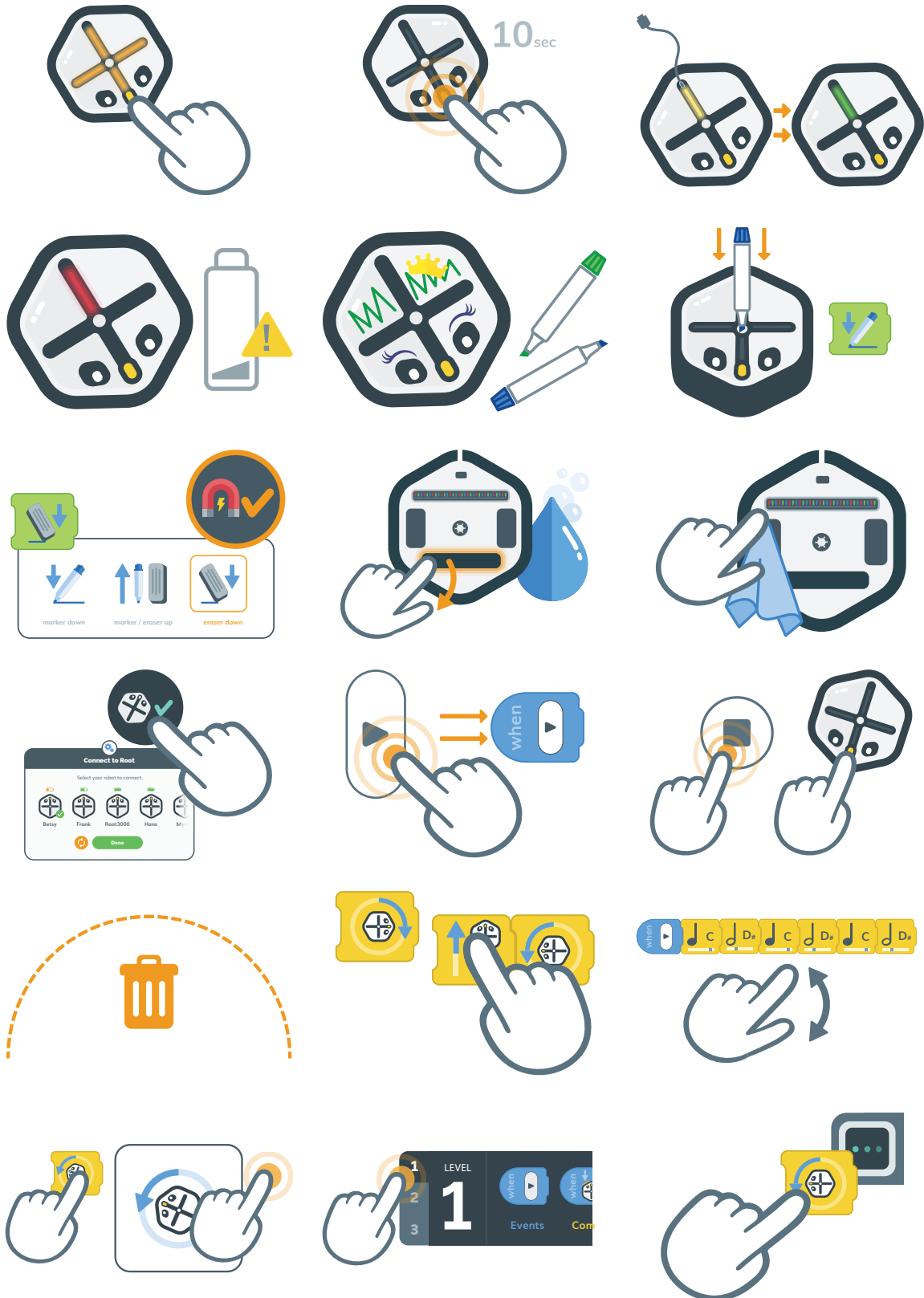


iRobot

iRobotコーディングヘルプガイド



電源を入れる/電源を切る

Rootの電源を入れるには、"鼻"を
しっかり押さえます。電源を切ると
きも同じようにします。



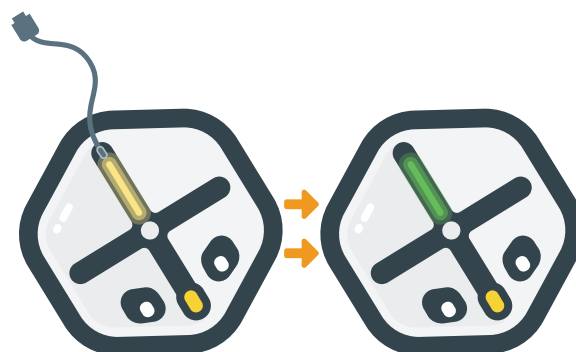
強制再起動

Rootが反応しなくなったとき
は、Rootの鼻を10秒間押さえます。



充電

充電を始めるには、充電ケーブル
を電源に接続します。充電中は黄
色のライトが点滅し、充電が終わ
ると緑色のライトが点灯します。



バッテリー残量低下の警告

バッテリーが少なくなると、赤色のライト
が点滅します。充電するには、Rootを電源
に接続します。



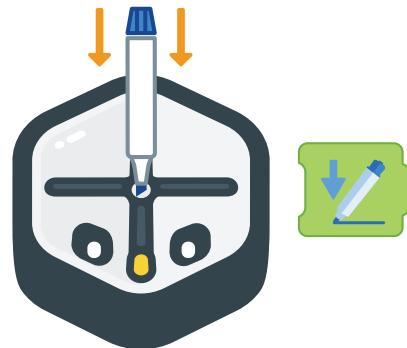
Rootを飾る

Rootの表側はホワイトボードになっています。ホワイトボードマーカーやシールで、ロボットを飾りましょう!



