ダウンロード / 組み立てガイド /

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。 すでにプログラミング領域にあるコードを消去するように指示します。ハブを接続します。

カート・アンダー・コントロール

距離センサーの使い方を探究します。

生徒たちにこのプログラムを復習してもらいます。 バグがあるかどうか話し合ってください。 そうでないことに気づいたら、なぜこのプログラムがうまく機能しないのかを尋ねます。 生徒は、モーターは永遠に動き続け、最終的には何かにぶつかることを理解する必要があります。

```python
#モーターペア、ポートをインポート
from hub import port
import runloop
import motor_pair

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)

    #指定の速度で前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)

runloop.run(main())
```

生徒たちに、カートが物体にぶつからないようにする方法を考えるように促します。 生徒は、これまでの経験に基づいて、プログラムに停止を含めたり、カートにカセンサーを追加したりすることを考えるかもしれません。 生徒たちに、別のセンサーを使用することを伝えます。

生徒に、カートの底に取り付けた距離センサーの位置を確認してもらいます。 距離センサーを使用してカートを移動するためのサンプル コードを生徒に提示します。 生徒はこのプログラムをプログラミング キャンバスに入力します。
生徒に、パートナーと一緒にコードの各行を確認し、プログラムの目的を決定します。
生徒たちにプログラムを実行してもらいます。

```python
#モーター、モーターペア、ポート、距離センサーをインポート
from hub import port
import motor
import runloop
import motor_pair
import distance_sensor

def distance_sensor_grater_than():
    #距離センサーの値が150mmより大きいどうか調べる
    #距離センサーの値をdistanceに取り込む
    distance = distance_sensor.distance(port.E)
    return distance == -1 or distance > 150

def distance_sensor_less_than():
    #距離センサーの値が100mmより近いかどうか調べる
    #距離センサーの値をdistanceに取り込む
    distance = distance_sensor.distance(port.E)
    return distance < 100 and distance > -1

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    #距離センサーの値が150より大きくなるまで待つ
    await runloop.until(distance_sensor_grater_than)
    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
    #距離センサーの値が100mmより小さくなるまで待つ
    await runloop.until(distance_sensor_less_than)
    #停止
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

生徒がプログラムを動作させるのに苦労している場合は、ヒントを与えてください。 生徒はプログラムを開始するときに、本や手などの物体を距離センサーの前にかざす必要があります。 プログラムを実行しオブジェクトを取り除くと、カートが動き始めるはずです。物体をセンサーの前に戻すと停止するはずです。 対象物は 100 mm より遠くにあることを確認してください。 生徒たちに、配送カートを対象物からさまざまな距離に置いてプログラムを複数回試してもらいます。

生徒たちに、オブジェクトとの距離を変更し、プログラムを数回再実行しながらプログラムを探究してもらいます。

注：モーターを始動する距離は、モーターを停止する距離と同じである必要はありません。

3. 説明する

グループで一緒にプログラムを見直します。
生徒に次のような追加の質問をします。

- await runloop.until(deistance_sensor_greater_than) および await runloop.until(distance_sensor_less_than) という行は、カートに何をするように指示していますか？
- プログラムを動作させるためになぜオブジェクトが必要だったのでしょうか？
- カートを物体からさらに離れたところからスタートするとどうなりますか？
- 距離センサーはどのように機能しますか？

カートの距離センサーが超音波センサーであることを生徒に説明します。 生徒に、ハブを接続する場所の近くのプログラミング キャンバスの左上のハブアイコンの右側を見てもらいます。ここでは、各ポートから入ってくるライブデータが表示されているはずです。モーター、センサーなどハブに接続されているすべてのハードウェアはライブ データを提供します。 距離センサーのライブ データは cm 表示ですが、プログラム内のセンサーの出力は mm 単位であることに注意する必要があります。

生徒に距離センサーのデータを見ながら、手を近づけたり遠ざけたりするよう指示します。 数字が変化し、手が遠ざかるにつれて増加します。片方の円（または「目」）から超音波パルスが発信し、センサーの前にある物体で反射し、もう片方の円（または「目」）に戻るというセンサーの仕組みを説明します。 センサーは、そのパルスが戻ってくるまでの時間を利用して、物体までの距離を「測定」します。 センサーは数式を使用して時間を距離測定値に変換します。

4. より詳しく探究する

生徒たちは、カートの後輪がカートの直線移動を妨げていることに気づいたかもしれません。 上記のプログラムを実行するとき、後輪が真っ直ぐでないとカートが片側に曲がってしまうのを見たかもしれません。 生徒向けにデモンストレーションを行うことを検討してください。

超音波パルスを使用して物体までの距離を検出する距離センサーにおいて、回転カートが問題となる理由を生徒に尋ねます。 カートをまっすぐに動かし続ける方法について生徒たちとアイデアを話し合います。

生徒は、後輪がモーターに取り付けられているため、カートをまっすぐに保つためにコードを追加できることを理解する必要があります。また、モーターを所定の位置に固定するために構造を変更することを考えることもできます。 それによってカートの動きがどのように制限され、代わりに生徒が再びコードに集中できるかについて話し合うことを検討してください。

バックモーターとホイールをまっすぐに設定するためのこのサンプル コードを生徒に提示します。 生徒は、大きなモーターを後輪で意図的に動かし、まっすぐになっていないことを確認する必要があります。 生徒たちにプログラムを実行してもらいます。

```python
#モーター、モーターペア、ポート、距離センサーをインポート
from hub import port
import motor
import runloop
import motor_pair
import distance_sensor

def distance_sensor_closer_than():
    #距離センサーの値が150mmより大きいどうか調べる
    #距離センサーの値をdistanceに取り込む
    distance = distance_sensor.distance(port.E)
    return distance < 150 and distance > -1

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    #カートがまっすぐに進むように後輪を0度の位置に動かす
    await motor.run_to_absolute_position(port.C, 0, 250, direction=motor.SHORTEST_PATH)
    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
    #距離センサーの値が150mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    #停止
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

プログラムについて生徒たちと話し合い、後輪をまっすぐに保つことでセンサーがどのようにより効果的に機能するかについて話し合います。 センサーが物体に対して垂直である場合、音波はより直接的に反射され、良好な読み取り値が得られます。

デバッグ

生徒に次のプログラムを確認して、受け取ったエラー メッセージに対処する方法を検討してもらいます。

デバッグアクティビティ # 1

```python
#モーター、モーターペア、ポート、距離センサーをインポート
from hub import port
import motor
import runloop
import motor_pair


def distance_sensor_closer_than():
    #距離センサーの値が150mmより大きいどうか調べる
    #距離センサーの値をdistanceに取り込む
    distance = distance_sensor.distance(port.E)
    return distance < 150 and distance > -1

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    #カートがまっすぐに進むように後輪を0度の位置に動かす
    await motor.run_to_absolute_position(port.C, 0, 250, direction=motor.SHORTEST_PATH)
    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
    #距離センサーの値が150mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    #停止
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

コンソール内エラーコード

