Scratch のこと
変数 と リスト

CoderDojo Aizu

齋藤文康

2021 年 6 月 13 日

概 要

Scratch の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

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

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、

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です。

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。

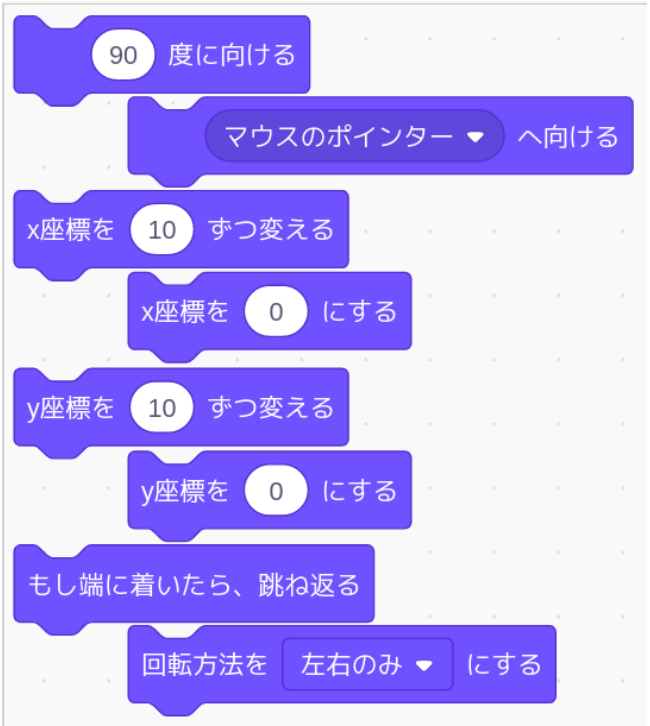
変数名に「スプライト 1:」のようにスプライト名が付いているものは、そのスプライト専用のものです。たとえば、「スプライト 1:音量」を 50 にしてもこれはこのスプライトだけのもので、他のスプライトの音を調整するにはそのスプライトのスクリプトで音量設定をしなければなりません。

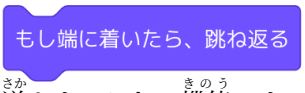
「x座標」「y座標」に画面上の座標の値を入れると、たとえば 

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 

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。



は、


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てしまって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。

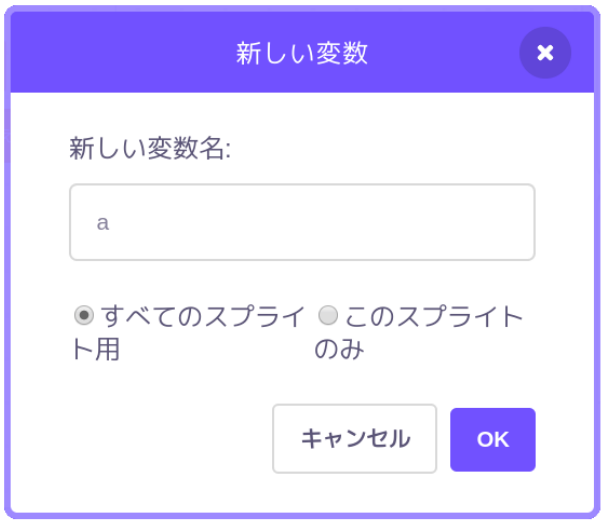
画面をきれいにするには、「ペン」のところにある「全部消す」をクリックしてください。「90度に向ける」「大きさを 100% にする」「色の効果を 0 にする」でもとに戻せます。

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。



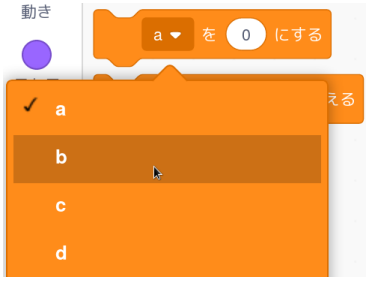
そうすると、この変数に対する操作のブロックが表示されて、値を入れたり増やしたりできるようになります。変数は、作成された時に 0 に初期設定されます。

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 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。

他の変数を選ぶには、



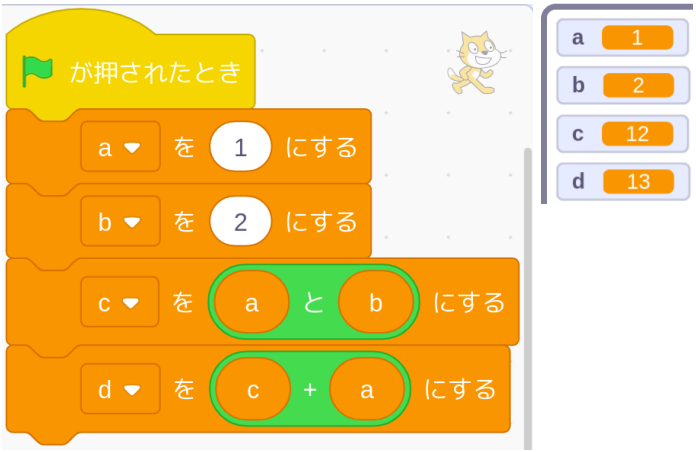
のようにすれば選べます。

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」のところにある もし なら でなければ 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。

