

**“実際に使える”  
プログラムの作成に  
挑戦しよう！  
レベル1**

# 次のステップに進もう！

Pepperを動かすことができるようになったので  
実際に使えるプログラムの作成に挑戦してみよう！



# 実際に使えるプログラムって？

生活の中にはプログラムで動いているものがたくさんある  
それらを自分で作ってみよう！

まずは生活の中にあるプログラムによって動いているものを  
思いつく限り書き出してみよう



# プログラムで動いているものの例



ゲーム



電化製品



駅の改札機

# プログラムで動いているものの例



ゲーム



電化製品



駅の改札機

# ゲームを作ろう！

Pepperをプログラミングしてゲームを作るなら  
どんなゲームがよいだろうか  
Pepperの特徴を活かしたPepperらしいゲームが良い



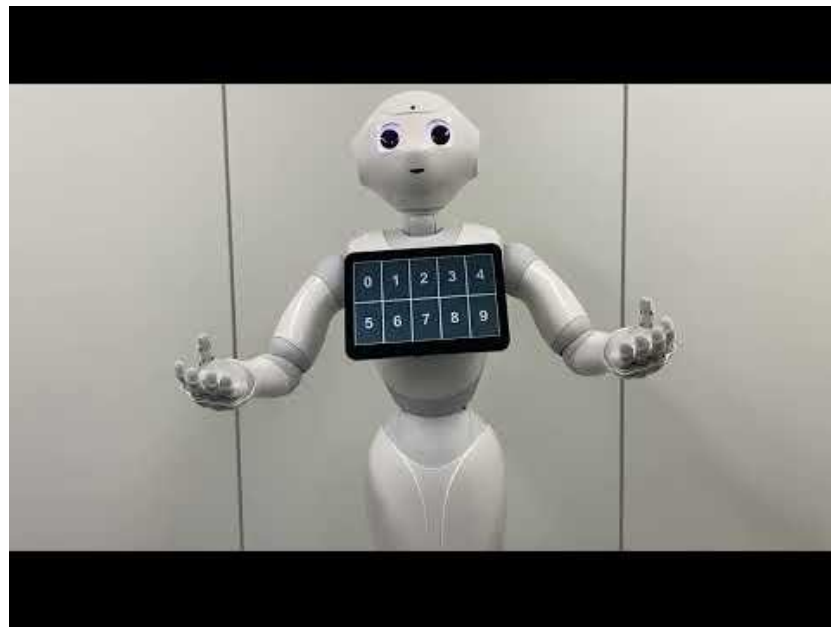
コントローラーは使えない



タッチできるディスプレイと  
音声認識ができるマイクがある



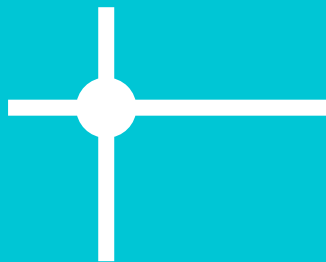
# 今回は数当てゲームを作ってみよう！



<https://youtu.be/4SSZpsJCyzU>

# STEP1

座標を使おう！



# STEP2

数字入力を作ろう！

1	2	3
4	5	6
7	8	9

# STEP3

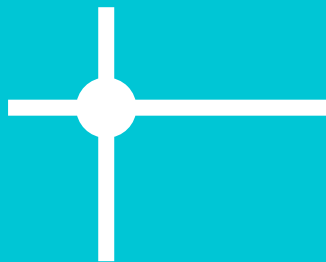
数当てゲームを作ろう！

1 2 3



# STEP1

座標を使おう！



# STEP2

数字入力を作ろう！

1	2	3
4	5	6
7	8	9

# STEP3

数当てゲームを作ろう！

1 2 3

# タッチパネル

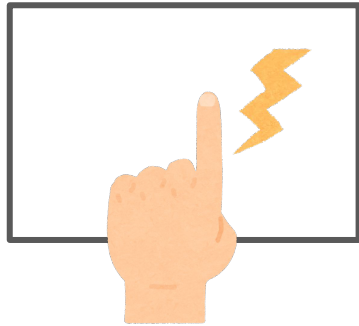
Pepperの胸にはディスプレイが付いていて  
タッチで操作することができる

タッチ操作ができるディスプレイのことをタッチパネルという

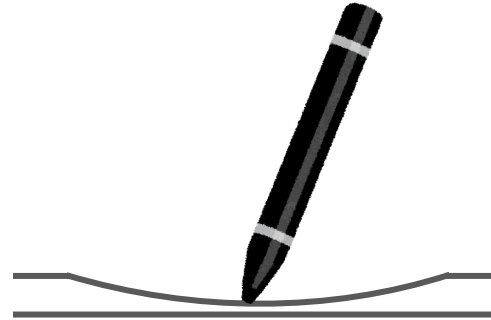


# タッチパネルの仕組み

タッチパネルには様々な方式があり、  
指だけでなくペンにも反応させたいかなど  
目的によって使い分けられている



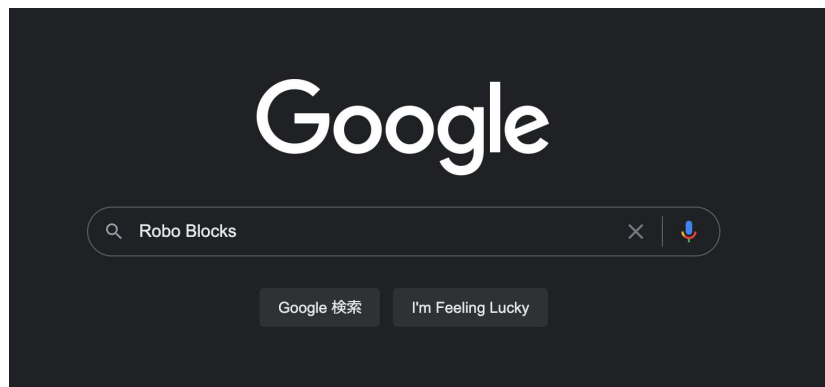
指先の静電気を検知する方式  
スマホなどで良く使われている



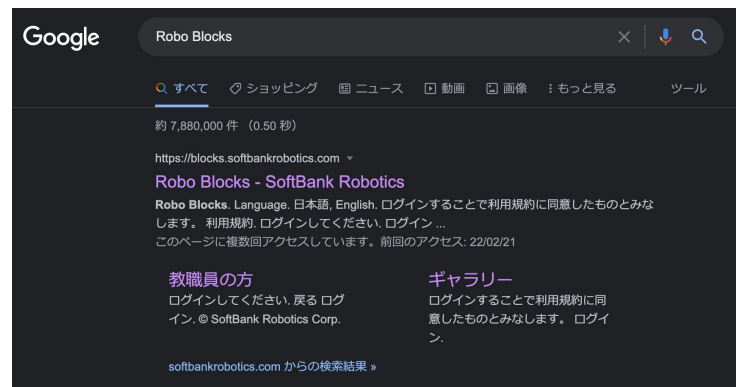
圧力で検知する方式  
ペンなどで押されても反応できる

# やってみよう！

早速Pepperのディスプレイで試してみよう！  
まずはRobo Blocksに ログインするために  
blocks.softbankrobotics.com にアクセス