```
Traceback (most recent call last):
  File "debagg #1", line 25, in <module>
  File "debagg #1", line 21, in main
  File "debagg #1", line 11, in distance_sensor_closer_than
NameError: name 'distance_sensor' isn't defined
```

エラー メッセージについて生徒と話し合います。 生徒は、エラーが 8 行目に関するものであることを認識する必要があります。ただし、エラーは 2 行～ 5 行目の間にあります。問題は、プログラムで距離センサー（ distance_sensor）が使用されているが、distance_sensor がインポートされていないことです。

デバッグアクティビティ #2

```
1  #モーター、モーターペア、ポート、距離センサーをインポート
2  from hub import port
3  import motor
4  import runloop
5  import motor_pair
6  import distance_sensor
7
8  def distance_sensor_closer_than():
9      #距離センサーの値が150mmより大きいどうか調べる
10     #距離センサーの値をdistanceに取り込む
11     distance = distance_sensor.distance(port.B)
12     return distance < 150 and distance > -1
13
14 async def main():
15     #左（ポートB）と右（ポートA）をペアにする
16     motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
17     #カートがまっすぐに進むように後輪を0度の位置に動かす
18     await motor.run_to_absolute_position(port.C, 0, 250, direction=motor.SHORTEST_PATH)
19     #前進
20     motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
21     #距離センサーの値が150mmより小さくなるまで待つ
22     await runloop.until(distance_sensor_closer_than)
23     #停止
24     motor_pair.stop(motor_pair.PAIR_1)
25
26 runloop.run(main())
```

コンソール内エラーコード

```
Traceback (most recent call last):
  File "debagg #2", line 25, in <module>
  File "debagg #2", line 21, in main
  File "debagg #2", line 11, in distance_sensor_closer_than
OSError: [Errno 19] ENODEV
```

エラー メッセージについて生徒と話し合います。 生徒は、エラーが 16 行目を指していることを認識する必要があります。問題は、MotorPair 変数が、モーターが接続されているポートとして定義されていないことです。生徒は、モーターが接続されているポートを確認し、正しいポート変数に変更する必要があります。ただし、このプログラムの場合モーターのポートは正しく、誤りは距離センサー（ distance_sensor）のポートです。ハードウェアの接続先ポートを指定する場合は必ず確認してからしてするよう指導します。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。

生徒に次のような質問をします。

- 距離センサーはカートを制御するためにどのように機能しましたか？
- 後輪のデザインはカートの動きにどのような影響を及ぼし、センサーの有効性にも影響を与える可能性がありますか？
- 距離センサーを使用するにはどのような方法がありますか？

生徒にノートに次のことを答えてもらいます。

- 距離センサーを使用して物体にぶつからないようにすることについて、今日何を学びましたか？

ユニット 3 レッスン 4

安全な配達

安全な配達

モーターとセンサーがどのように連携するかを学びます

探究への質問

安全性を提供するためにセンサー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
- 定規

準備

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

主な目的

生徒は次のことを行います：

- センサーを使用してモデルを安全に移動するようにプログラムします
- センサーを使用する場合はモーター出力の影響を検討してください

ボキャブラリー

Float (浮動小数点数)

int (整数)

1. 引きつける

ケイトとカイルは、遊び場に新しいツリーハウスを追加する作業をしています。 配送カートには必要な資材が入っていますが、カイルはカートが遊び場にある他の物を傷つけるのではないかと心配しています。

カートが遊び場で他の物にぶつからないようにするにはどうすればよいかについて議論します。 議論とその後の計画をサポートするために、例として遊び場の画像を表示することを検討してください。

2. 探究する

生徒たちに、どうすればカートをより安全に移動できるかを調べてもらいます。

カートが遊び場をより安全に移動する方法について生徒たちと話し合います。 距離センサーが「 認識」できる十分な高さに配置されたオブジェクトを使用して、遊び場を模倣する 2 つまたは 3 つのアイテムで小さな障害物コースを作成するよう生徒に指示します。

生徒たちに、カートを前進させ、物体にぶつかる前に停止できるようにするプログラムを作成してもらいます。これにより、カートが遊び場を安全に移動し、物体の前で停止できるようになります。距離センサーが物体に 15cm 以内に近づくまでカートのモーターをオンにし、その後モーターを停止するようにプログラムを作成します。

サンプルコード：

```
#モーター、モーターペア、ポート、距離センサーをインポート
from hub import port
import motor
import runloop
import motor_pair
import distance_sensor

def distance_sensor_closer_than():
    #距離センサーの値が150mmより大きいどうか調べる
    #距離センサーの値をdistanceに取り込む
    distance = distance_sensor.distance(port.E)
    return distance < 150 and distance > -1

async def main():
    #左（ポートB）と右（ポートA）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    #カートがまっすぐに進むように後輪を0度の位置に動かす
    await motor.run_to_absolute_position(port.C, 0, 250, direction=motor.SHORTEST_PATH)
    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=200)
    #距離センサーの値が150mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    #停止
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

生徒に、以下のような速度を変えて実行した結果を記録する表を作成してもらいます。

速　度	200	400	600	800	1000
距離センサーの値（mm）					

生徒は、毎回モーターの速度を変更しながらプログラムを 5 回実行する必要があります。 プログラムを実行し、SPIKE アプリの距離センサーの下に表示される距離を読み取ります。 生徒が各トライアルを完了し、チャートに記入する時間を与えます。

3. 説明する

さまざまな速度でプログラムを実行したときに発見したことを生徒と話し合います。

スピードとパワーの違いについて生徒たちと話し合います。 生徒がプログラムで速度を設定するとき、実際に設定しているのはモーターが回転する 1 秒あたりの度数であることを説明します。 これは、ロボットがどのくらいの速度で移動するのかを決定します。

生徒に次のような質問をします。

- パワーを上げるとカートの速度はどう変化しましたか？
- 速度を上げると、カートの停止方法にどのような影響がありますか？
- カートを最も正確に停止できる出力レベルまたは速度はどれですか？
- カートを安全に動かすためには、パワーレベルや速度をどのように考慮すればよいでしょうか？

4. より詳しく探究する

カートがどこに進入しても、物にぶつからずに遊び場を移動するように生徒に課題を与えます。資材を届けるためにカートが遊び場をどのように移動する必要があるかを生徒と話し合います。 カートは別の場所から入る可能性があります。 生徒はさまざまな角度からプログラムを実行して、距離センサーを使用して物に当たることなく遊び場を安全に移動できることを確認する必要があります。

注：配達カートは急旋回 (90 度の旋回など) はできませんが、円弧旋回を行います。

生徒は、カートの移動に必要な角度を考慮し、必要に応じて後輪の位置を変更(0度以外の角度に設定）する必要があります。

生徒たちに、遊び場内を安全に移動する方法を探る時間を与えてください。 生徒たちがどのようにして遊び場でカートを使用できたかについて話し合います。

生徒に次のような質問をします。

- センサーを使用して物体との衝突を回避するにはどうすればよいですか？
- センサーをどのように使用して、移動する物体の安全性を高めることができますか？
- カートの移動を助けるために、後部の追加モーターをどのように使用しましたか？
- この課題で難しかったことは何ですか？