が 0、つまり 0 と等しいならばのチェックは、


です。

が 0 よりも大きいならばのチェックは、


です。

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。

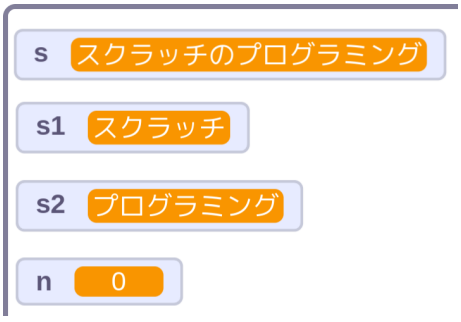
全体のプログラムです。

［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。  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。

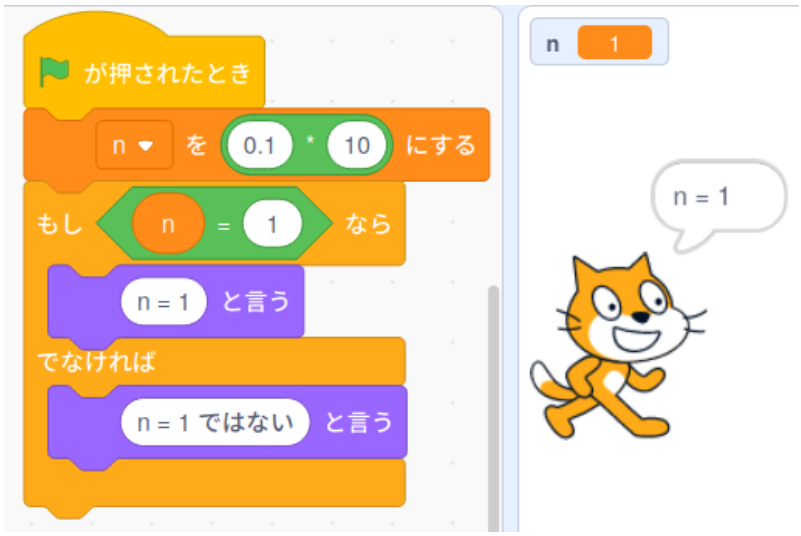
実行(じっこう)するとこのようになります。



［チャレンジ］ 上(うえ)の s に「123+456」を入(い)れて、「+」の左側(ひだりがわ)の数(かず)を s1 に、右側(みぎがわ)の数(かず)を s2 に取(と)り出(だ)して、足(た)した数(かず)を s3 という変数(へんすう)に入(い)れるように変更(へんこう)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。

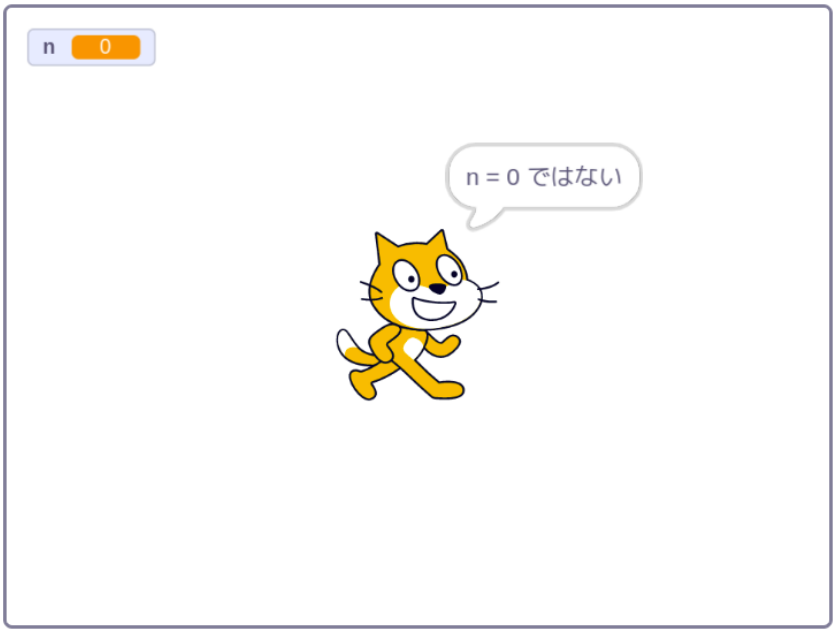
```
fn main() {
    let mut n = 0.1 * 10.0;
    print!("(n = {}) : ", n);
    if n == 1.0 {
        println!("n = 1 です");
    } else {
        println!("n = 1 ではありません");
    }
    n = 0.0;
    for i in 1 .. 11 {
        n += 0.1;
        println!("{0: >2} 回目 n = {1: <}", i, n);
    }
    print!("(n =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

[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
```

