

スクラッチ Scratch のこと 変数 と リスト

CoderDojo Aizu

齋藤文康

2021 年 6 月 13 日

概要

スクラッチ 変数やリストについて、性質や使い方をスクリプトを使って説明してみました。
変数やリストはいろいろな場面で使用されますから、「動き」「見た目」「音」「イベント」「制御」
「調べる」「演算」「ブロック定義」全部にかかわってしまいます。小学生には難しい内容かもし
れませんが、興味があるところだけでも読んでもらえればと思います。うまく説明できている
かは分かりませんが、自分でスクリプトを組んで確かめながらやってみてください。

なお、実行環境の違いにより、スクリプトのブロック表示が Windows のものとは違っている
かもしれません。

目次

1	変数 ^{へんすう} のこと	4
1.1	変数 ^{へんすう} の作成 ^{さくせい}	7
1.2	変数 ^{へんすう} の演算 ^{えんざん}	9
1.3	数 ^{かず} の不思議 ^{ふしぎ}	15
1.4	変数 ^{へんすう} の表示 ^{ひょうじ} について	18
1.4.1	「大きな表示 ^{おおひょうじ} 」について	18
1.4.2	「スライダー ^{ひょうじ} 」表示 ^{ひょうじ} について	20
1.5	変数 ^{へんすう} の値 ^{あたい} の保存 ^{ほぞん}	23
1.6	$0 + 1 = ???$	25
2	リストのこと	28
2.1	リストの作成 ^{さくせい}	28
2.2	ブレークポイント	31
2.3	一回 ^{いつかい} 押 ^お しただけなのに ...	34
2.4	リストへの間違 ^{まちが} った位置 ^{いち} 指定 ^{してい}	35
2.5	リストの内容 ^{ないよう} をファイルから読み込 ^よ み込む ^こ	35
3	変数 ^{へんすう} やリストのオプション	37
3.1	変数 ^{へんすう} による不 ^ふ 具 ^ぐ 合 ^{あい}	37
3.2	リストを使 ^{つか} った音楽 ^{おんがく} 演奏 ^{えんそう}	42
3.3	クローンと変数 ^{へんすう}	46
4	ブロック ^{ていぎ} 定義 ^{ひきう} の引数	47
5	変数 ^{へんすう} やリストのリミット	50

参考作品^{さんこうさくひん} : scratch.mit.edu のサイトの検索窓^{けんさくまど}から ezushi で検索^{けんさく}してください。

イベントについて

変数やリストのことの前にちょっとイベントのお話です。

Scratch に何かをさせるには、イベントを起こす、発生させる必要があります。イベントは次のような時に発生します。

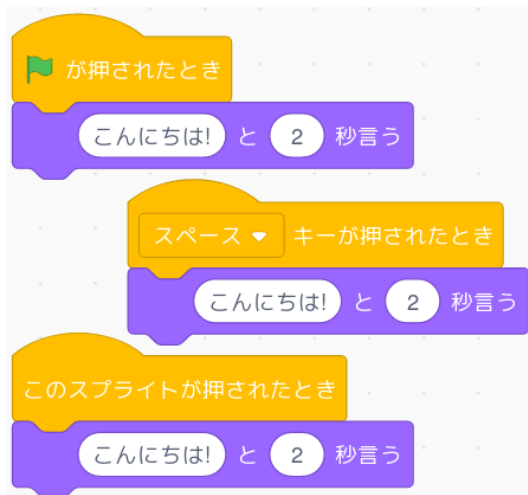
- ブロックやスクリプトをクリックする
- スプライトをクリックする
- 何かキーを押す
-  をクリックする
-  をクリックする
- 背景や音量が変わる
- メッセージを受け取る

見た目

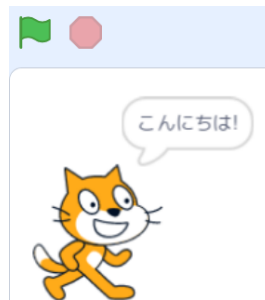
たとえば、



をクリックすると実行されます。



このようなスクリプトになっていたなら、スプライトをクリックしても、スペースキーを押しても、 をクリックしても同じように実行されます。



それぞれ、そのイベントごとに指定されされたスクリプトが実行されます。

勘違いしやすいのですが、 は、プログラムを実行させるボタンではなく、

 が押されたとき のところにあるスクリプトを実行させるイベントを発生させるものだという事です。

 が押されたとき がないスクリプトでは  をクリックしても何も実行されません。

変数やリストのこと

変数やリストは数などを記憶させておく入れ物です。文字も文字コード(数値)で扱うので、入れておくことができます。

1 変数のこと

Scratch では、スプライトごとに状態を表している変数が用意されています。

「x座標」「y座標」という変数は、スプライトの位置を表しています。画面の座標の値はセンターを0として横(x)の位置は-240 ~ 240、縦(y)の位置は-180 ~ 180の範囲になります。変数の左側のチェックボックスにチェックを入れると、変数の値が画面に表示されます。



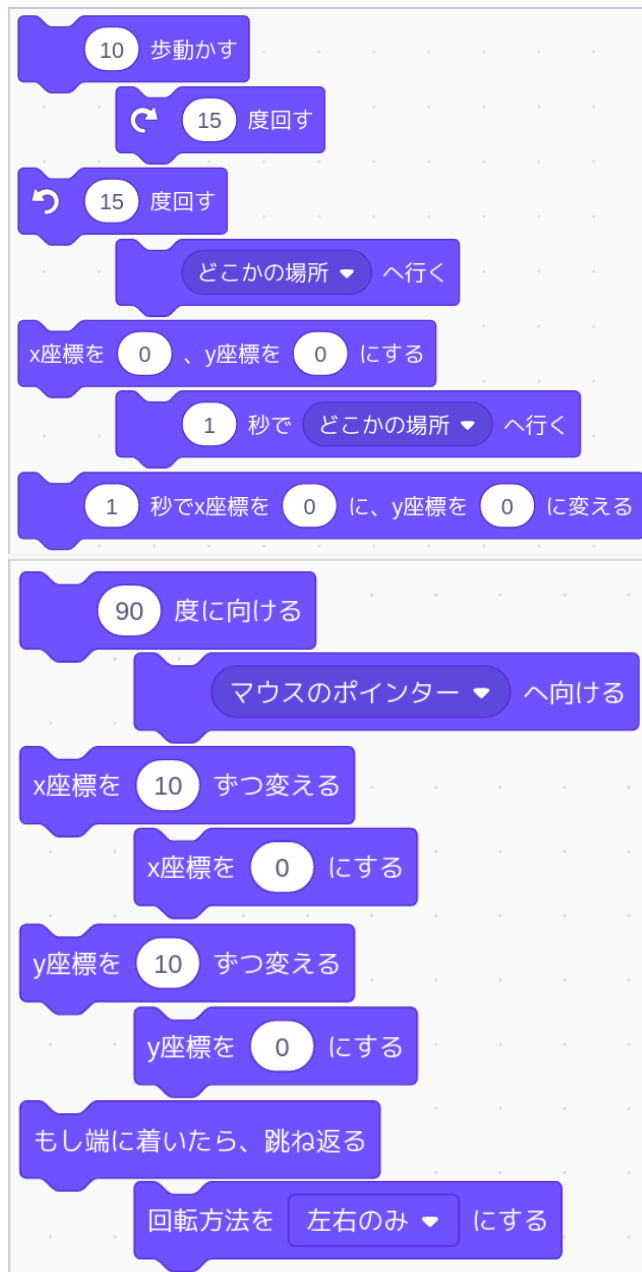
左側にチェックボックスが付いていて、変数の値を表示できるものをすべて表示してみます。



