

遊びで学んで  
デジタルコンテンツ  
の裏側を見てみよう



# まずは遊んで見よう！

準備する事

1. micro:bitをPCをUSBケーブルで繋ぐ
2. micro:bitに「プログラムファイル」を入れる
3. gameを開始する。





Aボタンで上に  
Bボタンで下に動きます。  
光の玉を沢山の集めましょう。

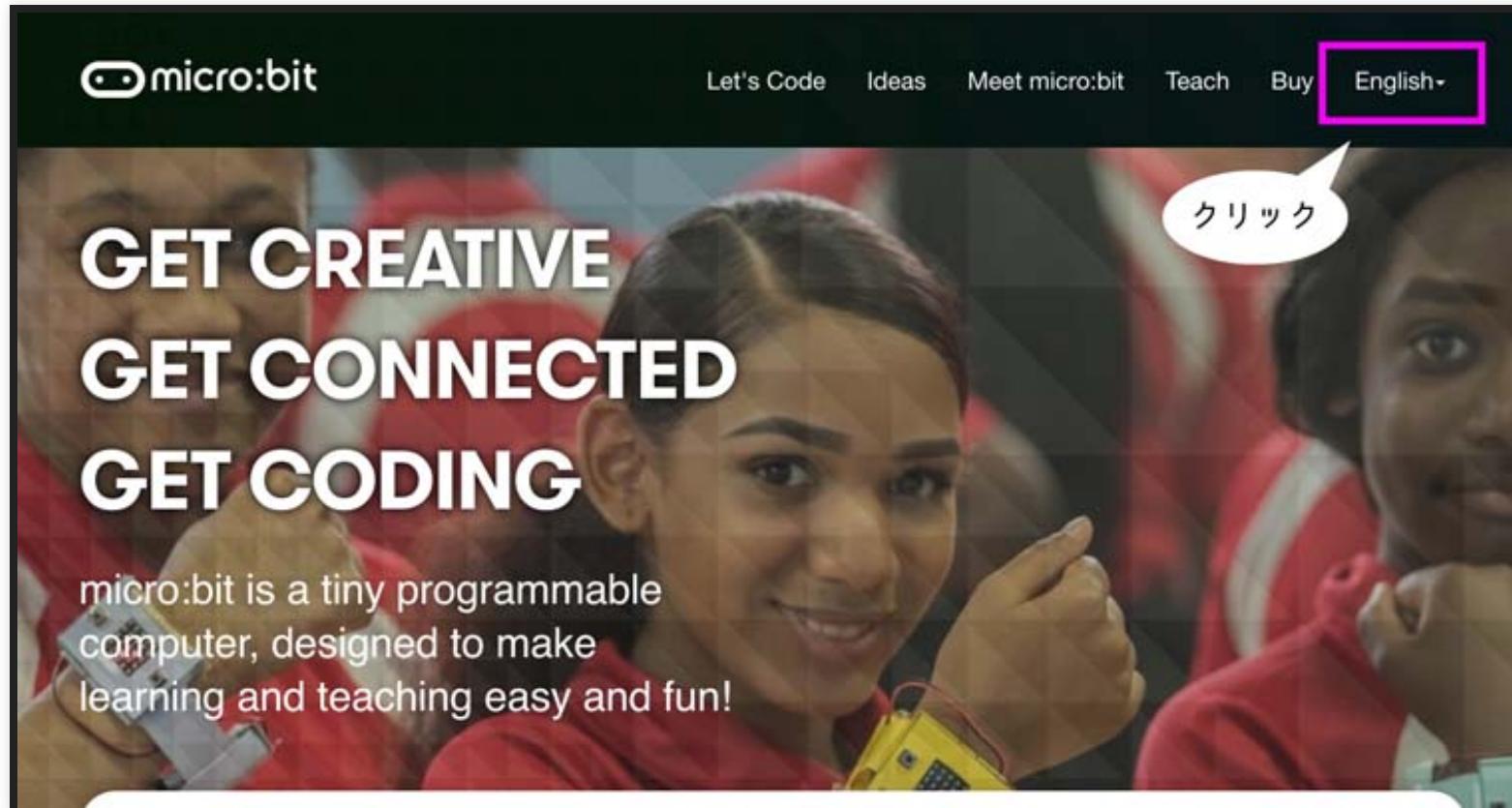
魚に当たったら GameOver !  
RELEASEPOWER が表じされ  
ている時に A+B ボタンを押し  
ながら、光センサーを隠すと  
...

早速！Micro:bitでプログラムをして見よう！

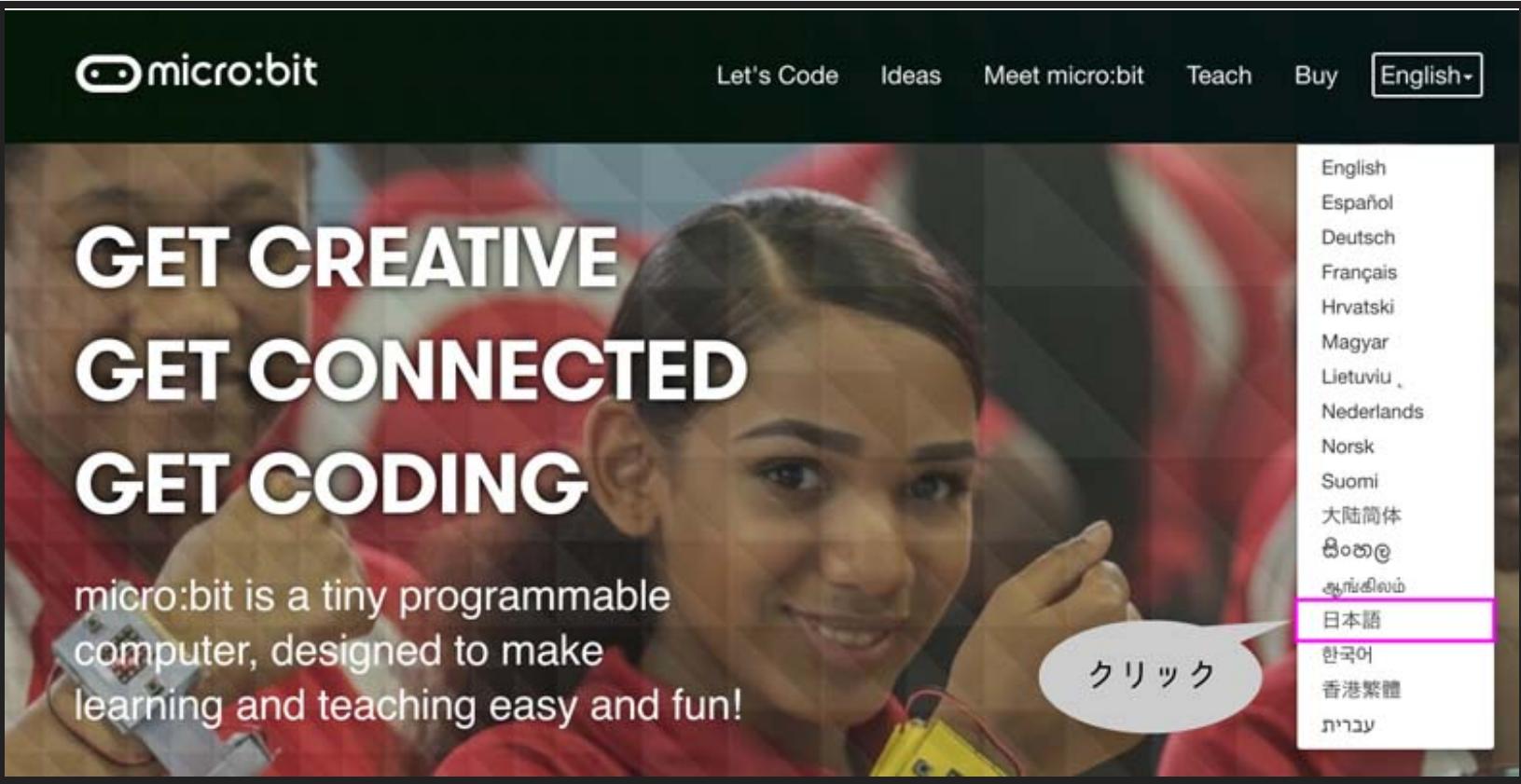
Micro:bit



# チュートリアル TUTORIAL(まずははじめに使い方の説明)



[HTTP://MICROBIT.ORG/](http://microbit.org/)



The screenshot shows the official micro:bit website homepage. At the top left is the micro:bit logo. To its right are navigation links: Let's Code, Ideas, Meet micro:bit, Teach, Buy, and English-. A dropdown menu for language selection is open, listing various languages including English, Español, Deutsch, Français, Hrvatski, Magyar, Lietuvių, Nederlands, Norsk, Suomi, 大陸簡體, ຜິ່ນໜາ, മലയാളം, 日本語 (which is highlighted with a pink border), 한국어, 香港繁體, and עברית.

**GET CREATIVE  
GET CONNECTED  
GET CODING**

micro:bit is a tiny programmable computer, designed to make learning and teaching easy and fun!