次のスクリプトは、 n にとても小さな数を入れます。

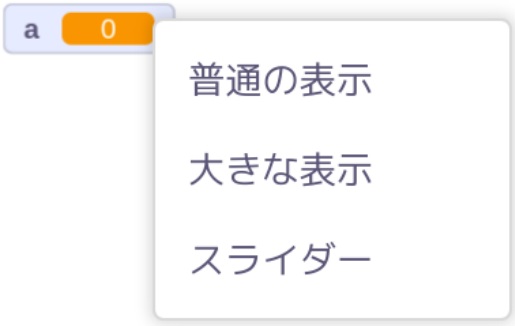
n = 0 かどうかのテストをすると、「n = 0 ではない」と表示されますが、 n の値 の表示は 0 になります。



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

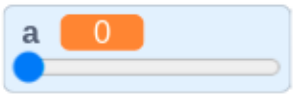
変数の値が表示してあるところにマウスカーソルをおいて右クリックすると、表示の仕方が選べます。



「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 

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。

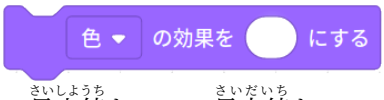
実行(じっこう)すると次(つぎ)のようになります。

1.4.2 「スライダー」表示(ひょうじ)について

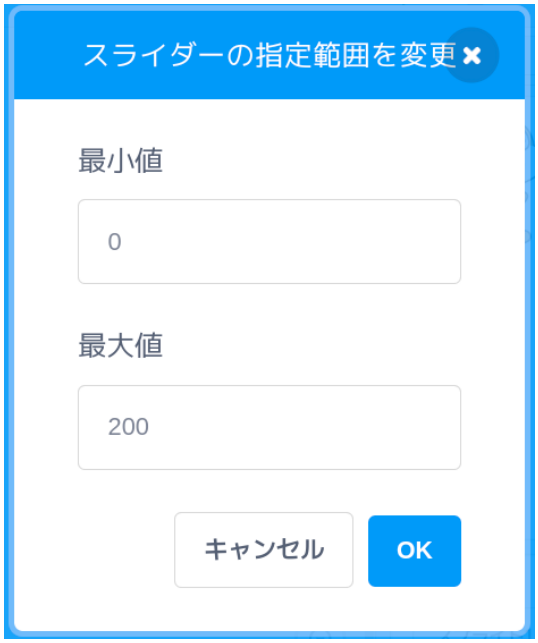
「いろ」という変数(へんすう)を用意(よう い)してください。

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。

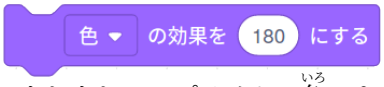
変数「いろ」のところにマウスポインターをおいて、右クリックするとスライダーの範囲が設定できます。



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、 

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。

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。

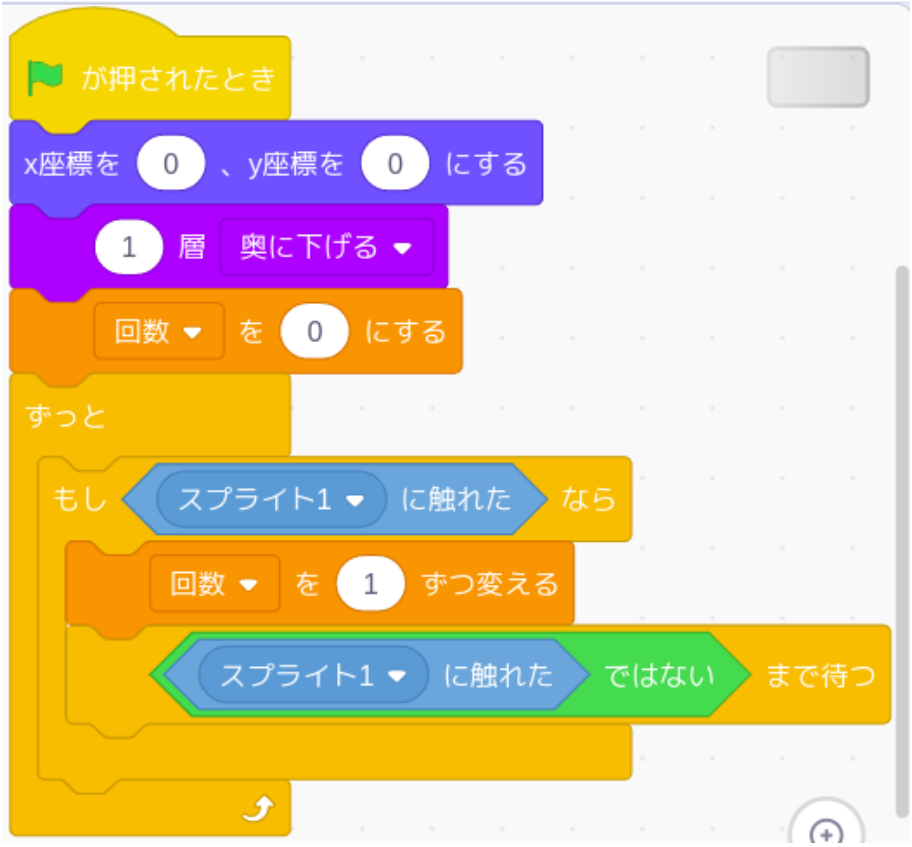
「ずっと」の中で、「もしスプライト 1 に触れたなら」のチェックを高速でやっているので、猫がボタンのところを動いている間に何度もこのチェックをするタイミングがやってきて、こういう結果になります。

