

スクラッチ
Scratch のこと
へんすう
変数 と リスト

齋藤文康

2024 年 2 月 8 日

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 ^{あたひ} 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 ^{さくせい} の作成	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 ^{まちが} への間 ^{まちが} 違った ^い 位置 ^ち 指定 ^{して}	35
2.5	リスト ^{ないよう} の内容 ^{ないよう} をファイル ^よ から読み ^よ 込む	35
3	変数 ^{へんすう} やリスト ^{へんすう} のオプション	37
3.1	変数 ^{へんすう} による ^{ふくあひ} 不 ^ふ 具 ^く 合 ^{あひ}	37
3.2	リスト ^{つか} を使 ^{つか} った ^{おんがくえんそう} 音楽 ^{おんがくえんそう} 演奏 ^{えんそう}	42
3.3	クローン ^{へんすう} と変数 ^{へんすう}	46
4	ブロック ^{ていぎ} 定義 ^{ひきすう} の引数	47
5	変数 ^{へんすう} やリスト ^{へんすう} のリミット	50
6	リスト ^{つか} を使 ^{つか} って ^{じつこうじゆんじよ} スクリプト ^{しら} の実行 ^{じつこう} 順序 ^{じゆんじよ} を調 ^{しら} べる	52

参考^{さんこう}作品^{さくひん} : scratch.mit.edu のサイト^{けんさく}の検索^{けんさく}窓^{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

変数やリストのこの前^{まえ}にちょっとイベントのお話^{はなし}です。
Scratch に何か^{なに}をさせるには、イベント^おを起こす^{おこす}、発生^{はっせい}させる必要^{ひつよう}があります。イベントは次のような時^{とき}に発生^{はっせい}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^{はいけい}や音量^{おんりょう}が変^かわる
- メッセージ^うを受け取る^と


見た目

たとえば、




をクリックすると実行^{じつこう}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、スペースキーをお押しても、 をクリックしても同じように実行^{じつこう}されます。



それぞれ、そのイベントごとに指定^{してい}されされたスクリプトが実行^{じつこう}されます。

勘違い^{かんちが}しやすいのですが、 は、プログラムを実行^{じつこう}させるボタンではなく、

 が押されたとき ^{じつこう} のところにあるスクリプトを実行^{じつこう}させるイベントを発生^{はっせい}させるものだということ

とです。 が押されたとき ^{ふくすう} が複数ある場合は作成^{ばあい}された ^{さくせい} 順^{じゆん} に実行^{じつこう}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 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。

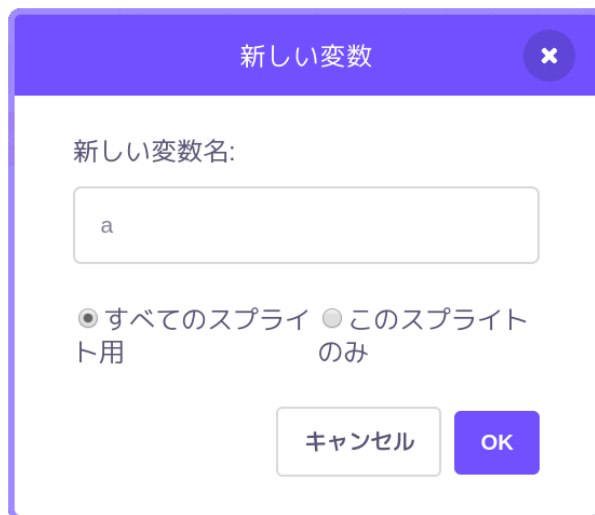
1.1 変数の作成

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。

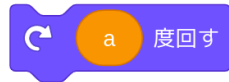
「すべてのスプライト用」「このスプライトのみ」のオプションがありますが、これについては「変数とリストのオプション」のところで説明します。



