

スクラッチ
Scratch のこと
へんすう
変数 と リスト

斎藤文康

2023 年 5 月 26 日

目 次

1 变数のこと	4
1.1 变数の作成	7
1.2 变数の演算	9
1.3 数の不思議	15
1.4 变数の表示について	18
1.4.1 「大きな表示」について	18
1.4.2 「スライダー」表示について	20
1.5 变数の値の保存	23
1.6 $0 + 1 = ???$	25
2 リストのこと	28
2.1 リストの作成	28
2.2 ブレークポイント	31
2.3 一回押しただけなのに	34
2.4 リストへの間違った位置指定	35
2.5 リストの内容をファイルから読み込む	35
3 变数やリストのオプション	37
3.1 变数による不具合	37
3.2 リストを使った音楽演奏	42
3.3 クローンと变数	46
4 ブロック定義の引数	47
5 变数やリストのリミット	50
6 リストを使ってスクリプトの実行順序を調べる	52

参考作品 : scratch.mit.edu のサイトの検索窓から ezushi で検索してください。

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



それぞれ、そのイベントごとに指定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、

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ことが分かります。
複数ある場合は作成された順に実行されます。

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。

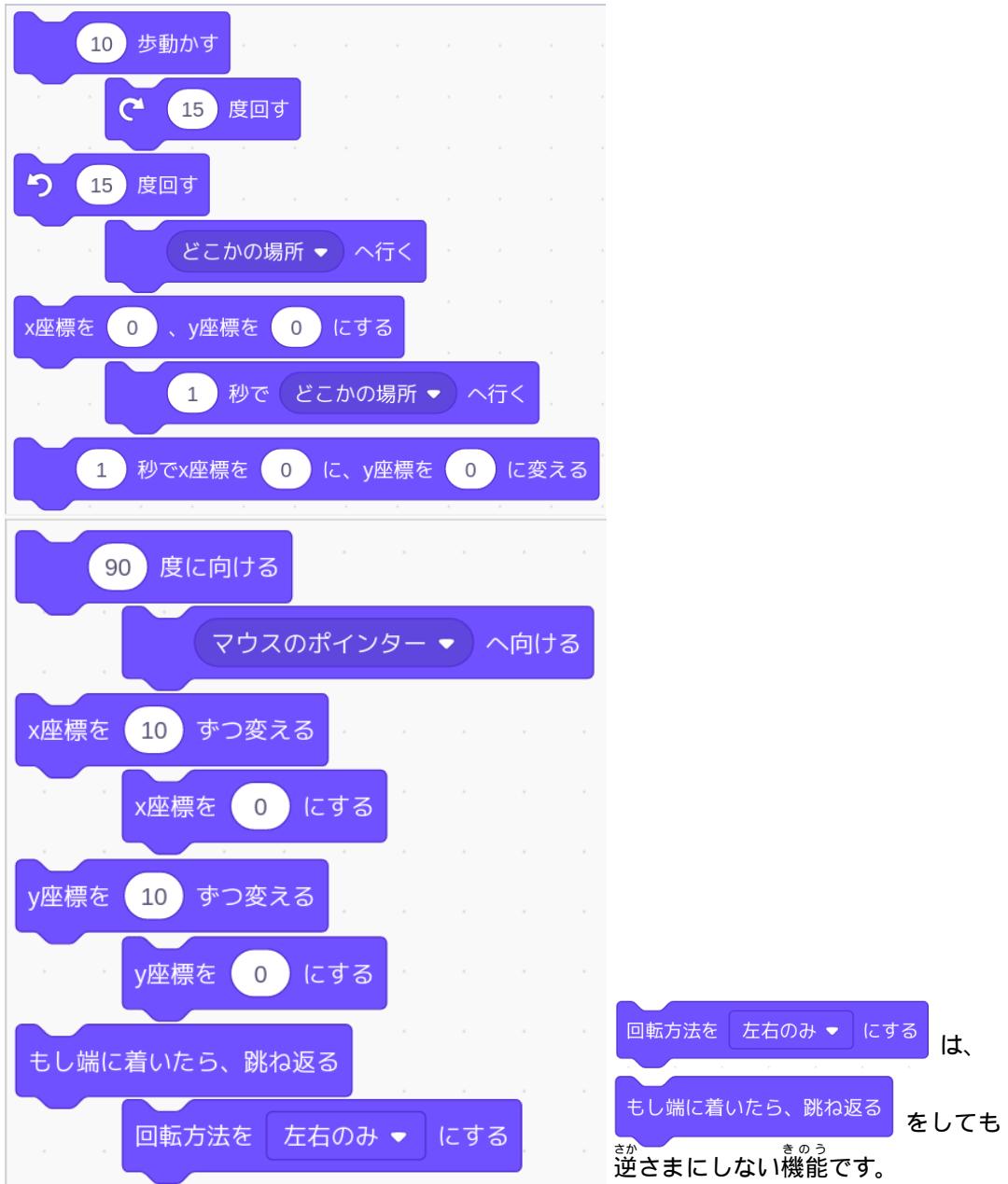


左側にチェックボックスが付いていて、変数の値を表示できるものをすべて表示してみます。



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
 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めば楽です。
 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。



「演算」の中に、□から□までの乱数 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。



 をクリックすることで、 が押されたとき 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、 どこかの場所 ▾  へ行く 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てしまって全身が表示されないことがあります。

すうち い とき にゆうりょく せんかく もじ スクラッチ なに はい
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
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