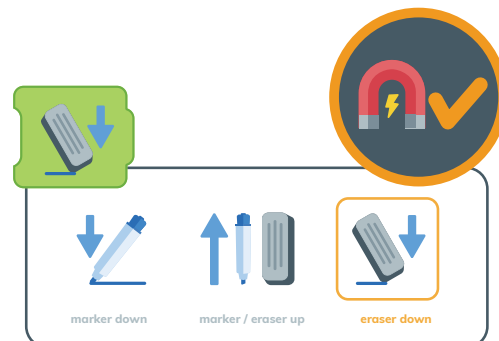
Rootで絵を描く

模様を描くには、Rootのマーカーホルダーに、マーカーをしっかりと差し込みます。マーカーを上げたり下げたりするには、"マーカー"ブロックを使います。



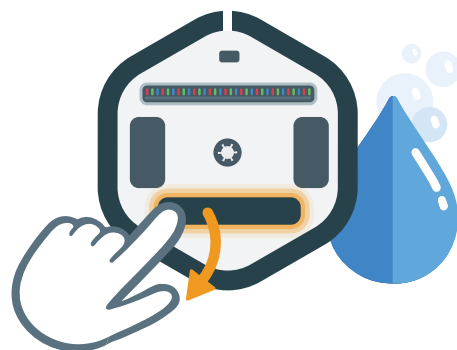
Rootで描いたものを消す

イレーサーを下げるには、"マーカー"ブロックを編集します。イレーサーを使える場所はマグネット式のホワイトボードなどだけです。



イレーサーの清掃

汚れを落とすには、イレーサーパッドを取り外し、水とせっけんで洗います。



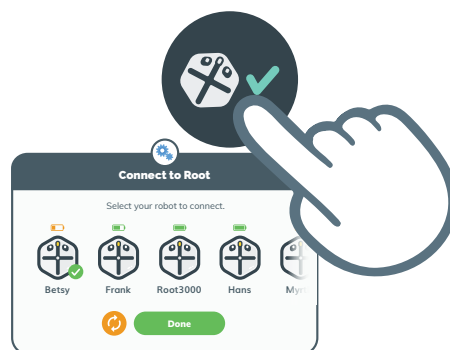
車輪の清掃

Rootの車輪が汚れたら、布できれいに拭き取ります。



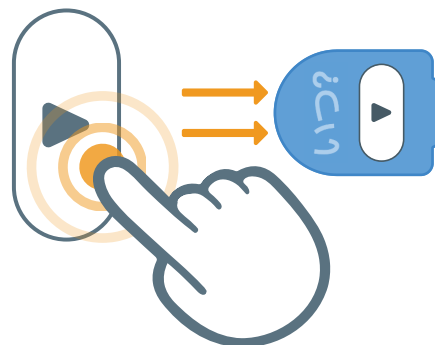
Rootへの接続

ロボットに接続するには、Rootアイコンにタッチして、自分のRootを選びます。



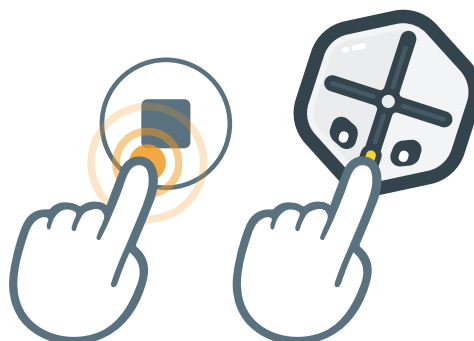
プロジェクトの実行

プロジェクトを実行するには、画面の左側にある再生ボタンを押します。プロジェクトはすべて、"再生を押したとき"ブロックから始まります。



プロジェクトの停止

プロジェクトを停止するには、Rootの鼻を押すか、画面の左側にある停止ボタンにタッチします。



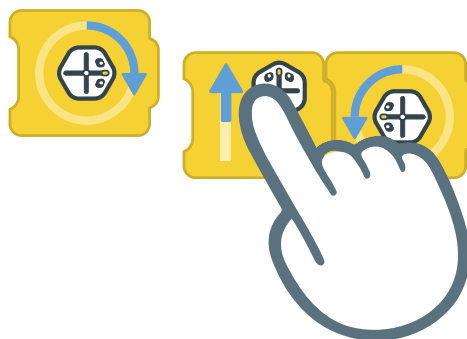
ブロックを削除する

ブロックをドラッグすると、ゴミ箱が現れます。ブロックを削除するには、ゴミ箱までドラッグします。



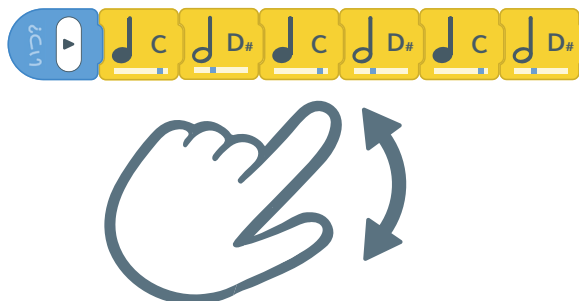
複数のブロックのドラッグ

ブロックを押さえると、右側に接続されたブロックをすべてドラッグできます。



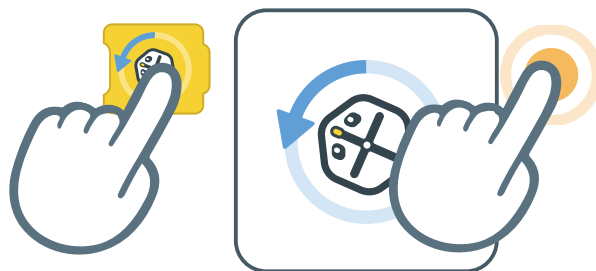
コーディング画面でのズーム

コードを見やすくするには、画面をピンチしてズームインまたはズームアウトします。



エディターを開く/閉じる

エディターを開くには、それぞれのブロックをタップします。エディター以外の場所をタップすると、エディターウィンドウが閉じます。



コードレベルの変更

1、2、または3をタップしてコーディングレベルを変更します。コードをそのレベルに変換できない場合は、警告が表示されます。



ブロックの説明

ブロックを正方形のアイコンにドラッグすると、そのブロックの機能が詳しくわかります。



レベル1ブロックの説明



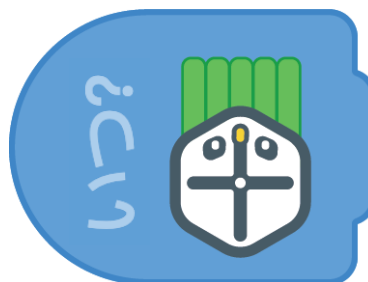
"再生を押したとき"ブロック

"再生を押したとき"ブロックより後のコードは、再生ボタンを押すとすぐに実行されます。



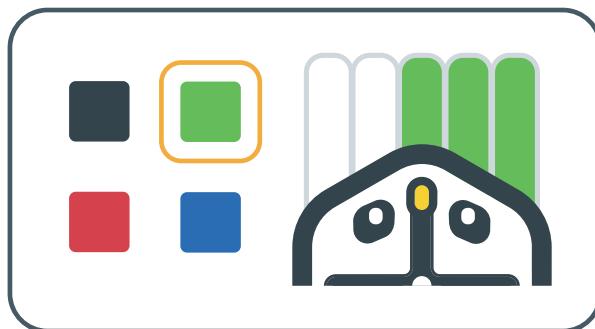
"色をスキャンしたとき"ブロック

選んだ色をRootが読み取って反応するようにコーディングするには、"色をスキャンしたとき"ブロックを使います。



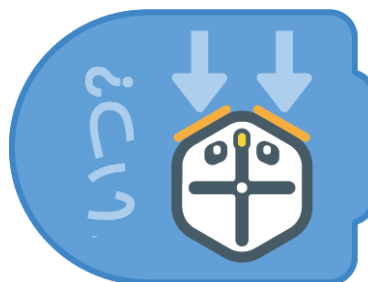
"色をスキャンしたとき"ブロックエディター

それぞれのゾーンをタップして、Rootに反応させる色を選びます。



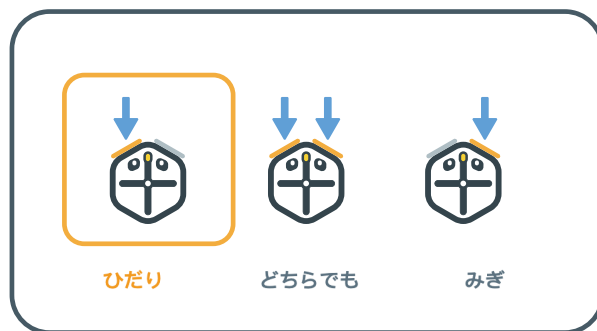
"バンパーがおされたとき"ブロック

"バンパーがおされたとき"ブロックは、バンパーセンサーが押されたことをRootに伝えます。



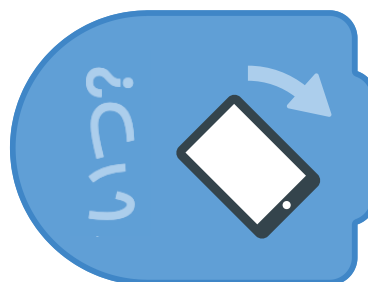
"バンパーがおされたとき"ブロックエディター

どのバンパーが押されたときにRootが反応するかを変更するには、このエディターを使います。



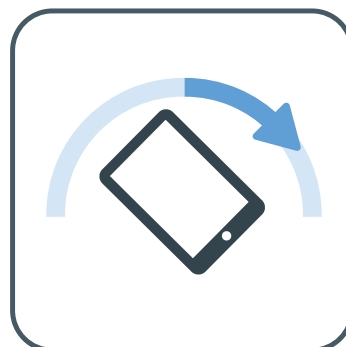
"かたむけたとき"ブロック

"傾けたとき"ブロックを使うと、デバイスをさまざまな方向に傾けたときにRootが反応するようにコーディングできます。



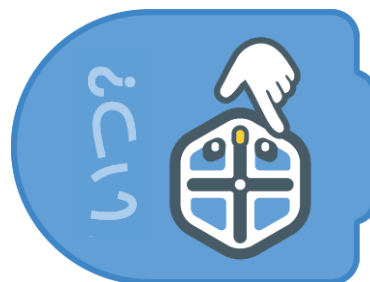
"傾けたとき"ブロックエディター

Rootに反応させる傾きの角度を選ぶには、矢印をドラッグします。



"おされたとき"ブロック

"おされたとき"ブロックを使うと、Rootの表側にある4つのゾーンをどれか押したときにRootが反応するようにコーディングできます。



"おされたとき"ブロックエディター

どのゾーンが押されたときにRootが反応するかを変更するには、このエディターを使います。



"音が聞こえたとき"ブロック

音量の変化にRootが反応するようにコーディングするには、"音が聞こえたとき"ブロックを使います。



"音が聞こえたとき"ブロック エディター

Rootが反応する音の大きさを変更するには、このエディターを使います。