「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。

「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。

次のようにすればうまくいきます。



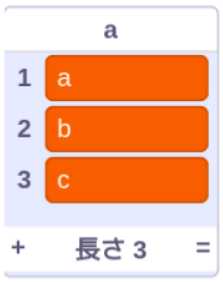
「スプライト 1 に触れたなら」の中に、「スプライト 1 に触れた　ではないまで待つ」を入れてあります。

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。

Scratch では、リストに数値でも 

、文字でも 

、


文字と数値をまぜ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 

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、 

を使ってリストを空にします。

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。

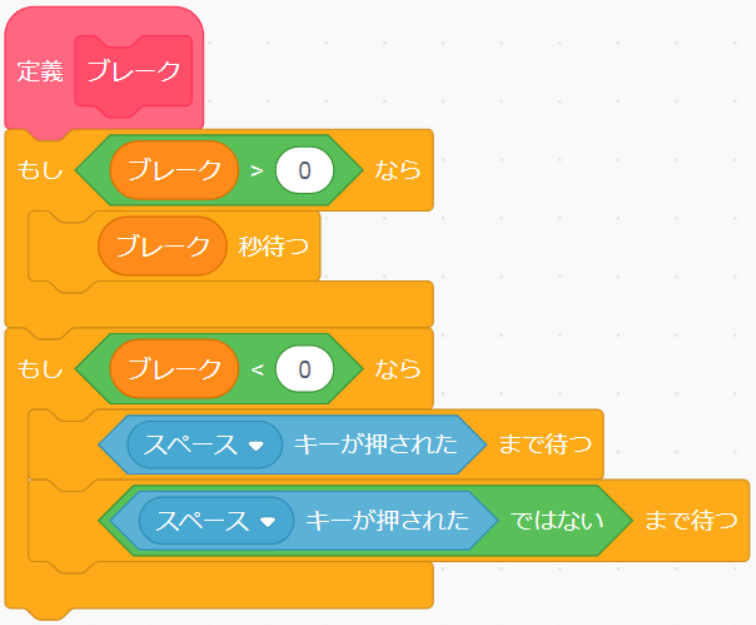
「ブロック」の中に

があります。これをクリックして、

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。

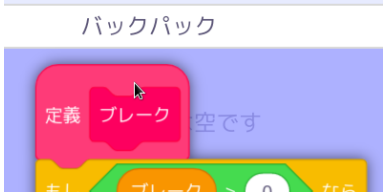
他のオプションは使いません。すると、 

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。