さい。上が半角モードで、下が全角モードで入れたようです。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。



度数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」^{よう}「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。



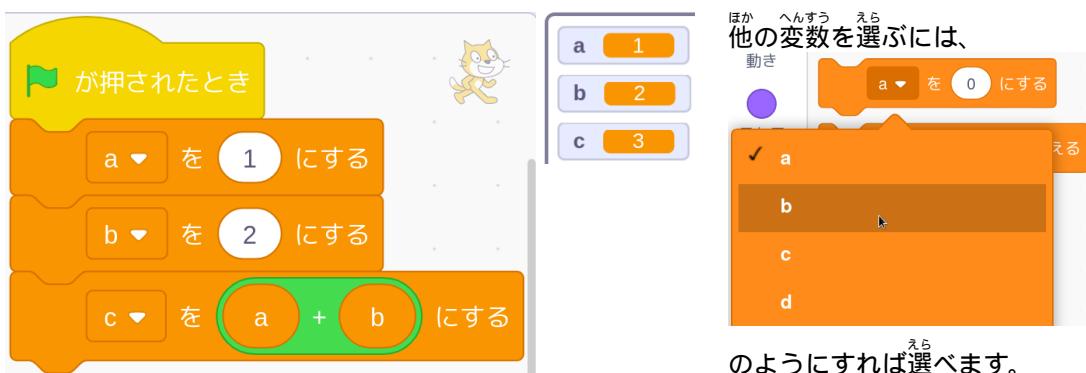
そうすると、この変数に対する操作のブロックが表示されて、値を入れたり増やしたりできるようになります。変数は、作成された時に 0 に初期設定されます。



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** をドラッグってきて、こんなふうに **a** 度回す入れてやります。

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。
かず 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、



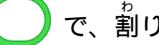
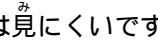
c は、12 になります。これは、a と b に入っている 1 と 2 を文字「1」「2」として扱った結果です。

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。
全角の数字はダメですが、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。



\square を \square 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 \square 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。



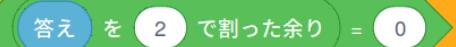
い入れると、テストの結果にしたがって別なスクリプトを実行させることができます。テストに使え
るブロックは六角形になっています。

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、

もし  なら

答え と は偶数 と言う

でなければ

答え と は奇数 と言う

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、

もし  なら

答え と は奇数 と言う

でなければ

答え と は偶数 と言う

になります。

ぜんたい
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。



つぎ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、文字を追加していくきます。  のように「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。