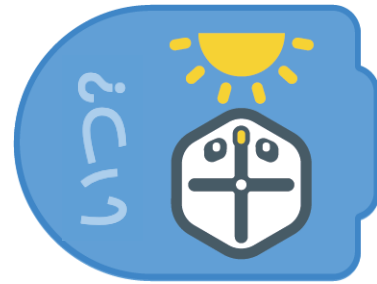
"明るさが変化したとき"ブロック

明るさの変化にRootが反応するようにコーディングするには、"明るさが変化したとき"ブロックを使います。



"明るさが変化したとき"ブロックエディター

Rootには、明るさの変化に反応するライトセンサーが2台あります。



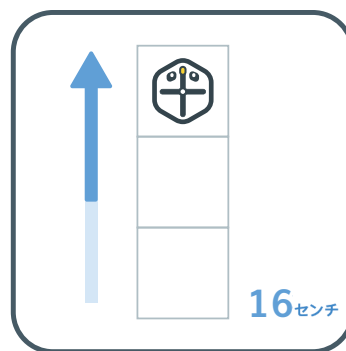
"いどうする"ブロック

前または後ろにRootが動くようにコーディングするには、"いどうする"ブロックを使います。Rootが移動する距離を設定することもできます。



"いどうする"ブロックエディター

前または後ろにRootが移動する距離を変更するには、Rootをドラッグします。



"かいてんする"ブロック

Rootが逆時計回りに回転するようにコーディングするには、この"かいてんする"ブロックを使います。



"かいてんする"ブロック

Rootが時計回りに回転するようにコーディングするには、この"かいてんする"ブロックを使います。



"かいてんする"ブロックエディター

Rootが回転する角度を変更するには、矢印をドラッグします。



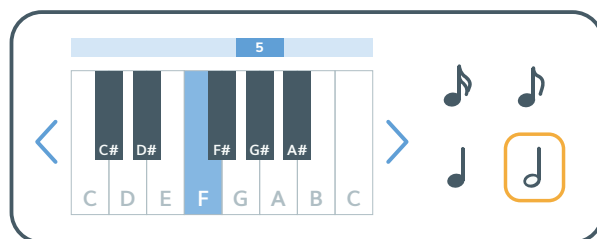
"音楽"ブロック

音が出るようにコーディングするには、"音楽"ブロックを使います。音の高さや長さを変えて、曲を作ることができます。



"音楽"ブロックエディター

エディターを使って、音階と音の長さを選びます。左か右の矢印をタップすると、オクターブを変更できます。



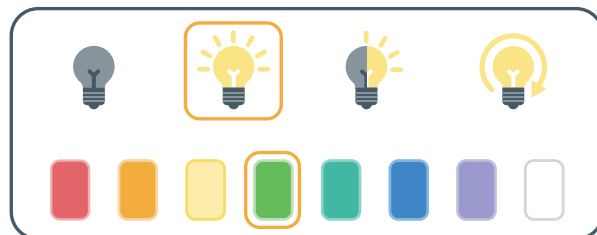
"ライト"ブロック

Rootの表側にあるライトを点灯して、色や点滅パターンを変更するには、"ライト"ブロックを使います。



"ライト"ブロックエディター

Rootのライトの色とパターンを変更するには、このエディターを使います。



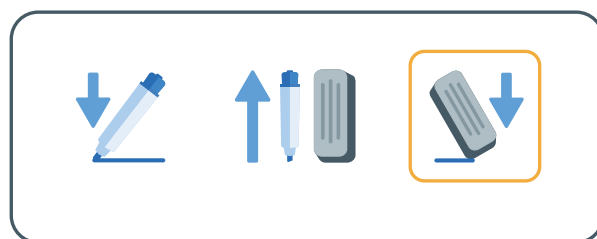
"マーカー"ブロック

マーカーとイレーサーを上げ下げするには、"マーカー"ブロックを使います。



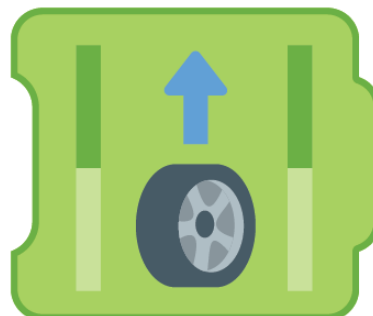
"マーカー"ブロックエディター

マーカーまたはイレーサーを上げたり下げたりするには、このエディターを使います。



"車輪の速度"ブロック

車輪が動く速さとRootが回転する範囲の広さを変更するには、"車輪の速度"ブロックを使います。



"車輪の速度"ブロックエディター

それぞれの車輪の速度と方向を直接変更することも、右側の共通アクション(その場で回転する、カーブ状に進むなど)を選ぶこともできます。



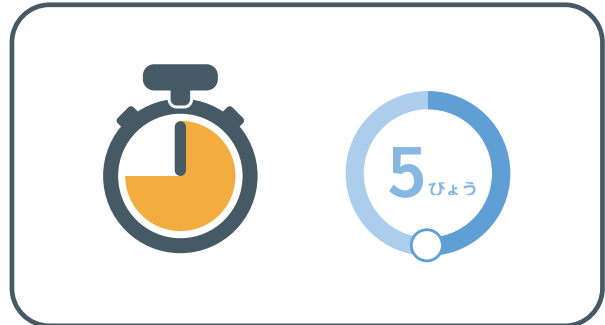
"まつ"ブロック

"まつ"ブロックを使うと、次のブロックに進むまでの時間の長さを設定できます。



"まつ"ブロックエディター

待つ時間の長さを変更するには、ダイヤルを動かします。



"くりかえす(かいすう)"ブロック

"くりかえす(かいすう)"ブロックは、再生を繰り返すために使います。

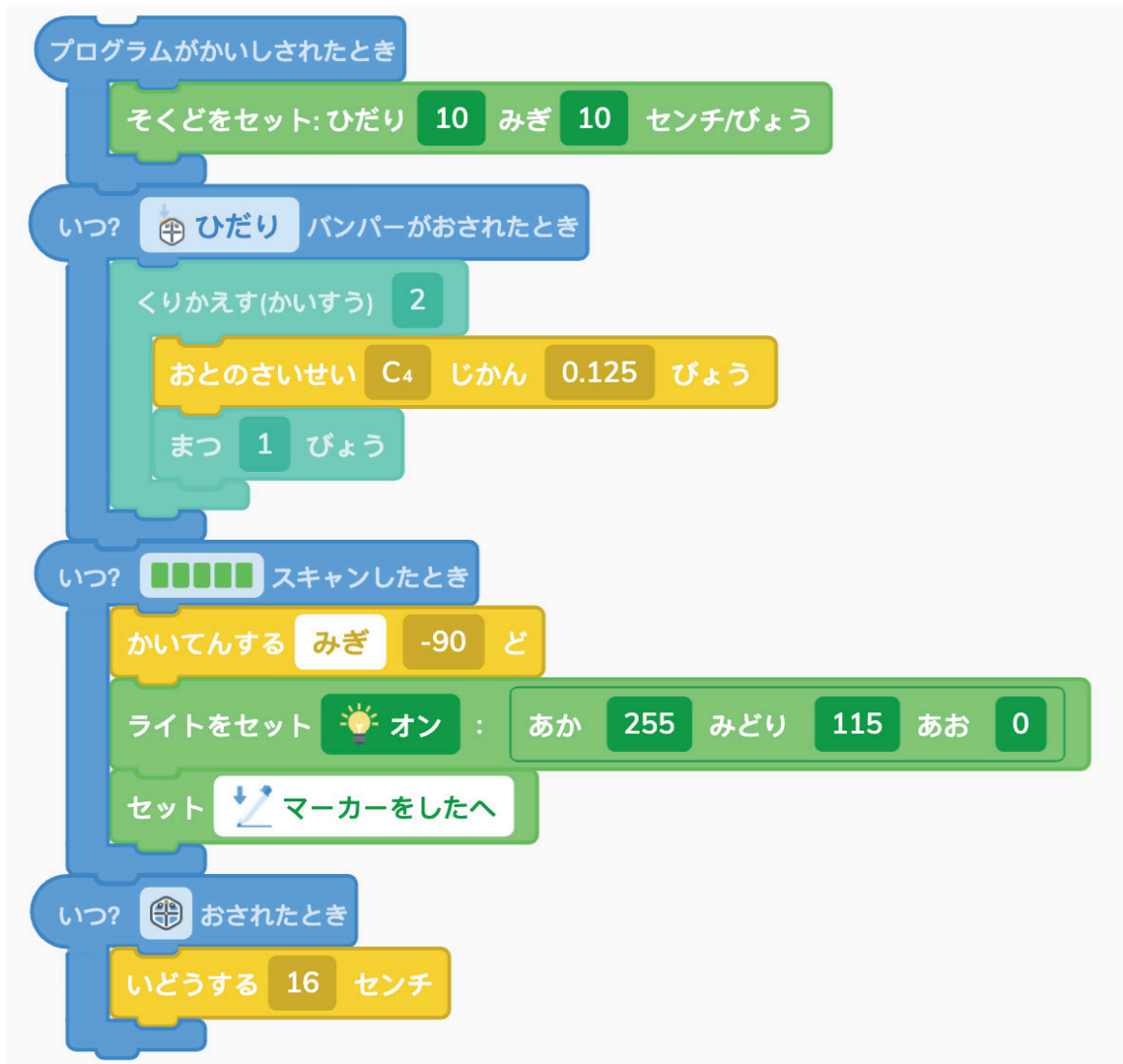


"くりかえす(かいすう)"ブロックエディター

プロジェクトのコードブロックを何回再生するかを指定するには、"くりかえす(かいすう)"ブロックを編集します。



レベル2ブロックの説明



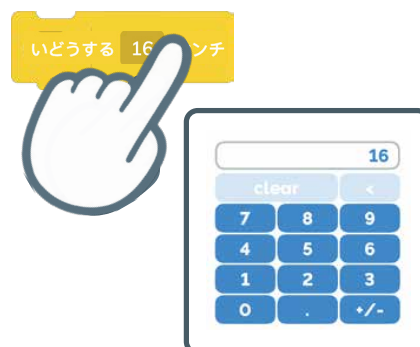
"いどうする"ブロック

前または後ろにRootが動くようにセンチメートル単位でコーディングするには、"いどうする"ブロックを使います。



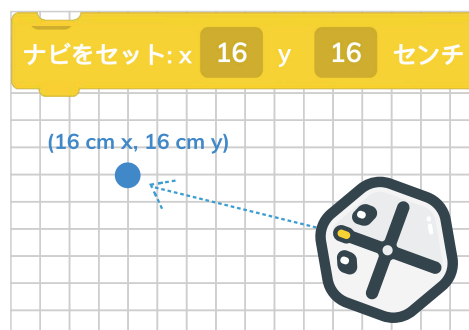
"いどうする"ブロックの編集

前または後ろにRootを何センチメートル動かすかを指定するには、"いどうする"エディターを使います。



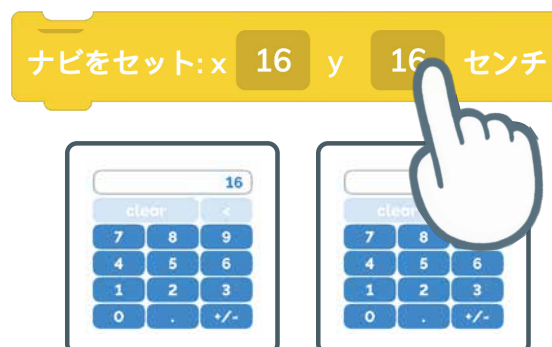
"ナビ"ブロック

Rootの移動には、目に見えない座標系が使用されます。Rootのスタート地点として原点(0センチ、0センチ)が設定されています。



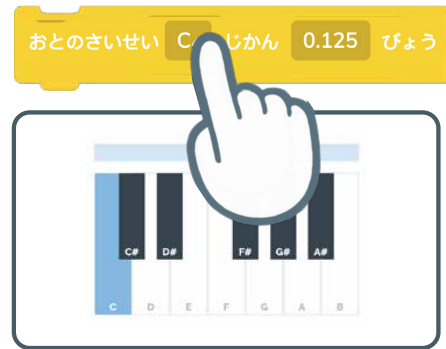
"ナビ"ブロックの編集

"ナビ"ブロックを編集すると、ある座標位置にRootが移動するように指定できます。



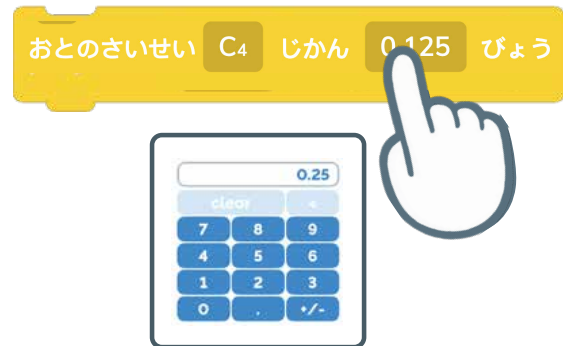
"音楽"ブロック

"音楽"ブロックを使うと、音を出すことができます。どの音階を鳴らすかを選ぶには、1つ目のエディターを開きます。



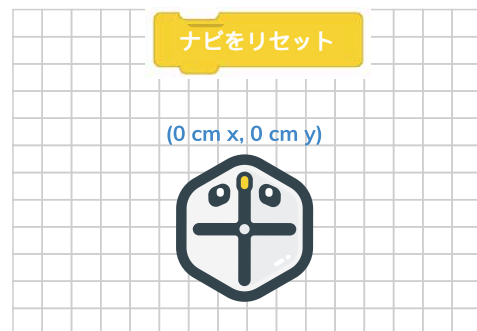
"音楽"ブロックの編集