Robo Blocks を 検索



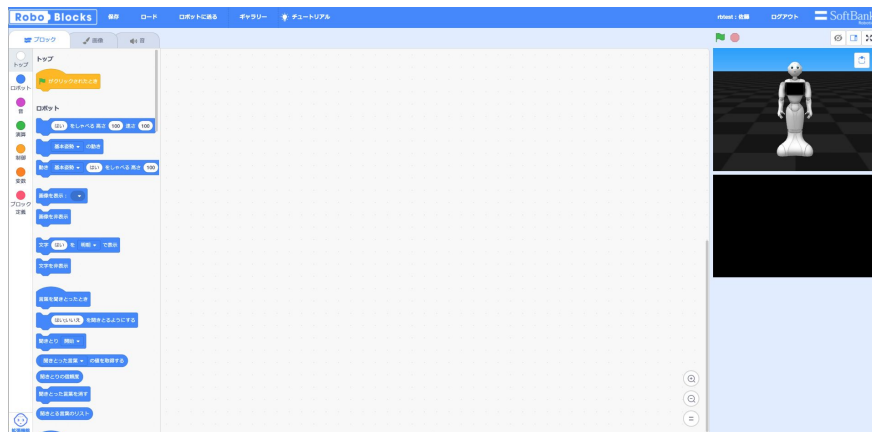
Robo Blocks にアクセス

# やってみよう！

ルーム名・ルームパスワード・名前を入力して  
ログインをクリック



ログインページ



Robo Blocks メイン画面

# やってみよう！

ディスプレイを使うためのブロックを探そう  
まずは“画面のタッチ待受を開始ブロック”と  
“画面がさわられたときブロック”を使う

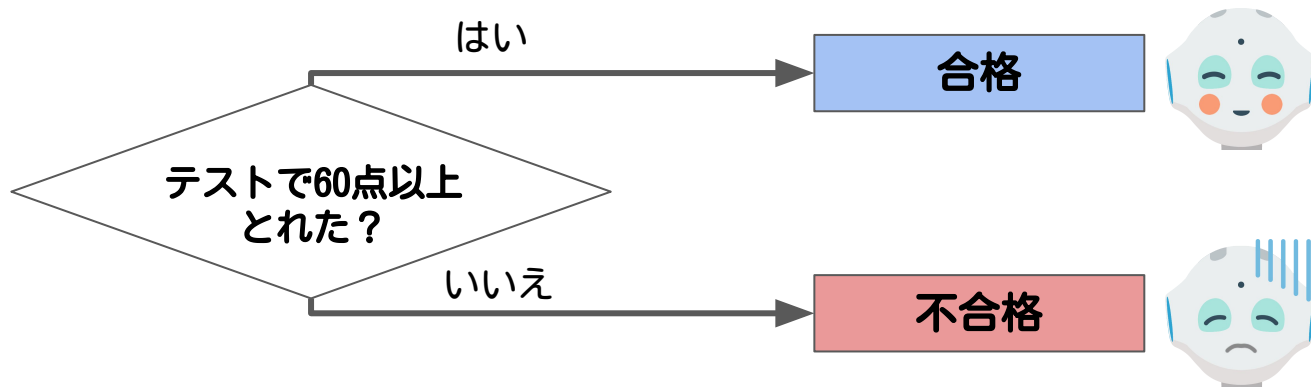
画面のタッチ待受を

開始 ▼

画面がさわられたとき

# 条件分岐

今のプログラムは画面のどの位置を触っても同じ動きをする  
条件によってプログラムの動きを変えたい時は条件分岐を使おう



普段の生活の中にも  
条件によってその後の行動が変わることは良くある



# やってみよう！

条件分岐は制御にある  
“もし～ならブロック” で作ることができる



目次から制御を選ぶ



もし～ならブロック

# やってみよう！

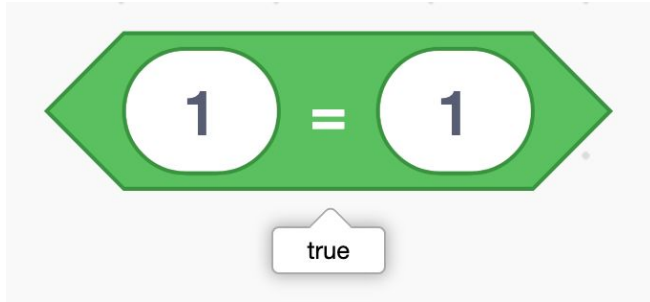
もし～ならブロックは六角形の空白に  
緑色の真偽ブロックを入れて使う



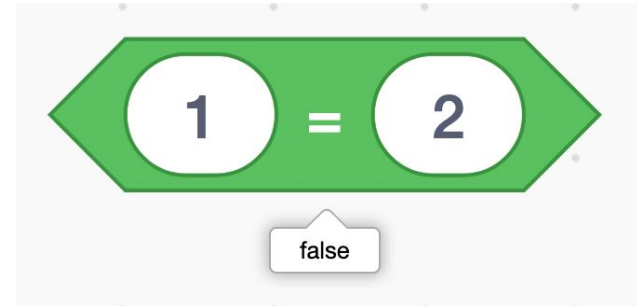
真偽ブロックは目次の中から緑色の演算を選ぶと見つけれれる

# 真偽ブロック

真偽ブロックは必ず真か偽のどちらかの値になる  
必ず真か偽のどちらかになるので条件を表すのに便利  
真はtrue、偽はfalseで表される



条件が満たされていれば  
値はtrue（真）になる

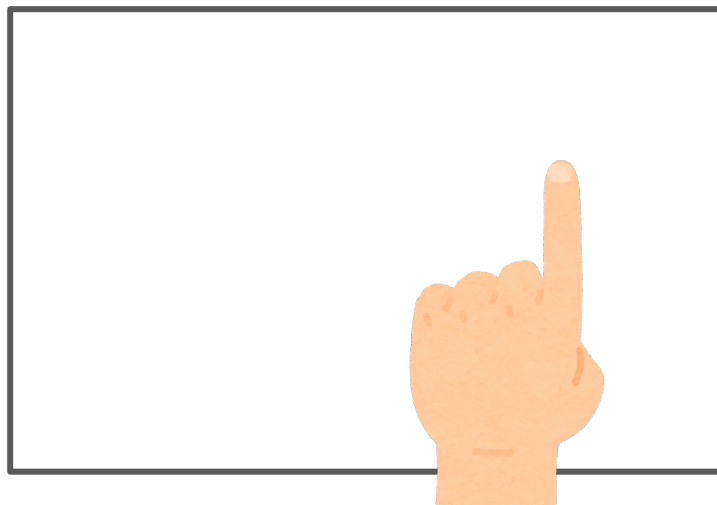


条件が満たされていない場合  
値はfalse（偽）になる

# ディスプレイの位置を表す条件

真偽ブロックを使って

ディスプレイのどこがタッチされたのか判定する方法を考えよう  
平面のある1点を表す方法には、どんなものがあるだろうか



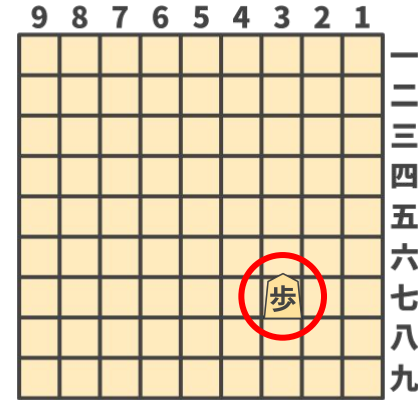
# 座標

ある一箇所を表現するための数値を座標という  
表現したい物によっていくつかの数値を組み合わせることもある



\*1

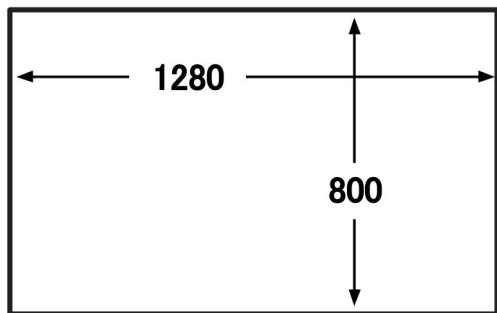
例えば東京タワーを表す座標は  
北緯35度39分31秒 東経139度44分44秒  
の2つの数字の組み合わせ



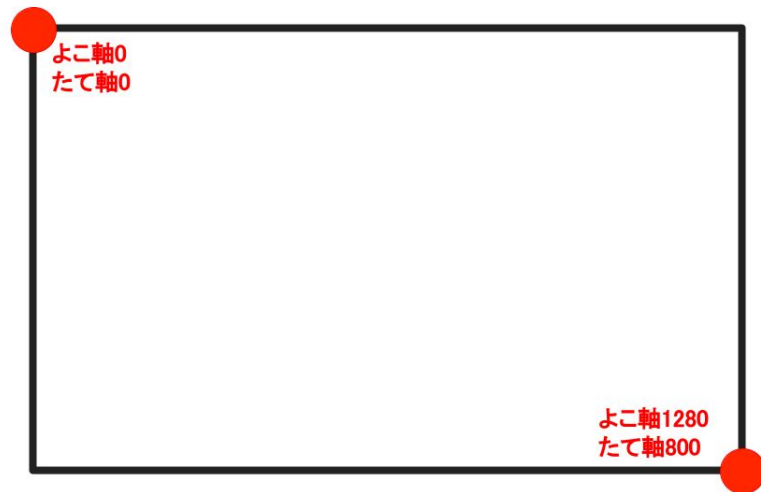
将棋では縦の位置と横の位置に加え  
コマの種類を表したいので3つの値を使う  
この場合は三七歩

# Pepperのディスプレイでの座標

Pepperのディスプレイでは位置を表すために  
よこの値とたての値の2つを使う  
左上が(0, 0)で、右下が(1280, 800)