s スクラッチのプログラミング

s1 スクラッチ

s2 プログラミング

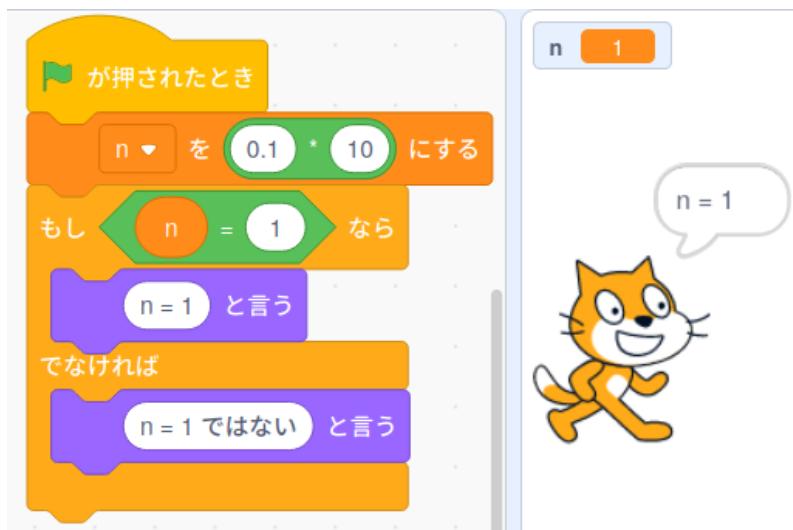
n 0

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の二進数で扱うためにおこることです。

これはこの仕組を使っている他のプログラミング言語でもおこる問題です。

```
fn main() {
    let mut n = 0.1 * 10.0;
    print!("(n = {}) : ", 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!("{} 回目 n = {}", i, n);
    }
    print!("(n = {}) : ", n);
    if n == 1.0 {
        println!("n = 1 です");
    } else {
        println!("n = 1 ではありません");
    }
}

[ 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
```

つぎのスクリプトは、 n にとても小さな数を入れます。

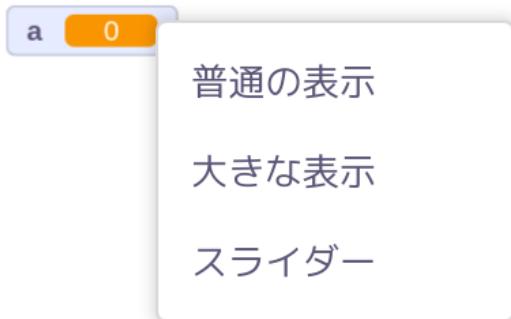
$n = 0$ かどうかのテストをすると、「 $n = 0$ ではない」と表示されますが、 n の値の表示は 0 になります。



かず
ふしき
せいけいよ
えんざん
ろつかつけい
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
い
ぱあい
すうち
してい
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
ぱあい
場合はあります。

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、表示の仕方が選べます。



「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」のところにある「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、猫のところにおきます。次のようなスクリプトにします。

と聞いて待つ と 答え は「調べる」のところに、 と 秒言う 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。



実行すると次のようになります。





1.4.2 「スライダー」表示について



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。



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。



ねこ
猫のスプライトで赤になったから、
あか
色▼ の効果を 180 にする
ちが
を違うスプライトに入れ
い
あか
で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「○ 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
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リプ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、次のようなスクリプ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。
じつこう 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
うはありません。

「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で、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。