さまざまなプログラムについて生徒と話し合い、モーターを動かすためのプログラムはどこか、どこでセンサーを使用しているのかを考えるよう促します。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。
生徒に次のような質問をします。

- カートをより安全に移動できるようにセンサーをどのようにプログラムしましたか？
- 距離センサーによる停止時の反応がモーターのパワーや速度に影響されるのはなぜですか？

生徒にノートに次のことを答えてもらいます。

- 安全性と精度を確保するためにセンサーを使用する際のモーターの出力レベルまたは速度の設定について、今日何を学びましたか？

ユニット 3 レッスン 5
ホッパーのトラブル

ホッパーのトラブル

生徒は、特定のタスクに適切なセンサーを選択する方法を調査します。

主な目的

生徒は次のことを行います：

- 適切なハードウェアの決定を行う
- センサーを追加するためにモデルを再設計します

探究への質問

- センサーはいつ使用するのが適切ですか？
- 特定のタスクにどのセンサーを使用するのが最適かをどのように判断しますか？

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

特に Bluetoorh 経由で接続している場合は、SPIKE Prime ハブが充電されていることを確認してください。

1. 引きつける

仕事に最適なツールを決定する方法についてディスカッションを始めます。 生徒に、完了するタスクや画像などの例をいくつか提示します。 作業は、穴を掘る、鉛筆を削る、ロボットのコーディングに至るまで多岐にわたります。 タスクごとに、その仕事に最適なツールについて話し合います。 たとえば、シャベルを使って鉛筆を削るロボットをコーディングすることはありません。 自分の名前を書くことは鉛筆、ペン、マーカー、クレヨンでできますが、シャベル、ハンマー、ジョウロで書くのは非常に難しいなど、使用できるツールが複数ある可能性のある例を挙げるようにしてください。

2. 探究する

生徒はホッパー モデルを組み立て、センサーを使用してさまざまな移動方法を調べます。
ホッパーモデルを組み立てるよう指示します。
Python プログラミングキャンバスで新しいプロジェクトを開くように生徒に指示します。プログラム領域にすでにあるコードを消去するように指示します。 ハブを接続します。
生徒は、ホッパーモデルが 50% の力、つまり速度 500 で前進できるプログラムを作成する必要があります。
サンプルコード：

```
#モーターペア、ポートをインポート
from hub import port
import runloop
import motor_pair

async def main():
    #左（ポートE）と右（ポートF）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)

    #前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)

runloop.run(main())
```

ホッパーモデルがどのように動くかを生徒たちと話し合います。

ホッパーが公園内を移動する際に直面する危険について生徒たちとディスカッションを始めます。ホッパーは踏まれてしまうかもしれない足にぶつかりたくありません。ホッパーが物にぶつからないようにするには、力または距離のどのセンサーが最適であるかを生徒に考えてもらいます。生徒がグループ内でさまざまなアイデアを話し合って、どのセンサーを使用するかを決定できるようにします。モデルが現在どのように機能しているかを注意深く観察するよう促します。

生徒は、モデルの基本設計を変更せずに、選択したセンサーをホッパーモデルに取り付ける必要があります。センサーをホッパーモデルに取り付ける時間を与えます。

生徒に、プログラムを変更して新しく追加されたセンサーを組み込むように指示します。生徒はモデルをテストして、選択したセンサーが機能するかどうかを判断する必要があります。プログラムを数回テストし、モデルが期待どおりに動作することを確認するために必要に応じて変更するように伝えます。# を使用してプログラムの手順を説明するコード内コメントを追加するように指示します。

3. 説明する

生徒が最終的なホッパーモデルとプログラムを共有できるようにします。 どのセンサーを選んだか、そしてその理由について話し合います。

生徒に次のような質問をします。

- どのセンサーを選択しましたか？その理由は何ですか？
- センサーはどの程度うまく機能しましたか？ホッパーモデルは何かオブジェクトに遭遇しましたか？
- センサーを追加する際に解決しなければならなかった問題は何ですか？
- デバッグが必要な問題は何でしたか？
- この課題で難しかったことは何ですか？

4. もっと詳しく探究する

ホッパーが物に到達したときにモーターをオフにするのではなく、逆方向または向きを変えるようにプログラムを変更するように指示します。 生徒は、コードのどの部分が同じままで、コードのどの部分を変更する必要があるかを検討する必要があります。

生徒たちに、ホッパーをどのように動かしたいかを計画してもらいます。コードを変更する前に、生徒に疑似コードを作成してもらいます。新しいプログラムに以下を含める必要があります。

- バッタがオブジェクトを発見し止まったときに画像をハブに表示します
- 方向を逆にするか向きを変えて開始位置に戻ります
- 新しいサウンドを再生します
- プログラムのどこかにセンサーの 2 回目の使用を含めます

オプション：コンソールにメッセージを表示することもできることを生徒に伝えます。

この課題では、各ステップのコードに # を使用したコード内コメントを含めて、モデルの動き (つまり、真っすぐに移動、停止、後退) を説明するように指示します。

生徒たちに一度に 1 つのステップを完了するよう促します。 この課題では、プログラムのテストと反復が重要になります。コンソールでエラーメッセージを確認し、必要に応じてサポートが必要な場合はナレッジベースを参照するよう伝えます。

生徒が最終的なプログラムを共有し、話し合うことができるようにします。

5. 評価する

教師の観察

生徒たちとプログラムについて話し合います。

生徒に次のような質問をしてください。

- どのセンサーを使用するかをどのように決めましたか？
- センサーを最も効果的に使用するためのプログラムはどのように作成しましたか？
- ホッパーモデルがオブジェクトから安全に離れるようにプログラムにどのような変更を加えましたか？