音を出す長さを秒数で指定するには、2つ目のエディターを開きます。



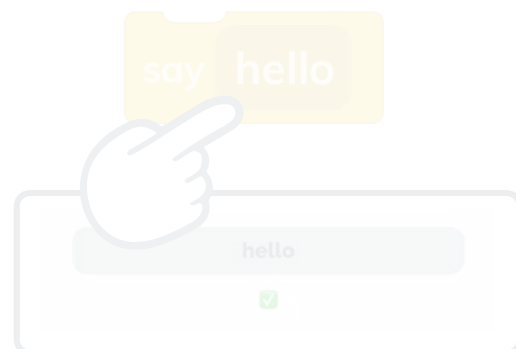
"ナビをリセット"ブロック

"ナビをリセット"ブロックは、目に見えない座標系の原点 (0センチ、0センチ) をRootの現在地点に再設定します。



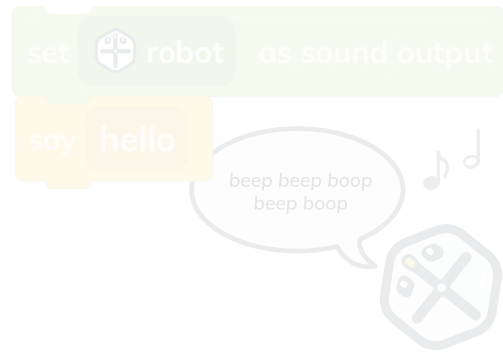
"しゅつりよく"ブロック

Rootまたは自分のデバイスが何か言葉を言うようにコーディングするには、"しゅつりよく"ブロックを使います。



Root言語

Rootが話すようにコーディングした場合、Rootは言葉をRoot言語に翻訳します。



"かいてんする"ブロック

"かいてんする"ブロックを使うと、Rootが左回りまたは右回りに回転するようにコーディングできます。



"かいてんする"ブロックの編集

Rootが左回りまたは右回りに回転するように指定するには、1つ目のエディターを使います。



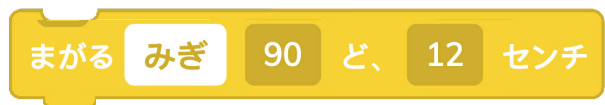
"かいてんする"ブロックの編集

Rootがどこまで回転するか、その角度を指定するには、2つ目のエディターを使います。



"まがる"ブロック

"まがる"ブロックを使うと、Robotが円などの形に沿って走行するように指定できます。



"まがる"ブロックエディター

"まがる"ブロックの角度は、Robotが円に沿ってどこまで移動するかを示します。



"まがる"ブロックエディター

"まがる"ブロックの半径は、円の大きさを示します。



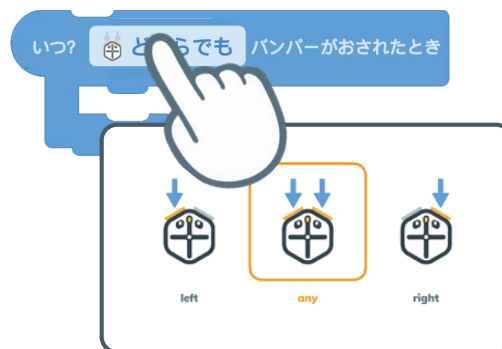
"バンパーがおされたとき"ブロック

"バンパーがおされたとき"ブロックは、指定のバンパーセンサーが押されたときに実行されます。



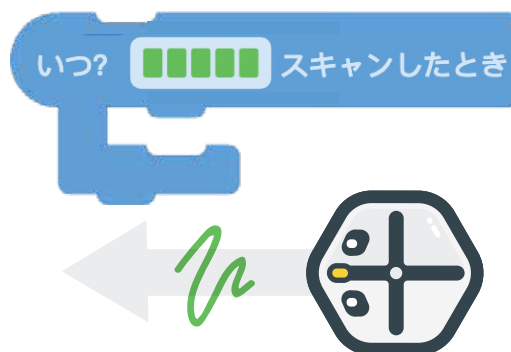
"バンパー"エディター

どのバンパーが押されたときにRootが反応するかを変更するには、このエディターを使います。



"色をスキャンしたとき"ブロック

選んだ色をRootが読み取って反応するようにコーディングするには、"色をスキャンしたとき"ブロックを使います。



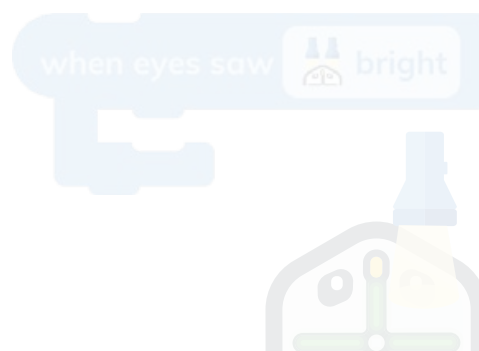
"カラーセンサー"エディター

このエディターを開くと、どのゾーンで何色を読み取るかを指定できます。



"明るさが変化したとき"ブロック

明るさの変化にRootが反応するようにコーディングするには、"明るさが変化したとき"ブロックを使います。



"ライト"エディター

Rootには、明るさの変化に反応するライトセンサーが2台あります。



"プログラムがかいしされたとき"ブロック

"プログラムがかいしされたとき"ブロックより後のコードは、開始ボタンをタップするとすぐに実行されます。



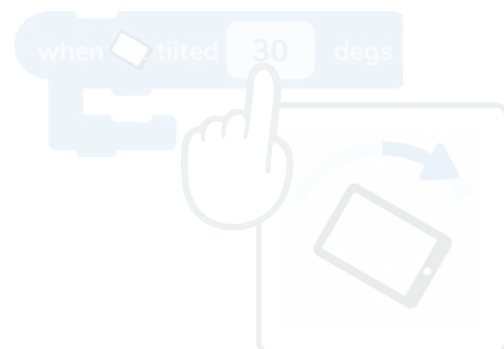
"かたむけたとき"ブロック

"傾けたとき"ブロックを使うと、デバイスをさまざまな方向に傾けたときにRootが反応するようにコーディングできます。



"傾き"エディター

Rootに反応させる傾きの角度を選ぶには、矢印をドラッグします。



"おされたとき"ブロック

"おされたとき"ブロックを使うと、Rootの表側にある4つのうち1つのゾーンを押したときにRootが反応するようにコーディングできます。



"タッチ"エディター

どのゾーンが押されたときにRootが反応するかを変更するには、このエディターを使います。



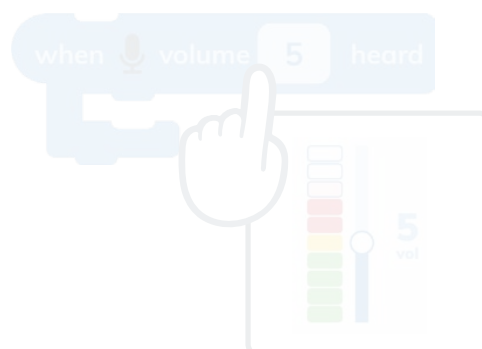
"音が聞こえたとき"ブロック

音量の変化にRootが反応するようにコーディングするには、"音が聞こえたとき"ブロックを使います。



"音"エディター

Rootが反応する音の大きさを変更するには、このエディターを使います。



条件

条件がtrueのときまたはfalseのときにどうするかを指定するには、条件ブロックを使います。



もし

ブロック内の式がtrueの場合は、このブロック内のコードが実行されます。falseの場合、プログラムはこのブロックを無視して、次へ進みます。



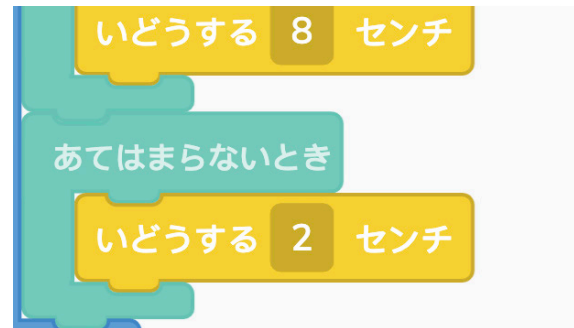
あてはまらないとき、もし

"もし"ブロック内の式がfalseの場合、プログラムは次の"あてはまらないとき、もし"に進みます。式がtrueの場合は、コードが実行されます。それ以外の場合、プログラムは次のブロックに進みます。



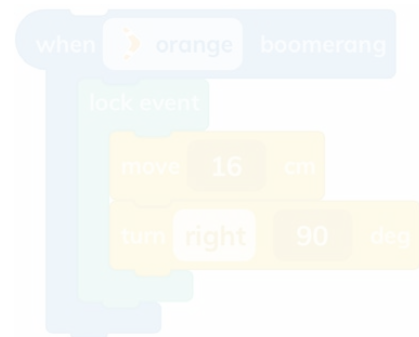
あてはまらないとき

これより上の"もし"ブロックとすべての"あてはまらないとき、もし"ブロックに含まれる式がfalseの場合は、"あてはまらないとき"ブロック内のコードが実行されます。