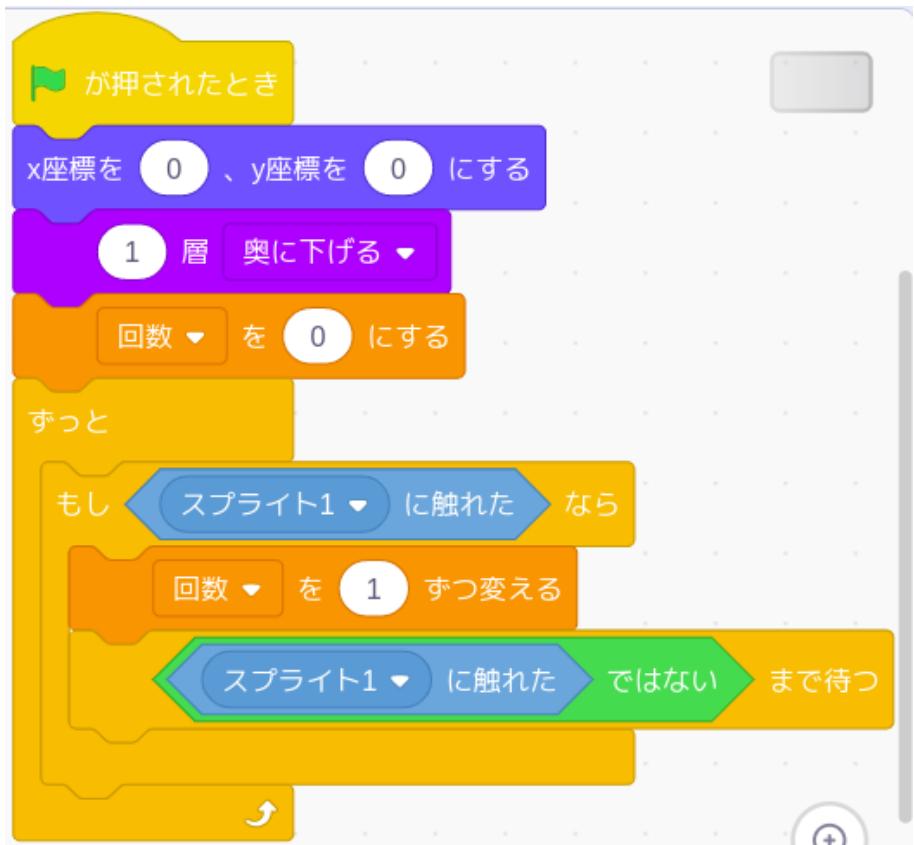
「1秒待つ」でチェックのタイミングを調整してやると回数が1回になるようです。



1秒を 0.8, 0.6, ... と変えてやってみてください。

「○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
かたちがおもい方に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。

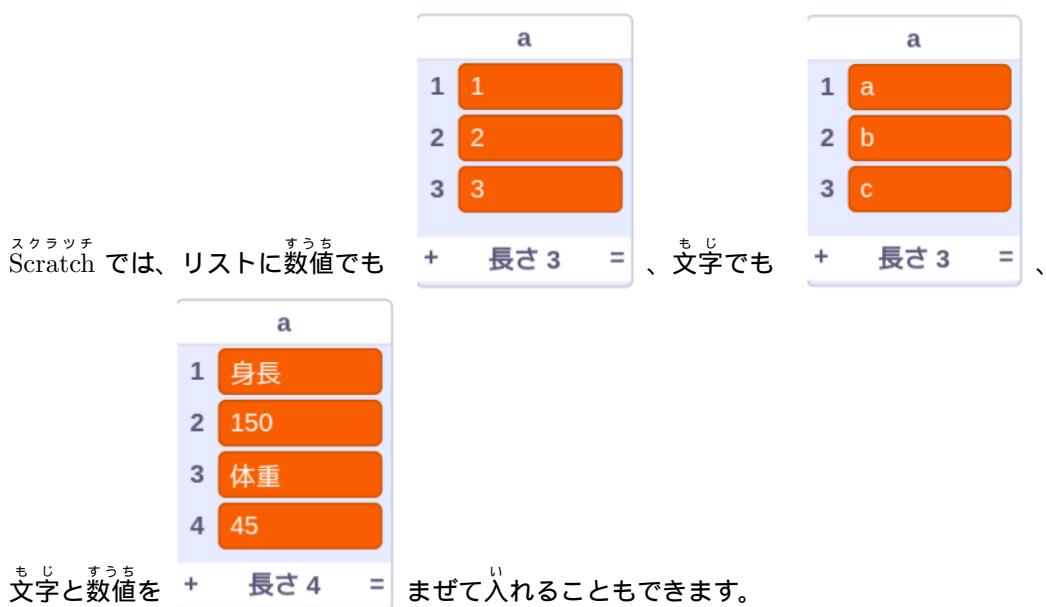


「スプライト 1 に触れたなら」の中に、「スプライト 1 に触れた ではないまで待つ」を入れてあります。

これは、ねこがボタンのところを通り過ぎるのをとおりすぎるのをまつということです。

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。



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んで実行すると、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。



から空のリストに **[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 ○ 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実行すると、前回作成されたリストに追加されました。



変数の場合と同じで、リストも初期化をしなければなりません。

リストでは、**a ▾ のすべてを削除する** を使ってリストを空にします。



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、プログラムの間違が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」を
クリックします。「ブレー
ク」という名前にしてく
ださい。



他のオプションは使いません。すると、**定義 ブレーク** が現れるので、その下に定義のスクリプトを作ります。「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リスト 5 を記録するスクリプトを作ってください。