変数名に「スプライト 1:」のようにスプライト名が付いているものは、そのスプライト専用のものです。たとえば、「スプライト 1: 音量」を50にしてもこれはこのスプライトだけのもので、他のスプライトの音を調整するにはそのスプライトのスクリプトで音量設定をしなければなりません。

「x座標」「y座標」に画面上の座標の値を入れると、たとえば x座標を 100、y座標を 100 にする で、スプライトがそこに移動します。逆に、スプライトをドラッグして表示させたい位置に移動させると、「動き」の中にある x座標を 、y座標を にする などにその位置の値が入ります。それをドラッグしてきてスクリプトを組みれば楽です。

スプライトの位置や動かし方に関する操作は「動き」の中のブロックでします。



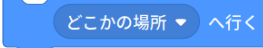
回転方法を 左右のみ にする は、もし端に着いたら、跳ね返る をしても逆さまにしない機能です。

「演算」の中に、 というのがあります。乱数というのは、さいころをふって出た目のようなものです。乱数で「x座標」や「y座標」の画面上の範囲を指定してやると、画面上のどこかにスプライトを移動させることができます。



「イベント」のところにある  を最初にもってきます。「x座標」や「y座標」を指定するブロックの後に、「制御」のところにある  をおいて、これを繰り返すために  で囲みます。

 をクリックすることで、 のところにあるスクリプトが実行されます。 をクリックすると、プログラムを終了させることができます。

「動き」の中のブロックには、 というブロックがあるのでそれでもよいのですが、これだとスプライトの一部が画面からはみ出してしまうと全身が表示されないことがあります。

数値を入れる時に、入力モードが全角だと文字になってしまうので、Scratchは何も入っていない、つまり0として扱うようです。日本語入力をローマ字変換で使っている人は注意してください。上が半角モードで、下が全角モードで入れたようすです。大きさがちがいます。



乱数をいろいろなブロックに入れてみます。画面の左下隅の  をクリックしてペンの機能を追加してください。



画面をきれいにするには、「ペン」のところにある「全部消す」をクリックしてください。「90度に向ける」「大きさを 100% にする」「色の効果を 0 にする」でもとに戻せます。

1.1 変数の作成

次に、新しい変数を作ってみます。「変数」をクリックすると、「変数を作る」「リストを作る」が表示されます。「変数を作る」をクリックしてください。



変数名のところに a という名前を入れて、a という変数を作ります。名前は、それを見ると変数の使い方が分かるようにつけるのが理想です。たとえば、ゲームの得点を記録するためなら「得点」とか「スコア」とかです。ここではただちょっと変数の説明をするだけなので、a, b, c, d を使います。

「すべてのスプライト用」「このスプライトのみ」のオプションがありますが、これについては「変数とリストのオプション」のところで説明します。



そうすると、この変数に対する操作のブロックが表示されて、値を入れたり増やしたりできるようになります。変数は、作成された時に 0 に初期設定されます。

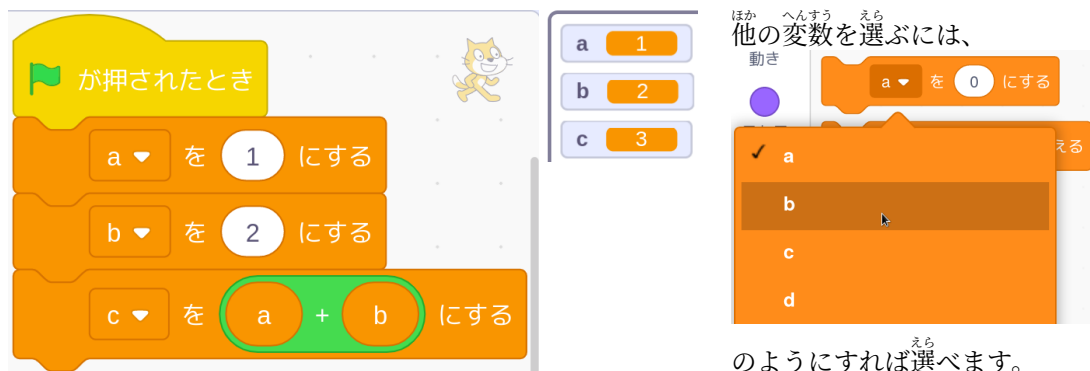


変数 a の左側にチェックが入っています。これは、この変数が画面上に表示されることを表しています。クリックしてこのチェックをはずすと、表示されなくなります。この変数を使う場合は、命令ブロックを使う時のようにこの a をドラッグしてきて、こんなふうに入れてやりま



1.2 変数の演算

a に続き b, c, d という変数を作ってください。次のようにスクリプトを組み、 a を 1 にし、 b を 2 にしてください。c に「演算」の中の $+$ を使って、 $a + b$ を入れます。



実行すると、 c は 3 になります。
数の 1 と 2 を足したのですから 3 ですね。

cに「演算」の中の  を使って「a と b」を入れると、



cは、12になります。これは、aとbに入っている1と2を文字「1」「2」として扱った結果です。

dにc + aを入れると、今度はcを数として扱って、12 + 1なので13になります。
全角の数字はだめですが、半角のものは変換できるようです。



他のプログラミング言語では数値を記憶させる変数は数値だけ、文字を記憶させる変数は文字だけにしか使えないようになっているものがあります。

数の足し算は  で、文字の足し算（文字をつなげる）は  を使います。
数の掛け算は  で、割り算は  です。掛け算は見にくいですが「*(アスタリスク)」です。他にも次のようなものがあります。