クリック







クリック

## JavaScript ブロックエディタ

マイクロビット JavaScript ブロックエディターを使えば、BBC micro:bit を、ブロックまたは JavaScript で簡単にプログラミングできます。

MakeCode で開発。もしアクセスに問題があるなら、あなたの学校でエディターが ブロック されていないか確認してください。アイデアやひらめきが必要なら、これらの プロジェクト を参考にしてください。

[プログラムしましょう](#)

[リファレンス](#)

[Lessons](#)



きほん  
「基本」をクリックしましょう。



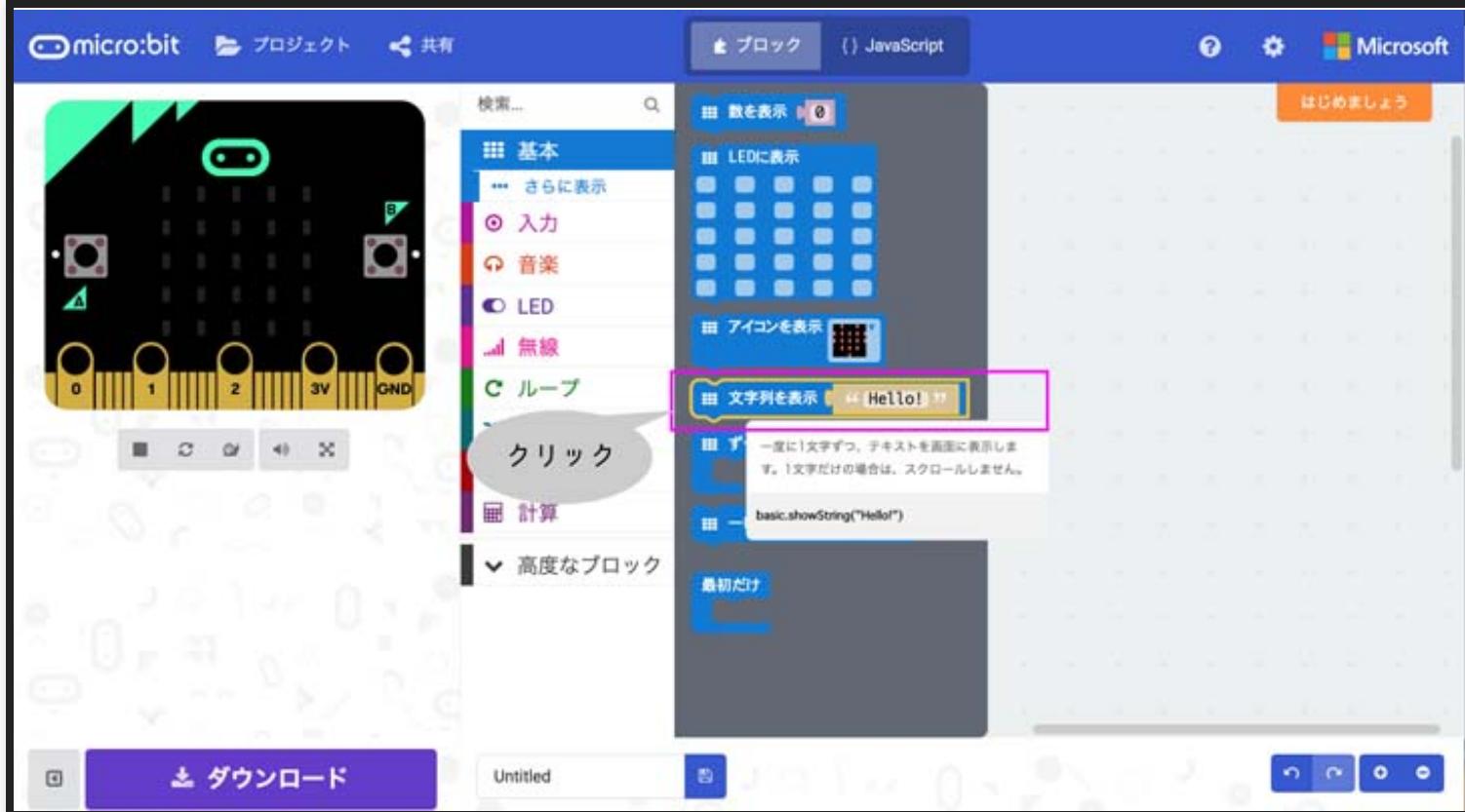
さいしょ

「最初だけ」をクリックしましょう。



きほん

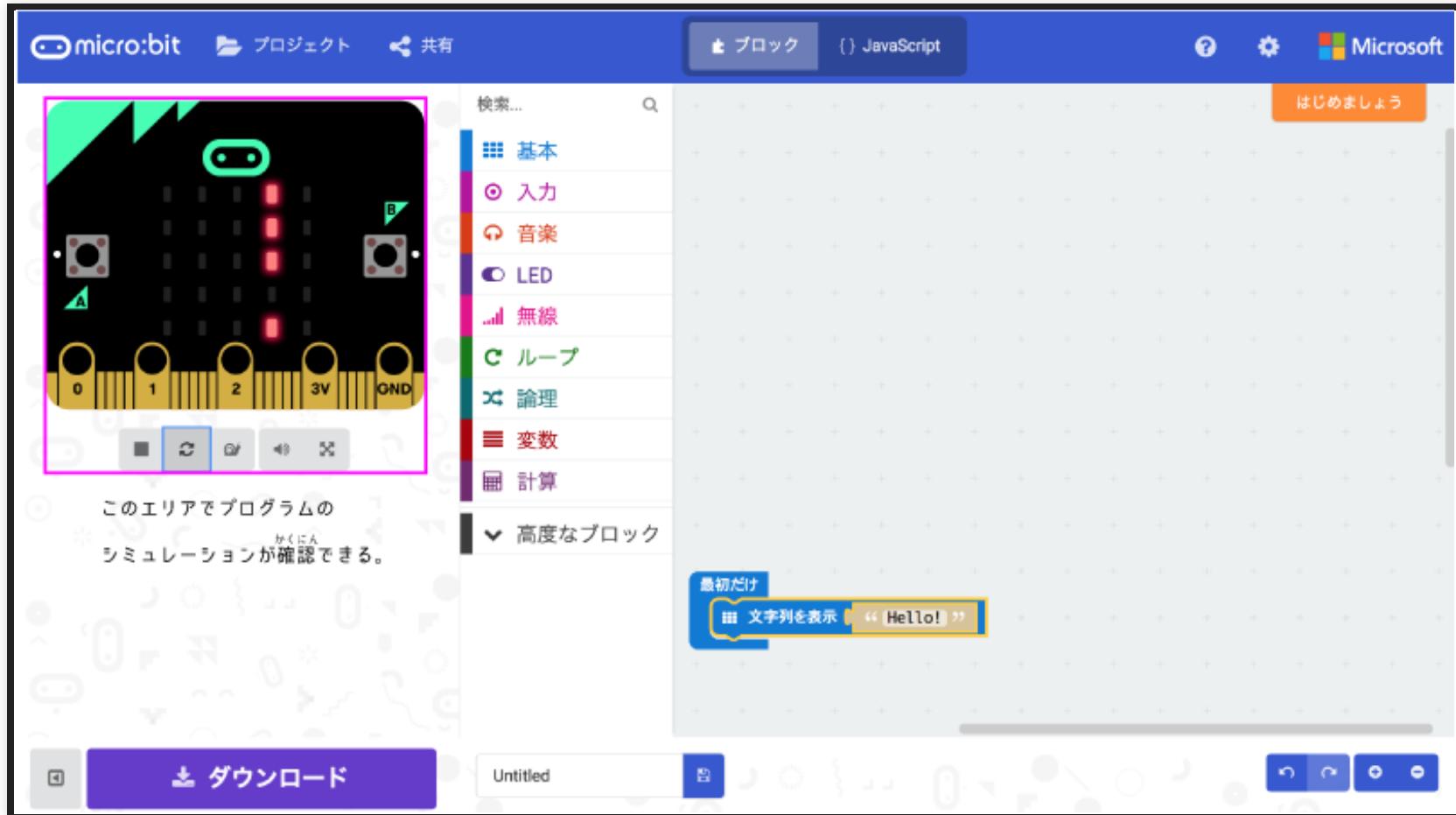
次に、また「基本」をクリックしましょう。



もじれつひょうじ  
「文字列を表示」をクリックしましょう。



もじれつひょうじ  
「文字列を表示」をドラッグして「最初だけ」の  
中に入れよう。 さいしょ



一回だけ「HELLO!」という文字が表示されること  
かくにん  
を左のシミュレーターで確認しよう。



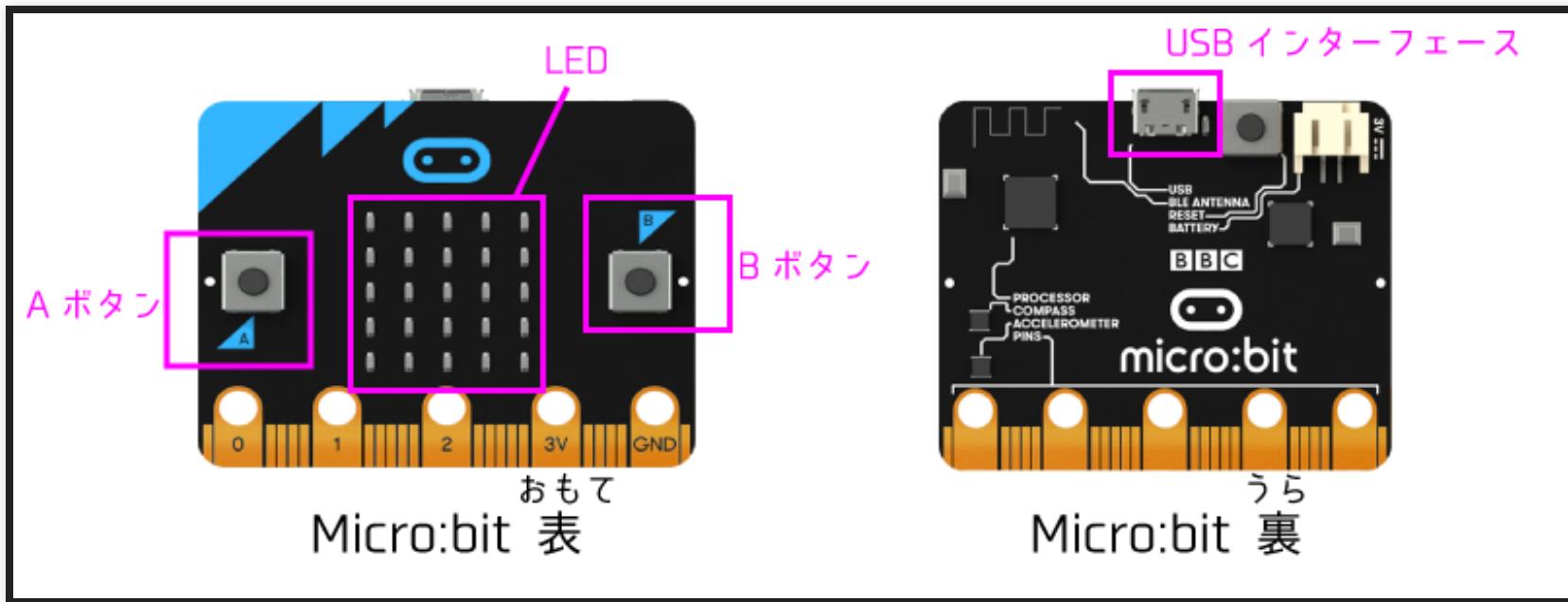
左のエリアにドラッグしてブロックを削除しよう。

レッスン  
**LESSON**

そうさほうほう りかい  
操作方法は理解しましたか？

理解ができたところで、実際にmicrobitを自分の  
思い通りに動かせるようにプログラムを組んで  
いきましょう！

# ハードウェアの紹介 :MICRO:BIT



にゅうりょく しゅつりょく そな  