まえ 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

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

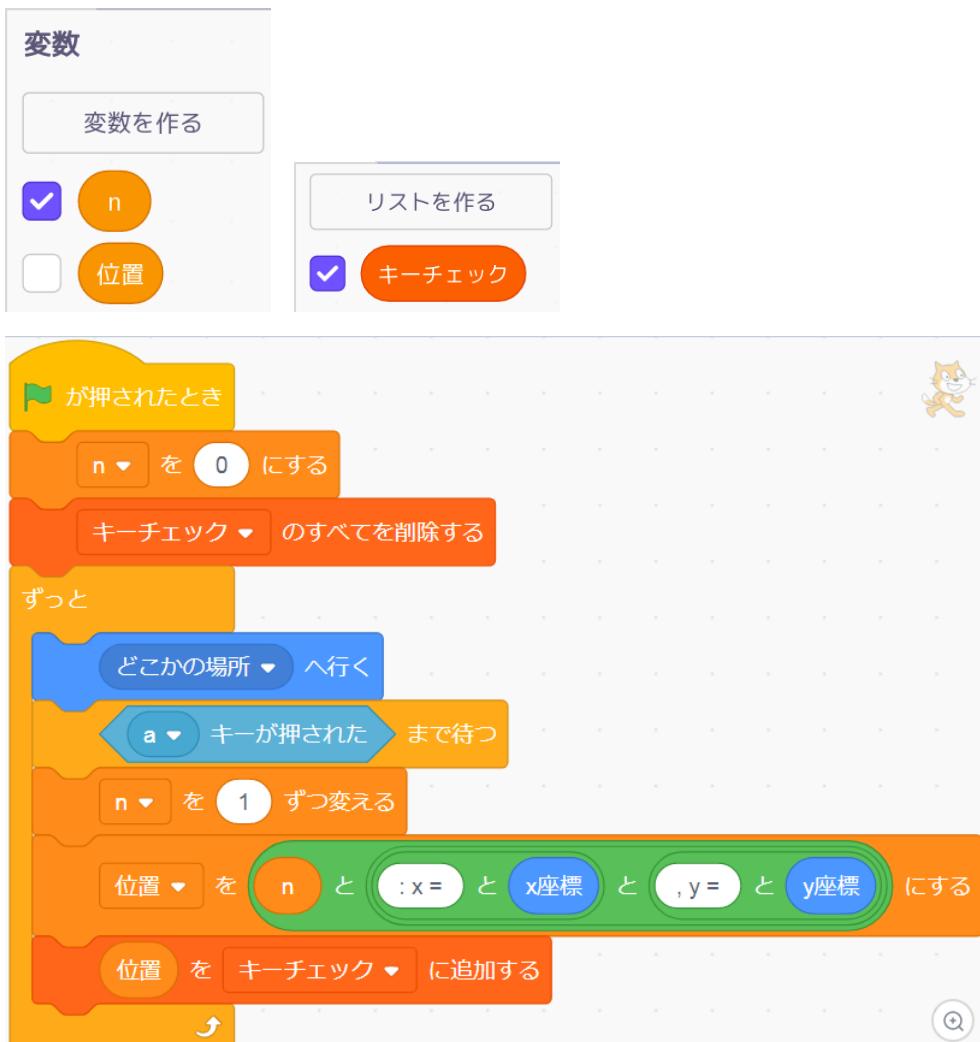
変数を使わなくてもよかったです。でも、あちこちにブレークを入れる場合はそれに引数を設定しなければなりません。それぞれのポイントごとに設定を変えられる利点はあります。

[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。

大きさのテストをする「>」を「<」に変えたり、あともうすこし。

2.3 いつかい お 一回押しただけなのに ...

つぎ へんすう さくせい
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りのものです。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」のリストにあのように、一回キーを押しただけなのに何回もキーを押したようになっています。見えませんがあちこちに移動しているようです。

位置 を キーチェック に追加する の次に
a キーが押された ではない まで待つ
を入れてやるとうまくいくようです。

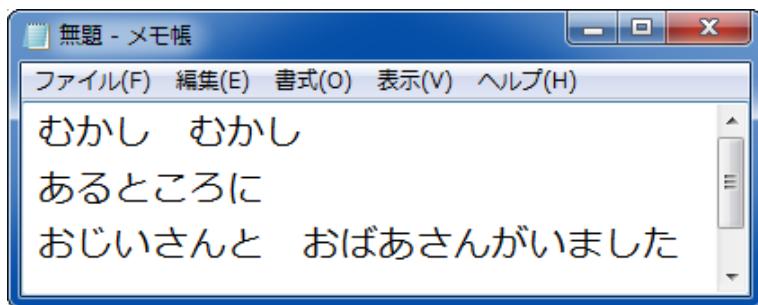
2.4 リストへの間違った位置指定

次のスクリプトでは、空のリストの 5 番目に 値 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。間違ったことを教えてくれるといいのですが。