を で割った余り を使うと、奇数偶数のテストができます。2 で割り切れる数は偶数で、割り切れない数は奇数です。2 で割った余りが 0 ならば偶数ということです。

「調べる」のところの と聞いて待つ を使うと、キーボードから入力したものが 答え という変数に入ります。 答え を 2 で割った余りが 0 ならば偶数、そうでないならば奇数です。



「制御」のところにあり、 のところにテストをする項目を

入れると、テストの結果にしたがって別なスクリプトを実行させることができます。テストに使えるブロックは六角形になっています。

答え が 0、つまり 0 と等しいならばのチェックは、 です。

答え が 0 よりも大きいならばのチェックは、 です。

答え が 0 よりも小さいならばのチェックは、 です。

もし、 でチェックする場合は、



になります。 は、「見た目」のところにあります。

もし、 でチェックする場合は、



になります。

全体のプログラムです。



[チャレンジ] 「数当てゲーム」というものがあります。出題する側の A 君(スクリプト)が 1 ～ 100 までの数を考え、答える側の B 君が適当な数を言います。A 君はその数より、もっと大きいとか、もっと小さいとヒントを出します。これを当たるまで繰り返します。ここまでに出てきたもので作れると思います。

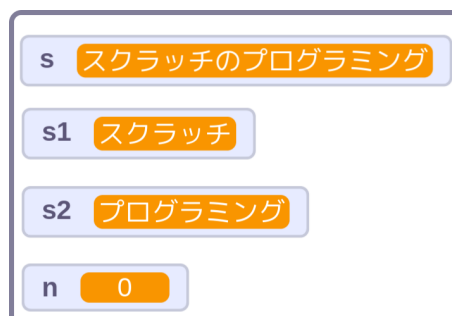
次の例は、「スクラッチのプログラミング」ということばが入っている変数 s から、「の」の左側「スクラッチ」を変数 s1 に、「の」の右側「プログラミング」を変数 s2 に入れていくものです。変数 s, s1, s2, n を作成してください。



次のようにスクリプトを組んでください。変数 s1, s2 の値を「」空にしてから、文字を追加していきます。s1 を 0 にするように「0」のところをクリックすると色が変わりますから、BS キーか DEL キーを押すと「」空の文字になります。



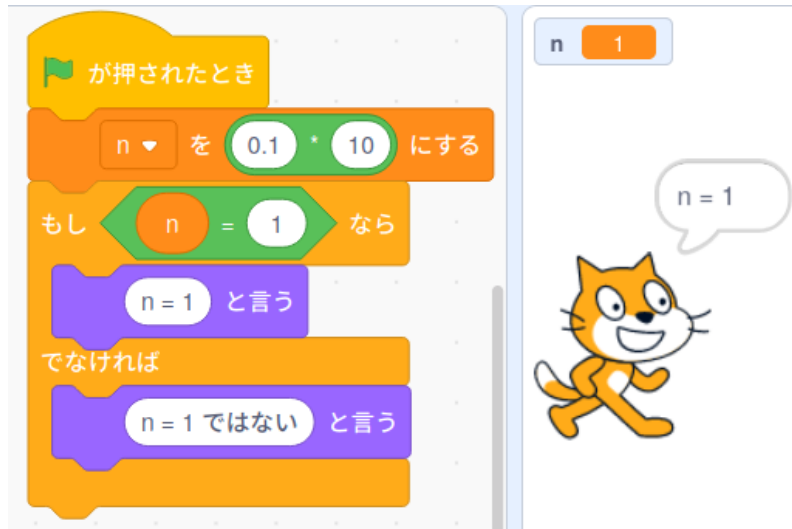
じつこう
実行するとこのようになります。



〔チャレンジ〕 上の s に「123+456」を入れて、「+」の左側の数を s1 に、右側の数を s2 に取り出して、足した数を s3 という変数に入れるように変更してください。

1.3 数の不思議

パソコンでは小数を正しく扱えない場合があります。次のスクリプトは 0.1×10 が 1 になるかを調べるものです。



これはだいじょうぶでした。

次のスクリプトは 0 に 0.1 を 10 回加えて 1 になるかを調べるものです。



変数 n の値の表示は 1 になっているのに「 $n = 1$ ではない」と表示されます。これはパソコンの中で、数を浮動小数点という仕組みの二進数で扱うためにおこることです。

これはこの^{しくみ}仕組み^{つか}を使っている他の^{ほか}プログラミング^{げんご}言語^{もんたい}でもおこる問題です。

```
fn main() {
    let mut n = 0.1 * 10.0;
    print!("{}", n);
    if n == 1.0 {
        println!("n = 1 です");
    } else {
        println!("n = 1 ではありません");
    }
    n = 0.0;
    for i in 1 .. 11 {
        n += 0.1;
        println!("{0: >2} 回目 n = {1: <}", i, n);
    }
    print!("{}", n);
    if n == 1.0 {
        println!("n = 1 です");
    } else {
        println!("n = 1 ではありません");
    }
}
```

^{じつこうけつ}
[実行結果]

```
(n = 1) : n = 1 です
1 回目 n = 0.1
2 回目 n = 0.2
3 回目 n = 0.30000000000000004
4 回目 n = 0.4
5 回目 n = 0.5
6 回目 n = 0.6
7 回目 n = 0.7
8 回目 n = 0.7999999999999999
9 回目 n = 0.8999999999999999
10 回目 n = 0.9999999999999999
(n = 0.9999999999999999) : n = 1 ではありません
```


つぎのスク립トは、 n にとても小さな数を入れます。

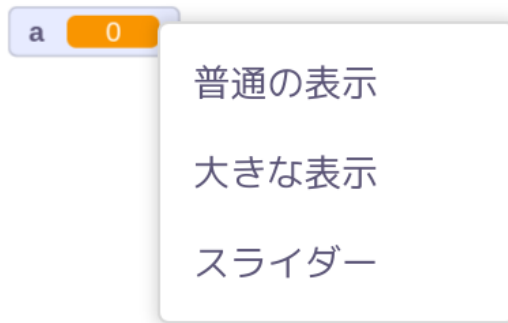
$n = 0$ かどうかのテストをすると、「 $n = 0$ ではない」と表示されますが、 n の値の表示は 0 になります。



このように数には不思議なことがあるため、「制御」のブロックに「演算」の六角形のブロックを入れてテストをする場合に、「 $n = 1$ 」とピンポイントで数値を指定してしまうとうまくいかないことがあります。「 $n = 1$ または $n > 1$ 」とか「 $n = 1$ または $n < 1$ 」としたほうが良い場合があります。