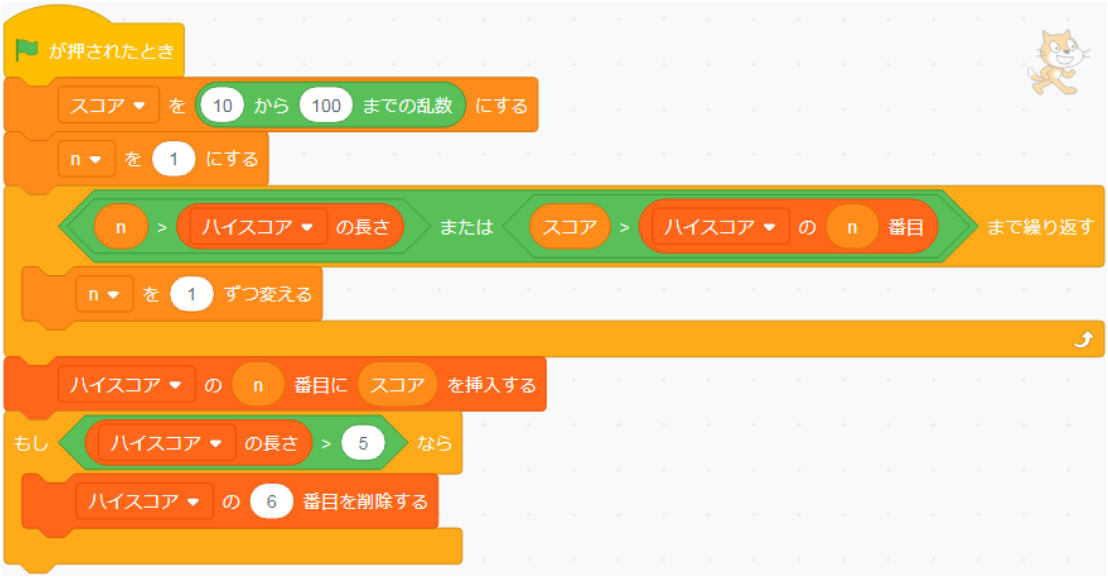


次のようにプログラムを止めてチェックしたいとろに入れて使います。

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。

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りします。

「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

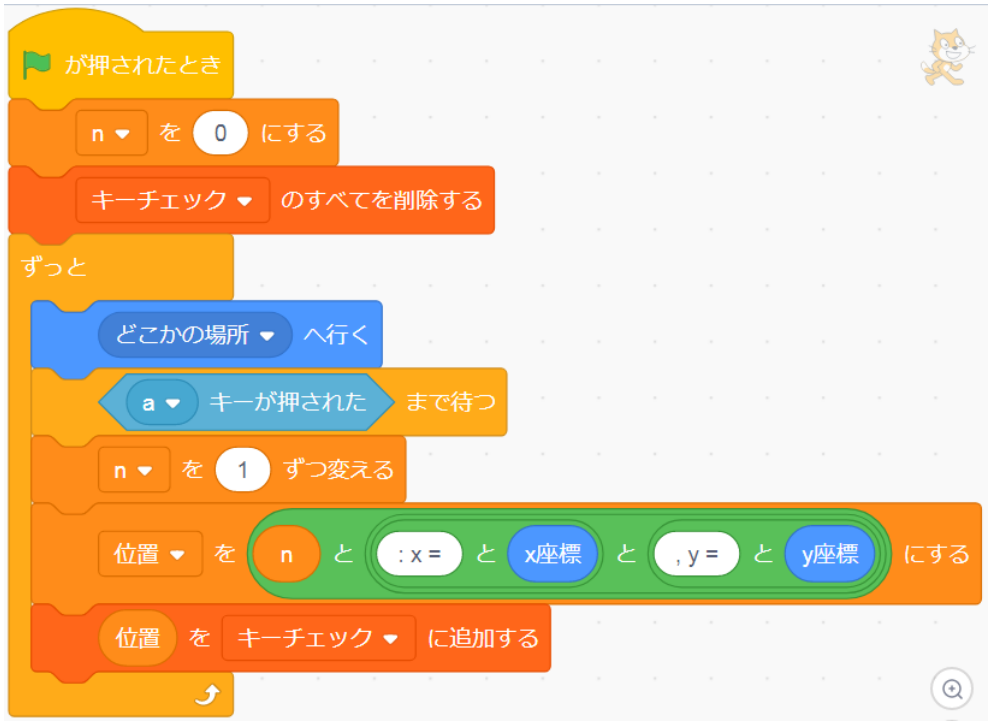
を入れる場合はそれぞれに引数を設定しなければなりません。それぞれのポイントごとに設定を変えられる利点はありますが。

［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。

大きさのテストをする「>」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」のリストにあるように、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。

の次に

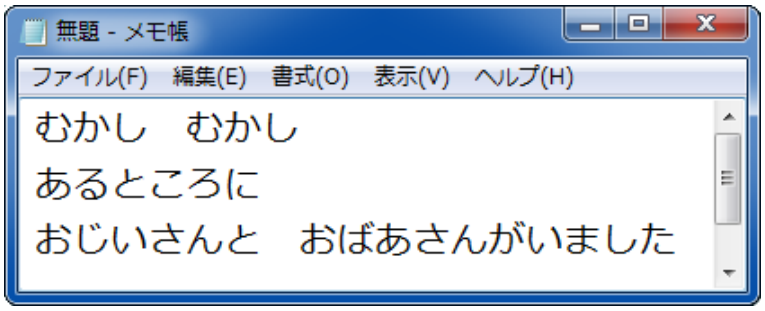
を入れてやるとうまくいくようです。

2.4　リストへの間違った位置指定

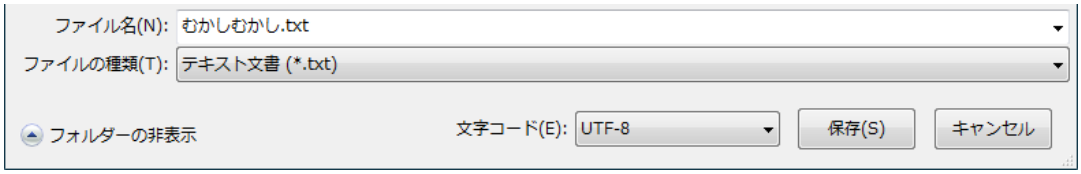
次のスクリプトでは、空のリストの 5 番目に値を入れようとしています。ですが、実行してもなにもおこりません。間違ったことを教えてくれるといいのですが。