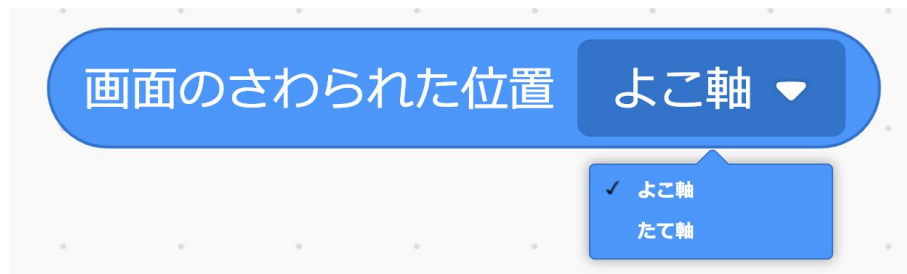


よこが0~1280  
たてが0~800 のサイズ



# やってみよう！

まずは、画面を半分に分けて、左半分か右半分かを判定してみよう  
“画面のさわられた位置ブロック”を使うと  
触られた位置の座標を取得することができる

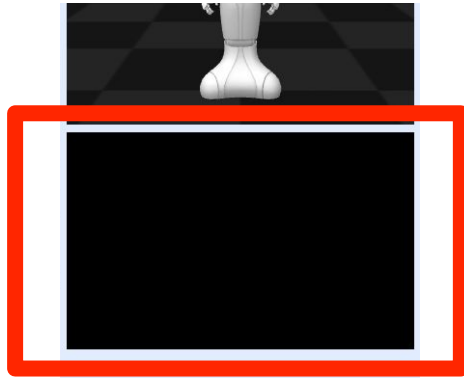


左か右かを判定したいのでよこ軸の値を使う



# やってみよう！

バーチャルロボットの下にある  
ディスプレイのプレビューエリアをクリックすると値が変わるので  
いろいろな場所を触って値を見てみよう



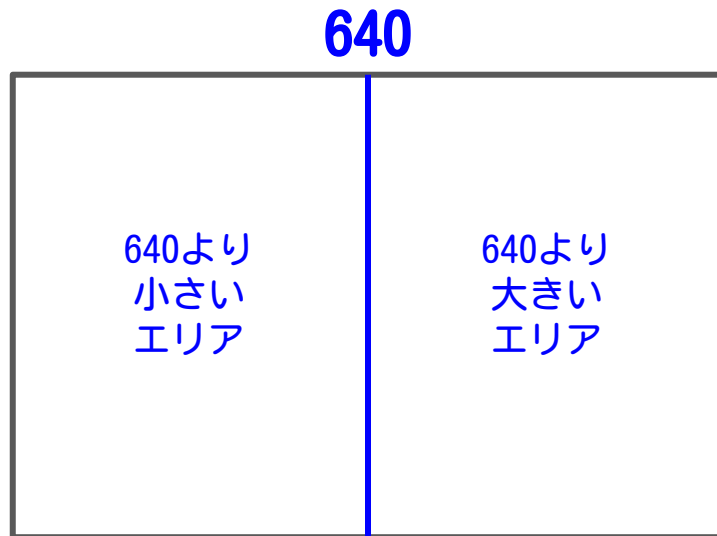
ディスプレイのプレビューエリア



タッチ待受を開始した後に画面を触ると“画面の触られた位置”の値が変化する

# ディスプレイを横半分にするなら

よこの長さは0~1280なので真ん中は640  
この640より大きい小さいか判定すれば良い



# やってみよう！

つまり条件は“640より大きい（小さい）”となる  
数字の大小を比べる以下の真偽ブロックを使って  
条件分岐を作ろう



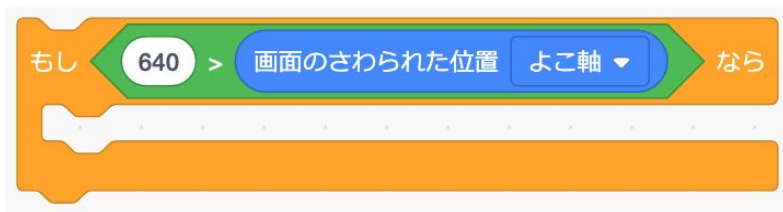
A大なりB  
AはBより大きい



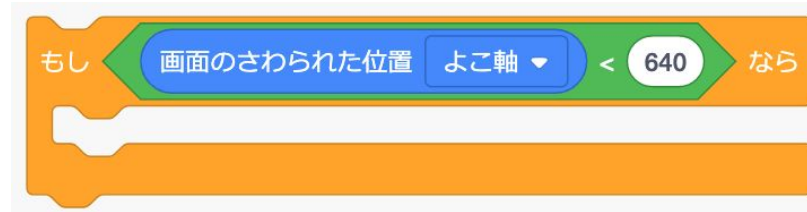
A小なりB  
AはBより小さい

# やってみよう！

条件（真偽ブロック）ができれば  
制御ブロックと組み合わせて条件分岐にしよう  
条件分岐「もしよこ軸が640より小さいなら」は以下のようになる



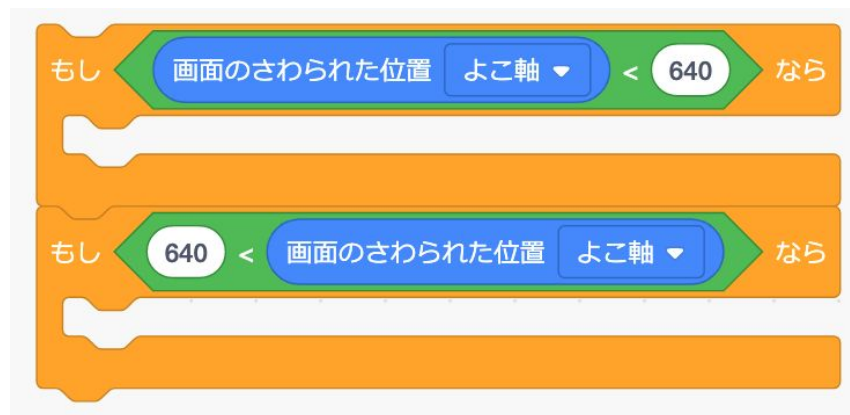
A大なりBを使って表現する場合



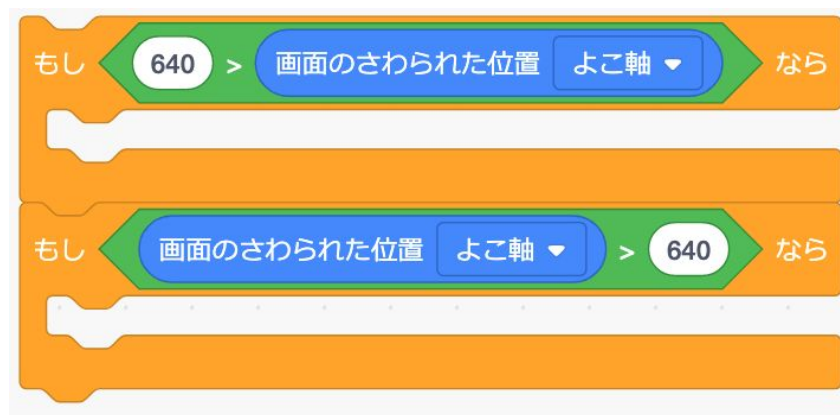
A小なりBを使って表現する場合

# やってみよう！

「もしよこ軸が640より大きいなら」の条件も作って重ねたら  
条件分岐が完成



A大なりBを使って表現する場合



A小なりBを使って表現する場合

# やってみよう！

分岐の中にブロックを入れて、  
タッチした位置によって違う動きをするPepperを作ろう



# “<” と “>” どちらを使うべきか

“<” と “>” のどちらを使っても同じ条件を表現することができるが、どちらを使うべきか。これには正解がなく、どちらをどう使っても良いが、使い方はルールを定めて統一されていると良い。

複数人で共同作業を行う場合はもちろん、自分一人でコードを書く場合もルールを定め決まった考え方のもとで条件を作ること、プログラムの可読性（読みやすさ）や保守性（修正しやすさ）などが向上するためである。

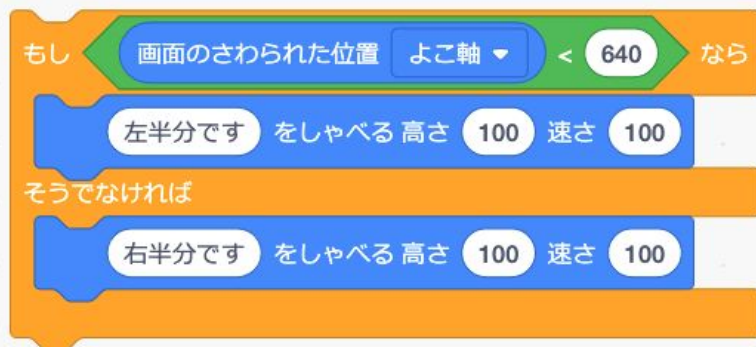
例えば「変数と定数の比較なら必ず変数を左にする」や「<しか使わない」など様々なルールが考えられる。

このプログラムの作り方を決めたルールのことをコーディング規約という。プログラミングに慣れてきたらプログラムが読みやすくなるコーディング規約を考えてみよう。



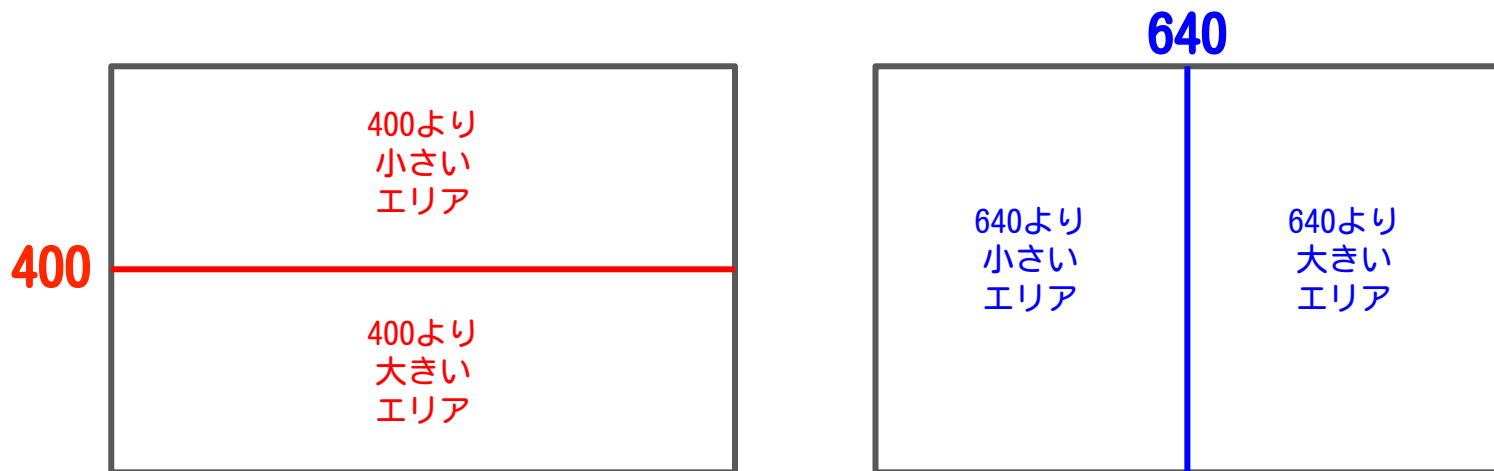
# 条件分岐の作り方 その1

もし()ならブロックを2個重ねて条件分岐を作成したが、  
もし()なら そうでなければブロックを使っても条件を作ることができる  
この場合、真偽ブロックを1つ減らすことができる  
ただ、必ずしもブロック数が少ない方が良いというわけではないので、  
自分が読みやすいと思う方を使おう