マイクロコンピュータと入力や 出力を 備えた  
きばん  
基盤です。

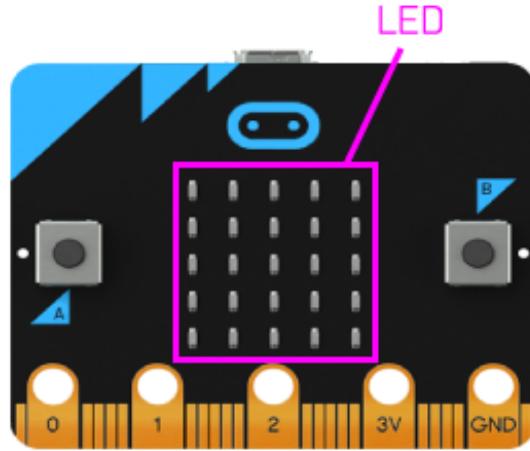
とりあつか

ちゅういてん

# 取扱いの注意点！ 濡れた手でさわらない



マイクロビット  
MICRO:BITの 機能紹介 LED



エル イー ディー はっこう  
LED :は 発行ダイオード(LIGHT EMITTING DIODE)  
という意味です。MICRO:BITには25個のLEDがあり、  
これを使って数字や文字を表現します。

# LESSON1 「出力」

Micro:bitのLEDを自由に表示できるようにしよう。



# LESSON1 の目的

- LEDの表示を自分の意図した通りに表現する。
- 「最初だけ」 「ずっと」 「ミリ秒」 を理解する。
- プログラムをmicro:bitに入れて動かす。

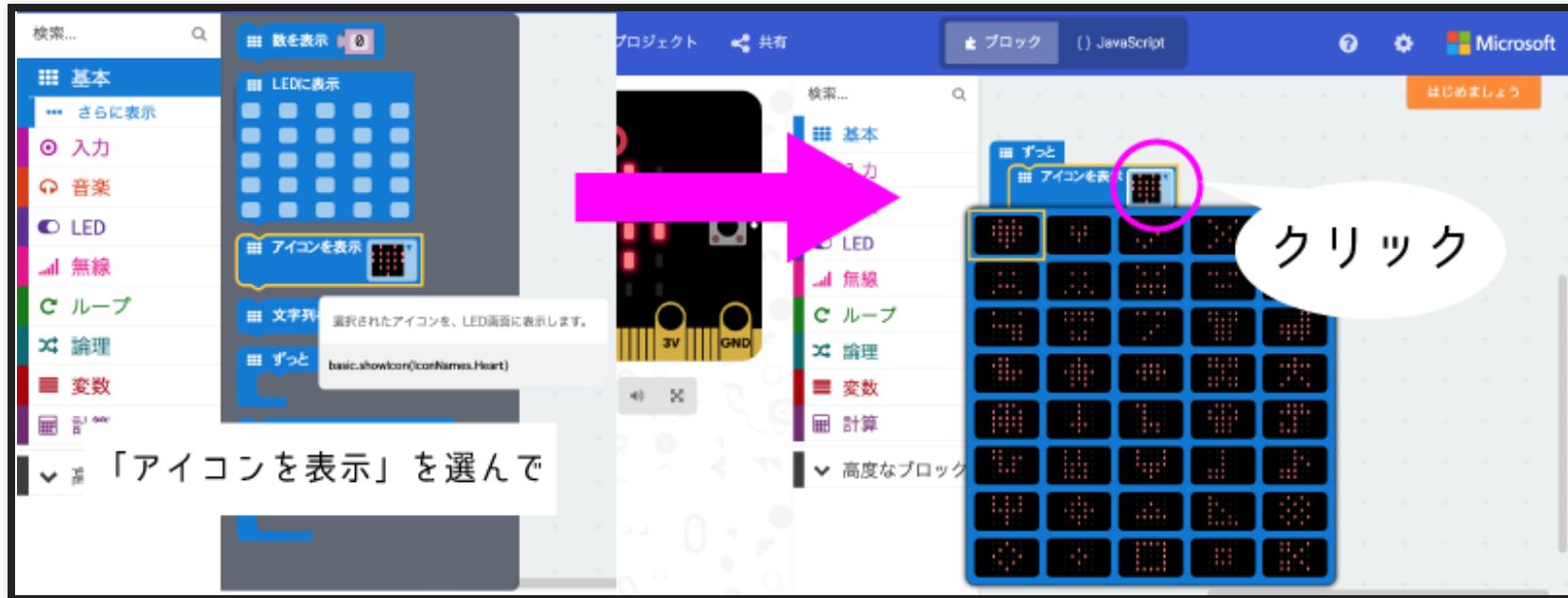
≡

# MICRO:BITのLEDを制御して色々な表示をしよう



えら  
「基本」→「ずっと」を選んでクリックします。

# アイコンを表示しよう



「アイコンを表示」を選んで「ずっと」ブロックの中に入れ、アイコンをクリックして好きなアイコンを表示するプログラムを作りましょう。

# プログラムとは

プログラムは、コンピュータを動かす為に「いつ」「どうする」を書いた命令書みたいなものだよ。

## 「いつ」を表すブロック

最初だけ

最初だけ  
(この中にある「どうする」)  
を動かしてね。

ずっと

ずっと  
(この中にある「どうする」)  
をくりかえし動かしてね。

## 「どうする」を表すブロック

アイコンを表示



選んだアイコンを表示してね

一時停止(ミリ秒)