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。

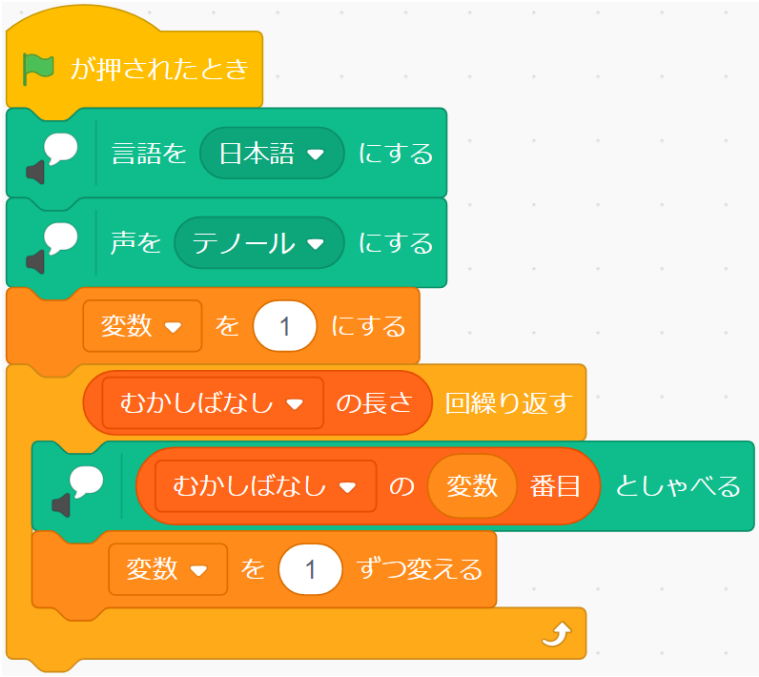


ファイルを保存する時に、次のようにファイルの種類を「テキスト文書」、文字コードを「UTF-8」にしてください。



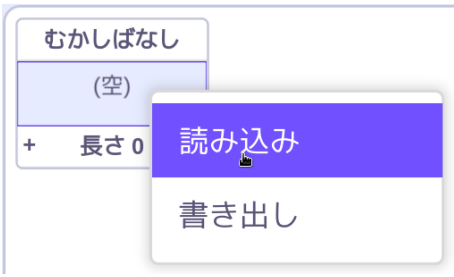
読み込んだ文章をしゃべらせるので拡張機能の音声合成を追加してください。

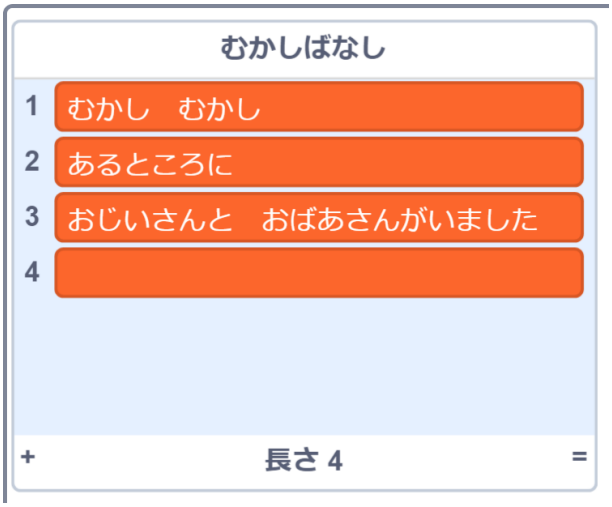
リストとして「むかしばなし」を作成し、変数は「変数」を使います。次のようにスクリプトを組んでください。



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。

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く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く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 