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。

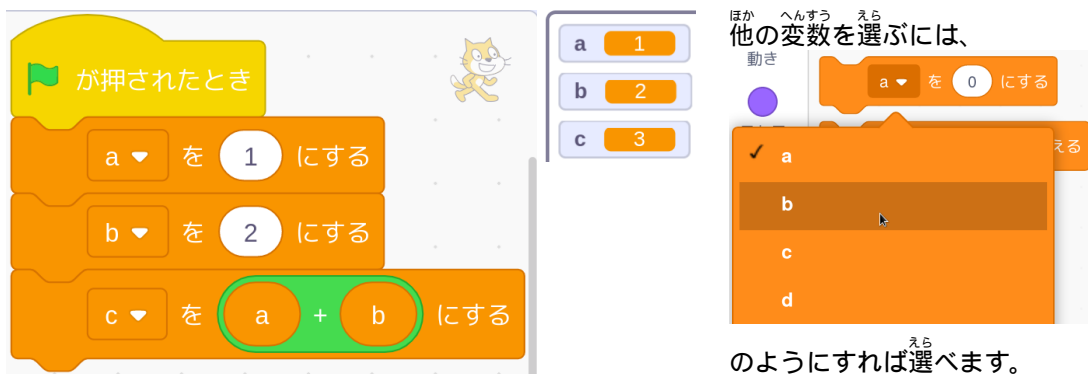


変数 a の左側にチェックが入っています。これは、この変数が画面上に表示されることを表しています。クリックしてこのチェックをはずすと、表示されなくなります。この変数を使う場合は、命令ブロックを使う時のようにこの **a** をドラッグしてきて、こんなふうに入れてやります。



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 という変数に入ります。 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」のところにある を使い、 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、



もし  なら

でなければ


になります。  という 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

でなければ


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。

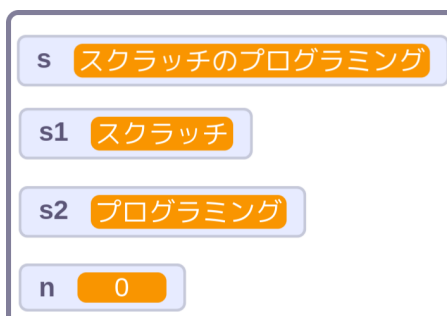
次の例は、「スクラッチのプログラミング」ということばが入っている変数 s から、「の」の左側「スクラッチ」を変数 s1 に、「の」の右側「プログラミング」を変数 s2 に入れていくものです。変数 s, s1, s2, n を作成してください。



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 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。



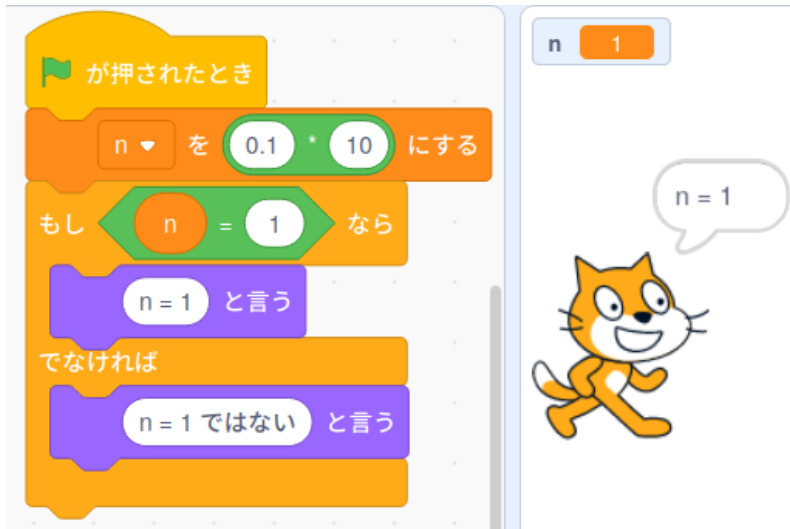
実行するとこのようになります。



[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。

これはこの仕組みしくみを使つかっている他のプログラミング言語ほかにでもおこる問題もんだいです。

```
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!("{0: >2} 回目 n = {1: <}", 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");
    }
}
```

じつこうけつが
[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 次のスクリプトは、 n にとても小さな数を入れます。

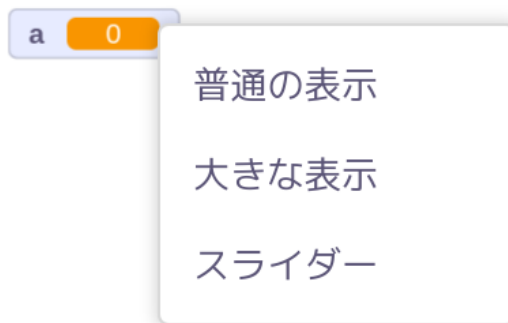
$n = 0$ かどうかのテストをすると、「 $n = 0$ ではない」と表示されますが、 n の値の表示は 0 になります。