"イベントのロック"ブロック

Rootのプログラムが"イベントのロック"ブロックまで進むと、ブロック内のコードが完全に実行されます。他のイベントで遮られることはありません。



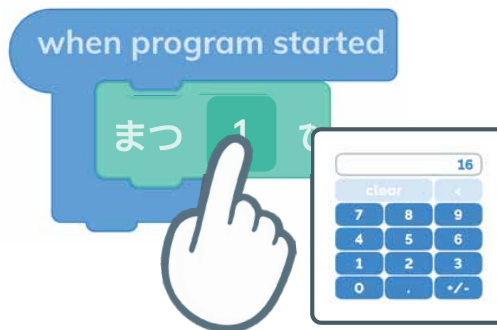
"くりかえす(かिसう)"ブロック

指定の回数だけ繰り返して実行するコードのループを作成するには、"くりかえす(かिसう)"ブロックを使います。



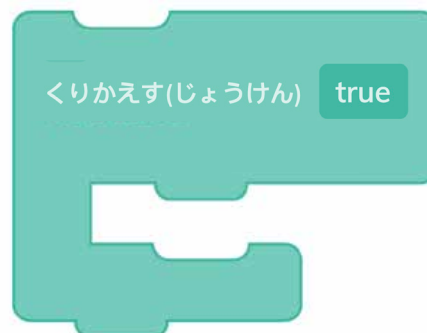
"まつ"ブロック

"まつ"ブロックでは、次のブロックに進むまでの時間の長さを設定できます。



"くりかえす(じょうけん)"ブロック

"まつ"ブロック内には、trueまたはfalseの式を設定できます。



"くりかえす(じょうけん)"規則

"くりかえす(じょうけん)"ブロック内の式がtrueの場合は、式がfalseになるまで、コードが繰り返されます。



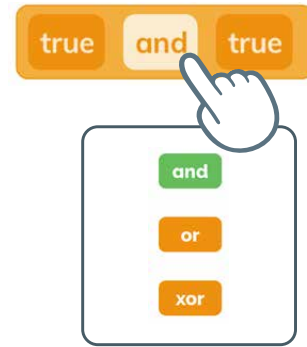
"あてはまらない"ブロック

"あてはまらない"ブロックでは、式の値が反転し、trueの場合はfalseに、falseの場合はtrueになります。



二重演算子ブロック

二重演算子ブロックは、内容の状態によってtrueまたはfalseになります。



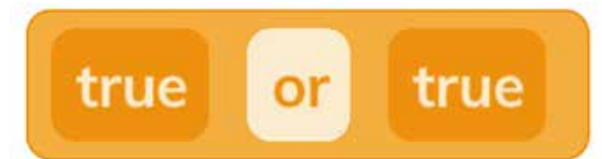
and

andブロックは、ブロック内のすべての式がtrueの場合にtrueを返します。



or

orブロックは、ブロック内のいずれかの式がtrueの場合にtrueを返します。



xor

xorブロックは、ブロック内の式の一方だけがtrueの場合にtrueを返します。



条件の使用

二重演算子ブロックと式を使うと、ブロック内の式がtrueのときまたはfalseのときにどうするかを指定できます。



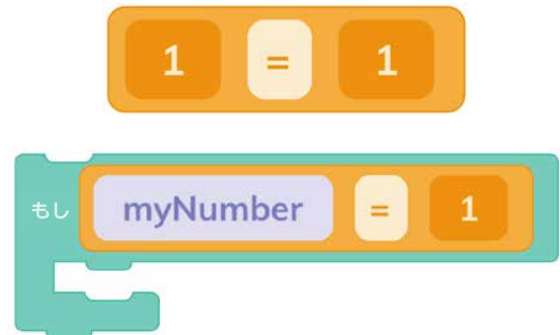
比較ブロック

比較ブロックは、両辺の大小関係によってtrueまたはfalseを返します。



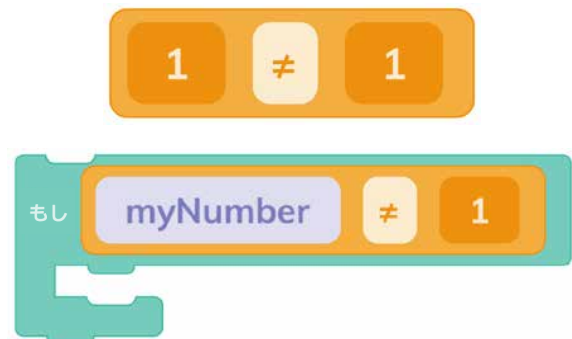
等しい

"等しい"ブロックは、1つ目の値が2つ目の値と等しいときにtrueを返します。



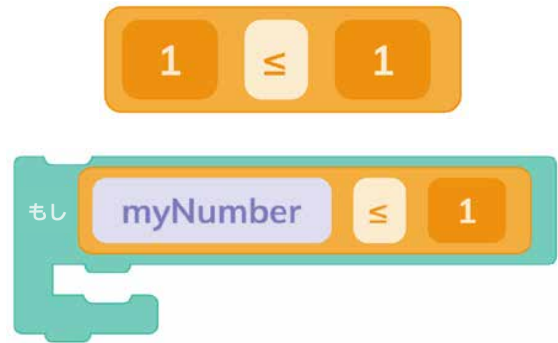
等しくない

"等しくない"ブロックは、1つ目の値が2つ目の値と等しくないときにtrueを返します。



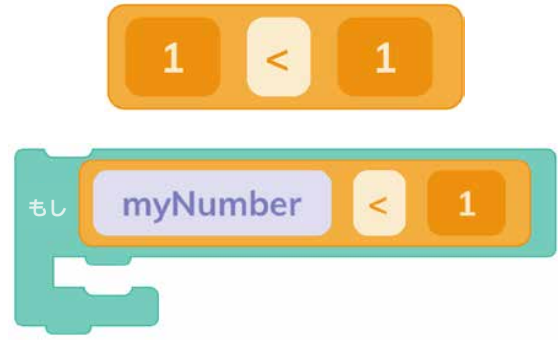
以下

"以下"ブロックは、1つ目の値が2つ目の値以下のときにtrueを返します。



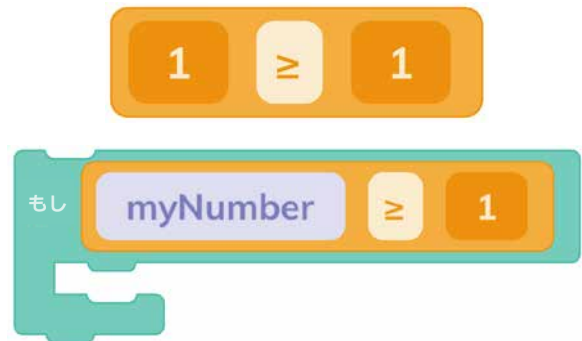
より小さい

"より小さい"ブロックは、1つ目の値が2つ目の値より小さいときにtrueを返します。



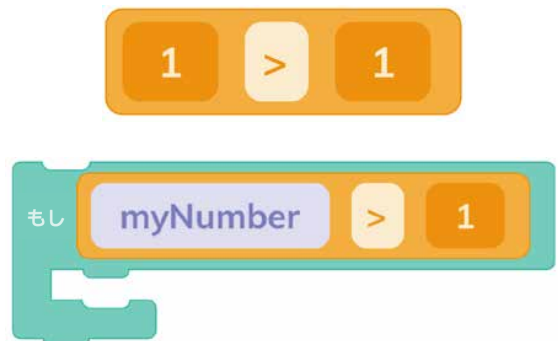
以上

"以上"ブロックは、1つ目の値が2つ目の値以上のときにtrueを返します。



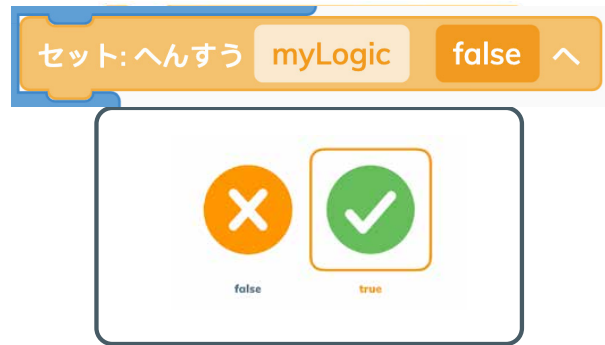
より大きい

"より大きい"ブロックは、1つ目の値が2つ目の値より大きいときにtrueを返します。



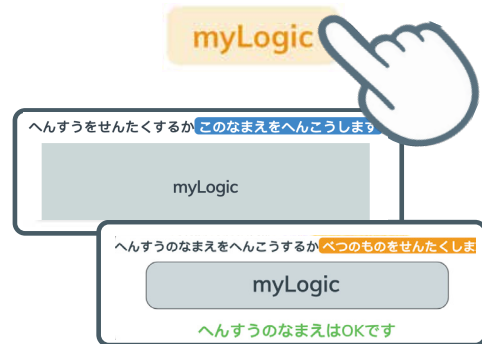
ブール変数の設定

何かが起こったとき(たとえばバンパーがタップされたとき)に、ロジック変数をtrueまたはfalseに変化させることができます。



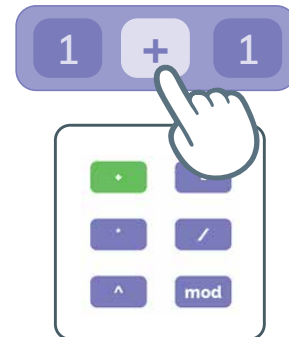
ブール変数

ロジック変数を使うと、trueとfalseを切り替えることができます。これによってRootは、プロジェクト内で何が起きているのかを記録できます。



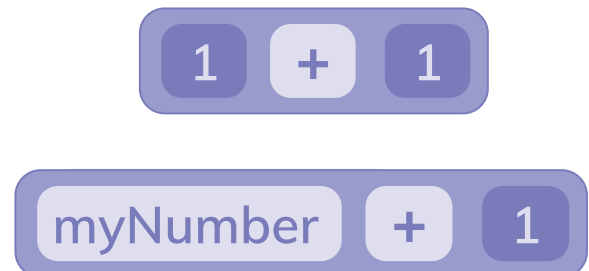
"等しい"ブロック

数学演算子ブロックは、両辺の間で選択された数学演算の結果を返します。



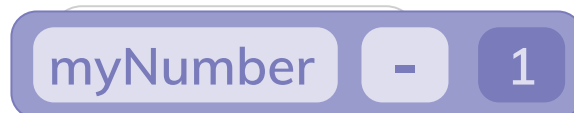
足し算

"足し算"ブロックは、2つの値の和に等しくなります。



引き算

"引き算"ブロックは、2つの値の差に等しくなります。



掛け算

"掛け算"ブロックは、2つの値の積に等しくなります。



累乗

"累乗"ブロックは、1つ目の値を2つ目の値で累乗した値と等しくなります。



剰余

"剰余"ブロックは、1つ目の値を2つ目の値で割ったときの余りと等しくなります。



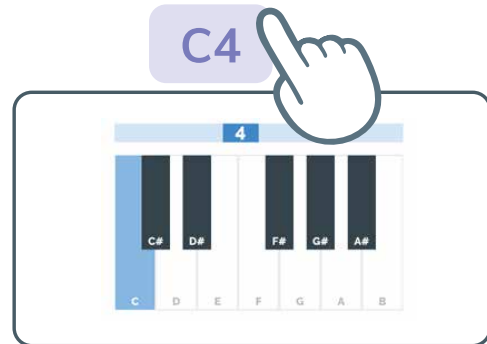
割り算

"割り算"ブロックは、1つ目の値を2つ目の値で割ったときの商と等しくなります。



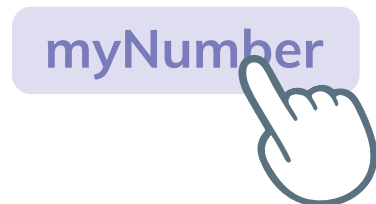
"音階周波数"ブロック

"音階周波数"ブロックは、指定された音階の周波数(ヘルツ単位)に等しくなります。



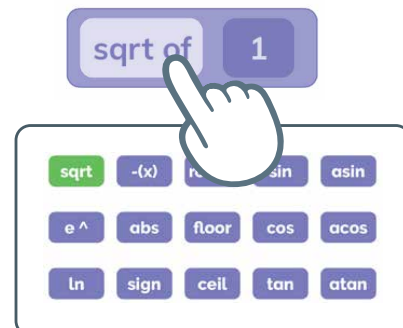
演算ブロック

演算ブロックは、ブロック内の値に対して実行された演算の解になります。



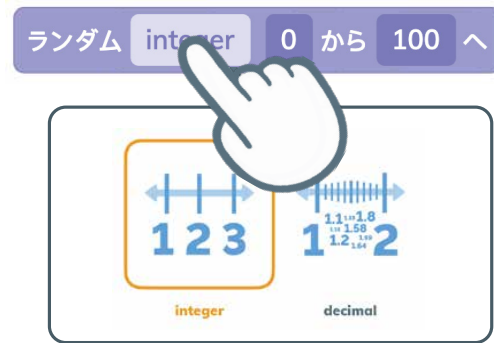
演算エディター

数学演算エディターを開くと、実行する数学演算を選ぶことができます。



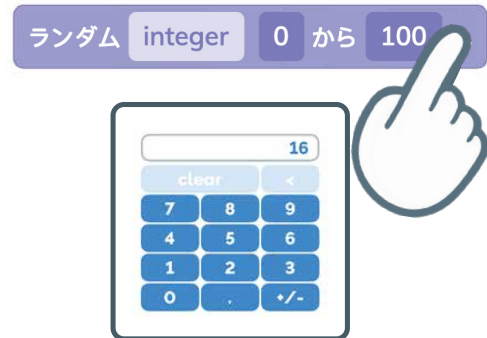
"ランダム"ブロック

"ランダム"ブロックは、指定された2つの値の間から、無作為の整数または小数を取り出します。



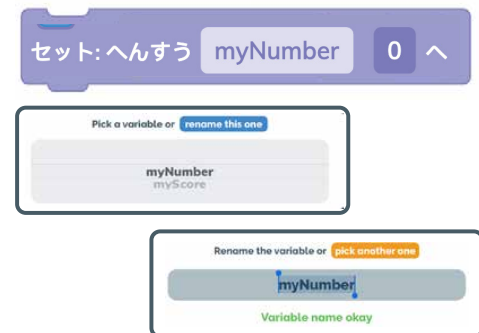
乱数の範囲

"ランダム"ブロックの範囲は、2つの数値エディターで変更できます。



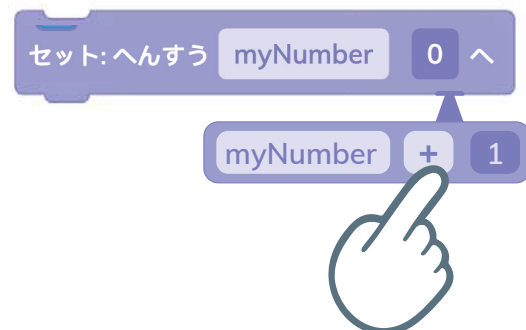
"セット: へんすう"ブロック

"セット: へんすう"ブロックは、"数値変数"ブロックに設定する数値を示します。



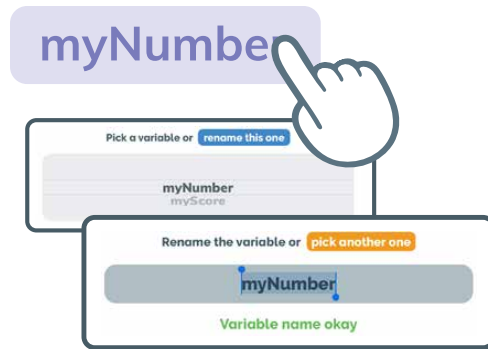
スコアの保持

"セット: へんすう"ブロックでは、数値または数学演算(スコアに1ポイントを加算する"足し算"ブロックなど)の結果を保持できます。



"変数"ブロック

数値変数には値を保持できます。
これによってRootは、プロジェクト
内で何が起きているのかを記録
できます。



"ライト"ブロック

"ライト"ブロックでは、Rootの表側
にあるライトについて、色や点滅パ
ターンを設定します。



"ライト"エディター

光の色はすべて、赤 (Red)、緑
(Green)、青 (Blue) の光の混合で
作られています。R、G、Bの値を別
の数値に設定すると、別の色を作る
ことができます。



"マーカー"ブロック

マーカーとイレーサーを上げ下げ
するには、"マーカー"ブロックを使
います。



"マーカー"エディター

マーカーまたはイレーサーの位置を選択するには、このエディターを使います。



"サウンド出力設定"ブロック

"サウンド出力設定"ブロックでは、Rootまたは自分のデバイスでサウンドを再生することを、コードに指定できます。



