学習日：

Lesson 5 ピクさんのブランチ

条件によって、ピクさんの動きに変化を与えよう。

1 ピクさん判断する

☑ 条件式

条件式とは、式の中でも条件を表す式のことを言います。例えば「もし変数の値　角度　が 50 より大きかったら実行する」というような表現を命令で表します。そのためには、もし命令を使います。

真：その式が成り立つこと。

命令の様式	処理
もし　式1	もし 式1 が真ならば対応するおわりまでの命令を実行する。
終わり	条件文または繰返しの終了。

もし命令の引数には比較演算子を使った式を記述します。比較演算子には以下のものがあります。

例えば、以下の通りです。

☑ 比較演算子

比較演算子の様式	評価
A ＞ B	A が B より大きい
A ＞＝ B	A が B より大きいか等しい　≧と同じ意味
A ＜ B	A が B より小さい
A ＜＝ B	A が B より小さいか等しい　≦と同じ意味
A ＝＝ B	A と B が等しい　＝と同じ意味
A ！＝ B	A と B が等しくない　≠と同じ意味

例 手を振るかもしれない

左手を挙げた後、1/2 の確率で手を振ります。

```
回転待ち 左上腕 －１４０ １                    // 左手を挙げる
もし 「乱数（1,6）＞＝ 4」 // もし 1~6 までの乱数が 4 以上なら
繰り返し 3                                  // 3 回、左手を振る
回転待ち 左前腕 －６０ ０.３
回転待ち 左前腕 ６０ ０.３
終わり
終わり
```

◰　条件式（ではないとき）

もし命令で「もし○○のとき」の命令しました。今回は「もし○○ではないとき」について説明します。

命令の様式	処理
もし 式1	条件式 式1 が真ならば対応する　*他でもし*　または　*他*　または　*終わり*　までの命令を実行する。
他でもし 式1	もし対応する先述の　*もし*　または　*他でもし*　の条件が全て満たされなくて、かつ条件式 式1 が真ならば対応する　*他でもし*　または　*他*　または　*終わり*　までの命令を実行する。
他	もし対応する先述の　*もし*　または　*他でもし*　の条件が全て満たされない場合、対応する　*終わり*　までの命令を実行する。

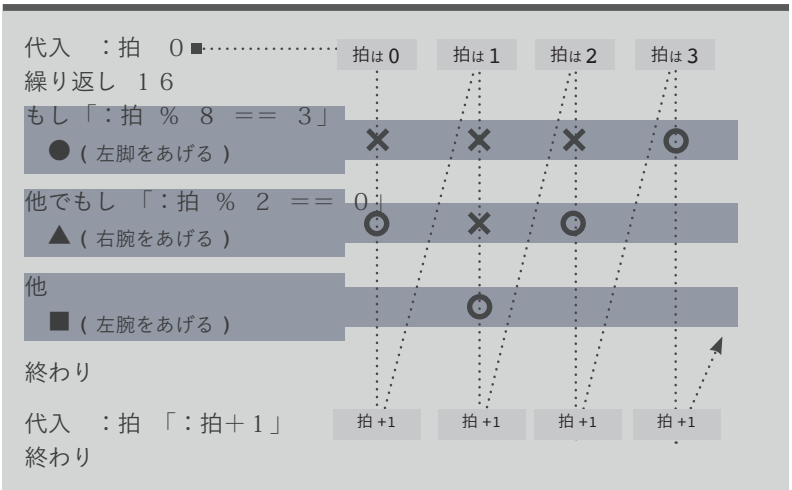
例 ピクさんの動きを計算式で書いてみます。

16 拍子の踊りをします。何拍目かによって動きの変わるプログラムです。

```
代入　：拍　０
繰り返し　１６
もし　「：拍　％　８　＝＝　３」　　// 拍を 8 で割った余りが 3 なら
回転待ち　左大腿　９０　０．２　　// 左脚を上げ下げする
回転待ち　左大腿　－９０　０．２
他でもし「：拍　％　２　＝＝　０」　　// その他の場合で拍が偶数なら
回転待ち　右上腕　９０　０．２　　// 右腕を上げ下げする
回転待ち　右上腕　－９０　０．２
他　　// その他の場合は
回転待ち　左上腕　９０　０．２　　// 左腕を上げ下げする
回転待ち　左上腕　－９０　０．２
終わり
代入　：拍「：拍＋１」
終わり
```