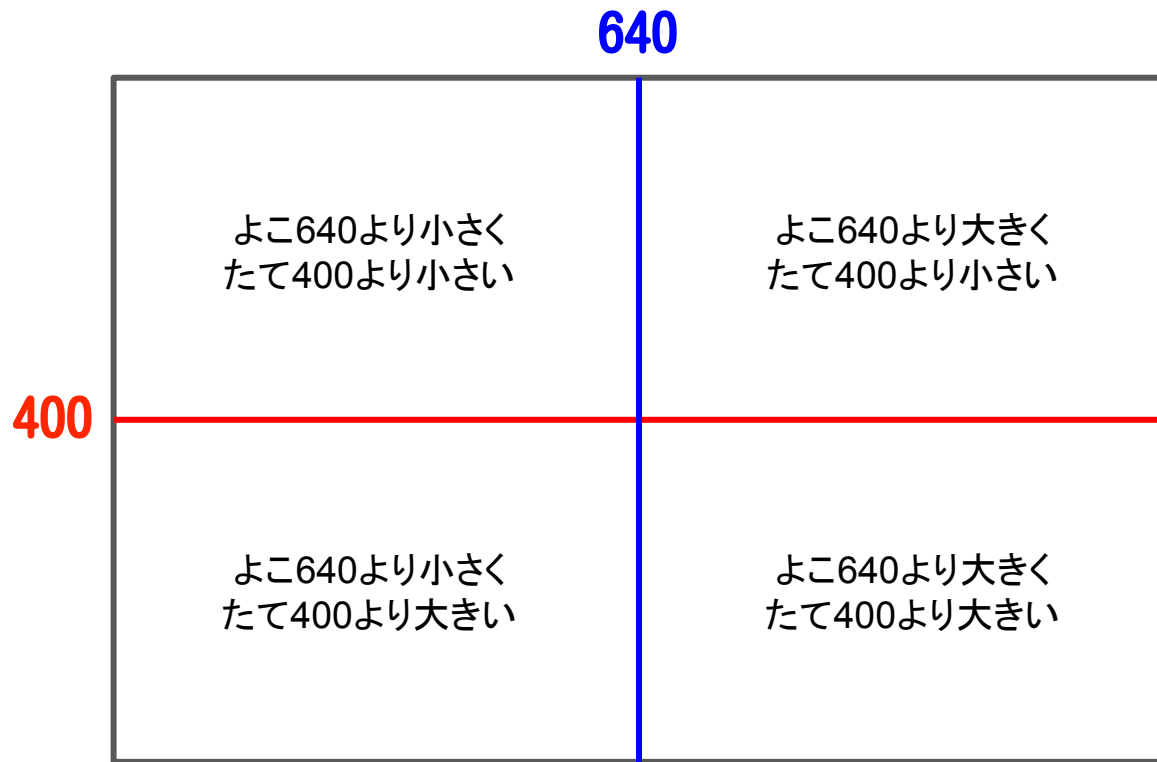


# ディスプレイを4等分にするなら

たての長さは0~800なので真ん中の値は400  
よこの真ん中は640だから400と640の二つの値を使って  
条件分岐を作る必要がある



# ディスプレイを4等分にするなら



## 2つの条件を使った条件分岐

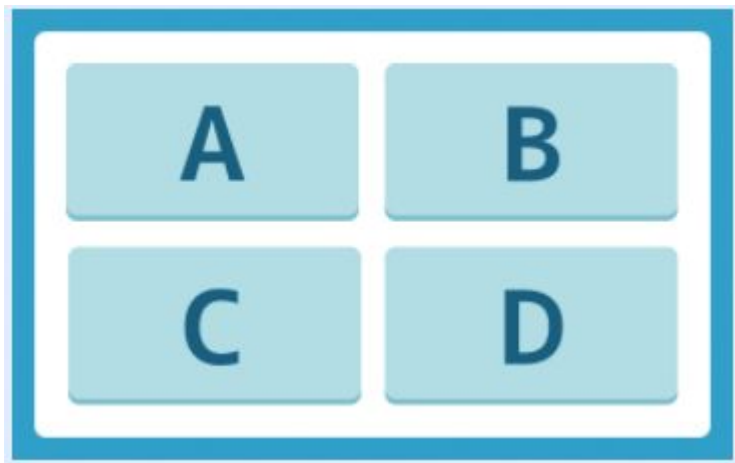
「よこ640より小さくたて400より小さい」のような2つの真偽値を使った条件分岐の作り方を考えよう



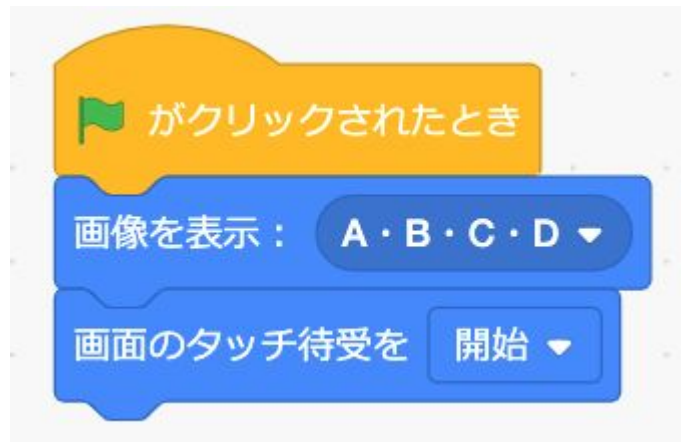
左上が押されたかどうか判定するにはこの2つの真偽値が必要になる

# やってみよう！

4択用の画像が用意されているので  
画像タブから選んでディスプレイに表示しよう



AからDの4択のボタンをイメージした画像



ブロックの例

# やってみよう！

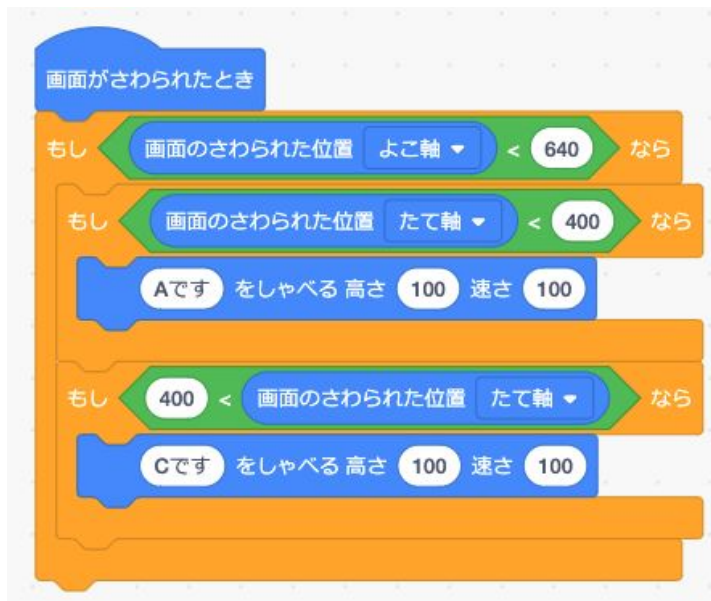
もし ( ) ならブロックの中にもう1つもし ( ) ならブロックを入れて  
条件分岐を2つ組み合わせてみよう



Aのエリア（よこ軸が640より小さく、たて軸が400より小さい）を判定する条件分岐

# やってみよう！

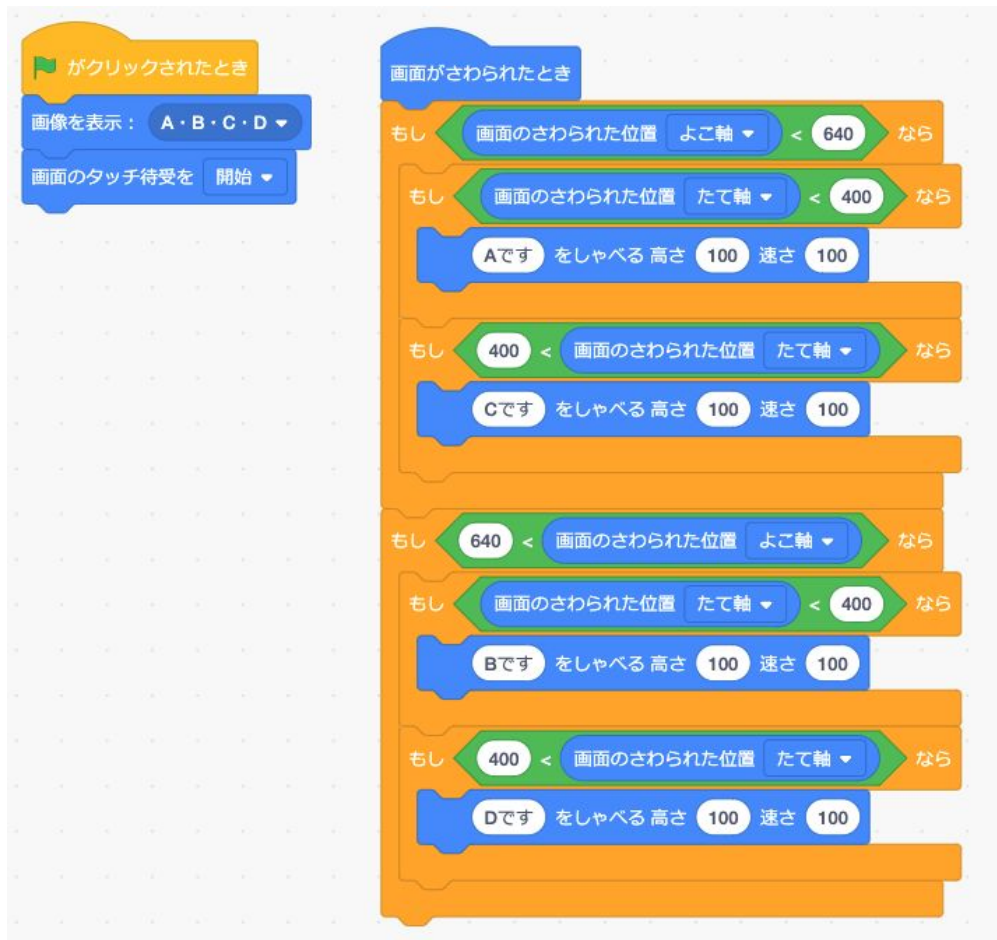
## 4つのエリアを判定する条件分岐を作ってみよう



先ほどのブロックにCを判定するブロックを追加した例



# ディスプレイを 4分割したタッチ処理の イメージ



## 条件分岐の作り方 その2

同じ条件分岐でも表現方法は様々あり、その1で紹介したもし()なら そうでなければブロックを使う方法や変数を使う方法の他にも、複数の条件を一つにまとめる方法などがある。例えば()かつ()ブロックや()または()ブロックを使えば複数の真偽ブロックを繋ぎ、1つの真偽ブロックにすることができる。