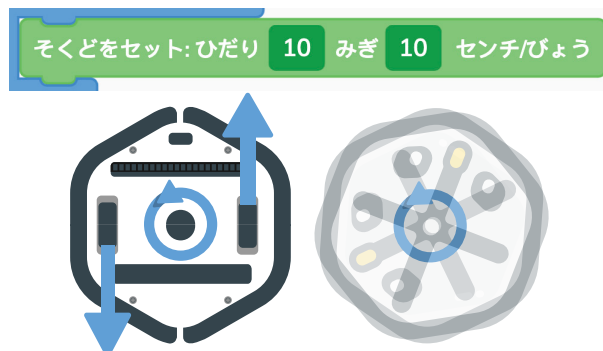
"車輪の速度"ブロック

車輪が動く速さとRootが回転する範囲の広さを変更するには、"車輪の速度"ブロックを使います。



"車輪の速度"エディター

Rootの車輪を逆回転させるには、速度を負の値に設定してみましょう。Rootはどのような動きになりますか？



レベル3ブロックの説明

robot.move(16)

robot.move(16)

robot.navigateTo(x:16 , y:16)

robot.navigateTo(x:16 , y:16)

play(Tone(freq:16 , duration:16))

play(Tone(freq:16 , duration:16))

robot.resetNavigation()

robot.say("hello")

robot.setSoundOutput(to: .robot)

robot.turn(right , 90)

robot.turn(right , 90)

robot.turn(right , 90)

boomerang(orange)

robot.whenBoomerang(.orange)

robot.whenBoomerang(.orange)

robot.whenBumperPressed([true, true])

robot.whenBumperPressed([true, true])

robot.whenColorScanned([color, color, color, color, color])

robot.whenColorScanned([green, green, green, green, green])

robot.whenEyesSee(.bright)

robot.whenEyesSee(.bright)

robot.whenProgramStarted()

robot.controllerTilted(30)

robot.controllerTilted(30)

robot.whenTouched([true, true, true, true])

robot.whenTouched([true, true, true, true])

robot.whenControllerHearsVolume({ louderThan: 5 })

robot.whenControllerHearsVolume({ louderThan: 5 })

robot.whenBoomerang(.orange)

robot.lockEvent {
 robot.move(16)
 robot.turn(right , 90)
}

for _ in 1...4 {
 robot.move(10)
 robot.turn(right, 90)
}

wait(1)

while true {

if myNumber > 10 {
 robot.move(10)
}

if myNumber > 10 {
 robot.move(10)
}

else if myNumber > 5 {
 robot.move(8)
}

else if myNumber > 5 {
 robot.move(8)
}

! true

1 == 1

1 != 1

1 == 1

1 < 1

1 >= 1

1 > 1

myLogic = true

myLogic

1 + 1

myNumber + 1

myNumber - 1

myNumber * 1

myNumber ^ 1 1 %

myNumber ^ 1

myNumber / 1

C4

sqrt(1)

sqrt(1)

myNumber

myNumber

myNumber = 0

myNumber + 1

myNumber

robot.lights(.on , Color(red: 1, green: 1, blue: 1))

robot.markerDown()

robot.markerDown()

robot.setSoundOutput(to: .robot)

robot.speeds(left: 10 , right: 10)

robot.speeds(left: 10 , right: 10)

move()

前または後ろにRootが動くようにセンチメートル単位でコーディングするには、move()メソッドを使います。



```
robot.move( 16 )
```

センチメートル

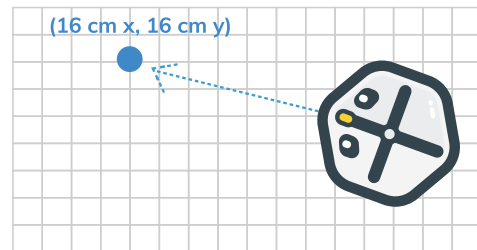
前または後ろにRootを何センチメートル動かすかを指定するには、このエディターを使います。



```
robot.move( 16 )
```

navigateTo()

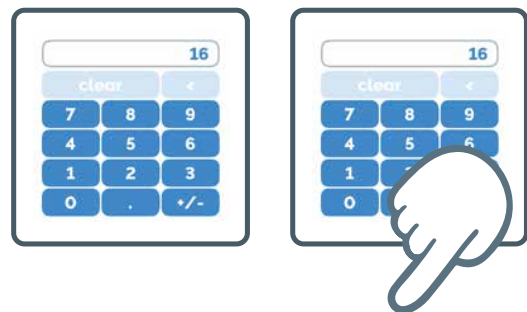
Rootの移動には、目に見えない座標系が使用されます。Rootのスタート地点として原点(0センチ, 0センチ)が設定されています。



```
robot.navigateTo( x:16 , y: 16)
```

navigateTo()

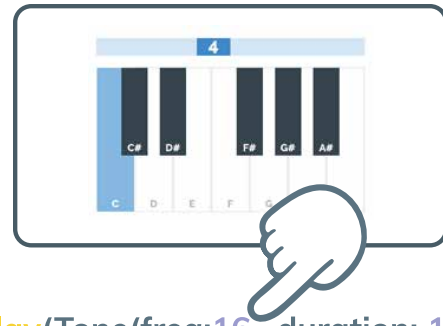
navigateTo()メソッドを使うと、ある座標位置にRootが移動するように指定できます。



```
robot.navigateTo( x:16 , y: 16)
```

play()

"音楽"メソッドを使うと、音を出すことができます。どの音階を鳴らすかを選ぶには、1つ目のエディターを開きます。



```
play(Tone(freq:16 , duration: 16))
```

duration

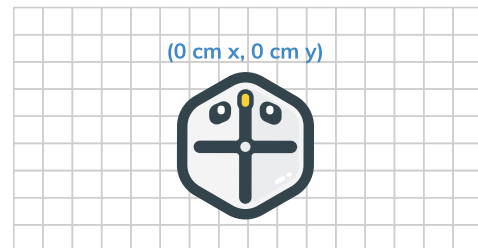
音を出す長さを秒数で指定するには、2つ目のエディターを開きます。



```
play(Tone(freq:16 , duration: 16))
```

resetNavigation()

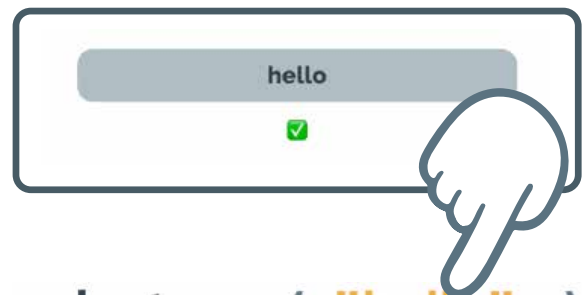
resetNavigation()メソッドは、目に見えない座標系の原点(0センチ、0センチ)をRootの現在地点に設定します。



```
robot.resetNavigation()
```

say()

Rootまたは自分のデバイスが何か言葉を言うようにコーディングするには、say()メソッドを使います。

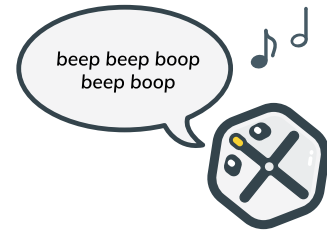


```
robot.say( "hello" )
```

Root言語

Rootが話すようにコーディングした場合、Rootは言葉をRoot言語に翻訳します。

```
robot.say( "hello" )  
robot.setSoundOutput(to: .robot )
```



turn()

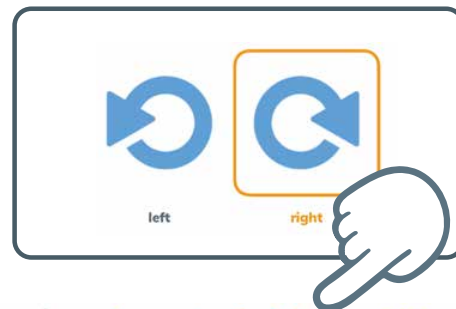
turn()メソッドを使うと、Rootが左回りまたは右回りに回転するようにコーディングできます。



```
robot.turn( right , 90 )
```