生徒にのーとに次のことを答えてもらいます。

- タスクに適したセンサーの選択について今日何を学びましたか？

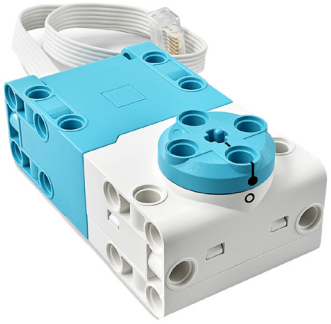
Worksheets for Student
生徒用ワークシート

生徒用ワークシート

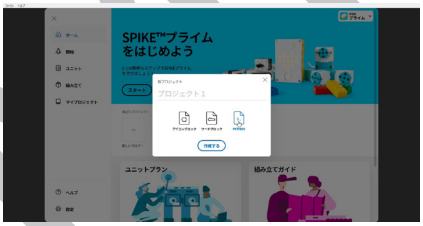
UNIT I Lesson I

ハードウェア&ソフトウェア

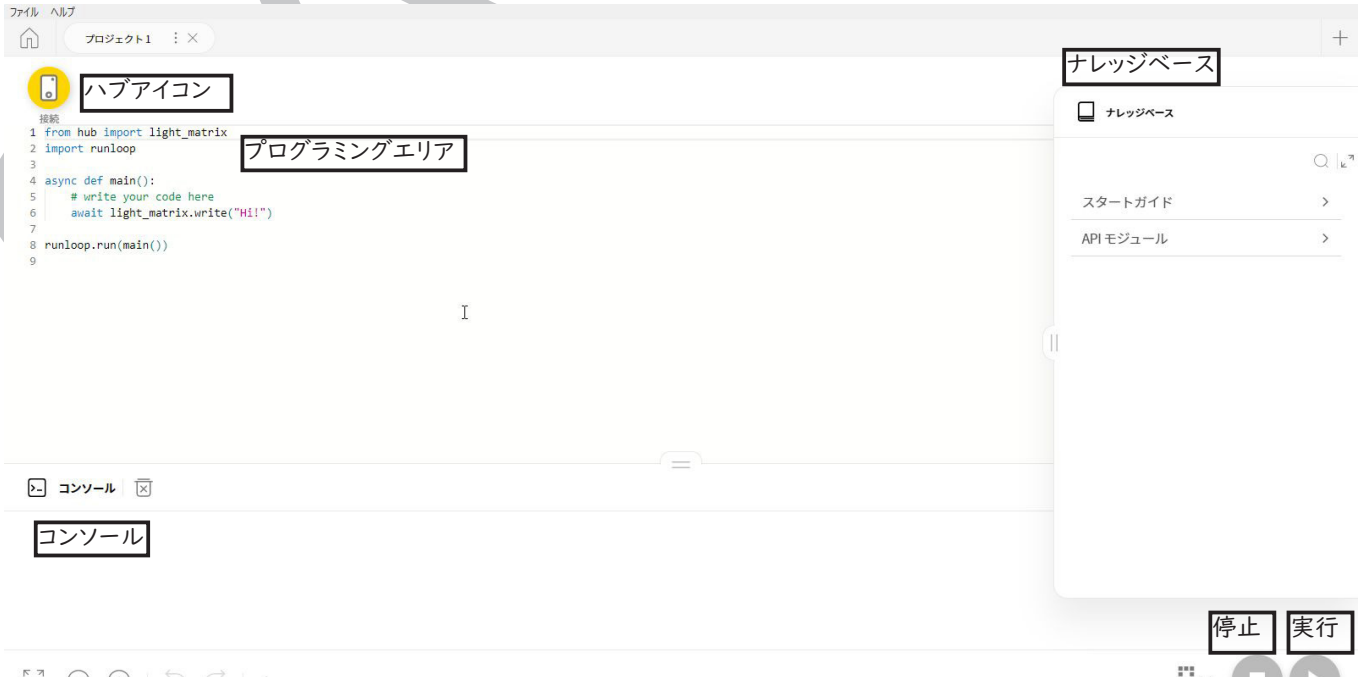
セットの中から次のハードウェアを出して並べましょう。それぞれの名前を記入しましょう。



SPIKE App を立ち上げて、Python プロジェクトを開きましょう。



Python プログラムキャンバスの各部分の名前を覚えましょう。



Python プログラムを見てみよう。

```
from hub import light_matrix
import runloop
async def main():
    #write your code here
    await light_matrix.write("Hi!")
runloop.run(main())
```

重要

import _____

プログラムにハードウェアを認識させるために必要なライン
import の後ろに認識させたいハードウェアを指定します

ナレッジベースでどんなハードウェアがインポートできるのか調べましょう。

ハードウェアの種類

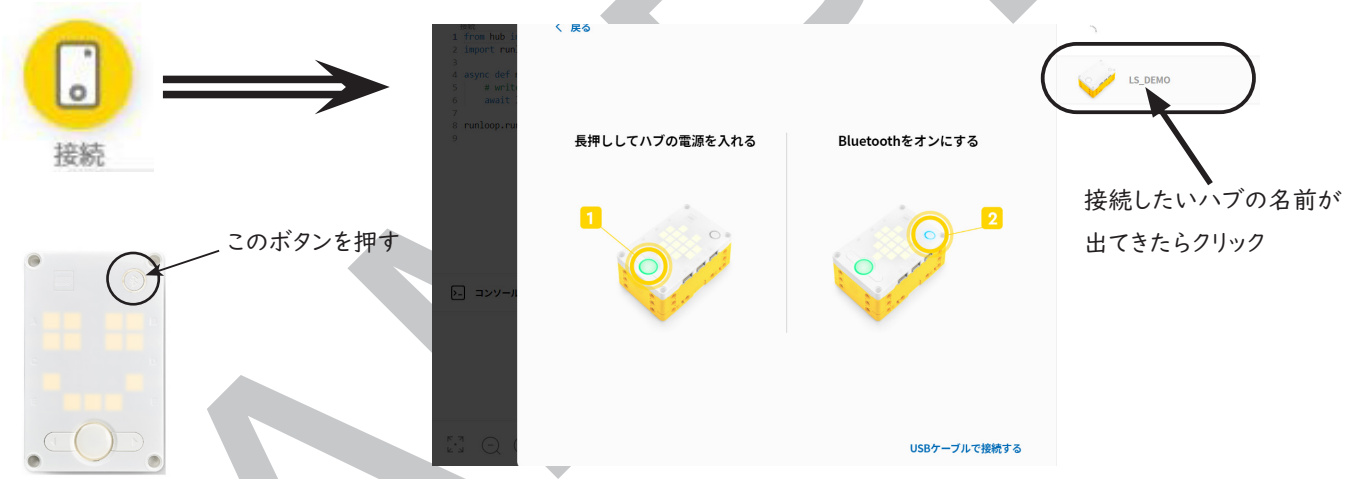
light_matrix _____ button _____ light _____