2.5 リストの内容をファイルから読み込む

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。



読み込んだ文章をしゃべらせるので拡張機能の音声合成を追加してください。

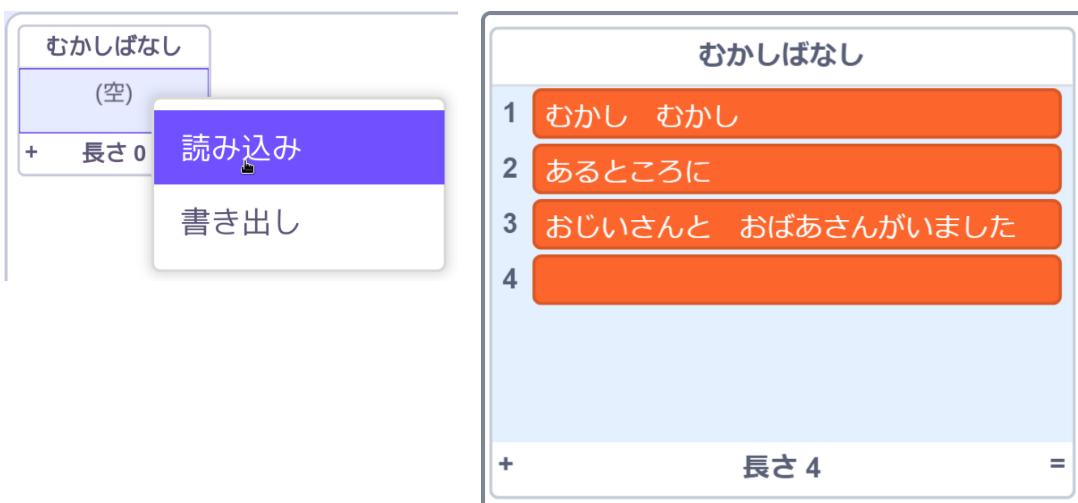
音声合成
言葉をしゃべるプロジェクトを作ろう。

必要なもの
協力
Amazon Web Services

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。



から空の「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 の数がセットされます。



b というリストを作成してください。



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 `旗が押されたとき` を入れてください。



旗をクリックして、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なことになっています。

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うかもしれません。ひとつの変数を同時に違うことに使ってしまったのでおかしなことになりました。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。

```

when green flag clicked
  [Delete all of [a v] forever
    set [m v] to [1]
    [Add [m v] to [a v] v
      change [m v] by [1
      wait [1 sec]
    ]
  ]

```

The Scratch script consists of the following blocks:

- A yellow hat block: **when green flag clicked**
- An orange control block: **[Delete all of [a v] forever]**
- An orange control block: **[set [m v] to [1]]**
- A yellow control block: **[10 (repeat)]**
- An orange control block: **[m v to a v add] (repeat)**
- An orange control block: **[change [m v] by [1]] (repeat)**
- A yellow control block: **[wait [1 sec]] (repeat)**

On the left, a new variable dialog shows:

- New variable name: m
- Scope: This sprite only (radio button selected)
- Buttons: キャンセル (Cancel) and OK