方向

Rootが左回りまたは右回りに回転するように指定するには、1つ目のエディターを使います。



```
robot.turn( right , 90 )
```

角度

Rootが何度まで回転するか、その角度を指定するには、2つ目のエディターを使います。



```
robot.turn( right , 90 )
```

arc()

Rootのプログラムがブーメラン関数まで進むと、そのときの"ブーメランに到達したとき"イベントに応じて、関数内のコードが実行されます。

```
robot.arc( right , angle:90 , radius:12 )
```

arc() angle

"まがる"ブロックの角度は、ロボットが円に沿ってどこまで移動するかを示します。

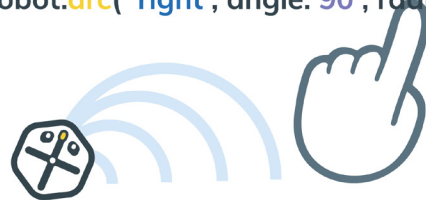
```
robot.arc( right , angle: 90 , radius: 12 )
```



arc() radius

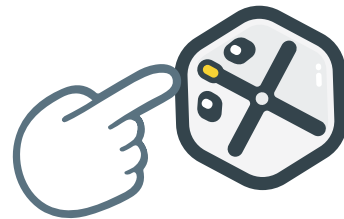
"まがる"ブロックの半径は、円の大きさを示します。

```
robot.arc( right , angle: 90 , radius: 12 )
```



whenBumperPressed()

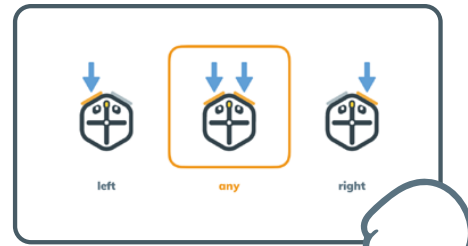
"バンパーがおされたとき" (whenBumperPressed()) イベントは、バンパーセンサーが押されたことをRootに伝えます。



```
robot.whenBumperPressed  
( [true, true] ) {
```

"バンパー"エディター

どのバンパーが押されたときにRootが反応するかを変更するには、このエディターを使います。



```
robot.whenBumperPressed  
( [true, true] ) {
```

whenColorScanned()

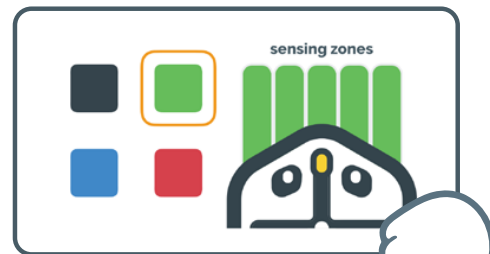
選んだ色をRootが読み取って反応するようにコーディングするには、"色をスキャンしたとき"イベントを使います。



```
robot.whenColorScanned  
( [color, color, color, color, color] ) {
```

"カラーセンサー"エディター

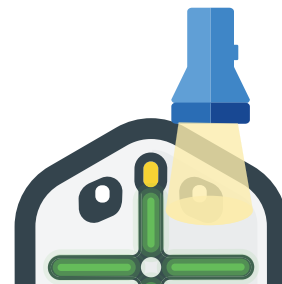
このエディターを開いて、どのゾーンで何色を読み取るかを指定します。



```
robot.whenColorScanned  
( [green, green, green, green, green] ) {
```

whenEyesSaw()

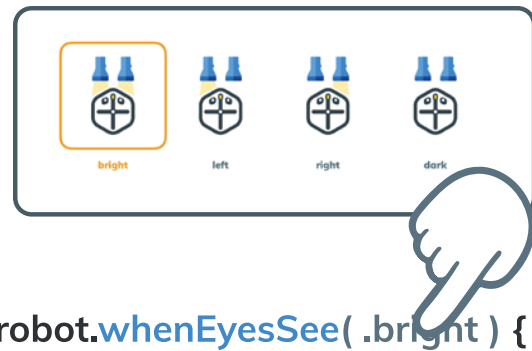
明るさの変化にRootが反応するようにコーディングするには、"いつ? あかるさ" (whenEyesSaw()) イベントを使います。



```
robot.whenEyesSee( .bright ) {
```

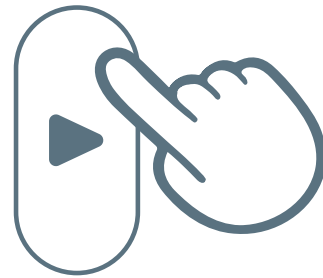
"ライト"エディター

Rootには、明るさの変化に反応するライトセンサーが2台あります。



whenProgramStarted()

"プログラムがかいしされたとき" (whenProgramStarted()) イベントより後のコードは、開始ボタンをタップするとすぐに実行されます。



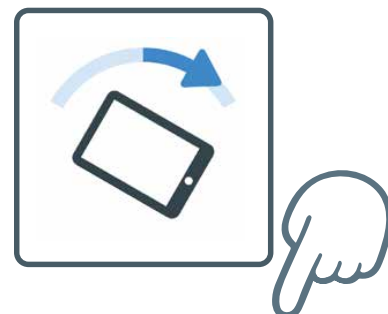
whenControllerTilted()

"コントローラーが傾いたとき" (whenControllerTilted()) イベントを使うと、デバイスをさまざまな方向に傾けたときにRootが反応するようにコーディングできます。



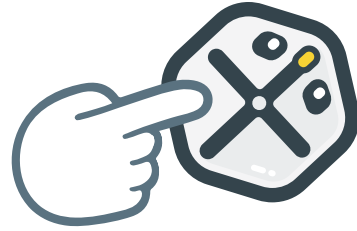
"傾き"エディター

Rootに反応させる傾きの角度を選ぶには、矢印をドラッグします。



whenTouched()

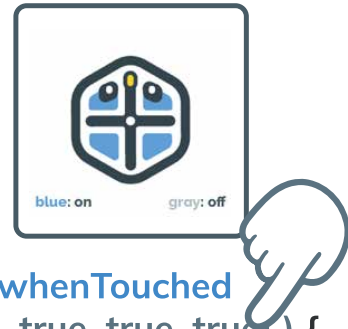
"おされたとき" (whenTouched()) イベントを使うと、Rootの表側にある4つのうち1つのゾーンを押したときにRootが反応するようにコーディングできます。



```
robot.whenTouched  
( [true, true, true, true] ) {
```

"タッチ"エディター

どのゾーンが押されたときにRootが反応するかを変更するには、このエディターを使います。



```
robot.whenTouched  
( [true, true, true, true] ) {
```

whenControllerHeardVolume()

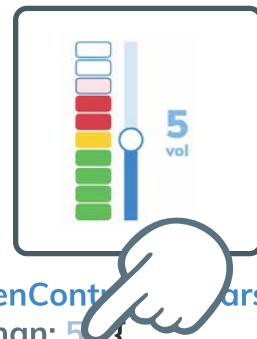
音量の変化にRootが反応するようにコーディングするには、"コントローラーにボリュームが聞こえたとき" (whenControllerHeardVolume()) ブロックを使います。



```
robot.whenControllerHearsVolume  
( louderThan: 5 ) {
```

"音"エディター

Rootが反応する音の大きさを変更するには、このエディターを使います。



```
robot.whenControllerHearsVolume  
( louderThan: 5 ) {
```

条件文

条件がtrueのときまたはfalseのときにどうするかを指定するには、条件文を使います。



```
if random (integer, from: 0, to: 100) > 5 {  
  robot.lights( .on, Color(red: 100, green: 45, blue: 0))  
}  
else {  
  robot.lights( .on, Color(red: 0, green: 0, blue: 100))  
}
```

if true { }

if文内の式がtrueの場合、プログラムはかっこ内の文を実行します。式がfalseの場合、プログラムは次の文に進みます。

```
if myNumber > 10 {  
  robot.move( 10 )  
}
```


else if true { }

if文内の式がfalseの場合、プログラムは次のelse if文に進みます。式がtrueの場合は、コードが実行されます。それ以外の場合、プログラムは次の文に進みます。

```
if myNumber > 10 {  
  robot.move( 10 )  
}  
else if myNumber > 5 {  
  robot.move( 8 )  
}
```

else { }

これより上のif文とすべてのelse if文に含まれる式がfalseの場合は、else文内のコードが実行されます。

```
else if myNumber > 5 {  
  robot.move( 8 )  
}  
else {  
  robot.move( 2 )  
}
```

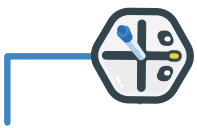
lockEvent { }

RootがlockEvent文を見つけると、この文に含まれるコードは、他のイベントで遮られることなく完了されます。

```
robot.whenBoomerang( .orange ) {  
  robot.lockEvent {  
    robot.move( 16 )  
    robot.turn( right , 90 )  
  }  
}
```

for _ in 1 ... x { }

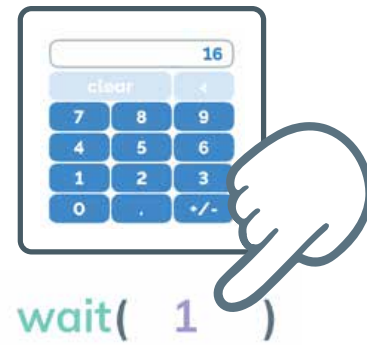
x回繰り返されるコードのループを作成するには、forを使います。



```
for _ in 1 ... 4 {  
  robot.move( 10 )  
  robot.turn( right, 90 )  
}
```

wait()

wait文を使うと、次のブロックに進むまでの時間の長さを設定できます。



while true { }

while文内には、trueまたはfalseの式を設定できます。



whileループ

含まれている式がtrueの場合は、式がfalseになるまで、コードが繰り返されます。

```
robot.whenProgramStarted {  
  while myNumber == 1 {  
    robot.speeds(left: 10 , right: 10 )  
  }  
}
```

!x

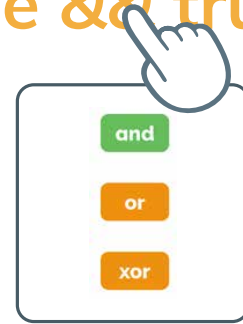
not演算子は、式の値を反転させます。その結果、trueの場合はfalseに、falseの場合はtrueになります。