sound _____ app _____ motion_sensor _____

force_sensor _____ color_sensor _____

distance_sensor _____ motor _____ motor_pair _____

ハブアイコンをクリックしてハブを接続しましょう。

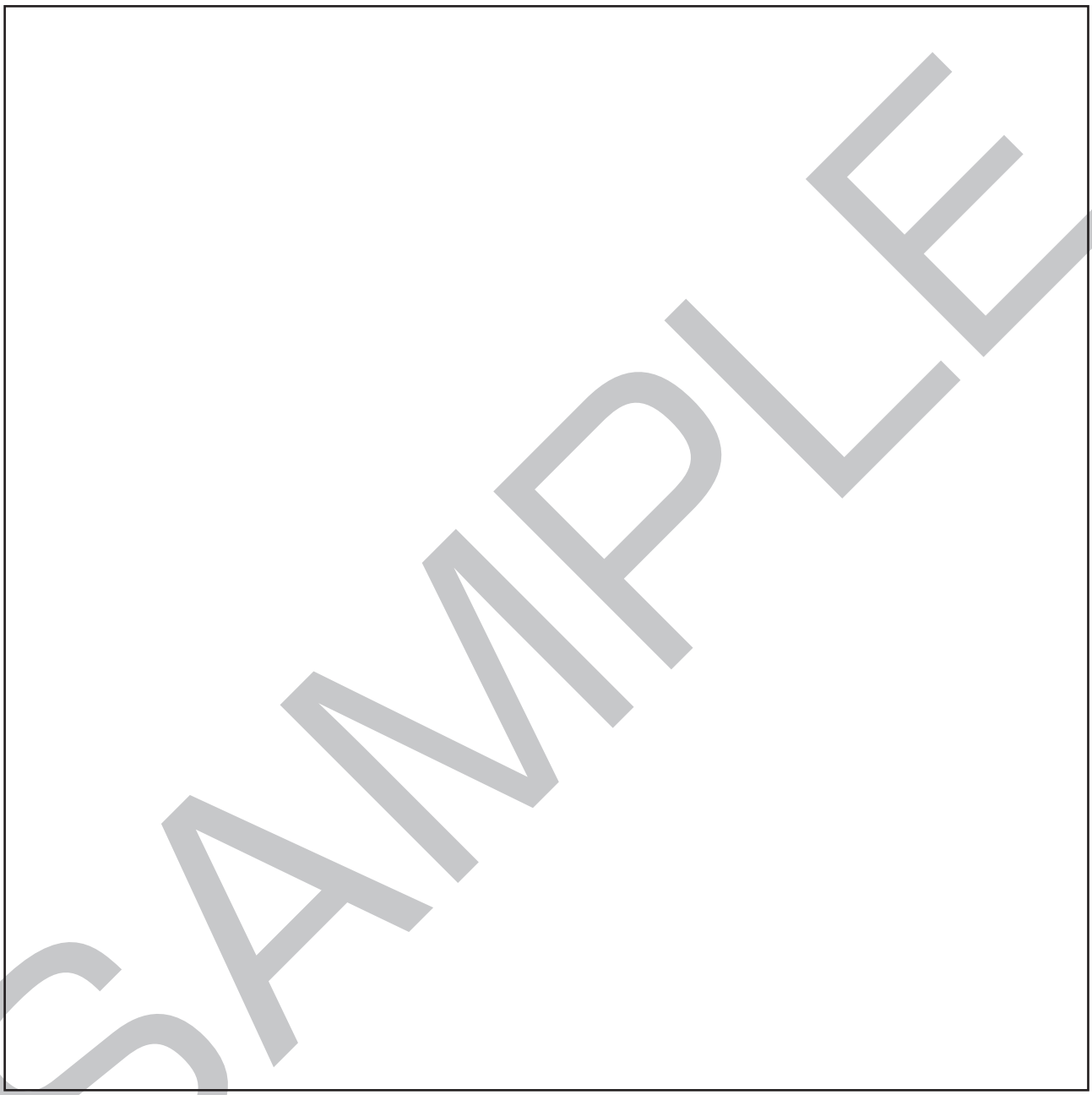


接続できたら、実行ボタンをクリックして、プログラムを実行してみましょう。
何が起きましたか？

新しいプロジェクトを開いて、ナレッジベースを参考にしながらいろいろなフレーズを試してみましょう。

UNIT I Lesson 2
光でコミュニケーション

色紙を 2 ～ 3 cm角の小さな正方形に切ります。25 枚の正方形ができたら、5 x 5 のマトリクスを書いて、色紙を置き、いろいろな画像を作ってみましょう。どんな画像が作れるでしょうか?



ライトマトリクスに画像を表示するプログラムを作るため、疑似コードを書きましょう。「 疑似コード」とは、 目的のタスクをコンピュータで実行するための手順をステップごとに言葉で書き出した一連の指示のことを言います。

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.

新しいプロジェクトを開いて次のプログラムを入力しましょう。

```
from hub import light_matrix
import runloop
async def main():
    # ハッピー顔を表示する
    light_matrix.show_image(light_matrix.matrix.IMAGE_HAPPY)
    await ruloop.sleep_ms(5000)
    light_matrix.clear()
runloop.run(main())
```

入力出来たらハブを接続してプログラムを実行してみましょう。何が起きましたか？

ナレッジベースの light_matrix.() の項目を調べて、ほかにいろいろな画像を表示してみましょう。

どんな画像を表示できましたか？

2 つの画像を表示してみよう

ライトマトリックスにハートとスマイル顔を表示してみましょう。

```
from hub import light_matrix
import runloop
async def main():
    # ハートを表示する
    light_matrix.show_image(light_matrix.matrix.IMAGE_HEART)
    await ruloop.sleep_ms(5000)
    light_matrix.clear()
    # ハッピー顔を表示する
    light_matrix.show_image(light_matrix.matrix_IMAGE_HAPPY)
    await ruloop.sleep_ms(5000)
    light_matrix.clear()
runloop.run(main())
```

コンソールにエラーが表示されたら、エラーメッセージをよく見てデバッグしましょう。

文字（ 言葉）を表示してみよう

ライトマトリックスに”Hello !”と表示させてみましょう。 どのように表示されるでしょう？

```
from hub import light_matrix
import runloop
async def main():
    #”Hello!” を表示する
    await light_matrix.write(“Hello!“)
runloop.run(main())
```

画像と文字（ 言葉）を表示してみよう

ライトマトリックスに画像を表示させた後に何かの言葉（ 自分の名前でもペットの名前でもなんでも OK）を表示させてみましょう。

学んだことを書きましょう

今日はライトマトリックスのプログラミングについて何を学びましたか？

UNIT I Lesson 4
音でコミュニケーション

Beep（ビープ音）を出してみよう

ハブのスピーカーからビープ音を足してみましょう。

```
# スピーカーをインポート
from hub import sound
import runloop
async def main():
    # ビープ音を 1 秒間鳴らす
    await sound.beep(400, 1000, 100)
runloop.run(main())
```

ナレッジベースで3つのパラメーターの意味を調べてみましょう

400：

1000：

100：

デバッグ

周波数のパラメーターを”100”にしたらどうなるかやってみましょう。

```
# スピーカーをインポート
from hub import sound
import runloop
async def main():
# ビープ音を 1 秒間鳴らす
    await sound.beep(100, 1000, 100)
runloop.run(main())
```

周波数（1つ目）のパラメーターの数値を変えて、どのくらいの周波数から音が出るのか調べてみましょう。

また、どのくらいの音まで聞こえるのかも調べてみましょう。

曲を作ろう

3つのパラメーターをいろいろ工夫して、簡単な曲を作ってみましょう。（サンプル）

```
# スピーカーをインポート
from hub import sound
import runloop
async def main():
# 曲を鳴らす
    await sound.beep(400, 1000, 100)
    await sound.beep(450, 500, 100)
    await sound.beep(500, 1000, 100)
runloop.run(main())
```