1.4 変数の表示について

変数の値が表示してあるところにマウスカーソルをおいて右クリックすると、表示の仕方を選べます。



「普通の表示」は、変数名とその値を示す表示の仕方です。



「大きな表示」は、変数の値だけを大きく表示します。



「スライダー」は、プログラムの実行中にスライダーを使って変数の値を変更することができます。



1.4.1 「大きな表示」について

「大きな表示」は、「変数」のところに「変数を表示する」「変数を隠す」を使ってメッセージを表示するのに使えます。「メッセージ」という変数を作って「大きな表示」にし、猫のところに置き、次のようなスクリプトにします。



と、変数 '秒言う' (Say for seconds) は「見た目」のところにあります。「こんにちは」ではなく、「こんにちは」とスペースを入れると「こんにちは ○○ さん」になります。



じつこう
実行すると次のようになります。





1.4.2 「スライダー」^{ひょうじ}表示について

「いろ」^{へんすう ようい}という変数を用意してください。



変数「いろ」を「スライダー」表示にしてください。そして、つぎのようなスクリプトにします。



変数「いろ」のところにマウスポインターをおいて、右クリックするとスライダーの範囲が設定できます。



色 の効果を にする は、0 ～ 200 くらいでもとの色に戻るようなので、スライダーの最小値を 0 に、最大値を 200 にします。



スクリプトを実行して、スライダーで変数「いろ」の値を変化させるとスプライトの色を変更することができます。



猫のスプライトで赤になったから、色 の効果を 180 にする を違うスプライトに入れてやっても赤になるとはかぎりません。もともとのスプライトの色によります。

【チャレンジ】 「ずっと」の中に「〇度回す」を入れます。その角度を変数にしてスライダーで操作してください。スライダーの左端の位置が -50、中央が 0、右端が 50 になるように最小値最大値を入れてやると中央の位置で回転の方向が変わります。

1.5 変数の値の保存

「スコア」という変数を作って 10 回 1 を加えるスクリプトを用意します。



「スコア」という変数を作ると自動的に 0 になって、そこから 10 回 1 を加えるので、10 になります。



このスクリプトをもう一度実行すると、スコアは 20 になります。



変数は作られた時に 0 に初期設定されますが、スクリプトの実行で変化した値はそのままなので、前に実行した後の 10 にまた 10 が加わって 20 になります。

Scratch を保存して終了した場合は、変数の値も保存されます。ですから、変数はスクリプトで初期設定してやらなければなりません。



逆に、この性質を利用してハイスコアを記録することができます。



をクリックして、何回かスクリプトを実行させると、乱数のスコアの値によってハイスコアが変わってきます。

Scratch を終了する時に保存して終了すると、ハイスコアを記録することができます。

1.6 0 + 1 = ???

変数が原因ではないのですが。次のように猫のスク립トを組んでみてください。



「回数」という変数を作成します。ボタンのスプライトを用意し、次のようなスク립トを組んでみてください。これは猫が動いてボタンに触れた回数を数えるものです。



「1層奥に下げる」は、「見た目」のところにあります。これは、猫がボタンのスプライトで隠れ

てしまわないようにするものです。「スプライト 1 に触れた」は、「調べる」のところにあります。
実行してみてください。

「もし スプライト 1 に触れた なら」をテストして期待するのは、1回という答えですが、そうはなりません。

「ずっと」の中で、「もしスプライト 1 に触れたなら」のチェックを高速でやっているの、猫がボタンのところを動いている間に何度もこのチェックをするタイミングがやってきて、こういう結果になります。

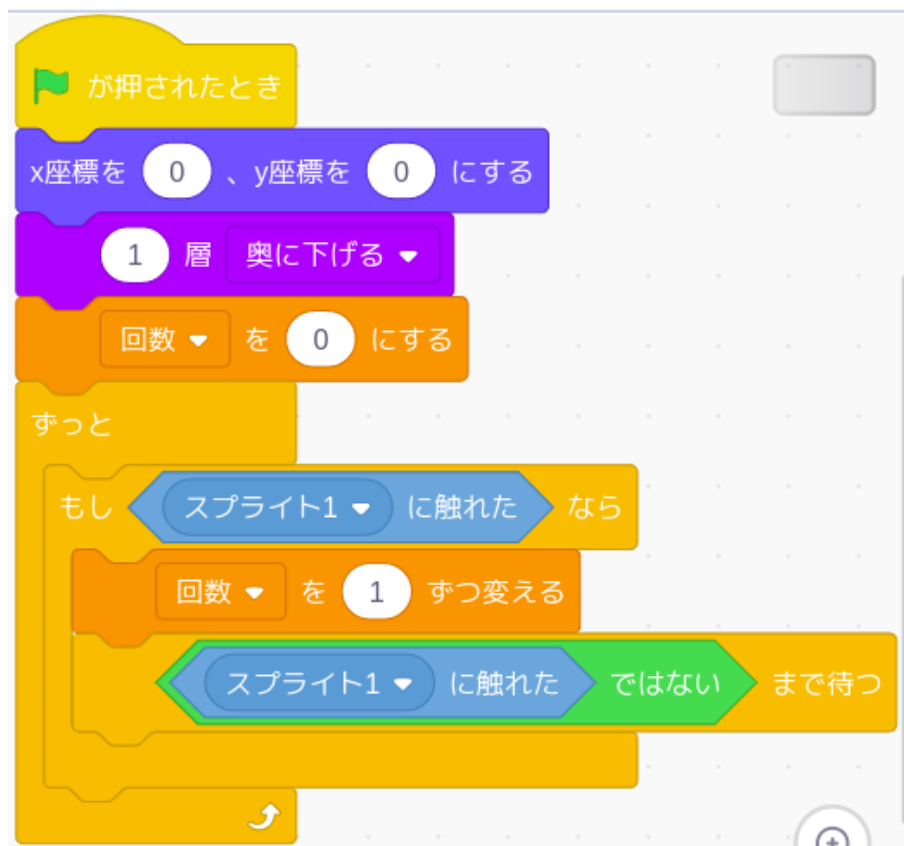
「1秒待つ」でチェックのタイミングを調整してやると回数が1回になるようです。



1秒を 0.8, 0.6, ... と変えてやってみてください。

「〇秒待つ」ではそのスプライトの大きさによって調整する必要がありますし、こういうやり方は違うかなと思います。

つぎ
次のようにすればうまくいきます。



「スプライト 1 に触れたなら」の中に、「スプライト 1 に触れた ではないまで待つ」を入れてあります。

これは、猫がボタンのところを通り過ぎるのを待つということです。

2 リストのこと

一つの変数は一つの値を記憶するということでした。リストは、変数をいくつもつなげることができる引き出しのようなものです。



2.1 リストの作成

「変数」「リストを作る」で、a という名前のリストを作ってみます。n という変数も作ってください。

新しいリスト

新しいリスト名:

☒ すべてのスプライト
ト用

☐ このスプライト
のみ

キャンセル

OK

新しい変数

新しい変数名:

☒ すべてのスプライト
ト用

☐ このスプライト
のみ

キャンセル

OK

すると、変数 n と空のリスト a が表示されます。



次のようなスクリプトを組み立てると、リストに値を入れることができます。
を入れてセットされていくようすが見られるようにしてあります。



空のリストに n を a に追加する で、 n の値がリストの 1 番目に、2 番目にと追加されていきます。リストに入っているものは、リストの $\bigcirc$ 番目と指定して、入っている値を利用したり変更したりすることができます。

このスクリプトをもう一度^{いちどじつこう}実行すると、^{ぜんかいさくせい}前回作成されたリストに^{ついか}追加されました。

The Scratch script on the left consists of the following blocks: a 'when green flag is clicked' block, an 'n set to 1' block, a 'repeat 5 times' loop containing an 'add n to a' block, an 'n increase by 1' block, and a 'wait 1 second' block. The list view on the right shows a variable 'n' with a value of 6 and a list 'a' containing the numbers 1 through 5. The list has a length of 10.

^{へんすう}変数の場合と同じで、^{おな}リストも^{しよきか}初期化をしなければなりません。

リストでは、^{つか}^{から}「a のすべてを削除する」を使ってリストを空にします。

The Scratch script on the left is modified to include a 'clear a' block at the beginning of the loop. The list view on the right shows the variable 'n' still at 6, but the list 'a' now only contains the numbers 1 through 5, with a length of 5.

つぎ
次のようにすると、逆順にすることができます。



2.2 ブレークポイント

ブレークポイントというのは、プログラムの間違いを探して修正する、デバッグということをする時などに使用します。チェックしたいところにブレークポイントを設定すると、そこでプログラムを一時停止させて、その時の動作のようすや変数などを確認することができます。

ブロック定義でそれをするブロックを作ってみます。

「ブロック」の中に



があります。これをクリックして、

ブロック定義

ブロックを作る

の「ブロックを作る」をクリックします。「ブレーク」という名前にしてください。



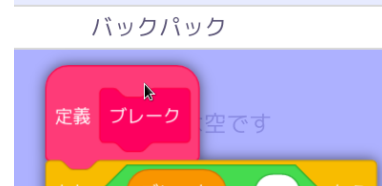
ほかのオプションは使いません。すると、 が現れるので、その下に定義のスク립トを作ります。「ブレーク」という名前の変数も作ってください。このスクリプトは、変数「ブレーク」の値で指定した秒数待つか、スペースキーが押されるまで待つものです。ブレークの時に音を出すように変更してもいいかもしれません。



次のようにプログラムを止めてチェックしたいところに入れて使います。



ブレークの定義のスクリプトを「バックパック」の中に入れておいてください。「バックパック」が使えない時は、ブレークの定義のあるスプライトのところに次のベスト5を記録するスクリプトを作ってください。



前にハイスコアを記録するスクリプトを作りました。リストを使うとベスト 5 を記録することができます。



なかなか込み入っていて分りづらいと思います。そういう時は、ところどころでプログラムを止めて変数の値をチェックしたりして考えます。

「バックパック」からブレークの定義のスクリプトを取り出してください。変数「ブレーク」も自動で取り込めるので値を指定します。変数「ブレーク」の値を指定するブロックはスクリプトの中に入れる必要はありません。

ブレーク

ブレークをスクリプトを止めてチェックしたいところに挿入します。

まあ、動作を理解するには紙に動作の流れを書いてみたりするのが一番よかったです。

「ブロック定義」を作る時に引数をつけることもできました。そうすれば「ブレーク」という

変数を使わなくてもよかったわけです。でも、あちこちにブレークを入れる場合はそれぞれに引数を設定しなければなりません。それぞれのポイントごとに設定を変えられる利点があります。

[チャレンジ] これを変更すると、10 個の乱数を作りながら、小さい → 大きい順になるようなリストを作るスクリプトにすることができます。

大ききのテストをする「>」を「<」に変えたり、あともうすこし。

2.3 一回押しただけなのに ...

次のように変数とリストとスクリプトを作成してください。



このスクリプトは「a」のキーが押されるまで待って、どこかへ移動するつものものです。

キーが押された時点の「x座標」、「y座標」の値を記録しています。「キーチェック」のリストにあるように、一回キーを押したただけなのに何回もキーを押したようになって見えます。見えませんがあちこちに移動しているようです。



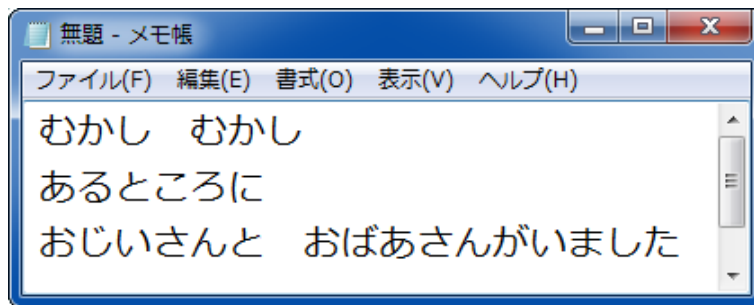
2.4 リストへの^{まちが}間違った^{いちしてい}位置指定

次のスクリプトでは、空のリストの 5 番目に 値^{ばんめ} ^{あた}い ^い を入れようとしています。ですが、実行してもなにもおこりません。^{まちが}間違ったことを^{おし}教えてくれるといいのですが。