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ス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「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、猫のところに置きます。次のようなスクリプトにします。  と  は「調べる」のところに、  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「○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。

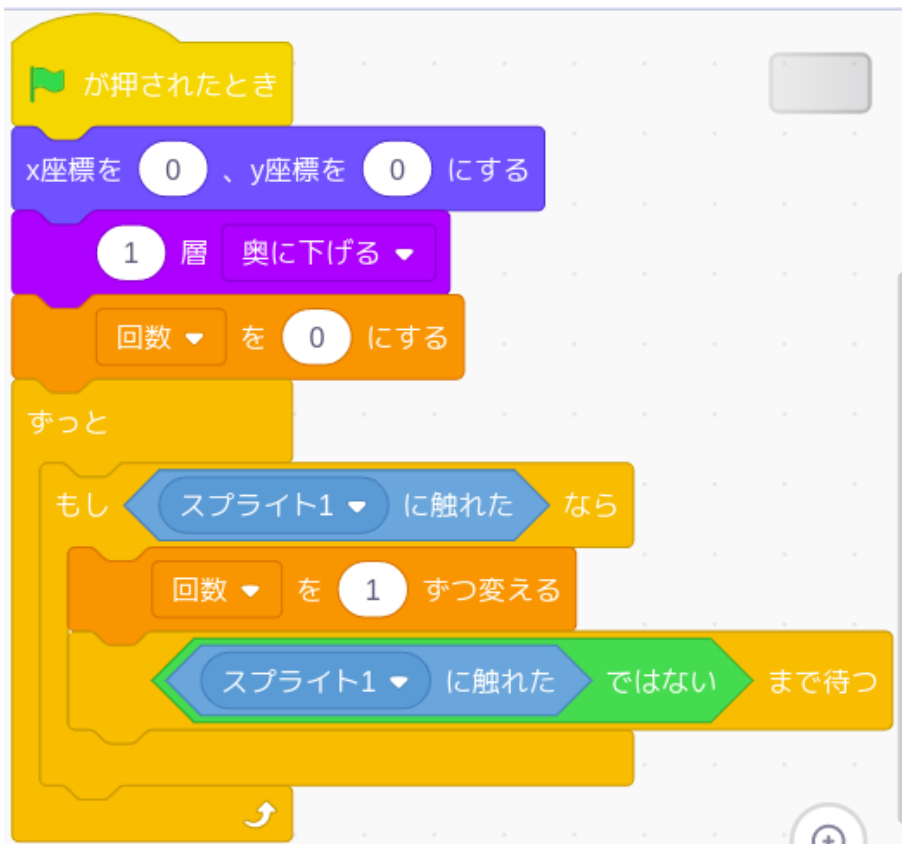
「1秒待つ」でチェックのタイミングを調整してやると回数が1回になるようです。



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。



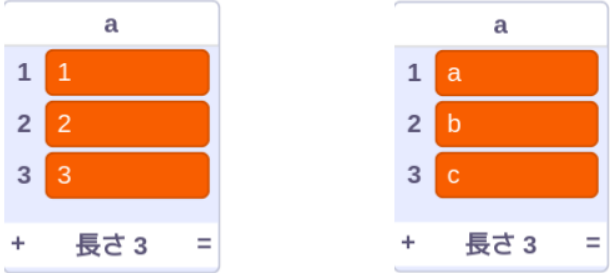
「^ふ「^{なか}スプライト 1 に触れたなら」の中に、「^ふ「^まスプライト 1 に触れた ではないまで待つ」を入れて
あります。

これは、^{ねこ}猫がボタンのところを^{とおす}通り過ぎるのを^ま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。

Scratch では、リストに数値でも + 長さ 3 = 、文字でも + 長さ 3 = 、



文字と数値を + 長さ 4 = まげて入れることもできます。



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。



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実行すると、^{いちどじつこう}前回作成されたリストに追加されました。

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' block, followed by 'set n to 1', a 'repeat 5 times' loop containing 'add n to a', 'increase n by 1', and 'wait 1 second'. The list view on the right shows a list named 'a' with 10 items, numbered 1 to 10, containing the values 1 through 5. The list length is shown as 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}「a のすべてを削除する」を使ってリストを^{から}空にします。

The image shows a modified Scratch script on the left and a list view on the right. The script now includes a 'clear a' block at the beginning of the loop. The list view on the right shows the list 'a' with 5 items, numbered 1 to 5, containing the values 1 through 5. The list length is shown as 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、