二重演算子ブロック

ブール演算子は、パラメーターの状態によってtrueまたはfalseになります。

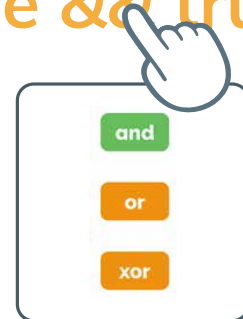
true && true



$x \&\& y$

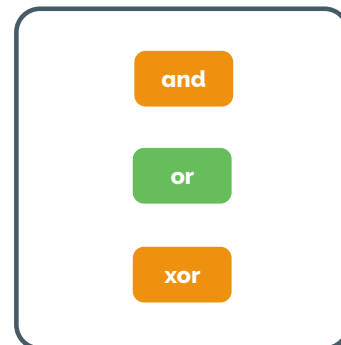
and演算子は、すべての式がtrueの場合にtrueを返します。

true && true



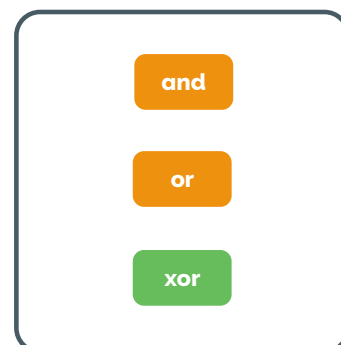
$x \parallel y$

or演算子は、いずれかの式がtrueの場合にtrueを返します。



$x \wedge y$

"xor"演算子は、式的一方だけがtrueの場合にtrueを返します。



条件の使用

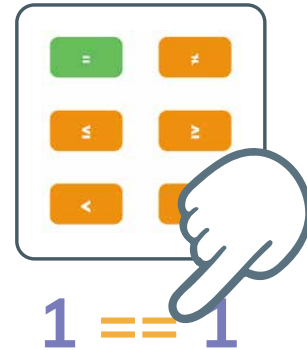
ブール演算子と条件を使うと、式がtrueのときまたはfalseのときにどうするかを指定できます。

```
if (myNumber < 1 && theirNumber > 10) {
```



比較演算子

比較演算子は、パラメーターの状態によってtrueまたはfalseを返します。



x == y

"等しい"演算子は、1つ目の値が2つ目の値と等しいときにtrueを返します。

```
if (myNumber == 1) {
```

1 == 1

x != y

"等しくない"演算子は、1つ目の値が2つ目の値と等しくないときにtrueを返します。

```
if (myNumber != 1) {
```

1 != 1

$x \leq y$

"以下"演算子は、1つ目の値が2つ目の値以下のときにtrueを返します。

if `myNumber` \leq 1 {

1 \leq 1

$x < y$

"より小さい"演算子は、1つ目の値が2つ目の値より小さいときにtrueを返します。

if `myNumber` $<$ 1 {

1 $<$ 1

$x \geq y$

"以上"演算子は、1つ目の値が2つ目の値以上のときにtrueを返します。

if `myNumber` \geq 1 {

1 \geq 1

$x > y$

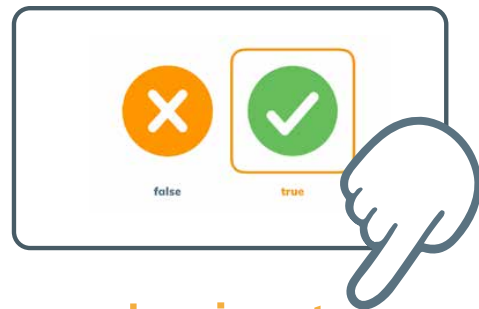
"より大きい"演算子は、1つ目の値が2つ目の値より大きいときにtrueを返します。

if `myNumber` $>$ 1 {

1 $>$ 1

myLogic = x

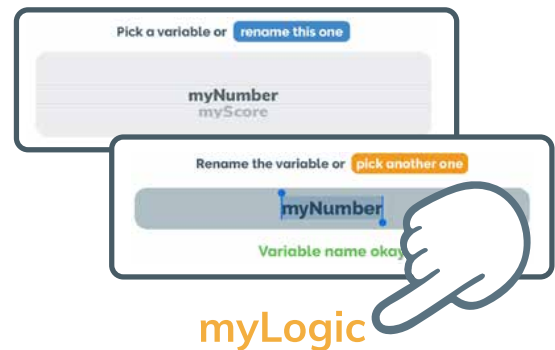
何かが起こったとき(たとえばバンパーがタップされたとき)に、ブール変数をtrueまたはfalseに設定するには、"ブール変数の設定"文を使います。



myLogic = true

ブール変数

ブール変数を使うと、trueとfalseを切り替えることができます。これによってRootは、プロジェクト内で何が起きているのかを記録できます。

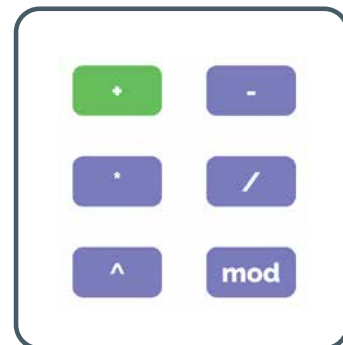


myLogic

数学演算子

数学演算子は、両辺の間で指定された数学演算の結果を返します。

1 + 1



$x + y$

"足し算"演算子は、2つの値の和を返します。

myNumber + 1

$x - y$

"引き算"演算子は、2つの値の差を返します。

myNumber - 1

$x * y$

"掛け算"演算子は、2つの値の積を返します。

myNumber * 1

$x ^ y$

"累乗"演算子は、1つ目の値を2つ目の値で累乗した値を返します。

myNumber ^ 1

$x \% y$

"剰余"演算子は、2つの値を使った乗算の余りを返します。

1 % (myNumber ^ 1)

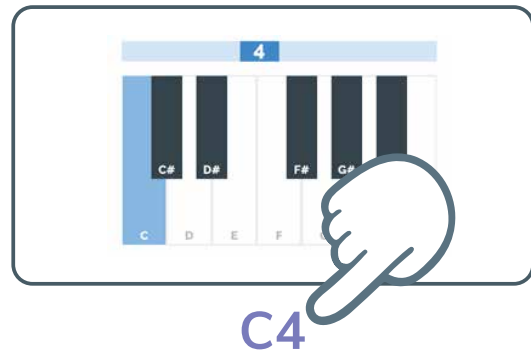
x / y

"割り算"演算子は、2つの値の商を返します。

myNumber / 1

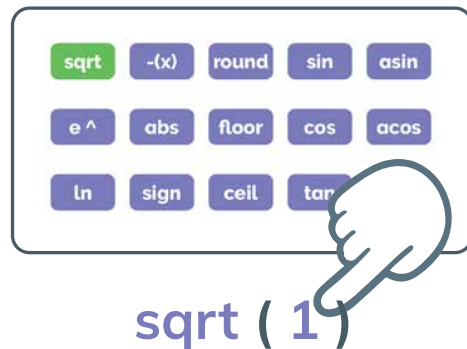
"音階周波数"ブロック

"音階周波数"定数は、指定された音階の周波数(ヘルツ単位)に等しくなります。



数学関数

数学関数は、関数内で値に対して実行された関数の解を返します。



sqrt (1)

関数エディター

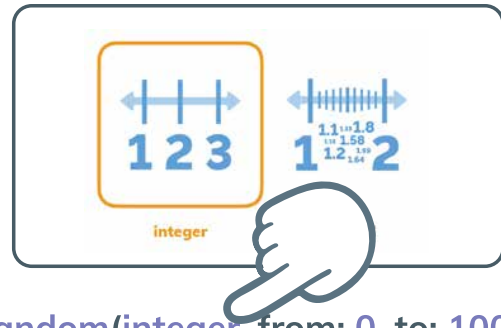
数学関数エディターを開くと、実行する関数を選ぶことができます。

sqrt (1)
myNumber



random()

random関数は、指定された2つの値の間から、無作為の整数または小数を取り出します。



`random(integer, from: 0, to: 100)`

乱数の範囲

random関数に使用する範囲は、2つの数値エディターで変更できます。



`random(integer, from: 0, to: 100)`

myNumber = x

"浮動変数の設定"文は、対応するmyNumber変数の値を定義します。



`myNumber = 0`

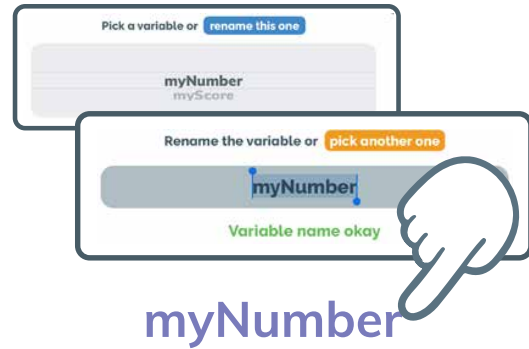
スコアの保持

浮動変数は、数値または演算（スコアに1ポイントを加算する"myNumber + 1"など）を保持できます。

`myNumber = 0`
`myNumber + 1`

浮動変数

浮動変数には値を保持できます。
これによってRootは、プロジェクト
内で何が起きているのかを記録
できます。



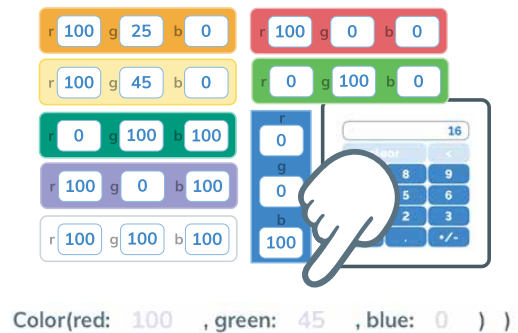
lights()

Rootの表側にあるライトについ
て、色や点滅パターンを設定する
には、lights()メソッドを使います。



"ライト"エディター

光の色はすべて、赤 (Red)、緑
(Green)、青 (Blue) の光の混合で
作られています。R、G、Bの値を別
の数値に設定して、どうなるか試し
てみましょう。



markerDown()

マーカーとイレーサーを上げ下げ
するには、markerDown()メソッド
を使います。



"マーカー"エディター

マーカーまたはイレーサーを上げたり下げたりするには、このエディターを使います。



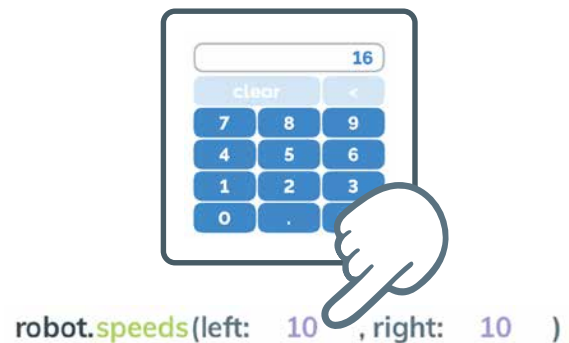
soundOutput()

soundOutput()メソッドでは、Rootまたは自分のデバイスでサウンドを再生することを、コードに指定できます。



speeds()

車輪が動く速さとRootが回転する範囲の広さを設定するには、speeds()メソッドを使います。



"車輪の速度"エディター

Rootの車輪を逆回転させるには、速度を負の値に設定してみましょう。Rootはどのような動きになりますか？