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。



初期設定です。

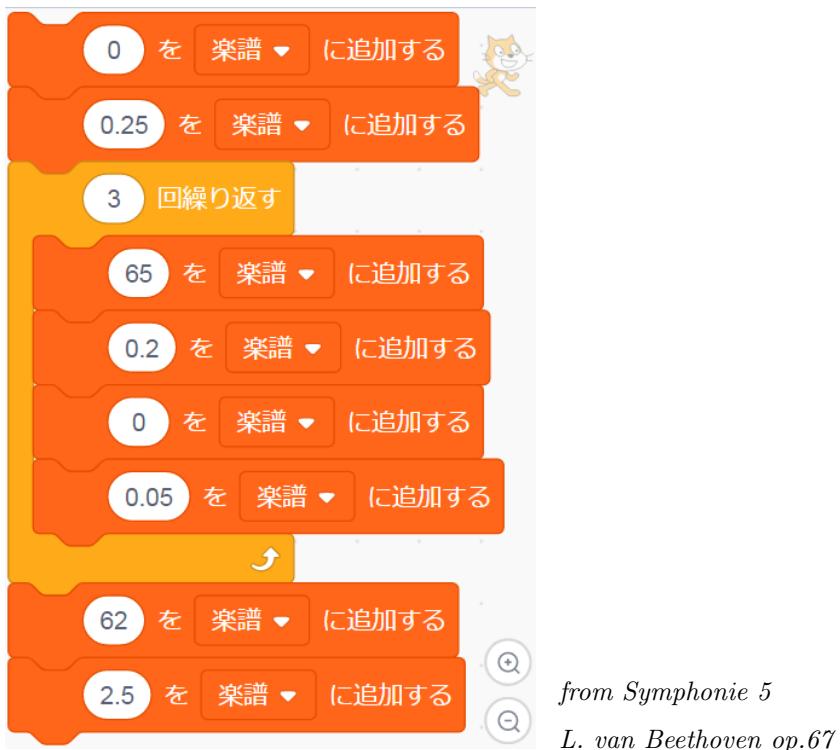


つづ
にしようせつぶん
続いて、二小節分のデータをつなげます。



from Symphonie 5
L. van Beethoven op.67

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
都合でこうしました。
残りの三小節分のデータをつなげます。



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リプトの中で、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。

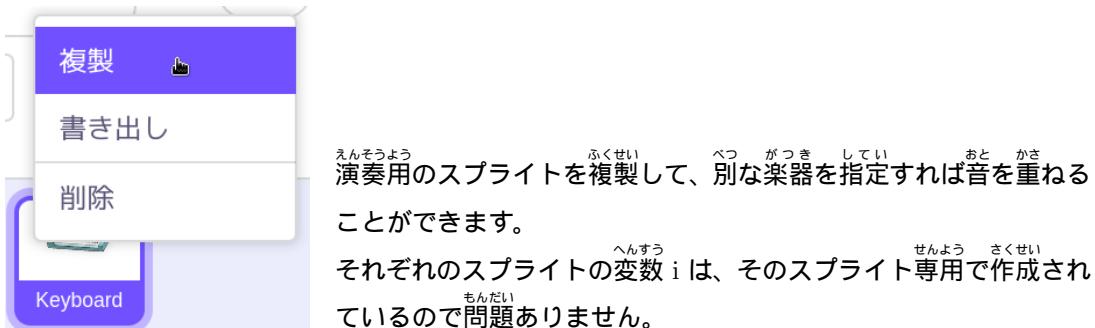
演奏用の楽器を指定します。「1秒待つ」は、楽譜のリストができるのを待つものもありますが、演奏の始まりでテンポがおかしい時があるので入れてみました。