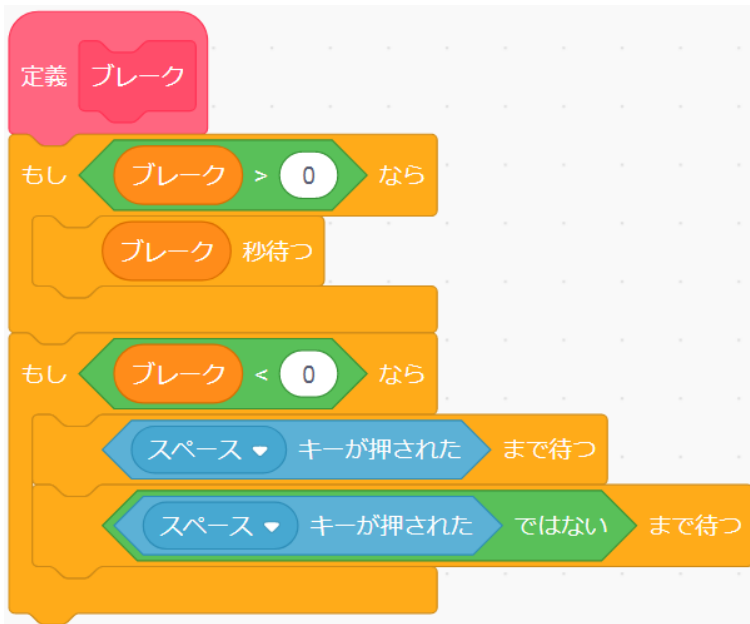
ブロック定義



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。



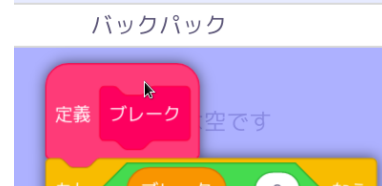
ほかのオプションは使いません。すると、**定義 ブレーク** が現れるので、その下に定義のスクリプトを作ります。「ブレーク」という名前の変数も作ってください。このスクリプトは、変数「ブレーク」の値で指定した秒数待つか、スペースキーが押されるまで待つものです。ブレークの時に音を出すように変更してもいいかもしれません。



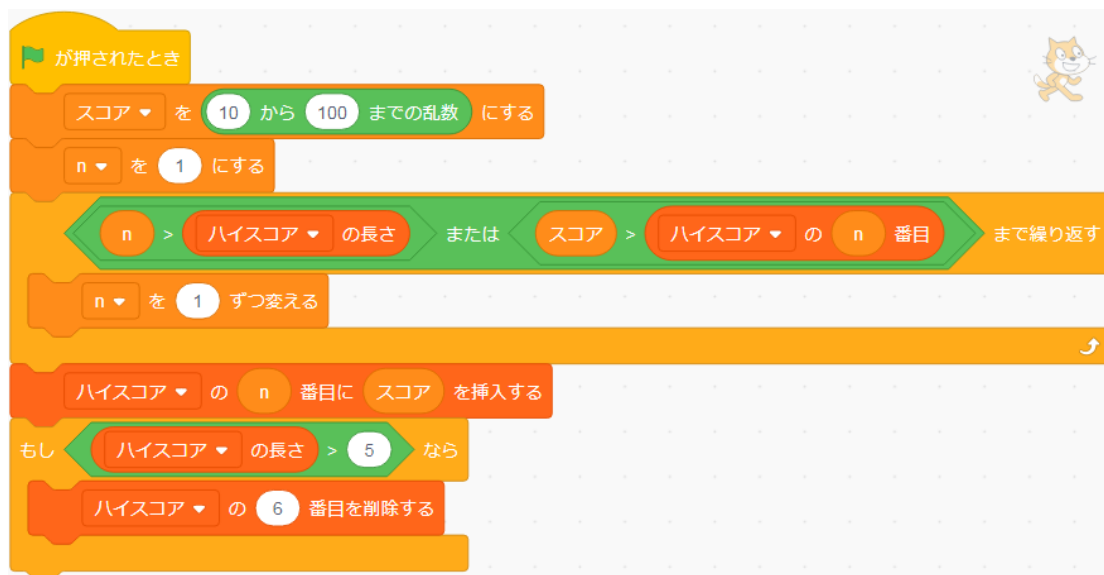
次のようにプログラムを止めてチェックしたいところに入れて使います。



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ベスト5を記録するスクリプトを作ってください。



まえ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はありますが。

[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。

The image shows the Scratch environment. On the left, the 'Variables' (変数) palette has 'Create variable' (変数を作る) and 'n' selected. The 'Lists' (リスト) palette has 'Create list' (リストを作る) and 'Key Check' (キーチェック) selected. The script area contains the following blocks:

- When green flag clicked (が押されたとき)
- Set n to 0 (n を 0 にする)
- Delete all key checks (キーチェック のすべてを削除する)
- Forever loop (ずっと):
 - Go to some location (どこかの場所 へ行く)
 - Wait for key 'a' (a キーが押された まで待つ)
 - Increase n by 1 (n を 1 ずつ変える)
 - Set position to n, x coordinate, y coordinate (位置 を n と :x= と x座標 と ,y= と y座標 にする)
 - Add key check to list (位置 を キーチェック に追加する)

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。

The image shows a close-up of the 'Add key check' block: '位置 を キーチェック に追加する'. Below it, the 'Wait for key' block is shown with 'a' selected and 'ではない' (not) checked: 'a キーが押された ではない まで待つ'. The text 'つぎの次に' (next time) is written above the block, and 'を入れてやるとうまくいくようです。' (if you put it in, it seems to work well) is written below it.

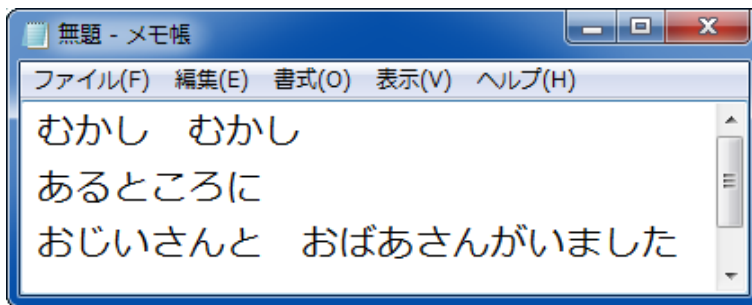
2.4 リストへの間違った位置指定

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。間違っただけを教えるといいたいのですが。

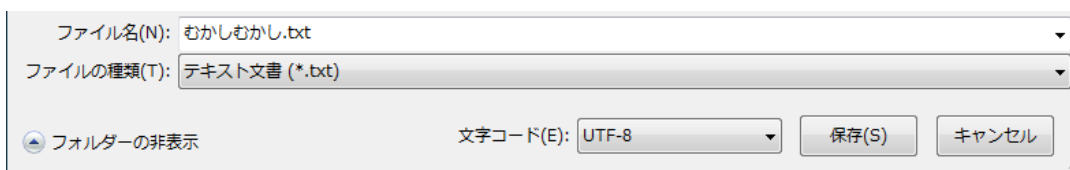


2.5 リストの内容をファイルから読み込む

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。

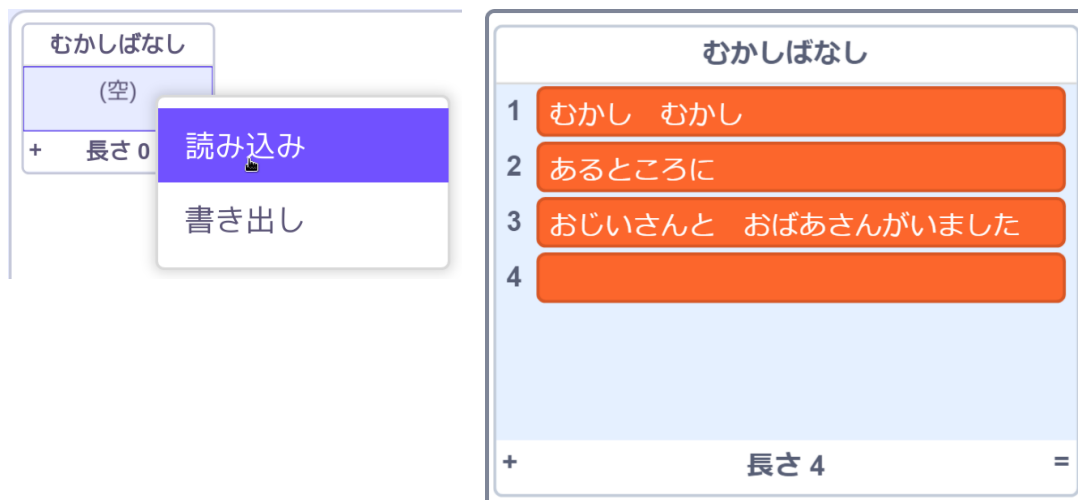


リストとして「むかしばなし」を作成し、変数は「変数」を使います。次のようにスクリプトを組んでください。



空の「むかしばなし」のリストのところでもうす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、設定が変更されていない場合は「ダウンロード」フォルダーです。