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っています。

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。

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。

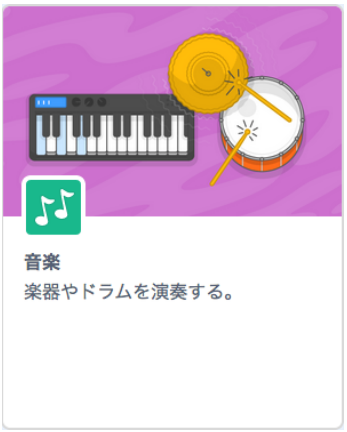
実行するとうまくいきます。

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。

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。

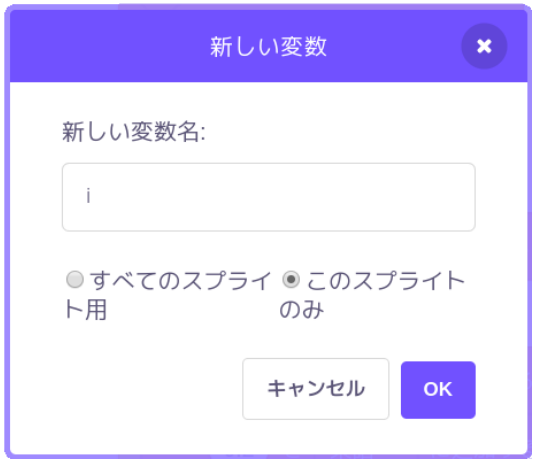
残りの三小節分のデータをつなげます。

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。

続いて、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。

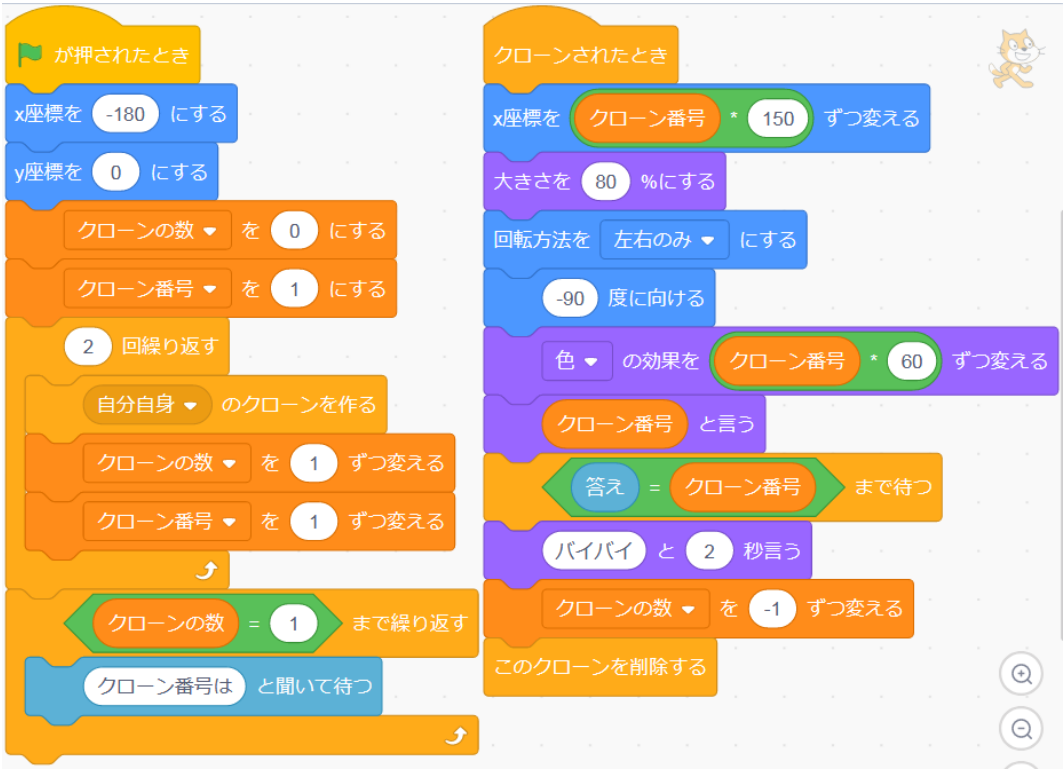
3.3 クローンと変数

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

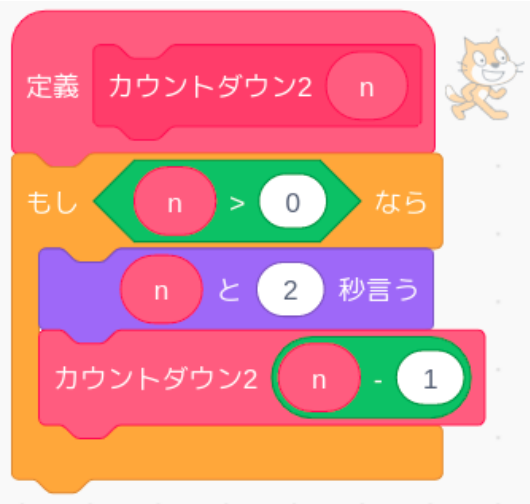
5, 4, 3, 2, 1 とカウントダウンするブロックを定義してみます。

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 

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。

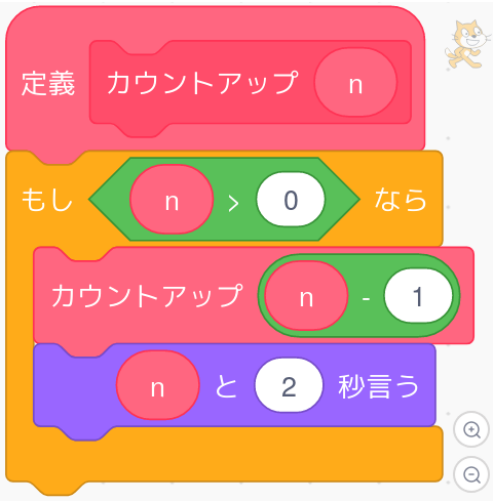


再帰呼出しでは必ず再帰からぬけるように作らなければなりません。この場合は n が 0 になった時です。再帰からぬけるように作らなかった場合は、ブラウザが不具合を起こしたり、電源をオフにするしかなくなるかもしれません。無限ループになっているようならば、すぐにストップボタンをクリックしてください。