()かつ()ブロックを使ってAのエリア（よこ軸が640より小さく、たて軸が400より小さい）を表現する場合、右下の図のようになる。



## 条件分岐の作り方 その2

()かつ()は両方の値がtrueの場合のみtrueになり、()または()はどちらかの値がtrueであればtrueになる。作りたい条件によってどちらを使えば良いか考えよう。

なお、>や=のような単純に値を比較する演算子（Robo Blocksの場合はブロック）を関係演算子、()かつ()などの条件を組み合わせることができる演算子を論理演算子と呼ぶ。

ブロック	Aの値	Bの値	ブロックの値
	true	true	true
	true	false	false
	false	true	false
	false	false	false
	true	true	true
	true	false	true
	false	true	true
	false	false	false

# STEP1

座標を使おう！



# STEP2

数字入力を作ろう！

1	2	3
4	5	6
7	8	9

# STEP3

数当てゲームを作ろう！

1 2 3

# 数字を入力するプログラムを考える

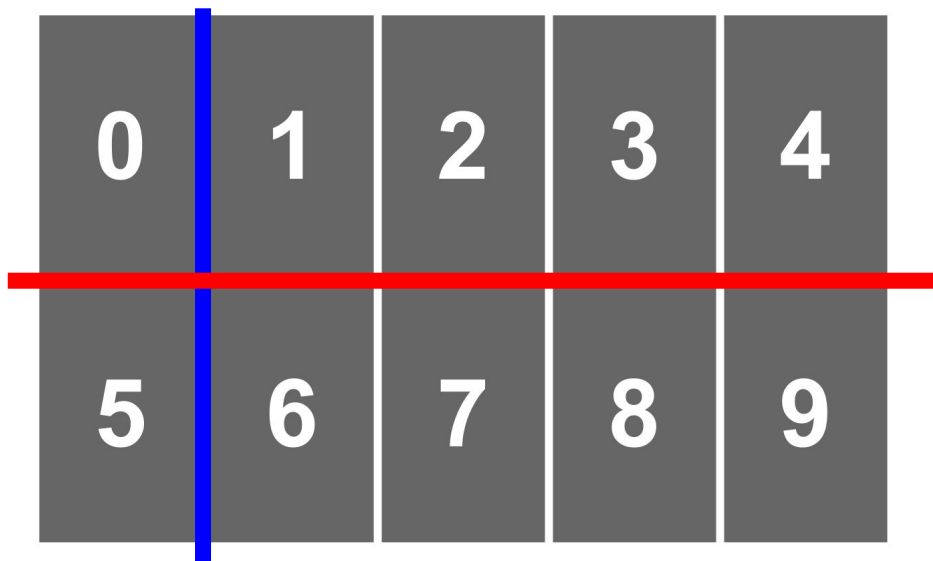
ディスプレイを分割する方法を学んだので、  
応用して数字を入力するプログラムを考えてみよう  
数字は0から9の10種類あるため、最低でもボタンが10個必要になる

0	1	2	3	4
5	6	7	8	9

Pepperのディスプレイにボタンを10個配置した例

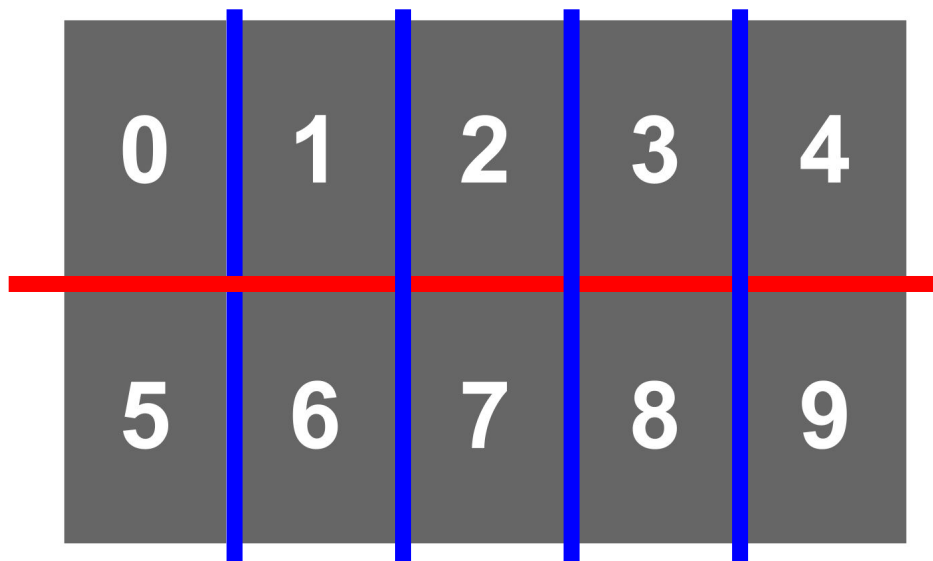
# 数字を入力するプログラムを考える

4分割と同じ考え方で、ディスプレイを分割してみよう  
必要になる境界線は何本あるだろうか



# 数字を入力するプログラムを考える

この条件分岐にはよこを分割する軸(青)が4本、  
たてを分割する軸(赤)が1本必要

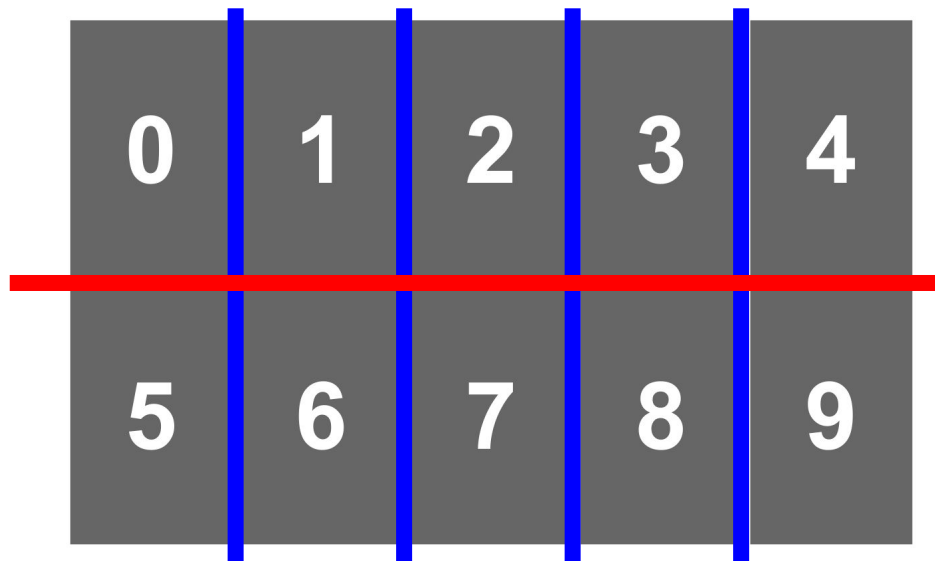




# やってみよう！

必要な5つの値を計算しよう

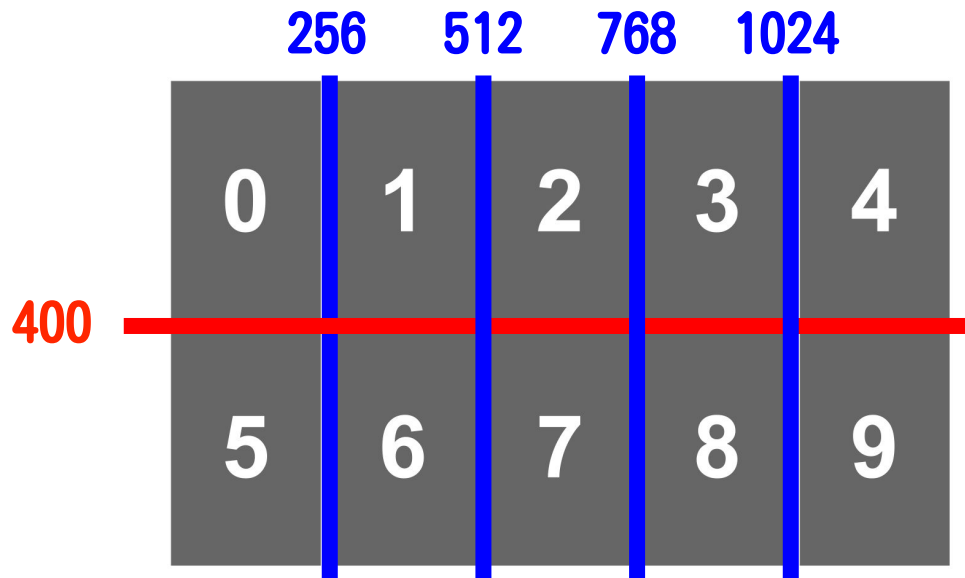
よこは画面を5分割しているので $1280 \div 5$ 、たては2分割なので $800 \div 2$