100

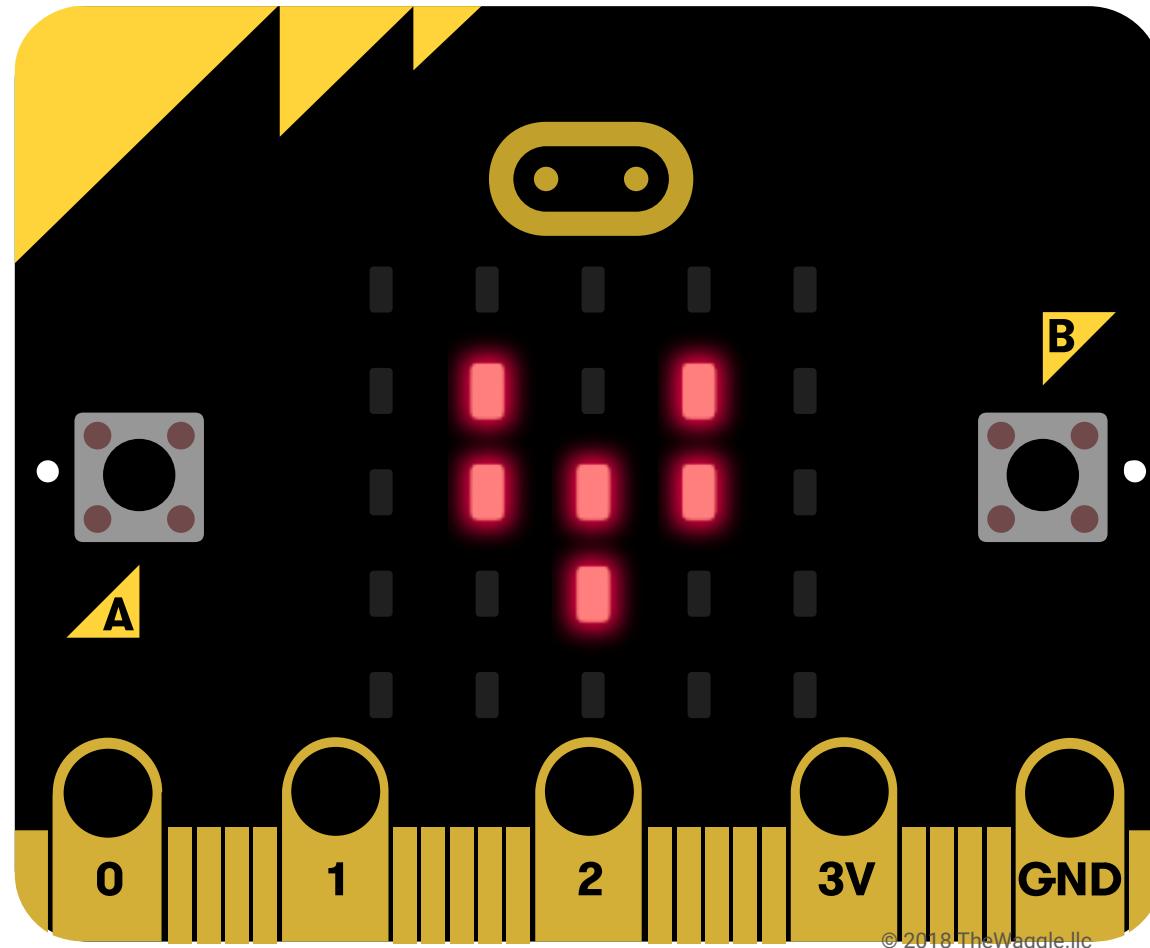
100ミリ秒動くの  
止めてね

+

ブロックは「いつ」 + 「どうする」の組み合わせで命令を作る

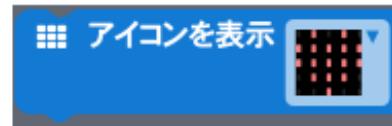
# やってみよう!

ハートがドキドキするアニメーションを作ってみよう！



# ヒント

## 利用するブロックの種類



大きいハートのアイコン

小さいハートのアイコン

# プログラムに名前をつけよう。



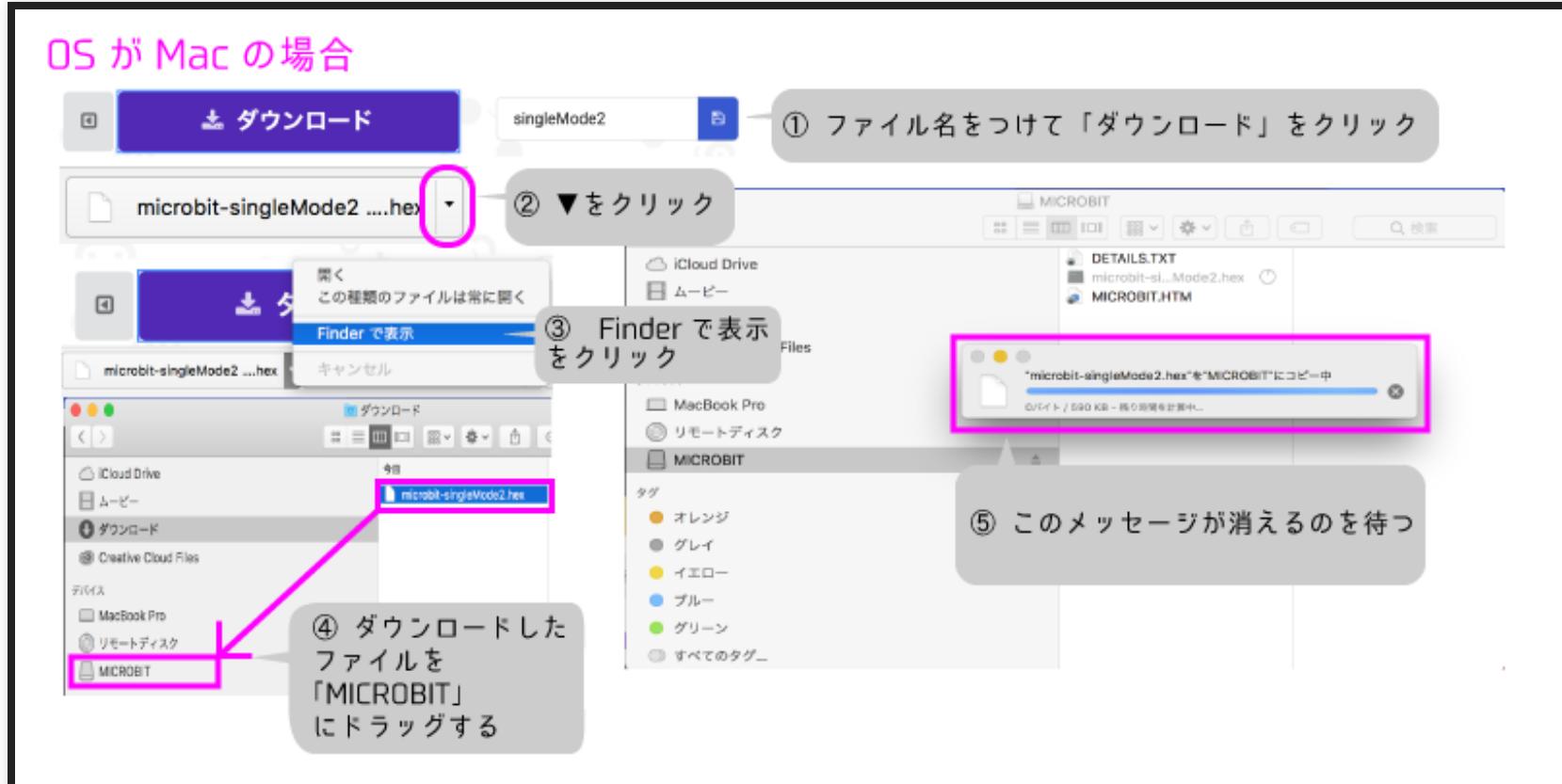
- 「クリック」
- 「delete」キーで「題名未設定」を削除
- 「Heart」と入力します。