2.5 リストの^{ないよう}内容を^よファイルから^こ読み込む

テキストファイルを読み込んで、Scratch で使ってみます。メモ帳などのエディターで次のようなファイルを作^{つく}ってください。



ファイルを保存する時に、次のようにファイルの^{しゆり}種類を「テキスト文書」、文字コードを「UTF-8」にしてください。



読み込んだ^{ぶんしやう}文章をしゃべらせるので^{かくちやうきのう}拡張機能の^{おんせいごうせい}音声合成を^{ついか}追加してください。

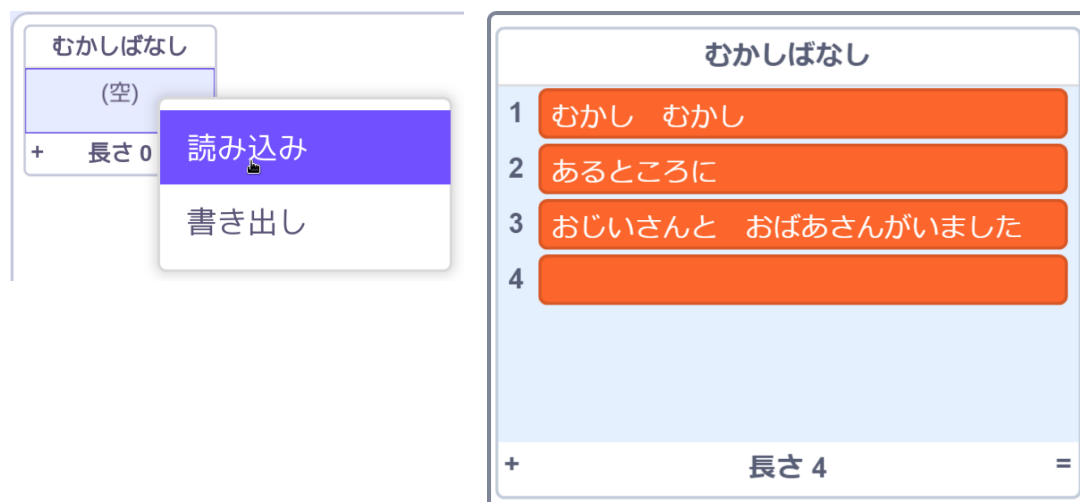


リストとして「むかしばなし」を作成し、変数は「変数」を使います。次のようにスクリプトを組んでください。



空の「むかしばなし」のリストのところでマウスを右クリックすると、「読み込み」「書き出し」が出ますから、「読み込み」をクリックしてください。

すると、ファイルを読み込むことができるので、作成しておいたテキスト文書を読み込んでください。リストにファイルの内容がセットされます。



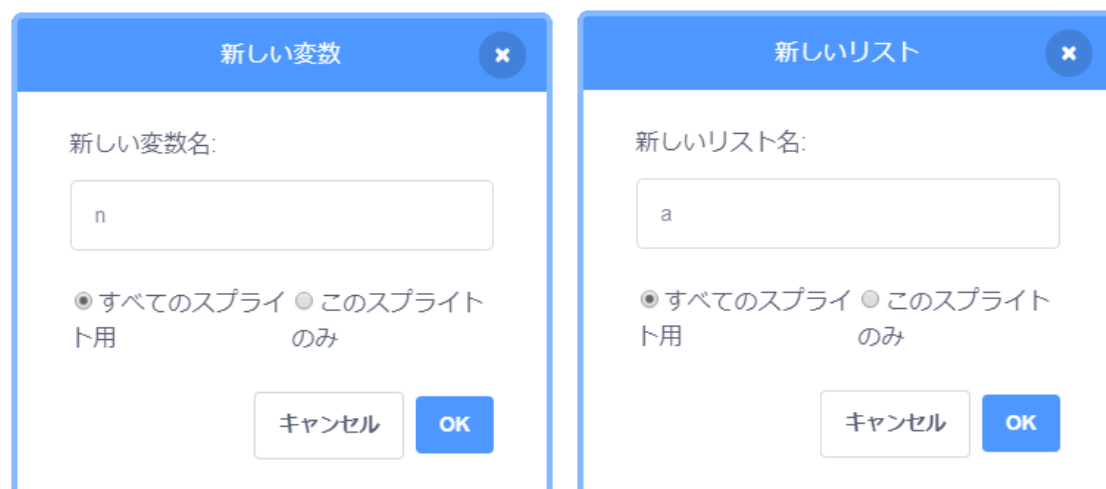
プログラムを実行してみてください。(スピーカーから音を出す設定はだいじょうぶでしょうか)
 リストの「書き出し」では、ファイル名は自動的にリスト名に「.txt」が付いたものになります。
 保存場所は、設定が変えられていなければ「ダウンロード」フォルダーです。

3 変数やリストのオプション

3.1 変数による不具合

変数やリストを作る時に、「すべてのスプライト用」「このスプライトのみ」のオプションが選べます。「すべてのスプライト用」を選んだ場合は、どのスプライトでもその変数の値を利用したり変更したりできます。しかし、このことが原因でプログラムがうまく動かないことがあります。

変数 n を「すべてのスプライト用」のオプションで作成してください。
 a というリストも作成してください。



つぎ
次のようなスクリプトを作成します。

このスクリプトのところをクリックして実行すると、リスト a に 1 ～ 10 の^{かず}数がセットされます。



The image shows a Scratch script on the left and a list named 'a' on the right. The script consists of the following blocks: an orange block 'a のすべてを削除する' (Delete everything in a), an orange block 'n を 1 にする' (Set n to 1), a yellow loop block '10 回繰り返す' (Repeat 10 times), an orange block 'n を a に追加する' (Add n to a), an orange block 'n を 1 ずつ変える' (Increase n by 1), and a yellow block '1 秒待つ' (Wait 1 second). The list 'a' on the right contains the numbers 1 through 10, and its length is shown as 10.

b というリストを^{さくせい}作成してください。



The image shows the '新しいリスト' (New List) dialog box in Scratch. It has a title bar with a close button. Inside, there is a label '新しいリスト名:' followed by a text input field containing the letter 'b'. Below the input field are two radio buttons: 'すべてのスプライトト用' (For all sprites) which is selected, and 'このスプライトのみ' (Only this sprite). At the bottom are two buttons: 'キャンセル' (Cancel) and 'OK'.

べつ
別なスプライトを用意してください。次のようなスクリプトを作成します。

このスクリプトのところをクリックして実行すると、リスト b に 10 ～ 1 の^{かず}数がセットされます。



このようにそれぞれのスクリプトを単独で実行すると、問題ない結果になります。

両方のスクリプトの最初に `が押されたとき` を入れてください。



をクリックして、これを実行すると次のようになります。一方のスクリプトでは `n` に 1 を加え、同時にもう一方のスクリプトでは `n` から 1 を引いているのでおかしいなになっています。

n 2	
a	b
1 1	1 10
2 1	2 2
3 1	3 2
4 1	4 2
5 1	5 2
6 1	6 2
7 1	7 2
8 1	8 2
9 1	9 2
10 1	10 2
+ 長さ 10 =	+ 長さ 10 =

ただし、実行するパソコンやタイミングによってセットされる値が違ふかもしれません。ひとつの変数を同時に違ふことに使ってしまったのでおかしいなとなりました。このような単純なスクリプトだと、このようなミスはすぐに気が付くと思いますが、複雑になると難しいです。