# やってみよう！

値は以下の通り

この条件分岐の境目になる値を境界値という

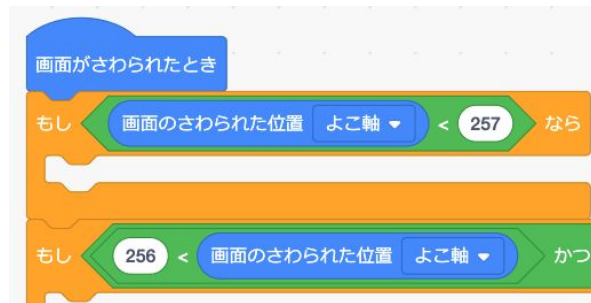
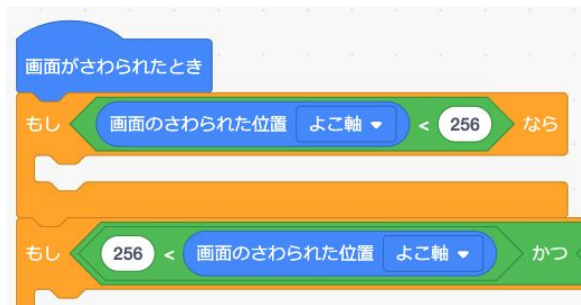


## 境界値と条件分岐

今回は256や512など境界値をそのまま条件に設定しているが、256より小さいか、256より大きいかという条件分岐であるため256ぴったりの位置をタッチしてしまった場合、どちらにも反応しない。

Robo Blocksには等号付き不等号 ( $\geq$ ・ $\leq$ ) が無いため、漏れがないように設定するなら、どちらかを1増減させるか、 $=256$ の条件を追加して256の場合に対応させる必要がある。

ただ、ボタン同士の境界線をタッチすることは稀である事、境界値ちょうどをタッチするのは困難である事から、本資料では分かりやすさを重視し条件には境界値をそのまま使用する。



## ディスプレイ上の位置の調べ方

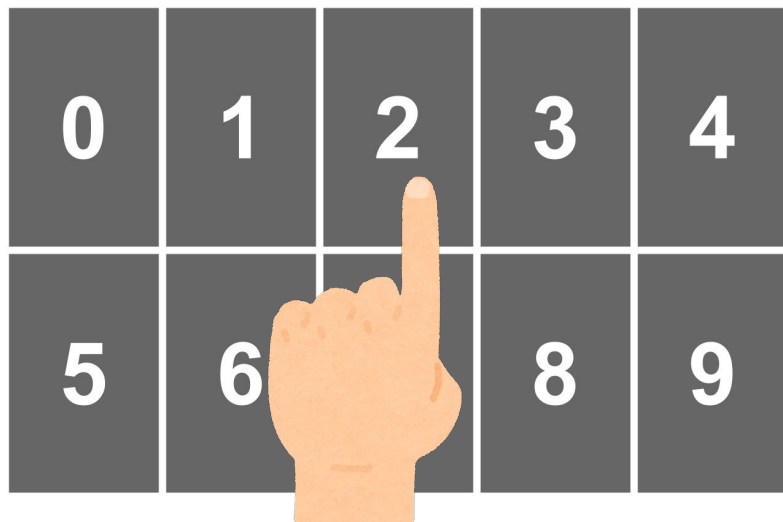
今回使用する画像はタブレットを等分しているため計算で境界値を求めることができるが、不規則なレイアウトで画面を作成した場合は画像を作成したツール上で測るか、実際にタッチして調べる必要がある。

タッチして調べる場合、左下の図のような簡単なブロックを作成してタッチの待受を開始し、ディスプレイのプレビュー上の調べたい箇所をクリックする。その後画面の触られた位置ブロックをクリックすると触った座標が格納されているので、その値を記録していくとよい。