# プログラムを保存しよう。

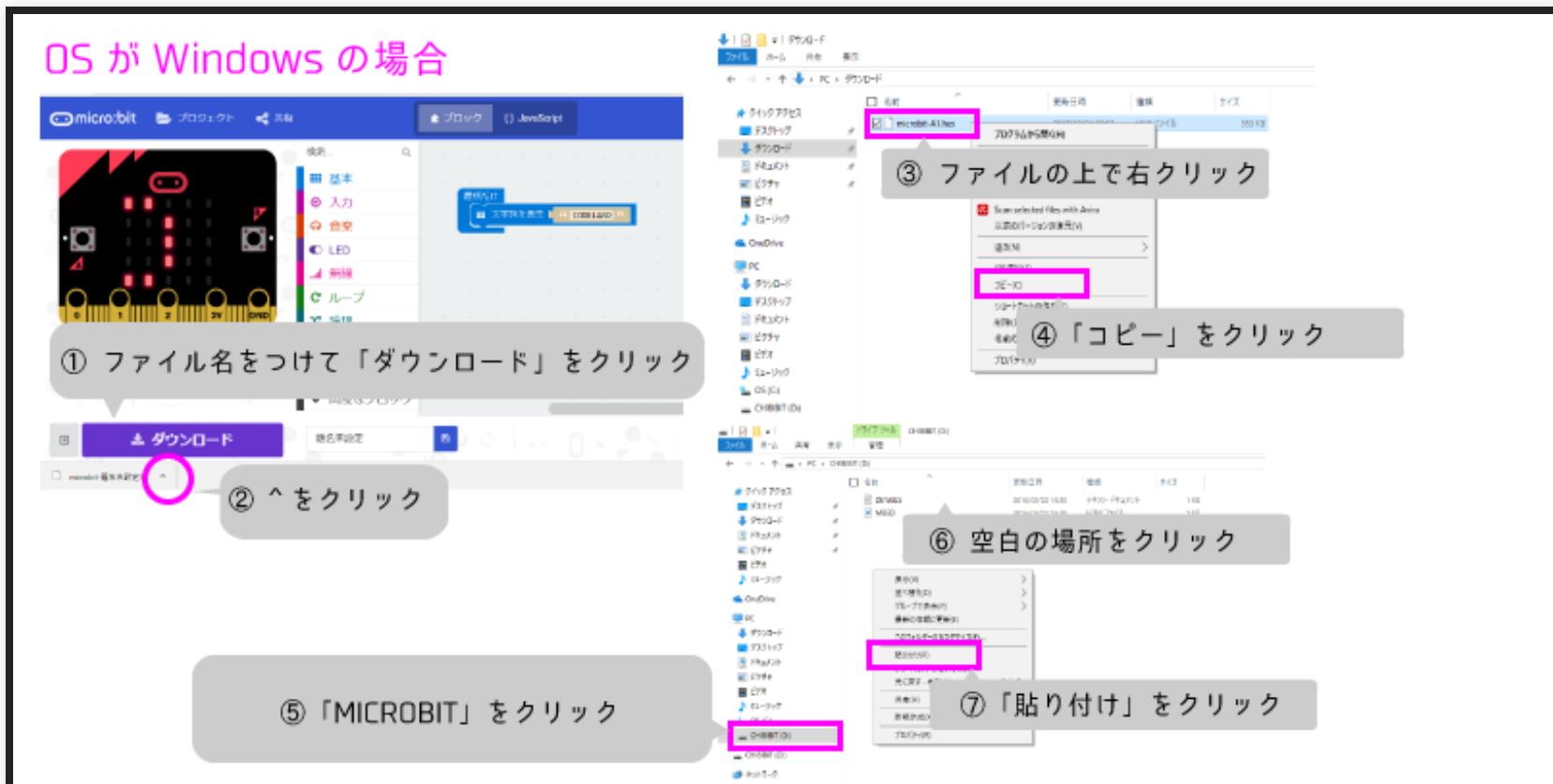


- 上をクリックするとプログラムファイルが  
ダウンロードフォルダに保存される。

# プログラムをMICRO:BITに入れてみよう！MACの場合



# プログラムをMICRO:BITに入れてみよう！ WINDOWSの場合



## LESSON2 [入力]

Micro:bitのLEDに色々な表示ができるようになりましたか？  
LESSON1では、Micro:bitの出力を変更する事をやりました。

それでは、続いてMicro:bitの「入力」を使って  
「出力」する物を変える事をやって見ましょう。

# LESSON2 の目的

- 入力条件について理解する
- 変数の使い方が分かる

≡

# ボタンを使って、表示する数値を 変えてみよう。



# Aボタンを押すと「0を表示する」を作る



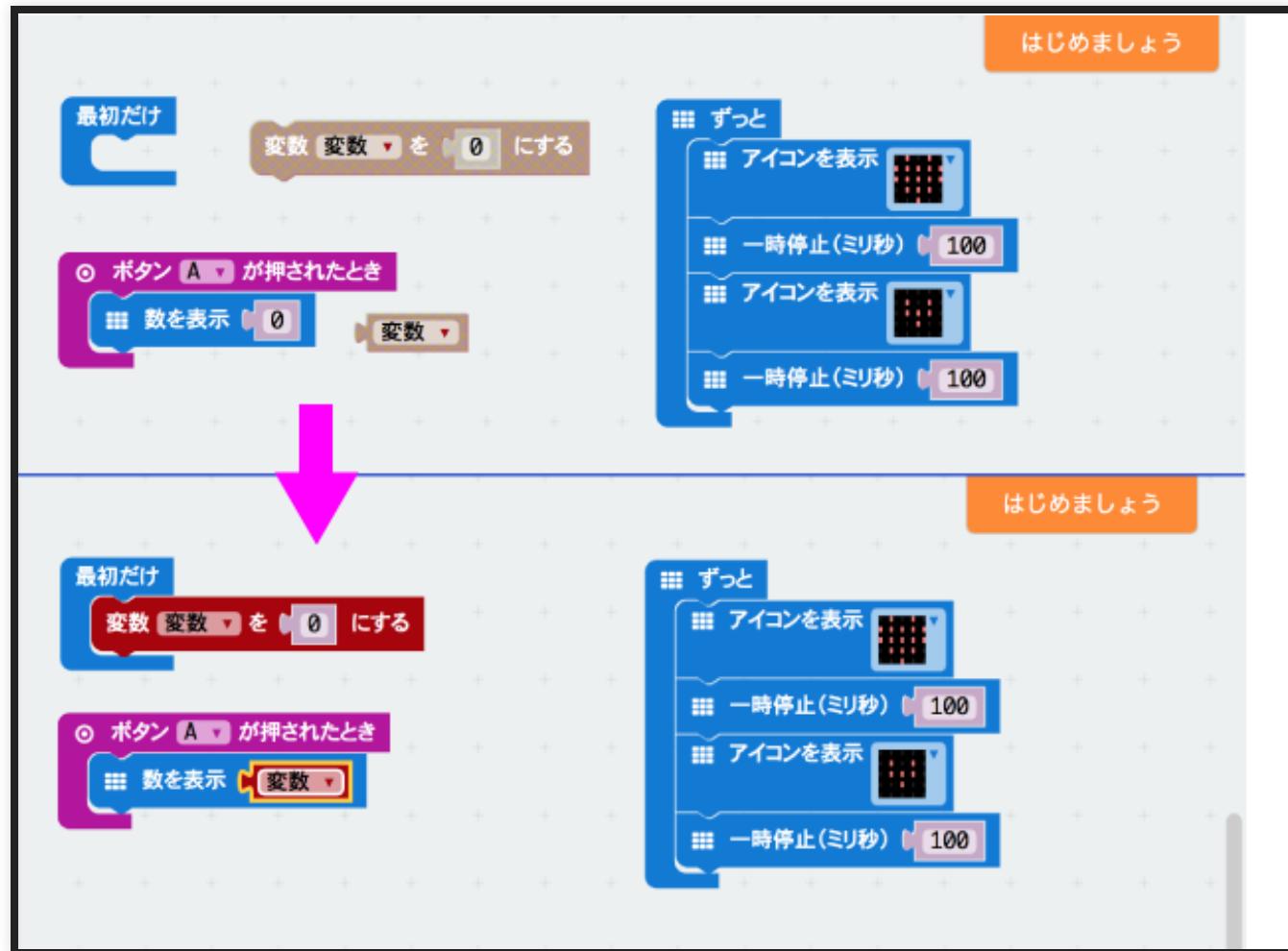
# 変数を使ってみよう[変数]をクリック



# 「変数を 0 にする」をクリック



# 「変数」を使って数字を表示しよう



# Aボタンを押して「0」が表示された



# 変数のイメージ

「変数」はこんなイメージで使います。