一番目のスプライトで、m という変数を「このスプライトのみ」のオプションで作成してください。スクリプトも変更してください。

新しい変数

新しい変数名:

m

☐ すべてのスプライト用 ☒ このスプライトのみ

キャンセル OK



```

when green flag clicked
  delete all of a
  set m to 1
  loop 10 times
    add m to a
    increase m by 1
    wait 1 sec
  
```


二番目のスプライトで、m という変数を「このスプライトのみ」のオプションで作成してください。スクリプトも変更してください。



実行するとうまくいきます。
同じ m という名前の変数ですが、「このスプライトのみ」のオプションで作成することで、それぞれのスプライトの中だけで有効なものになるので、影響を受けません。
繰り返しなどに使うちょっとした変数は、「このスプライトのみ」で作成したほうが間違いが減らせるかもしれません。

3.2 リストを使った音楽演奏

画面の左下隅のをクリックして音楽の機能を追加してください。



すると、音楽用のブロックが使えるようになります。

楽譜をリストにするのに、音の高さと長さのデータをならべていきます。

まずは、楽譜というリストを「すべてのスプライト用」のオプションで作成します。



しよきせつてい
初期設定です。



つづ
にしようせつぶん
続いて、二小節分のデータをつなげます。



from *Symphonie 5*
L. van Beethoven op.67

おと たか
音の高さ 0 のデータは休符を表します。「3 回繰り返す」とあります。同じデータなので紙面の
つごう
都合でこうしました。
のこ
さんしよせつぶん
残りの三小節分のデータをつなげます。



from Symphonie 5
L. van Beethoven op.67

これで楽譜のリストができました。
演奏用のスプライトを作成してください。なんでもかまいません。そのスプライトのスク립ト
の中で、変数 i を「このスプライトのみ」のオプションで作成してください。



初期設定の部分です。
演奏用の楽器を指定します。「1 秒待つ」は、楽譜のリストができるのを待つのもありますが、
演奏の始まりでテンポがおかしい時があるので入れてみました。



つづいて、演奏する部分をつなげます。



これで実行すれば、演奏できます。



演奏用のスプライトを複製して、別な楽器を指定すれば音を重ねることができます。

それぞれのスプライトの変数 i は、そのスプライト専用で作成されているので問題ありません。

3.3 クローンと変数^{へんすう}

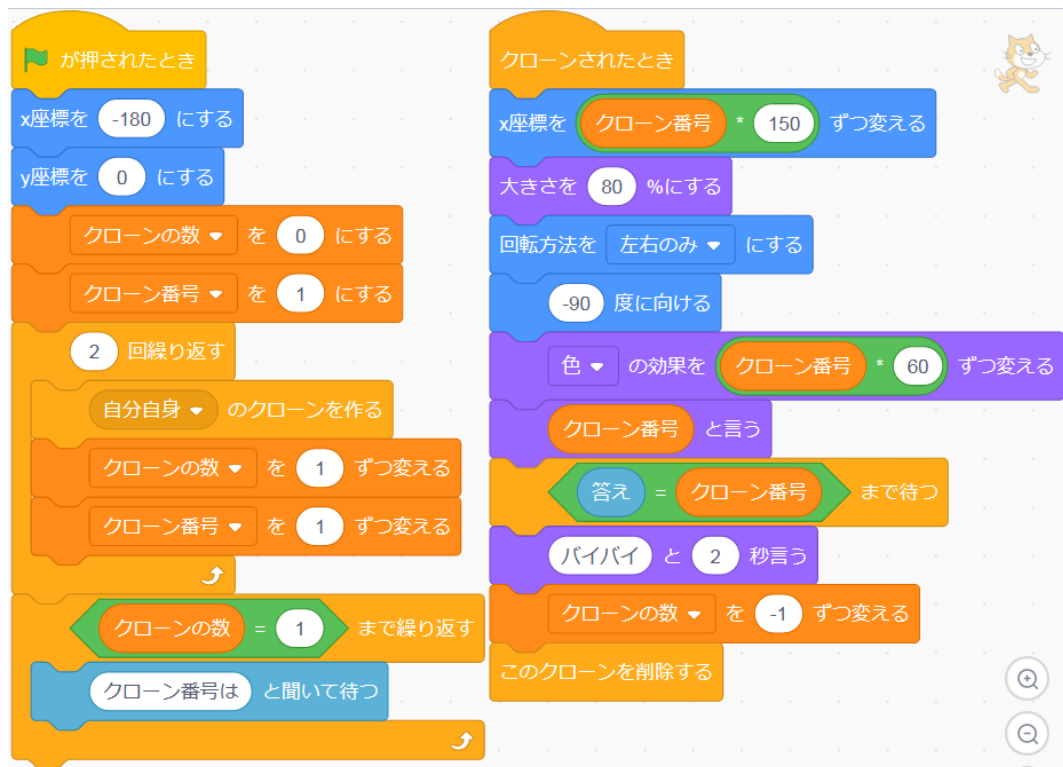
スクラッチ^{スクラッチ}ではクローン^{つか}が使えます。本体^{ほんたい}の分身^{ぶんしん}のようなものです。

「クローン番号^{ばんごう}」という変数^{へんすう}を「このスプライトのみ^{のみ}」のオプションで作成^{さくせい}してください。



「このスプライトのみ」のオプションにすることで、クローンを作成^{さくせい}するごとにそのクローン専用^{せんよう}の変数^{へんすう}ができます。

つぎ^{つぎ}次のスクリプトを作成^{さくせい}してください。



これを実行^{じっこう}すると、



にゅうりよく ばんごう さくじよ
入 力 した 番号 の クローン が 削除 されます。

がめん に「スプライト 1: クローン 番号」の 値 が 3 と 表示 されています。これは 本体 (左 端 の スプライト) の 変数「クローン 番号」で、クローン を 作成 する ごと に +1 さ せ た 結 果 だ け だ。一方、クローン を 作成 する ごと に その クローン 専 用 の 変数「クローン 番号」が 作成 されて、自 分 の 番 号 を 記 憶 して います。た と え ば、2 番 の クローンは 自 分 だ け の 変数「クローン 番号」を 持 っ て いて、2 を 記 憶 して います。この 変 数 は、2 と 番 号 が 入 力 さ れ て その クローン が 削 除 さ れ る ま で 存 在 し ま す。これは 変 数 を「この スプライト の み」の オプ シ ョ ン で 作 成 し た か ら で き る こ と で す。