つづ
続いて、えんそう 演奏する部分をつなげます。



じつこう これで実行すれば、えんそう 演奏できます。



3.3 クローンと変数

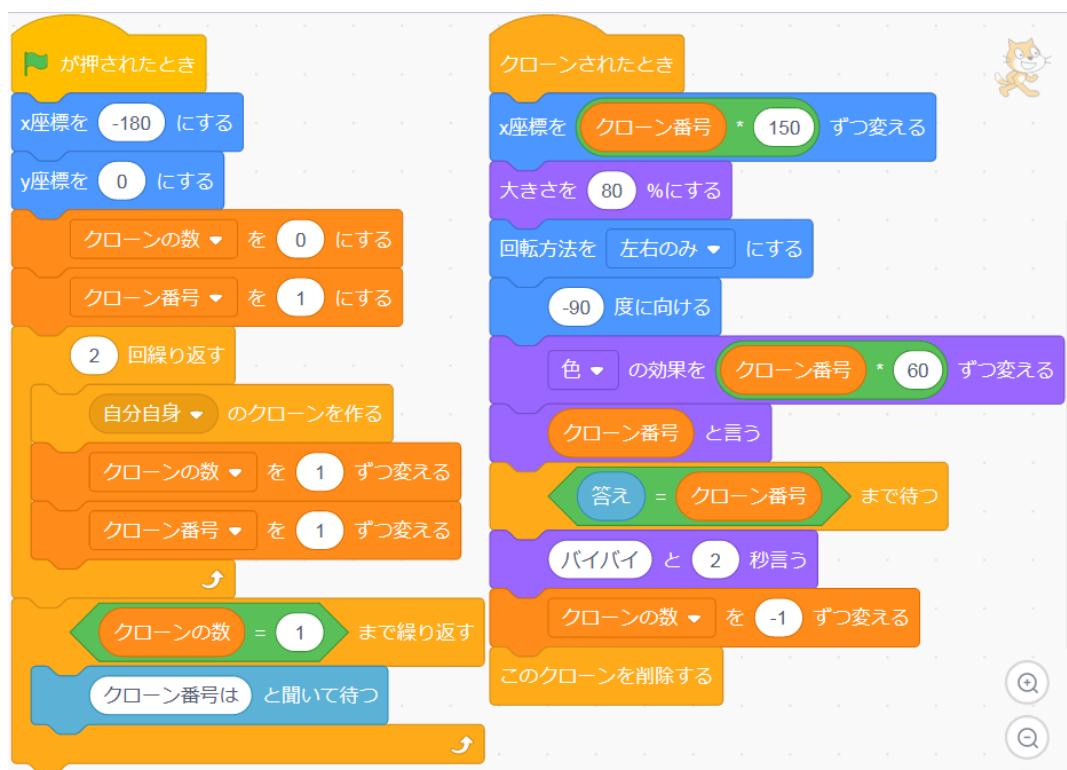
Scratch ではクローンが使えます。本体の分身のようなものです。

「クローン番号」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。



「このスプライトのみ」のオプションにすることで、クローンを作成するごとにそのクローン専用の変数ができます。

次のスクリプトを作成してください。



これを実行すると、



入力した番号のクローンが削除されます。

画面に「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させた結果です。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。

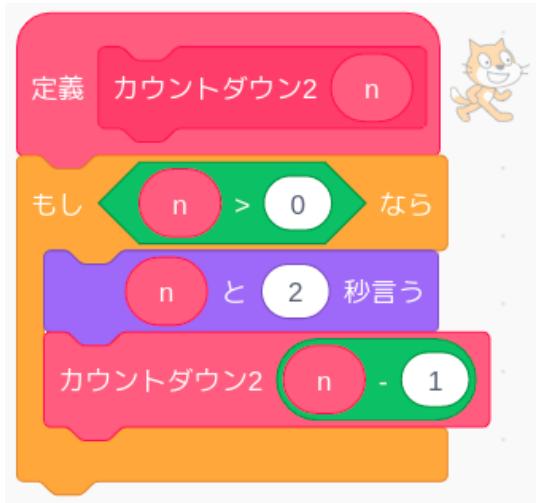


カウントダウン1 5

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。だからほか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な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ならば、すぐにストップボタンをクリックしてください。

カウントダウン2 5

として、クリックして

自分で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 자체は同じです。

こちらでは変数は必要ありませんでした。 n は呼出されるごとに作られて、引数の値が入れられます。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。



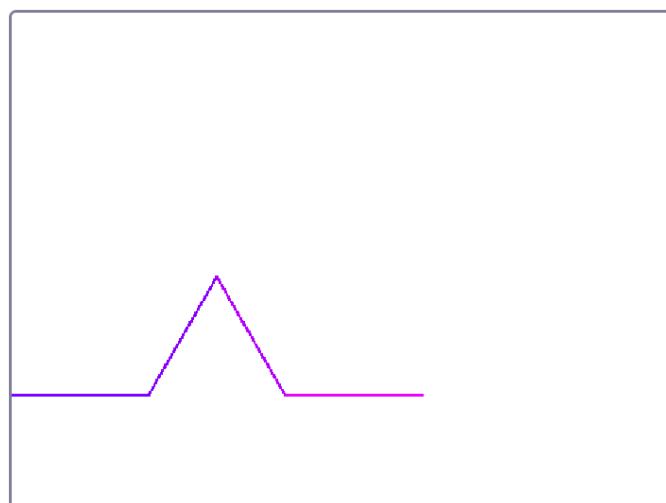
このようにするとカウントアップすることができます。

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、自分で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。



次のう か 次数を変えていくと、その辺に へん
まえ じすう すけい はい 前の次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]

か
と変えてやってみてください。

5 変数やリストのリミット

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。



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 9.99999999999998e+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 からはじ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。



じつは、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」で検索すると出できます。それを実際にリストを使って調べてみると、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。

6 リストを使ってスクリプトの実行順序を調べる

次のように二つのスプライトを用意し、「順序リスト」というリストに A?, B?, C?, D? (?) には 1 ~ 3 の数値が入ります。) が入る順番によってスクリプトの実行順序を調べます。

〔スプライト 1〕



〔スプライト 2〕



このようにした場合、作成された順に のところにあるスクリプトが実行されます。リストには [A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3] がセットされます。

[A1, B1, C1, D1, A2, B2, C2, D2, A3, B3, C3, D3] のようにセットさせるには、次のようにします。

[スプライト1]



[スプライト2]



繰り返しのためのブロックの端で実行するスクリプトが切り替わります。

ひと ひとつ じゅんばん じつこう つぎ
それぞれのスクリプトにあるブロックを一つずつ順番に実行させるには次のようにします。

[スプライト1]



[スプライト2]