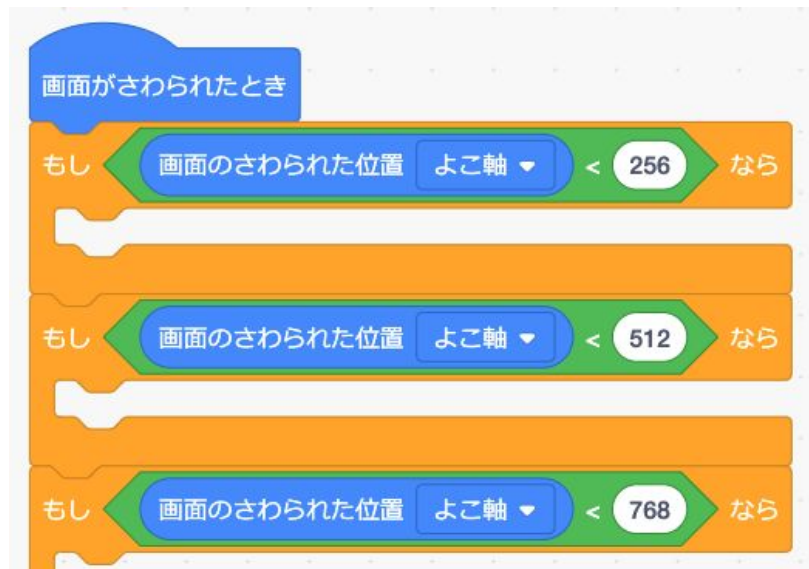
# やってみよう！

境界値を使って条件分岐を作り、  
触った位置の数字をPepperが喋るプログラムを作ろう



# ヒント！

図のように作るとうまく動かない  
例えばよこ100をタッチした時、全ての条件が真になってしまう



# ヒント！

触られた位置に対して、真になる条件が1つだけになれば良い  
例えば以下のように条件を2つ組み合わせる方法や・・・





# ヒント！

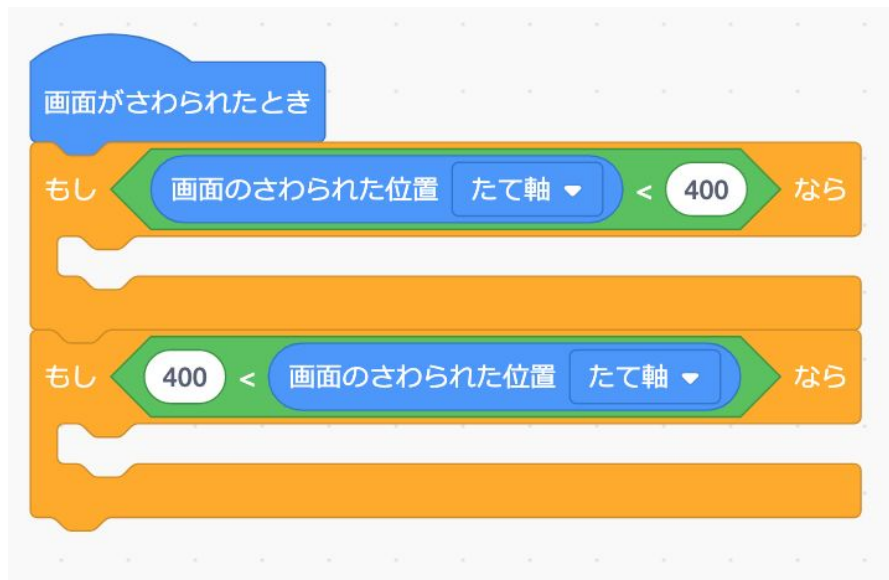
もし()なら そうでなければ を使う方法などがある



# やってみよう！

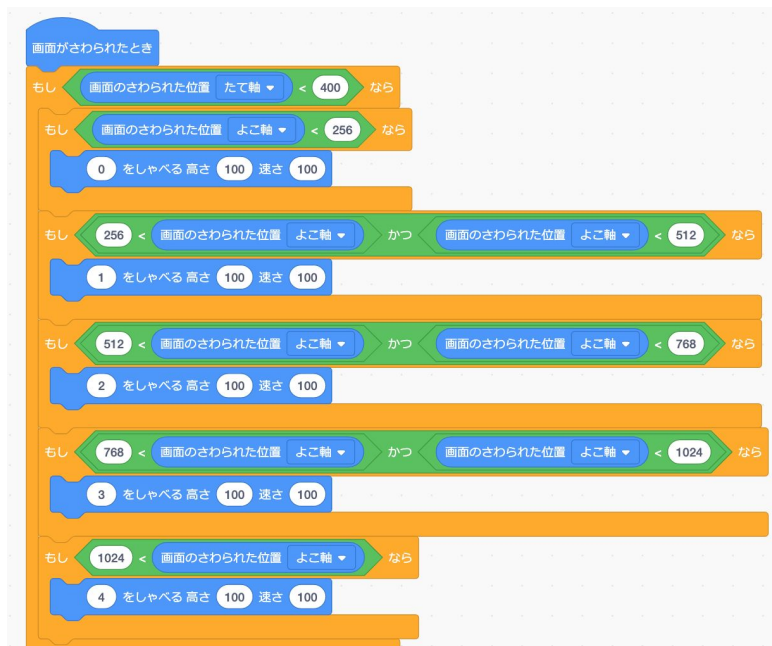
## 条件分岐の例

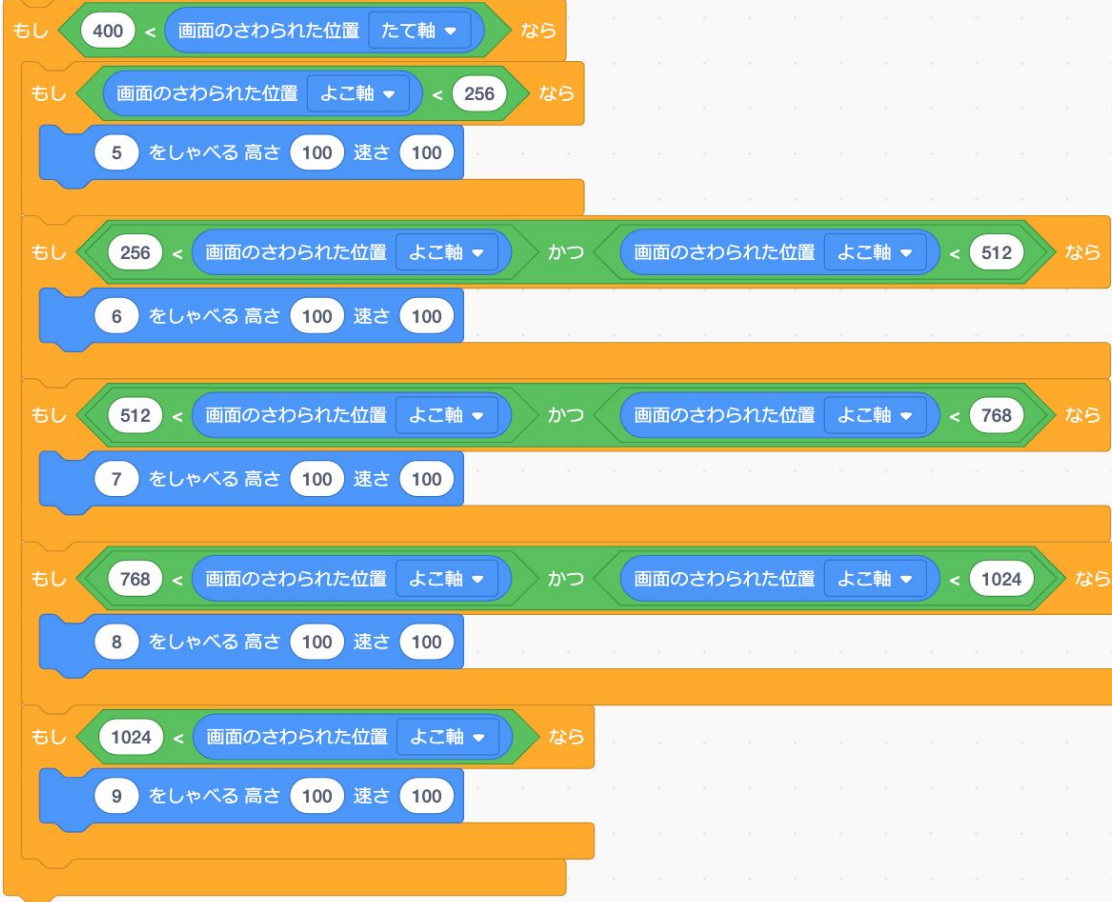
まずは分岐が少ないたて軸の分岐を作ってみよう



# やってみよう！

たて軸の分岐それぞれによこ軸の処理を入れる





# STEP1

座標を使おう！



# STEP2

数字入力を作ろう！

1	2	3
4	5	6
7	8	9

# STEP3

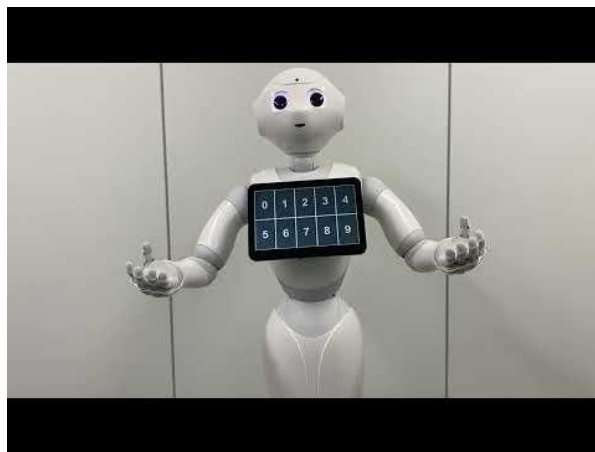
数当てゲームを作ろう！

1 2 3

# 数当てゲームの流れ

数当てゲームはお題を出す人が数を一つ思い浮かべて、  
何回目でその数を当てられるか競うゲーム

答えが外れだった場合、  
正解の数はその数より大きいのか小さいのか教える

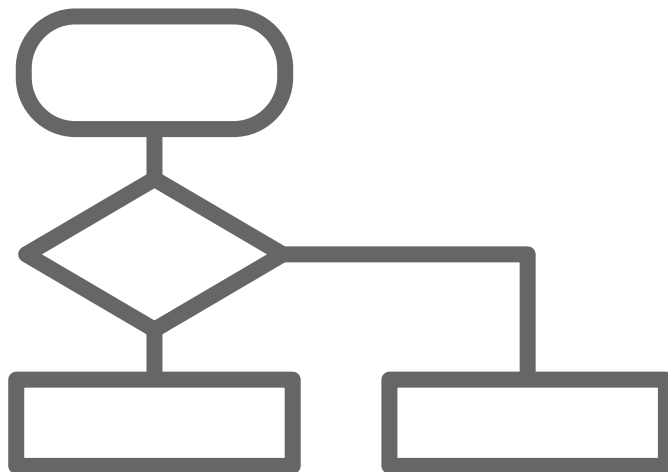


# 数当てゲームの流れ

ゲームをプログラミングするため処理の流れを考えてみよう

最初に用意しなければいけない数は何か

押された数によってどのように分岐すれば良いだろうか





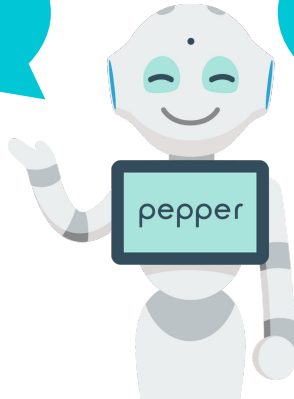
# 数当てゲームの流れ

- ① Pepperは0～9までの数を1つ思い浮かべる
- ② 人間は数を予想してディスプレイの数字をタッチする
- ③ Pepperは人間に正解か不正解かを伝える
  - ③-1 正解の場合、Pepperは人間が何回目で成功したかを伝えて終了
  - ③-2 不正解の場合、Pepperが思い浮かべた数がタッチした数より大きい小さいかを伝えたあと、もう1度②から挑戦させる

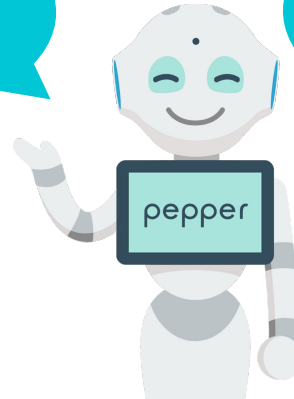
# 乱数

まず ①Pepperが0～9までの数字を1つ思い浮かべる を作ろう  
思い浮かべる数字を自分で入力してしまうと  
毎回同じ数が答えになってしまいゲームが面白くない