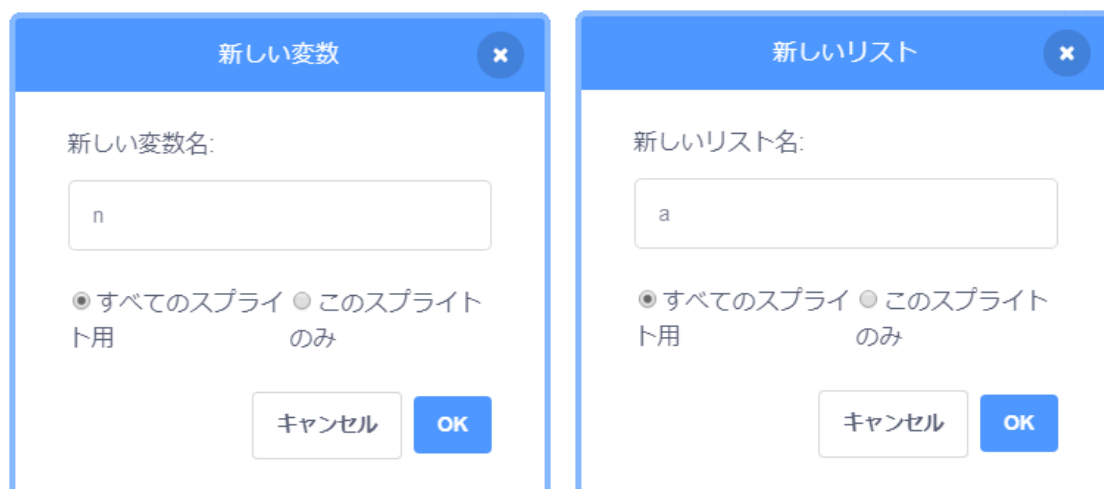
3 変数やリストのオプション

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^{じつこう}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: 'a のすべてを削除する' (Delete all of a), 'n を 1 にする' (Set n to 1), a loop '10 回繰り返す' (Repeat 10 times) containing 'n を a に追加する' (Add n to a), 'n を 1 ずつ変える' (Increase n by 1), and '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 a dialog box titled '新しいリスト' (New List). It has a text input field containing 'b'. Below the input field, there are two radio buttons: 'すべてのスプライト用' (For all sprites) which is selected, and 'このスプライトのみ' (Only for this sprite). At the bottom, there are 'キャンセル' (Cancel) and 'OK' buttons.

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^{じつこう}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なことになります。

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ってしまうかもしれません。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。

新しい変数 ✕

新しい変数名:

すべてのスプライト用
 このスプライトのみ

```

当緑旗がクリックされたとき
  a のすべてを削除する
  m を 1 にする
  10 回繰り返す
    m を a に追加する
    m を 1 ずつ変える
  1 秒待つ
          
```


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、^{えんそう}演奏する部分をつなげます。



これで実行すれば、^{えんそう}演奏できます。



^{えんそうよう}演奏用のスプライトを複製して、^{べつ がつき してい}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^{もんだい}問題ありません。