4 ブロック定義の引数

5, 4, 3, 2, 1 と カウ ン ト ダウ ン する ブロック を 定 義 し て み ま す。

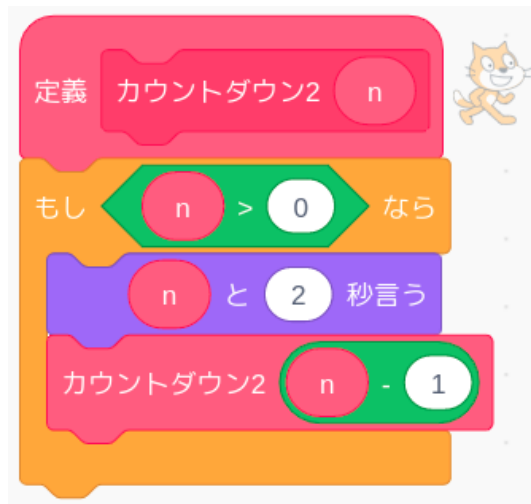


カウントダウン1 5

として、クリックしてみてください。

この定義の引数 n は、スクリプトの中では n をドラッグして変数のように使われています。しかし、これは引数の値を使用するためだけのもので、値を変更することはできません。だから他に変数を用意する必要がありました。

プログラミングには再帰呼出しという方法があります。それを使った二番目の定義です。



再帰呼出しでは必ず再帰からぬけるように作らなければなりません。この場合は n が 0 になった時です。再帰からぬけるように作らなかった場合は、ブラウザが不具合を起こしたり、電源をオフにするしかなくなるかもしれません。無限ループになっているようならば、すぐにストップボタンをクリックしてください。

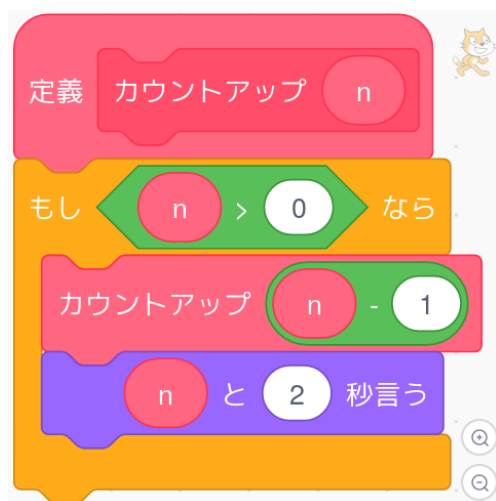
自分自身を呼出すという、不思議なものです。

みてください。どちらも見える動作自体は同じです。

こちらでは変数はありませんでした。 n は呼出されるごとに作られて、引数の値が入れます。一回目に呼出された時は n は 5 で、二回目は n は 4 になります。

カウントダウン2 5

として、クリックして



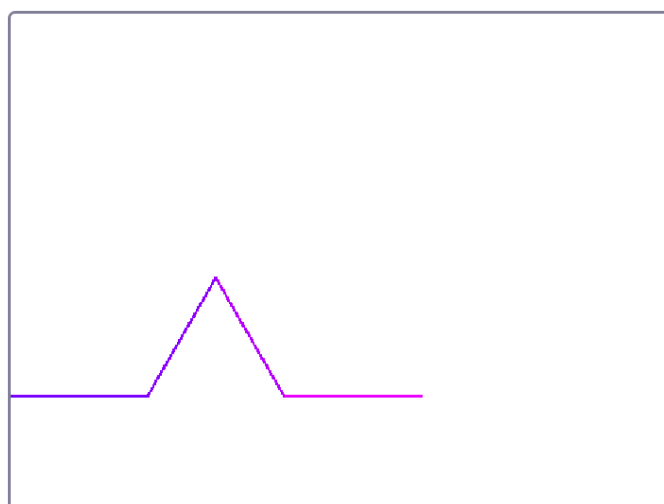
このようにするとカウントアップすることができます。

カウントアップ 5

として、クリックです。

使用するブロックはカウントダウンと同じですが、自分自身を呼出す位置が違うだけで反対の動作になります。

再帰呼び出しはフラクタル図形を描く時とか、「ハノイの塔」の問題を解く時とかなどにも使われることがあります。フラクタル図形の例です。



回数を減らしていくと、その辺に前の回数の図形が入ります。

コツホ	2	50
コツホ	3	18
コツホ	4	6
コツホ	5	2

と変えてやってみてください。

5 ^{へんすう}変数やリストのリミット

^{へんすう}変数 n とリスト a を作成して次のようなスクリプトを実行してみてください。

The image shows a Scratch script on the left and a list view on the right. The script starts with a 'when green flag clicked' event, followed by 'set n to 1', 'clear list a', and a 'repeat 310 times' loop. Inside the loop, 'n is multiplied by 10' and then 'n is added to list a'. The list view on the right shows the contents of list 'a' from index 300 to 310. Index 300 contains '1.0000000000000002e+300', and the values increase exponentially until index 309, which contains 'Infinity'. Index 310 also contains 'Infinity'. The list length is shown as 310.

Index	Value
300	1.0000000000000002e+300
301	1.0000000000000002e+301
302	1e+302
303	1e+303
304	1e+304
305	1e+305
306	9.999999999999999e+305
307	9.999999999999999e+306
308	9.999999999999998e+307
309	Infinity
310	Infinity

これは n に 1 を入れて、それをなんどもなんども 10 倍していくものです。結果をリスト a に追加していきます。

309 回目に Infinity (無限大) が入っています。無限大つまり大きすぎて扱えない数になってしまったということです。直前の数値は $9.999999999999998e+307$ です。これはおよそ 10 のうしろに 0 を 307 個付けた数ということです。

このあたりが Scratch で扱える大きな数の限界のようです。

つぎに、n を 1 から始めてどんどん大きくしながらリスト a に追加していきます。
すると、n は大きくなっていくのですが、リストが途中で追加されなくなっています。
リストのリミットは 200000 のようです。

The image shows a Scratch 3.0 script on the left and a list 'a' on the right. The script starts with a 'when green flag clicked' block, followed by 'set n to 1', 'clear list a', a 'repeat 210000' loop, and inside the loop, 'add n to list a' and 'increase n by 1'. The list 'a' on the right shows numbers from 199989 to 200000, with a total length of 200000. The variable 'n' is shown as 210001.

a	
199989	199989
199990	199990
199991	199991
199992	199992
199993	199993
199994	199994
199995	199995
199996	199996
199997	199997
199998	199998
199999	199999
200000	200000

+ 長さ 200000 =

実は、リストのリミットが 200000 ということは「scratch 3.0 リストのサイズ制限」で検索すると出てきます。それを実際にリストを使って調べてみるというのも、リストの使用例として参考になるかなと思ってのせてみました。