自分で考えたパラメーターのセットを記録しておきましょう。

1. ビープ音を鳴らすためにインポートするものは何ですか？

2. 3つのパラメーターの意味はそれぞれ、前から何だったでしょうか？

3. 何が難しかったですか？

音を再生しよう

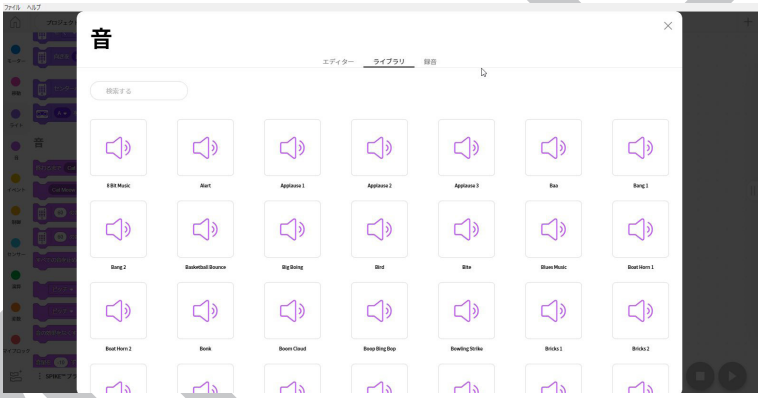
犬や猫の鳴き声など、あらかじめ録音された音を再生することもできます。
では、猫の鳴き声を再生してみましょう。

```
# サウンドをインポート
from app import sound
import runloop
async def main():
# 音を再生
await sound.play('Cat Meow 1')
runloop.run(main())
```

プログラムを実行して、猫の鳴き声が聞こえましたか？
さて、猫の鳴き声はどこから聞こえましたか？
ハブのスピーカー？

デバイスのスピーカー？

'Cat Mwow 1' の部分は、アプリにプリセットされているサウンドのファイル名です。どのような音がプリセットされているのかは、ワードブロックプログラミングの「 音を再生する」ブロックで確認することができます。



ワードブロックプログラミングの「 音を再生する」ブロックでいろいろなファイルの名前を調べて、メモをしておきましょう。
メモをした音を Python プログラムで再生してみましょう。

Pyrhon プログラミングで再生してみたい音ファイル

学んだことを書きましょう

今日はビープ音、プリセットサウンドのプログラミングについて何を学びましたか？

UNIT I Lesson 5
デジタルサイン

ケイトとカイルのためにポップコーンスタンドのデジタル看板を作ろう!!

ケイトとカイルはポップコーンスタンドをはじめましたがお客さんがなかなか来ません。ポップコーンスタンドを目立たせるためのデジタル看板を作ってお客さんにアピールしましょう!

デジタル看板製作の条件

- 看板は自立しません。フレームに取り付ける必要があります。
- 看板はポップコーンを宣伝するものでなければなりません。
- 看板のデザインスケッチを作成すること。
- 看板の動作を説明するために疑似コードを作成すること。
- コード内コメントを使ってコードの各行の説明をすること。

看板のデザインスケッチ

疑似コード

うまくいったところ・苦労したところ

UNIT 2 Lesson 1
モーターで動かす

モーターを動かしてみよう

ラージモーターをポートＡに接続します。コードを入力する前に、疑似コードを書きましょう。

1.

2.

3.

4.

5.

ナレッジベースを開き「スタートガイド」→「４．モーターの制御」と進みます。表示されたプログラムを入力し、実行してみましょう。

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 360 度モーターを回す
    motor.run_for_degrees(port.A, 360, 720)
runloop.run(main())
```

モーターを動かしてみよう

ミドルモーターをポートＢに接続します（モーターは基本的には度のポートに接続しても構いませんが、プログラム内でのポートの記述と一致していないとエラーになります）。２つのモーターを同時に動かすにはどうしたらよいか？プログラムを変えてみましょう。

同時に一回転しましたか？

では、「同時」ではなく、ひとつづつ順番に回転させるにはどうしたらよいと思いますか？次の二つのプログラムを動かしてみて比較してみてください。

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
# 毎秒 720 度の速さで 360 度 2 つのモーターを回す
    motor.run_for_degrees(port.A, 360, 720)
    motor.run_for_degrees(port.B, 360, 720)
runloop.run(main())
```

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 360 度 PortA のモーターを回す
    await motor.run_for_degrees(port.A, 360, 720)
    # 毎秒 720 度の速さで 360 度 PortB のモーターを回す
    await motor.run_for_degrees(port.B, 360, 720)
runloop.run(main())
```

二つのプログラムの違いはどこでしょう？

モーターを動かすコマンドの前に「await」がついていないか、ついているか。

ついていないとすぐに次のコマンドへ移動することで「ほぼ」同時にモーターが回転。「await」がついていると、コマンドが終了するまで（モーターの回転が終わるまで）「待って」から次のコマンドへ移動します。

ダンス・パーティー

ミドルモーター2つを使ってダンスパーティーをしましょう!

モーターのローター（ 回転する部分）の上にダンスさせるものを取り付けて回転ダンスパーティーをします。いろいろな方向、スピードで回転させて楽しいだ明日パーティーにしましょう。

コードを作る前に、疑似コードを書いてどのような動きにさせるかを確認してから、コードを入力、実行しましょう。

ヒント

スピードは、－ 1000 ～1000の間で設定できます。マイナスの値を設定するとどうなるでしょうか?

疑似コード

学んだことを書きましょう

今日はモーター制御のプログラミングについて何を学びましたか?

UNIT 2 Lesson 2
モーターによる新たな動き

1つのモーターを角度で動かす

ラージモーターをポートAに接続します。コードを入力する前に、疑似コードを書きましょう。

1.
2.
3.
4.
5.

ナレッジベースを開き「スタートガイド」→「4. モーターの制御」と進みます。表示されたプログラムを入力し、実行してみましょう。

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 360 度モーターを回す
    await motor.run_for_degrees(port.A, 360, 720)
runloop.run(main())
```

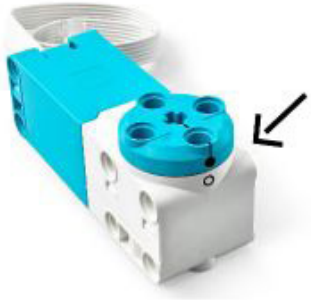
モーターのマーカーを合わせて、0度にセットします。

360よりも小さな値、大きな値、マイナスの値と変化させて、実行してみましょう。

1つのモーターを位置へ動かす

ラージモーターをポートAに接続します。位置0度へ移動するプログラムをコードを入力する前に、疑似コードを書きましょう。

1.
2.
3.
4.
5.