「変数」という名前のカラッポの箱を作りました。



「変数」に「0」を入れる

最初だけ

変数 **変数** を **0** にする



「変数」に1を足す

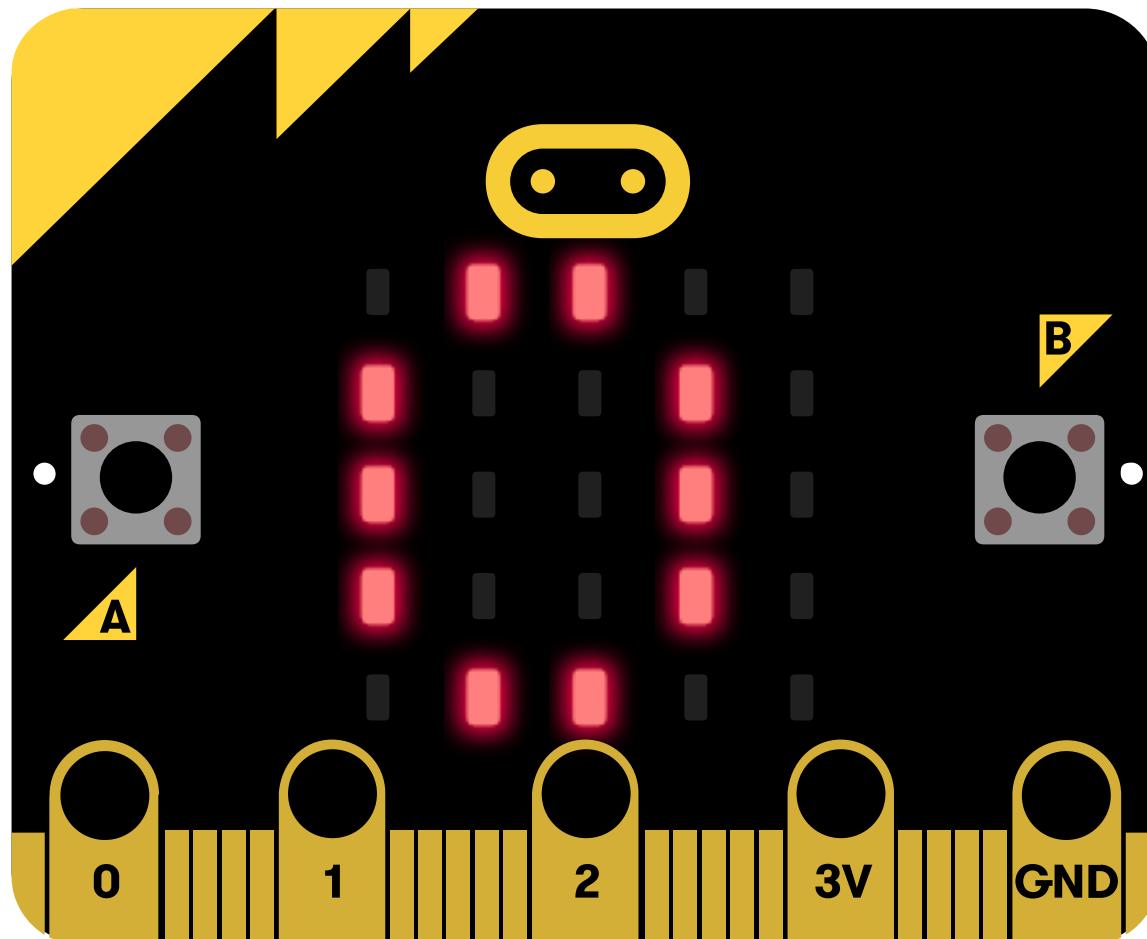
変数 **変数** を **+1** だけ増やす

# Bボタンを押して表示する数字を増やそう



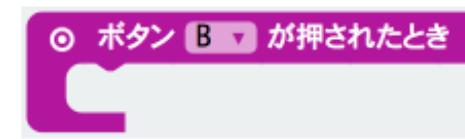
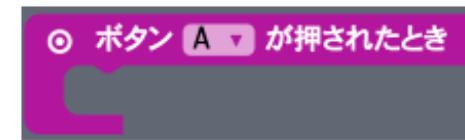
# やってみよう！

Aボタンを押すと、数字が増えて Bボタンを押すと、  
数字が0になる



# ヒント

## 利用するブロックの種類



## LESSON3 [制御]

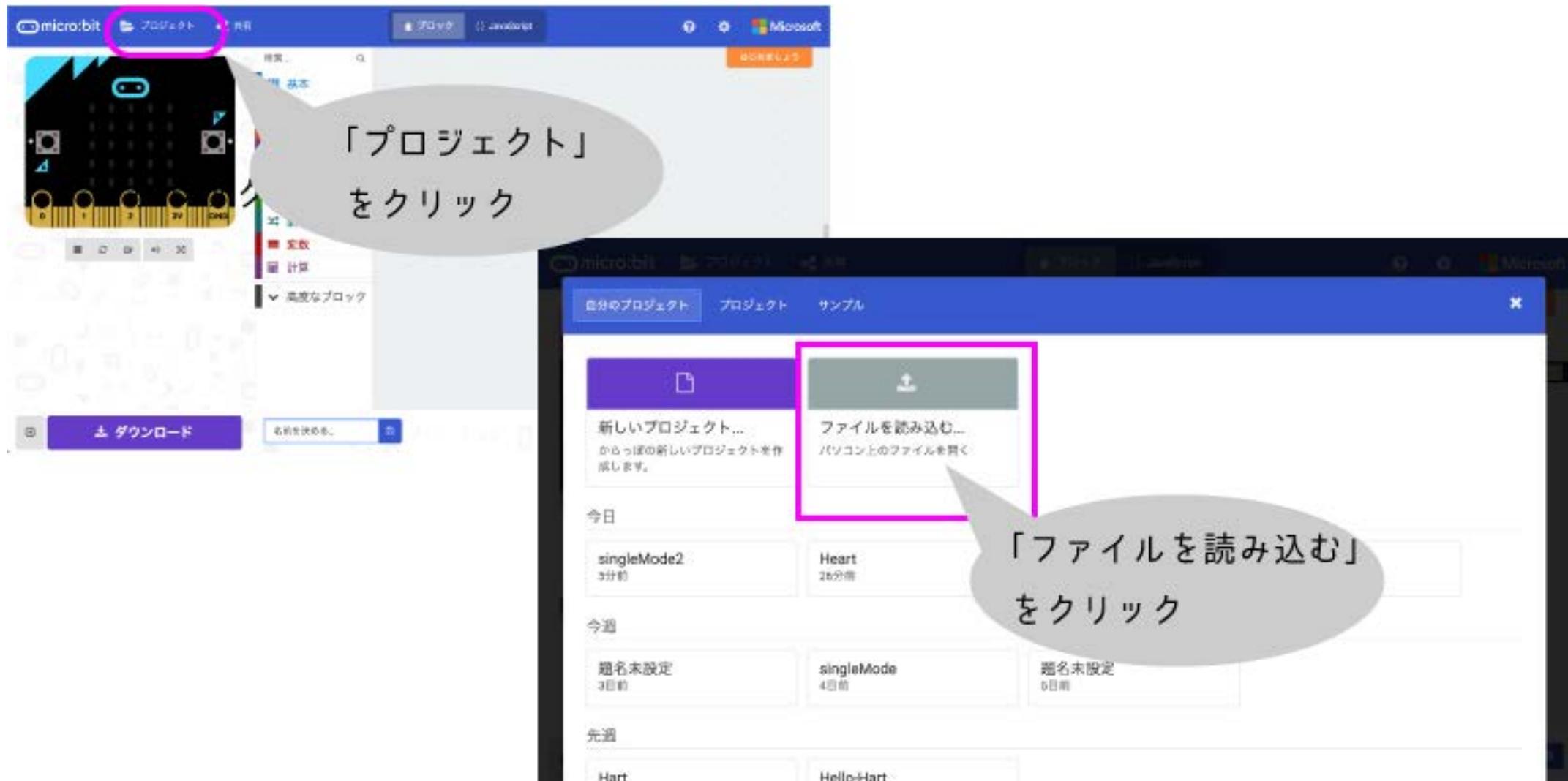
それでは、いよいよ、プログラムの制御について学びます。

実際に体験したゲームコントローラーのプログラムを改造しながら、学んで行きましょう。

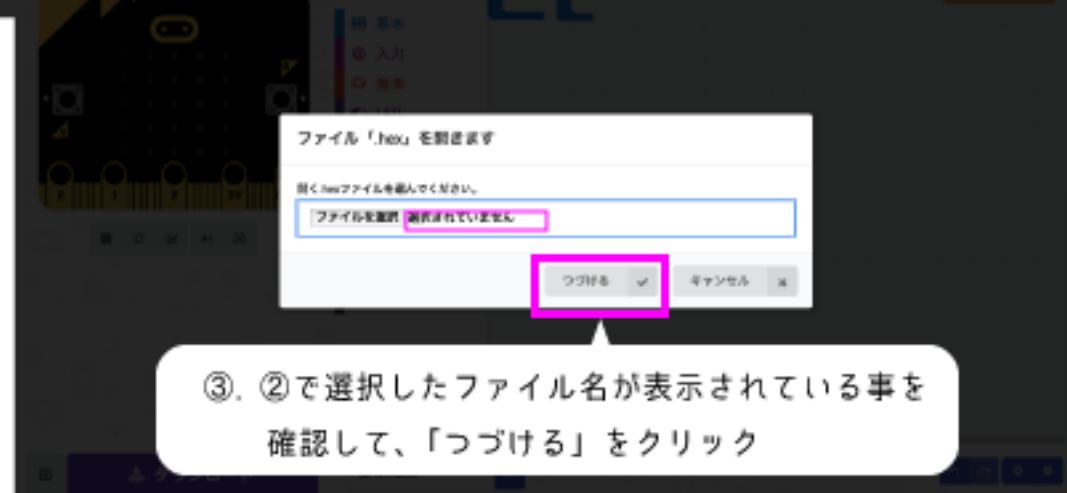
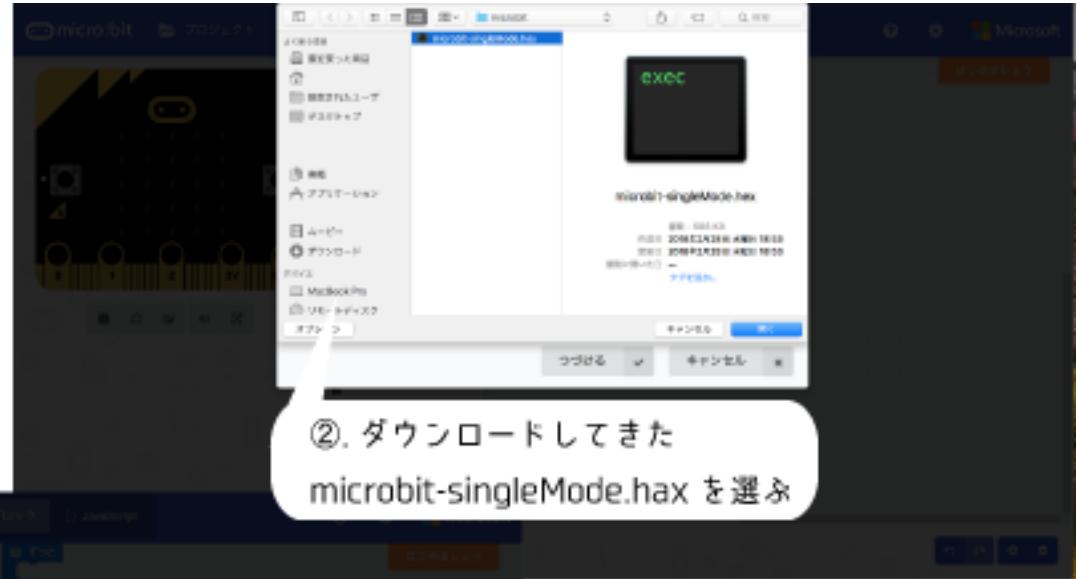
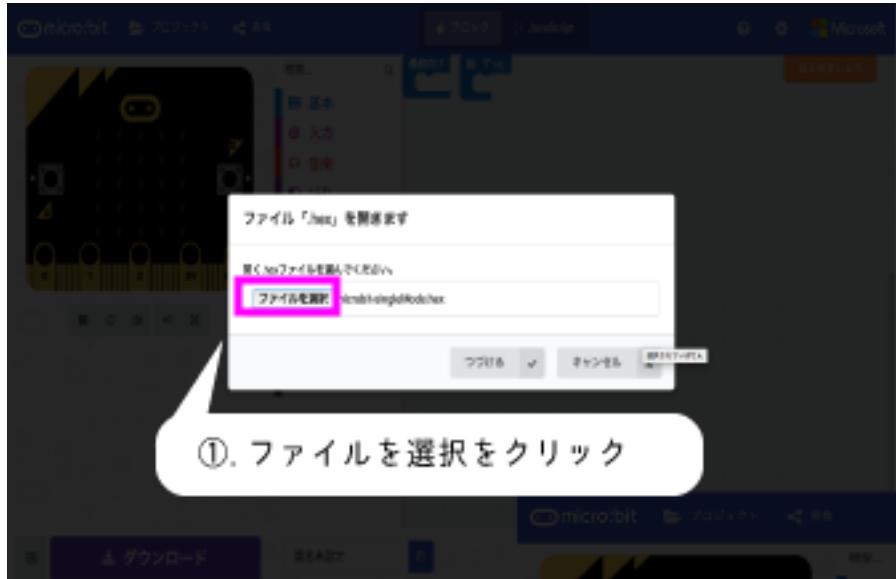
# LESSON3 の目的