正解は  
5です！



正解は  
5です！



正解は  
5です！



# 乱数

そこで、答えには毎回違う数がランダムで設定されるようにしよう  
ランダムな数のことを日本語で乱数という

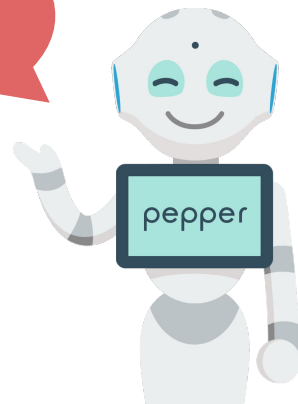
正解は  
5です！



正解は  
2です！



正解は  
9です！



# やってみよう！

乱数を使うためのブロックは緑色の演算ブロックの中になる  
乱数の上限と下限を指定すると  
実行するたびにその範囲のどれかの数になる



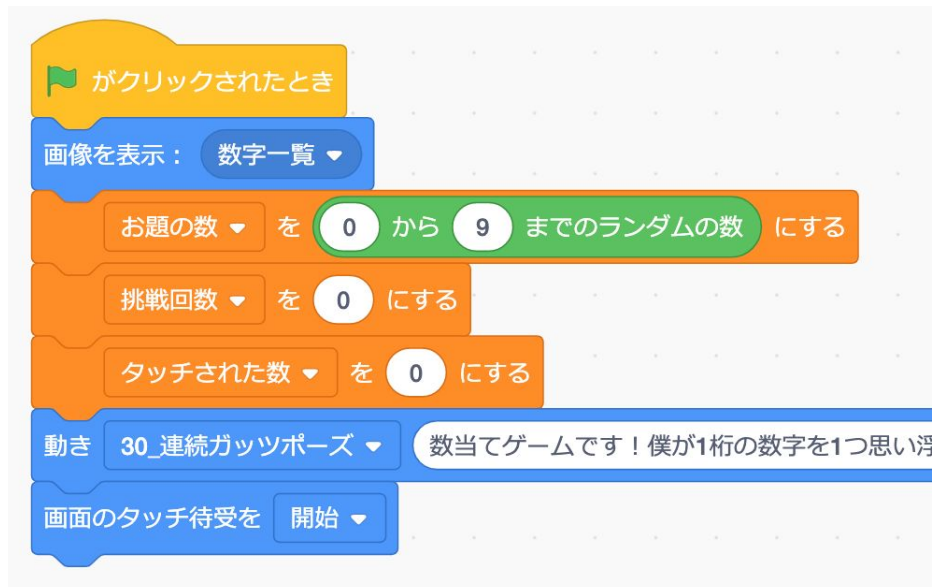
# やってみよう！

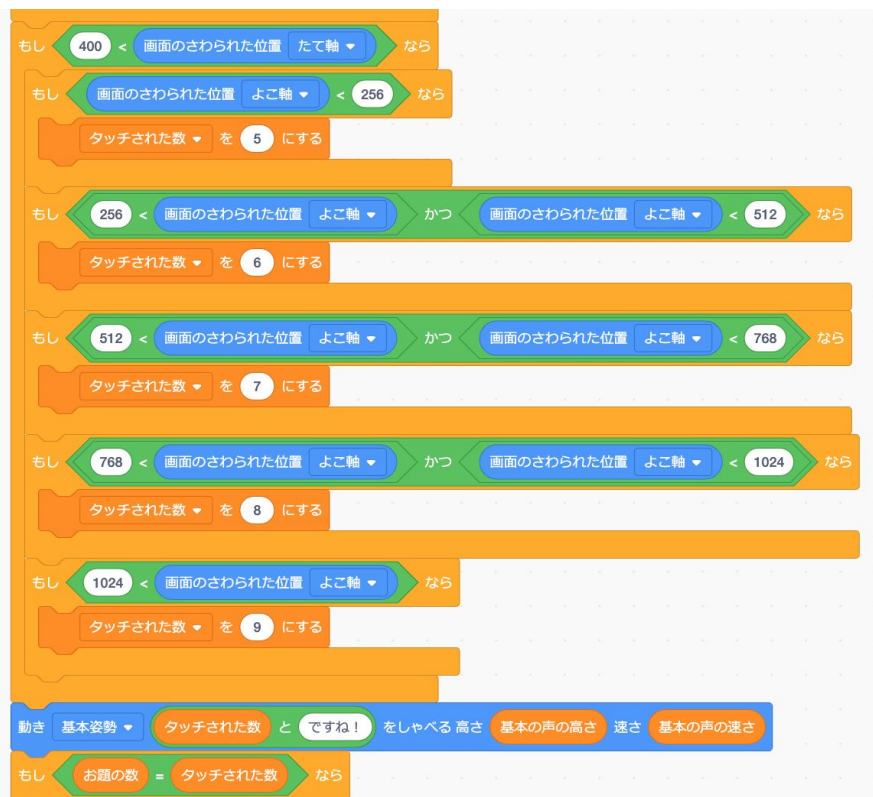
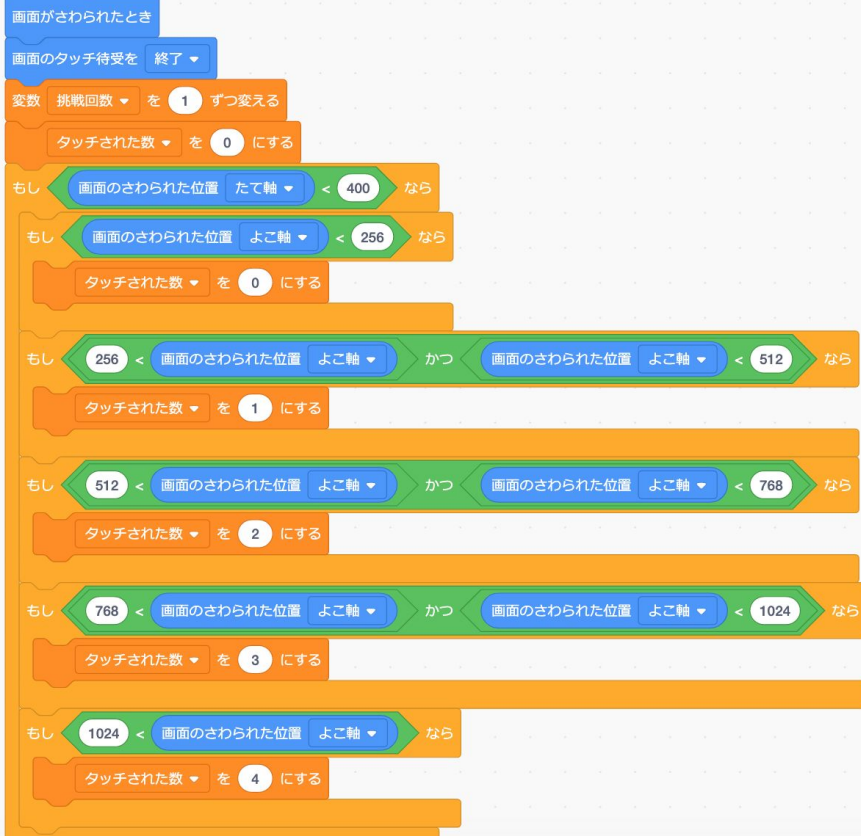
乱数ブロックは動くたびに値が変わるので、  
1度抽選した結果を使い続けたい時は変数に格納しよう



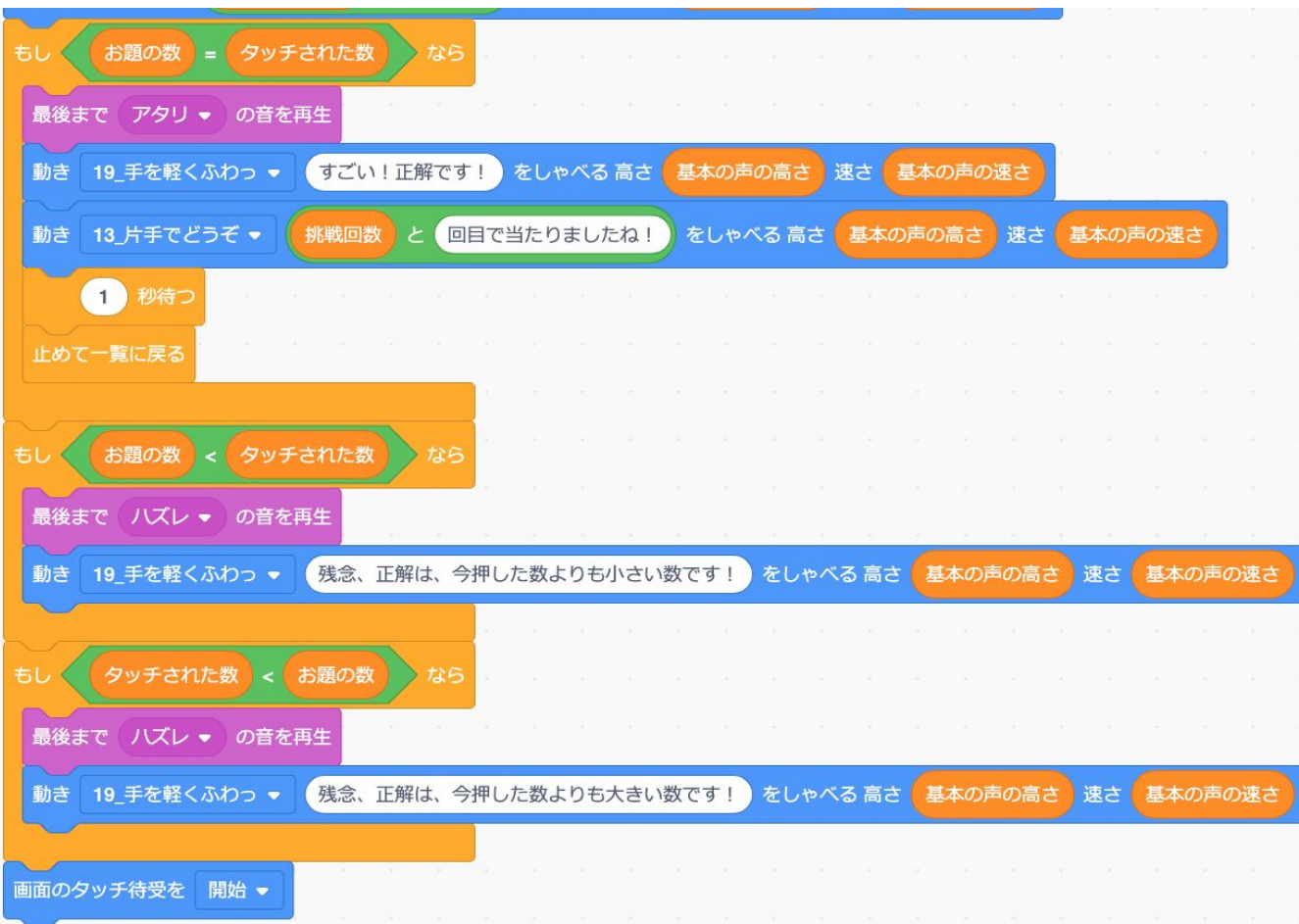
# やってみよう！

## 数当てゲームの例





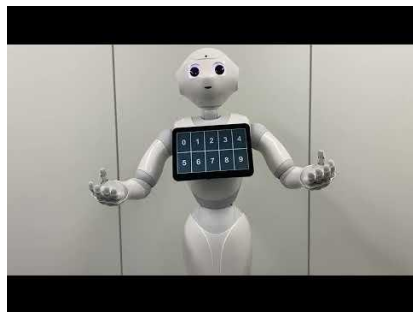




# 余裕がある人は機能を追加してみよう

- ・ 音声でも数字を選択できる
- ・ 2桁の数字まで取り扱える
- ・ 演出を豪華にする

など、プログラムにさらなる機能を追加してみよう！



<https://youtu.be/4SSZpsJCyzU>



悩んだら  
作成例の動画を  
真似してみよう