モーターのマーカーを合わせて、0度にセットします。

次のプログラムを入植して、実行してみましょう。どのように動くでしょうか？

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 0 度の位置へ最短経路でモーターを回す
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
runloop.run(main())
```

モーターのマーカーを0度にから少しずらしてセットすると？

次のプログラムを入植して、実行してみましょう。どのように動くでしょうか？

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 0 度の位置へ最短経路でモーターを回したあと、２秒間停止、時計回りで 90 度の位置へ回転
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
    await ruloop.sleep_ms(2000)
    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
runloop.run(main())
```

1. 何が足りない？

次のプログラムは何かが足りないようです、コンソールに表示されたエラーをもとにプログラムを修正しましょう。

```
# モーターとポートをインポート
import motor

import runloop
async def main():
    # 毎秒 720 度の速さで 0 度の位置へ最短経路でモーターを回したあと、２秒間停止、時計回りで 90 度の位置へ回転
    await motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
    await ruloop.sleep_ms(2000)
    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
runloop.run(main())
```

2. どこかの数値が違う？なぜ？

次のプログラムはどこかの数値がちがうようです、プログラムを修正しましょう。

```
# モーターとポートをインポート
import motor
from hub import port
import runloop
async def main():
    # 毎秒 720 度の速さで 0 度の位置へ最短経路でモーターを回したあと、２秒間停止、時計回りで 90 度の位置へ回転
    await motor.run_to_absolute_position(port.A, 0, 7200, direction=motor.SHORTEST_PATH)
    await ruloop.sleep_ms(2000)
    await motor.run_to_absolute_position(port.A, 90, 720, direction=motor.CLOCKWISE)
runloop.run(main())
```

3. 何かが違う？

次のプログラムはどこかが正しくないようです、プログラムを修正しましょう。

```
import motor
from hub import port
import runloop
async def main():
    await Motor.run_to_absolute_position(port.A, 0, 720, direction=motor.SHORTEST_PATH)
runloop.run(main())
```

UNIT 2 Lesson 3
アクションの自動化

アクションの自動化

アームの動きを自動化しましょう。
モーターをどこにつけ、どのように動かせばアームの動きを自動化することができるかよく考えてまずは組み立てのアイデア（ 設計図）を書いてから組み立てましょう。
コードを書く前に疑似コードを作成しましょう。

組み立てのアイデア

疑似コード

コード

うまくいったところ・くろうしたところ・くふうしたところ

UNIT 2 Lesson 4

ホッパーラン

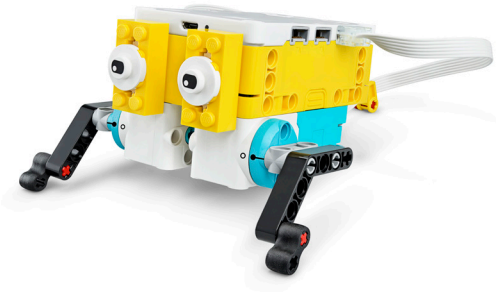
ホッパーの組み立て

ホッパーを組み立てましょう。

車輪以外でホッパーが移動できるようにホッパーの脚をつけましょう。

ホッパーを 5 秒間前進させてください!

まずは。疑似コードを書いてからコードの入力をしましょう。



疑似コード

```
# モーターペアとポートをインポート
from hub import port
import runloop
import motor_pair
async def main():
    #Port E と Port F のモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port_E, port_F )
    #5 秒間、初期設定の速度で直進
    await motor_pair.move_for_time(motor_pair.PAIR_1, 5000, 0)
runloop.run(main())
```

ホッパーが動き出す前に、「 3, 2, 1 」とライトマトリクスでカウントダウンを表示するようにしてみましょう。

疑似コード

```
# モーターペアとポートをインポート
from hub import port
import runloop
import motor_pair
import light_matrix
async def main():
    # カウントダウン
    await light_matrix.write("3")
    await runloop.sleep(1000)
    await light_matrix.write("2")
    await runloop.sleep(1000)
    await light_matrix.write("1")
    await runloop.sleep(1000)
    #Port E と Port F のモーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port_E, port_F )
    #5 秒間、指定の速度（verocity=500）で直進
    await motor_pair.move_for_time(motor_pair.PAIR_1, 5000, 0,
verocity=500)
runloop.run(main())
```

移動の部分を次のプログラムに変えてそれぞれ実行してみましょう、それぞれどのような動きをするかよく観察してください。

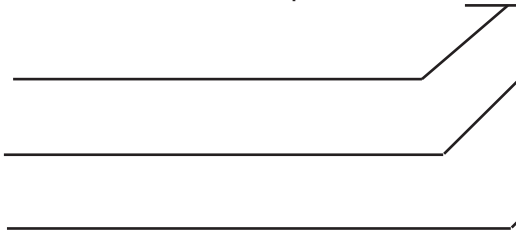
```
await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, -20, velocity=280)

await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, 90, velocity=280)

await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

それぞれのパラメーターの意味はなんでしょう？ 下に記入しましょう。

```
await motor_pair.move_for_time(motor_pair.PAIR_1, 3000, -20, velocity=280)
```



```
await motor_pair.move_tank_for_time(motor_pair.PAIR_1, 500, 1000, 3000)
```

学んだことを書きましょう

今日はモーター制御のプログラミングについて何を学びましたか？

UNIT 3 Lesson 1
センシング開始

モーターをポートA、フォースセンサーをポートBに接続します。

フォースセンサーを押したら、モーターが回転し、離したら止まるようにプログラムをしたい。まずは疑似コードでどのようなステップのプログラムにすればよいか、書きましょう。

疑似コード

```
# モーターとフォースセンサー、ポートをインポート
from hub import port
import runloop
import motor, force_sensor
# フォースセンサーが押されている
def is_force_sensor_pressed():
    return force_sensor.pressed(port.B)
# フォースセンサーが押されていない
def is_force_sensor_not_pressed():
    return not force_sensor.pressed(port.B)
async def main():
    # フォースセンサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)
    motor.run(port.A, 750)
    # フォースセンサーが解放されるまで待つ
    await runloop.until(is_force_not_pressed)
    motor.stop(port.A)
runloop.run(main())
```

```
# フォースセンサーが押されている
def is_force_sensor_pressed():
    return force_sensor.pressed(port.B)
# フォースセンサーが押されていない
def is_force_sensor_not_pressed():
    return not force_sensor.pressed(port.B)
```

関数

このような、特定の機能をプログラムすることを「関数を定義する」と言います。このように定義した関数はメインのプログラムから呼び出して使うことができます。

`await runloop.until(is_force_sensor_pressed)`

の部分のように。

それでは、このプログラムを次のように変えてみてください。

「プログラムを実行するとモーターが回転しはじめ、フォースセンサーを押すと止まる。」

それでは、このプログラムを次のように変えてみてください。

「 プログラムを実行するとモーターが回転しはじめ、フォースセンサーを押すと止まる。」

疑似コード

```
# モーターとフォースセンサー、ポートをインポート
from hub import port
import runloop
import motor, force_sensor
# フォースセンサーが押されている
def is_force_sensor_pressed():
    return force_sensor.pressed(port.B)
async def main():
    motor.run(port.A, 750)
    # フォースセンサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)
    motor.stop(port.A)
runloop.run(main())
```

コンソールはエラーが表示されるだけではなく、プログラム内からメッセージを表示することもできます。

次のプログラムを入力して実行してみましょう。実行するときはコンソールを開いておくのを忘れずに。