- 条件分岐を理解する
- 複数の変数を利用する

# プログラムの読み込み



# プログラムを選択する



# ボタンの機能を変更しよう

The image shows the Microsoft MakeCode editor interface for the micro:bit. On the left, there's a virtual representation of the micro:bit board with its components labeled: A, B, 0, 1, 2, 3V, GND, and a central circular button labeled A+B. Below the board is a purple button labeled "データを表示 シミュレーター". At the bottom left is a purple "ダウンロード" button. In the center, the code editor displays several Scratch-style blocks:

- A green "basic" category block: "アイコンを表示 [ ]"
- A blue "control" category block: "ずっと [もし [もし [A+B] 按されたと/or B 按されたと/or 3V が接続されたとき] なら [LSPlay > 1] でなければ [アイコンを表示 [ ]] [アイコンを表示 [ ]]]"
- A blue "control" category block: "シリアル通信 つぎのいずれかの文字を受信したとき [シャープ(#)]" followed by a red "data" category block: "数値 [lightVol] を [明るさ] にする"
- A pink "control" category block: "ボタン A+B が按されたとき [実数 [lightVol] を [明るさ] にする] [もし [A+B] 按されたと/or B 按されたと/or 3V が接続されたとき] なら [シリアル通信 1行書き出す [hide=0]] でなければ [シリアル通信 1行書き出す [hide=1]]]" which is highlighted with a pink border.

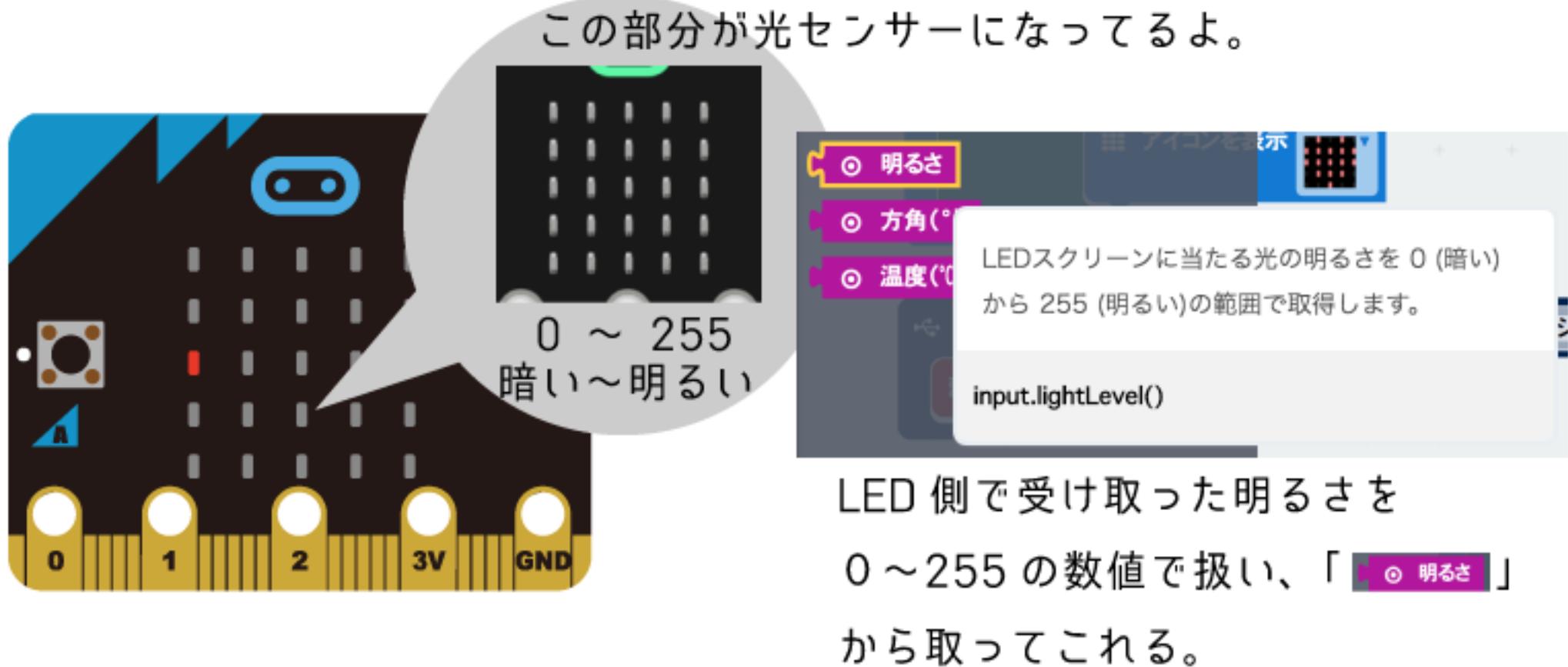
At the bottom right, a gray callout bubble contains the text: "ボタンの機能をプログラムしている部分をみてみよう。"