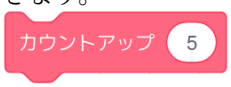
自分自身を呼出すという、不思議なものです。 

として、クリックしてみてください。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。

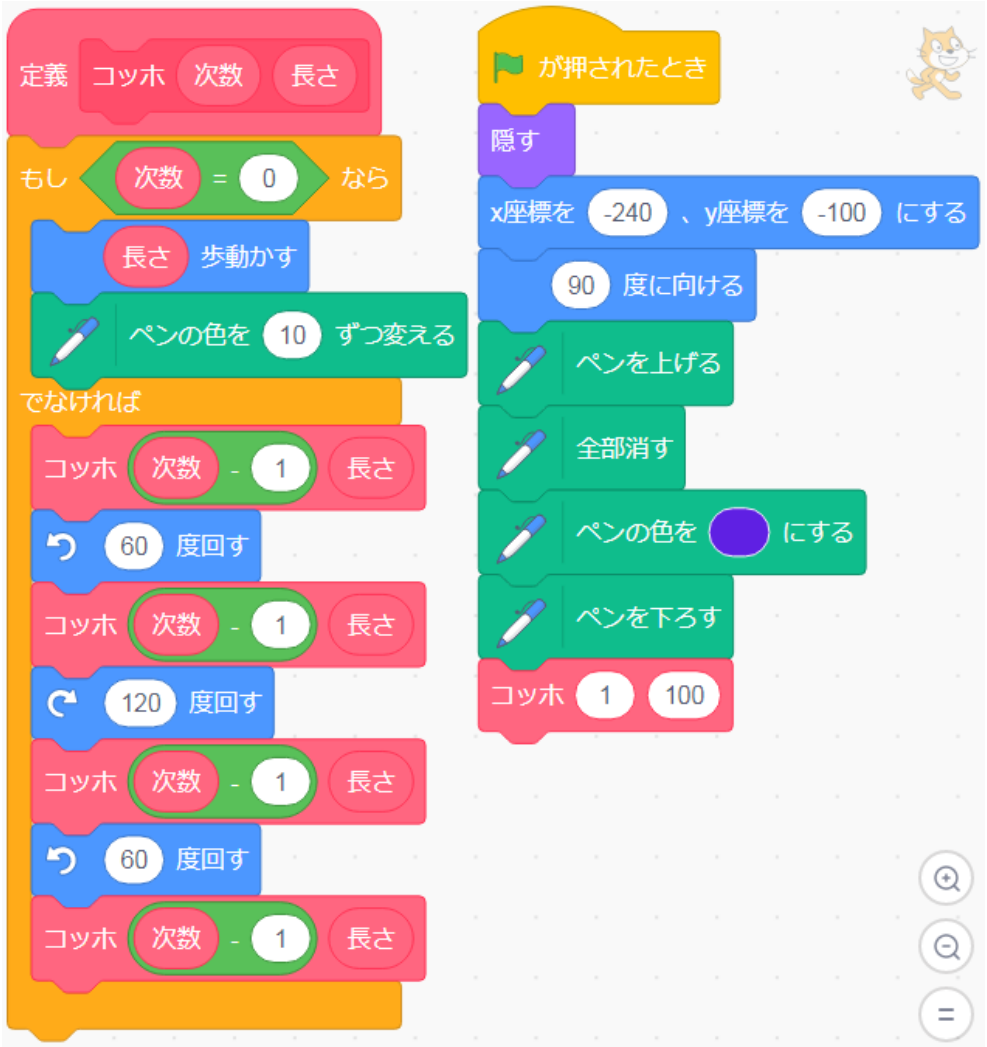


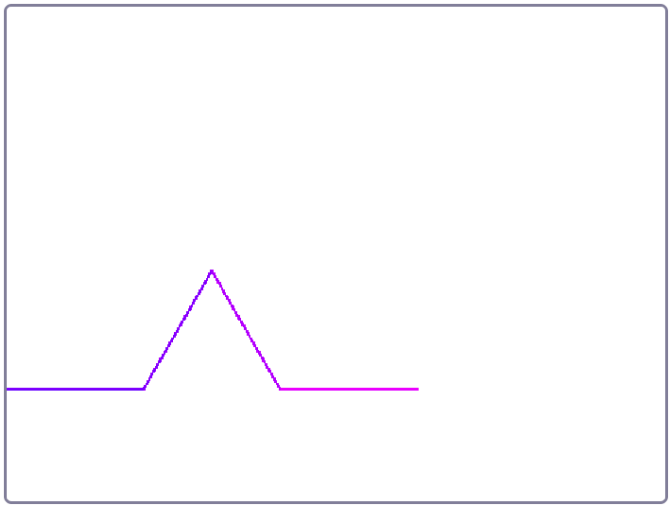
このようにするとカウントアップすることができます。



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。





次数を変えていくと、その辺に前の次数の図形が入ります。

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 9.999999999999998e+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。

次に、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。

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。