3.3 クローンと変数^{へんすう}

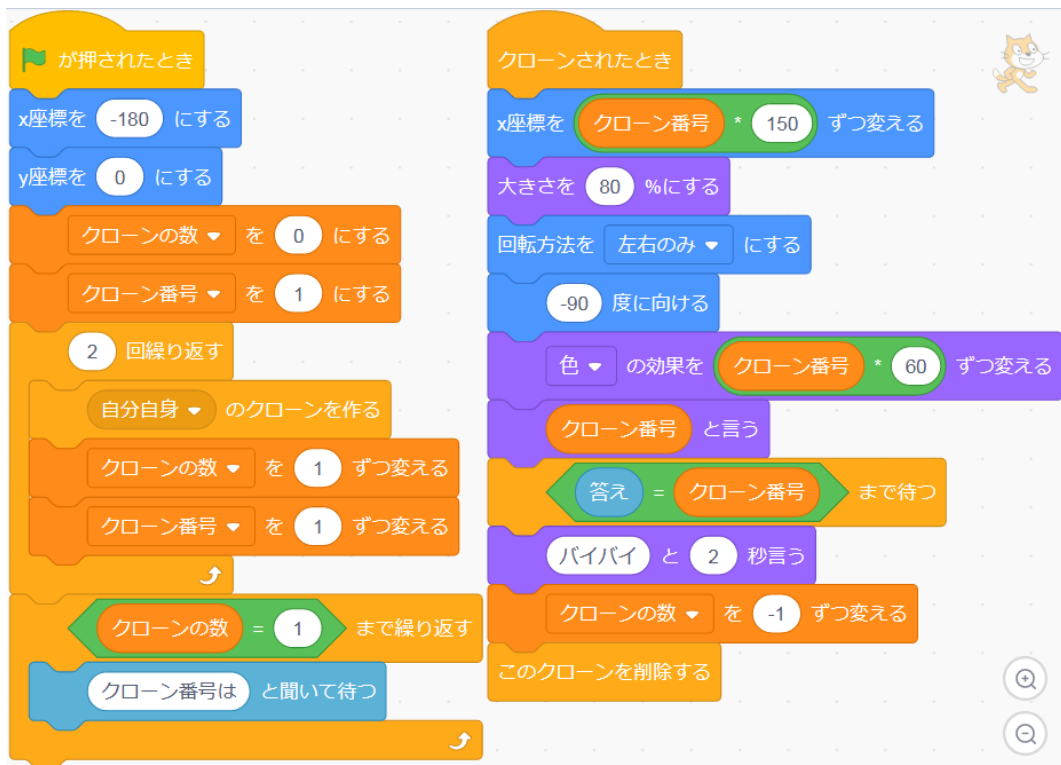
Scratch ではクローン^{つか}が使えます。本体の分身^{ほんたい ぶんしん}のようなものです。

「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」のオプション^{さくせい}で作成してください。



「このスプライトのみ」のオプション^{さくせい}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の変数^{せんよう へんすう}ができます。

次のスクリプト^{つぎ}を作成^{さくせい}してください。



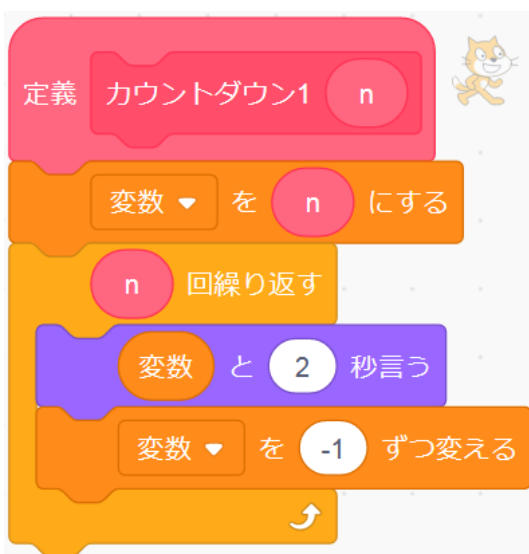
これを実行^{じっこう}すると、



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。

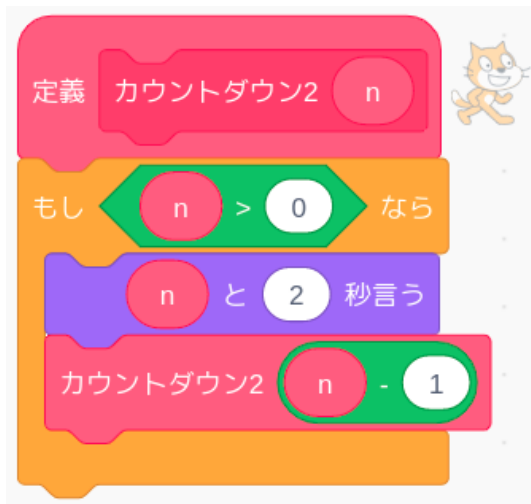


カウントダウン1 5

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のだけのもので、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。