# A+Bボタンの中を見てみよう



[`lightVal`]という変数に「明るさ」の中の数値を入れている。

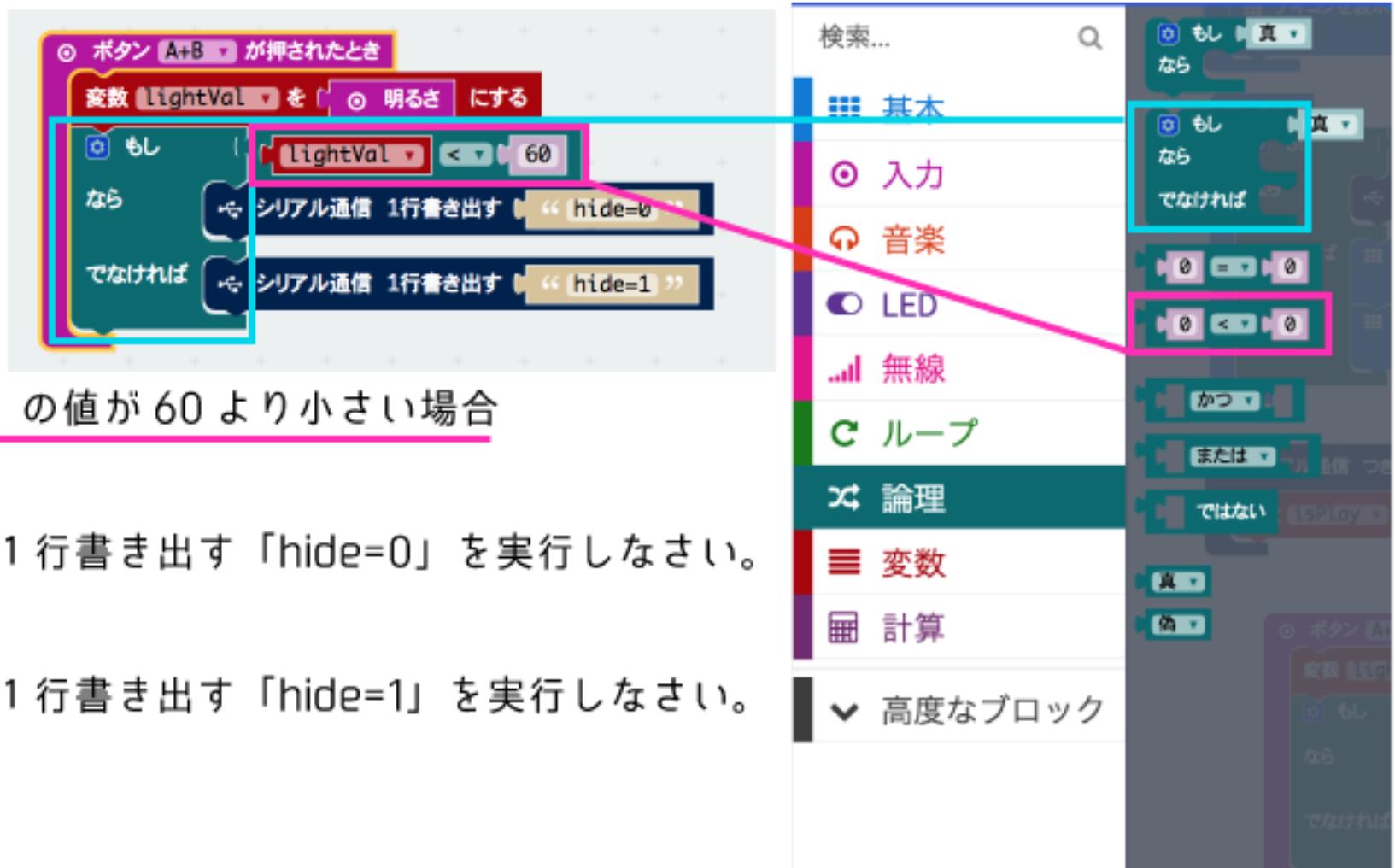
# 「明るさ」の仕組み



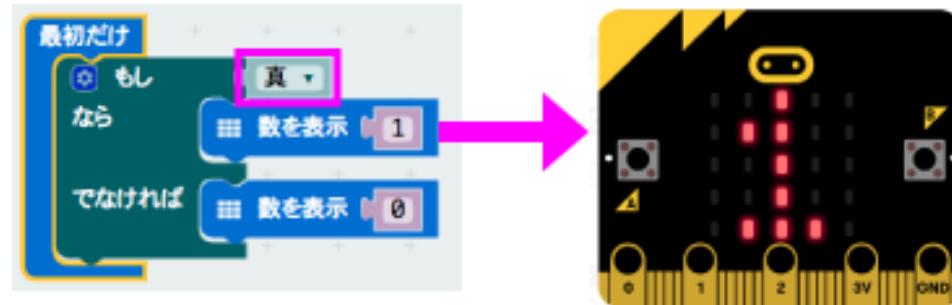
# 条件分岐を使ってみよう

## じょうけんぶんき 条件分岐

もし「lightVal」の値が 60 より小さい場合  
が成り立つなら  
 シリアル通信 1 行書き出す「hide=0」を実行しなさい。  
でなければ  
 シリアル通信 1 行書き出す「hide=1」を実行しなさい。



# もっと詳しく条件分岐を知ろう



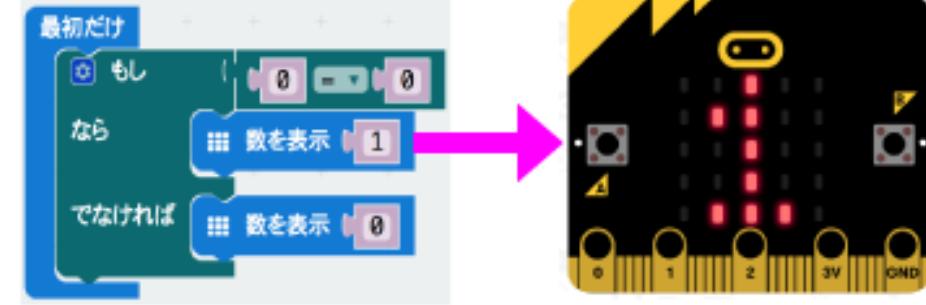
もし「真」なら1を表示する



変数も使える



もし「偽」なら0を表示する



条件式も使える「0」 = 「0」

この場合は、= が成り立つので1を表示

# やってみよう！

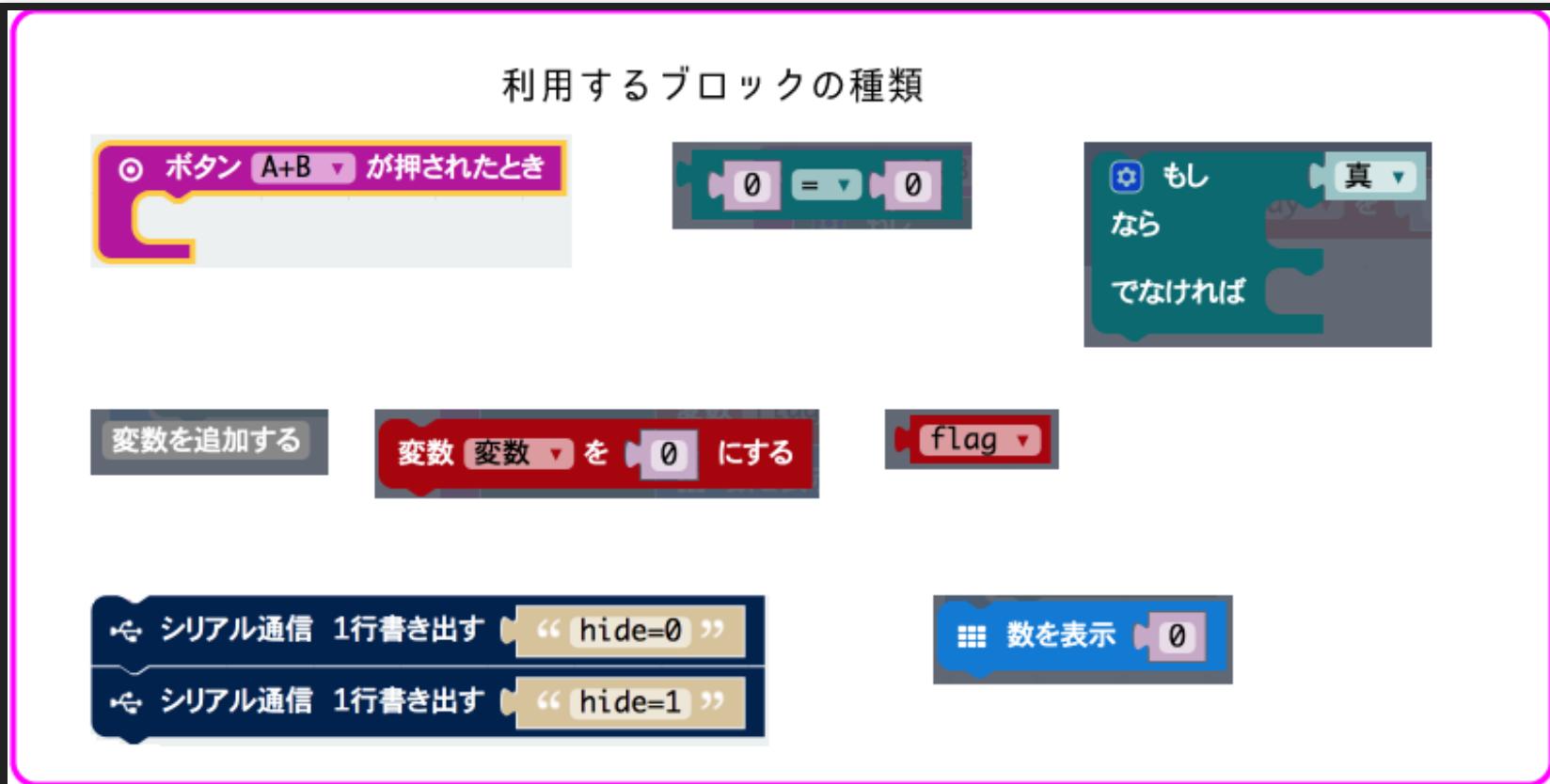
[A+B]を押すと、LEDの画面に1と表示されて、もう一度押すと0と表示される。

1と表示されている間は、プレイヤーが消えて、0と表示されている間はプレイヤーが表示されるように改造してみよう。

flagという変数を作ってやってみよう！

# ヒント

利用するブロックの種類



```
when green flag clicked
    [set [hide? v to 0]
    if [key A+B pressed?]
        then
            [if [hide? v = 0]
                then
                    [show [ ] v
                    say [ ]
                    [flag v]
                    [serial output v "hide=0"]
                    [serial output v "hide=1"]
                end
            end
        end
    end]
```

このスクリプトは、緑色の旗が押されたときに実行されます。ボタン A+B が押されたときの条件ブロックで、変数 hide? の値が 0 の場合に表示を現す、メッセージを出力する、flag を立てる、シリアル通信で "hide=0" と "hide=1" を書き出すなどの操作を行います。

# オリジナルのコントローラーに改造してみよう！

≡