```
# モーターとフォースセンサー、ポートをインポート
from hub import port
import runloop
import motor, force_sensor
# フォースセンサーが押されている
def is_force_sensor_pressed():
    return force_sensor.pressed(port.B)
async def main():
    motor.run(port.A, 750)
    # フォースセンサーが押されるまで待つ
    await runloop.until(is_force_sensor_pressed)
    motor.stop(port.A)
    # コンソールに Hello と表示する
    print('Hello')
runloop.run(main())
```

以下の文章の空欄を埋めましょう

print() 関数は、(　　) に、任意の (　　) を表示することができる。

表示したい (　　) は、print() 関数の () 内に’’ か”” ではさんで指定することができる。

学んだことを書きましょう

今日はセンサーのプログラミングについて何を学びましたか？ その他にも学んだことはありますか？

UNIT 3 Lesson 2
突進するサイ

次のプログラムを入力して実行してみましょう。サイはどうなるでしょうか？

```
# モーターペア、ポートをインポート
from hub import port
import runloop
import motor_pair

async def main():
    # 左（ポート B）ろ右（ポート A）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, portA)

    # 前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)
runloop.run(main())
```

サイは、

_____

カセンサーを使って、サイが壁にぶつかったら止まるようにしましょう。

```
# モーターペア、ポート、力センサーをインポート
from hub import port
import runloop
import motor_pair
import force_sensor

def is_force_sensor_pressed():
    # 力センサーの入力（押されている）を収集
    return force_sensor.pressed(port.E)

async def main():
    # 左（ポート B）ろ右（ポート A）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, portA)

    # 前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=750)

    # 力センサーが押されるまで待つ
    await ruloop.untill(is_force_senso_pressed)

    # モーターを止める
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

速度を変えて試してみましょう。速度を1000にするとどうなるでしょうか？ 250 にするとどうなるでしょうか？

```
# モーターペア、ポート、力センサーをインポート
from hub import port
import runloop
import motor_pair
import force_sensor

def is_force_sensor_pressed():
    # 力センサーの入力（押されている）を収集
    return force_sensor.pressed(port.E)

async def main():
    # 左（ポート B）ろ右（ポート A）をペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, portA)

    # 早い速度で前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=1000)

    # 力センサーが押されるまで待つ
    await ruloop.untill(is_force_senso_pressed)

    # モーターを止める
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```

```
    # 遅い速度で前進
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=250)
```

学んだことを書きましょう

今日はセンサーのプログラミングについて何を学びましたか？ その他にも学んだことはありますか？

UNIT 3 Lesson 3
カート・コントロール

次のプログラムをプログラムキャンパスに入力して、実行してみましょう。何が起こるかよく観察してください。

```
# モーターペアとディスタンスセンサー、ポートをインポート
from hub import port
import runloop
import motor_pair
import distance_sensor

def dustance_sensor_grater_than():
    # 距離センサーの値が 150 mmより大きいかどうかを返す
    distance = distance_sensor.distance(port.B)
    return distance == -1 or distance < 150

def dustance_sensor_less_than():
    # 距離センサーの値が 100 mmより小さいかどうかを返す
    distance = distance_sensor.distance(port.B)
    return distance < 100 and distance > -1

async def main():
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)
    # 距離センサーの値が 150 mmより大きくなるまで待つ
    await runloop.until(distance_sensor_grater_than)
    # モーター ON
    motor_pair.move(motor_pair.PAIR_1, 0, verocity = 500)
    # 距離センサーの値が 100 mmより小さくなるまで待つ
    await runloop.until(distance_sensor_less_than)
    # モーター STOP
    motor_pair.stop(motor_pair.PAIR_1)
runloop.run(main())
```

プログラムを実行した結果どんな動きになりましたか？

では、このプログラムを次のような動きに変えてみましょう。

「 プログラムを実行すると、まっすぐに進む為に後輪をまっすぐの位置（ 角度0度）に直し、動き出し（ 前進）し、距離センサーの値が150mm以下になったら止まる」

デバッグアクティビティー

次の2つのプログラムには誤りがあります、実行したときにコンソールに表示されるエラーを参照してエラーを正し(デバッグし) ましょう。

#Program 1

```
# モーターペアとディスタンスセンサー、ポートをインポート
from hub import port
import runloop
import motor_pair
import motor

def dustance_sensor_closer_than():
    # 距離センサーの値が 150 mmより小さいかどうかを返す
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)
    # 後輪モーターをまっすぐに進むように 0 度にする
    await motor.run_to_absolute_position(port.C, 0, 250)
    # モーター ON（前進）
    motor_pair.move(motor_pair.PAIR_1, 0, verocity = 200)
    # 距離センサーの値が 150 mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    # モーター STOP
    motor_pair.stop(motor_pair.PAIR_1)
runloop.run(main())
```

#Program 2

```
# モーターペアとモーター、ディスタンスセンサー、ポートをインポート
from hub import port
import runloop
import motor_pair
import motor
import distance_sensor

def dustance_sensor_closer_than():
    # 距離センサーの値が 150 mmより小さいかどうかを返す
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    # 後輪モーターをまっすぐに進むように 0 度にする
    await motor.run_to_absolute_position(port.C, 0, 250)
    # モーター ON（前進）
    motor_pair.move(motor_pair.PAIR_1, 0, verocity = 200)
    # 距離センサーの値が 150 mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    # モーター STOP
    motor_pair.stop(motor_pair.PAIR_1)
runloop.run(main())
```

UNIT 3 Lesson 4

安全な配送

次のプログラムを入力し、速度を「200」、「400」、「600」、「800」、「1000」と変更しながら、障害物を発見（距離センサーが150 mmより小さくなる）して停止した時の距離を計測して速度による違いを調べてみましょう。

```
# モーターペアとディスタンスセンサー、ポートをインポート
from hub import port
import runloop
import motor_pair
import motor
import distance_sensor

def dustance_sensor_closer_than():
    # 距離センサーの値が 150 mmより小さいかどうかを返す
    distance = distance_sensor.distance(port.B)
    return distance < 150 and distance > -1

async def main():
    # モーターをペアにする
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.A)
    # 後輪モーターをまっすぐに進むように 0 度にする
    await motor.run_to_absolute_position(port.C, 0, 250)
    # モーター ON（前進）
    motor_pair.move(motor_pair.PAIR_1, 0, verocity = 200)
    # 距離センサーの値が 150 mmより小さくなるまで待つ
    await runloop.until(distance_sensor_closer_than)
    # モーター STOP
    motor_pair.stop(motor_pair.PAIR_1)
runloop.run(main())
```

速　度	200	400	600	800	1000
距離センサーの値					

上の表からわかることは何ですか？

公園内を物にぶつからずに安全に資材を配送するためのプログラムを作成しましょう。まずは、疑似コードを作り、その疑似コードをもとにプログラムをコーディングしていきましょう。

疑似コード

プログラム

To Be Continued in Part 2

Unit 4 Loops & Variables

Unit 5 Conditions for Games

Unit 6 Troubleshooting and Debugging

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 1

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/


Learning
Systems™