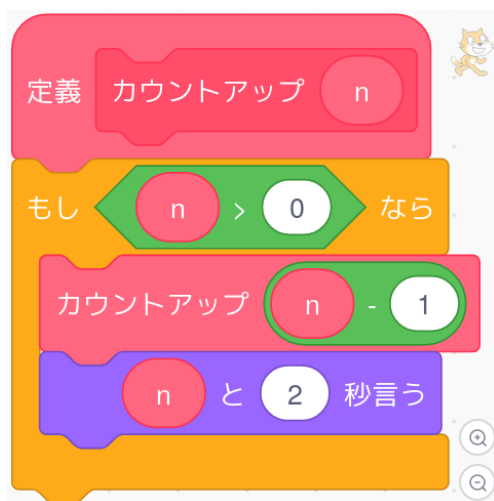
自分自身を呼出すという、不思議なものです。

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。

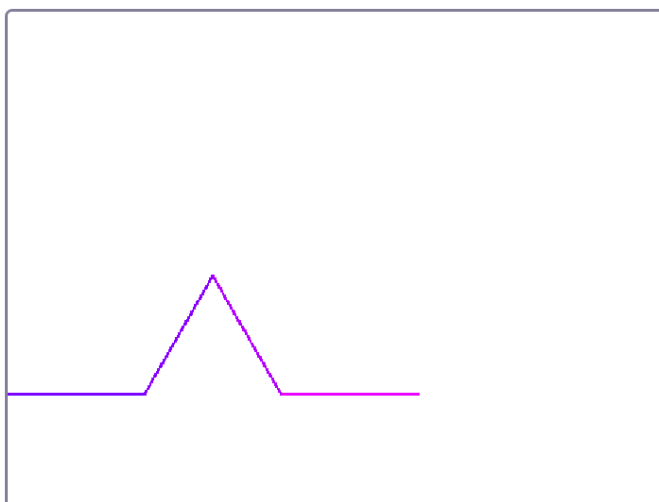
使用するブロックはカウントダウンと同じですが、自分自身を呼出す位置が違うだけで反対の動作になります。

さいきよびだ
再帰呼出しはフラクタル図形を描く時とか、^{すけい か と}「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^{すけい れい}の例です。

```

定義 コツホ 回数 長さ
もし 回数 = 0 なら
  長さ 歩動かす
  ペンの色を 10 ずつ変える
でなければ
  コツホ 回数 - 1 長さ
  60 度回す
  コツホ 回数 - 1 長さ
  120 度回す
  コツホ 回数 - 1 長さ
  60 度回す
  コツホ 回数 - 1 長さ
  コツホ 回数 - 1 長さ

が押されたとき
隠す
x座標を -240 、y座標を -100 にする
90 度に向ける
ペンを上げる
全部消す
ペンの色を 紫 にする
ペンを下ろす
コツホ 1 100
  
```



じすう か
回数を^か変えていくと、その^{へん}辺に
まえ じすう すけい はい
前の^{まへ}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

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。

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' block, followed by 'set n to 1', 'clear a', and a 'repeat 310 times' loop. Inside the loop, 'n is multiplied by 10' and 'n is added to a'. The list view shows the values of 'a' from index 300 to 310. Index 300 has the value '1.0000000000000002e+300'. The values increase exponentially, with index 307 being '9.999999999999998e+307'. From index 309 onwards, the value is '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 script on the left and a list view on the right. The script starts with a 'when green flag clicked' block, followed by 'set n to 1', 'clear a', a 'repeat 210,000' loop containing 'add n to a' and 'increase n by 1'. The list view on the right shows a list 'a' with items from 199989 to 200000. The list length is shown as 200,000.

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。

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  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。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。

ステージに同じようにスクリプトを作成した場合、ステージのスクリプトが最後に実行されます。したがって、ステージで変数などの初期化処理を行うとうまくいかないかもしれません。

このようにそれぞれのスクリプトごとに順番に実行すると、いくつかのスプライトが同時に動くようなゲームは作れません。Scratch では、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るようになっています。次のようにすると、それぞれのスクリプトの「リストに追加する」ブロックを順番に実行します。

〔スプライト1〕



〔スプライト2〕



〔 A1, B1, C1, D1, A2, B2, C2, D2, A3, B3, C3, D3 〕のようにセットされます。

つぎのように時間待ちブロックでも、切り替えることができます。

〔スプライト1〕



〔スプライト2〕



実際のところ、繰り返し処理はよく使われますし、イベントブロックの使用でも切り替わるので、あまり神経質になる必要はないでしょう。