それぞれが動くタイミング

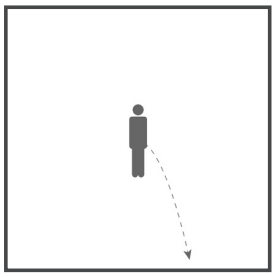
ピクさんは 拍 を記憶します。プログラムの最後の行で 拍 が 1 増えます。
繰り返し命令で、もう一度 一番上の行から実行されます。そのときには 拍 が変わっています。
ですから 実行される箇所が変わります。



◰ 式を書く

例 計算式を使ってみよう。

```
倍率　０.３
代入　：ｄｘ　１
代入　：ｄｙ　０
繰り返し　１０００
移動待ち　：ｄｘ　：ｄｙ　０.０２
代入　：ｄｙ「：ｄｙ　＋　０.０５」
終わり
```



実行結果のイメージ

ピクさんが落下します。
落下速度に注目しましょう。

式とは、値、変数、演算子の組み合わせのことを言います。値と変数はこれまでに出てきました。例えば、

```
代入　：ｄｘ　１
```

は変数ｄｘに値１を格納するという意味でした。演算子とは演算を表す記号のことをいいます。
例えば下にあるような＋、ー、＊、/、％などです。特に計算のために用いる演算子なので算術演算子と呼ばれています。

算術演算子の様式	評価
Ａ ＋ Ｂ	ＡとＢを足す
Ａ ー Ｂ	ＡからＢを引く
Ａ ＊ Ｂ または Ａ × Ｂ	ＡとＢを掛ける
Ａ / Ｂ または Ａ ÷ Ｂ	ＡをＢで割る
Ａ ％ Ｂ	ＡをＢで割ったあまり

繰り返しますが、式とは、値、変数、演算子の組み合わせです。 式の例を示します。

```
：ｄｙ　＋　０.０５
```

◰ 引数に式を使うとき

「」という記号が出てきました。先に関数のところで使いましたが、「から」までの間が一つの引数であるという意味です。引数の区切りは空白でしたが、式を書くと式の中に空白が入ることがあり、どこからどこまでが一つの引数かわからなくなるからです。よってピクトグラミングでは、式を表現するのに「 」で囲っています。

```
代入　：角度　「：角度　ー　１００　」
```

引数１　引数２

代入 命令の様式

命令の様式	処理
代入　：引数１　引数２	変数　引数１ に 引数２ を代入する。 引数１ の前には ： をつける。

式は、命令の引数には式を記述することができます DX ＋ ０.０５ を DY に代入するというのは、

```
代入　：ｄｙ「：ｄｙ　＋　０.０５」
```

左上腕に対して角度の２倍の角度を（回数 +1）の２倍の秒かけて回転するというのは、

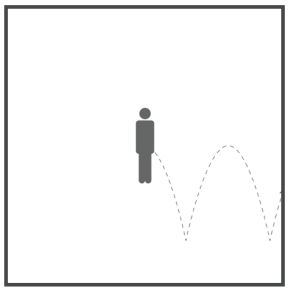
```
回転　左上腕「：角度　＊　２」「（：回数　＋　１）＊　２」
```

と書けます。

例 ピクさんにジャンプさせる

これまでに学んだ、変数、繰返し を使います。
条件式と関数を引数に使うことで以下の図形を描いてみましょう。

```
倍率　０.３
代入　：ｄｘ　１
代入　：ｄｙ　０
繰り返し　１０００
移動待ち　：ｄｘ　：ｄｙ　０.０２
代入　：ｄｙ「：ｄｙ　＋　０.０５」
もし「ｙ（）　＞　２００」
代入　：ｄｙ「－１　＊　：ｄｙ」
終わり
終わり
```



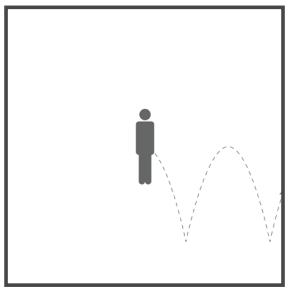
実行結果のイメージ

ピクさんが跳ねながら移動します。
3 行の追加前後でどのように変化する
でしょう。

例 ピクさんにジャンプさせる (3 行追加)

3 行加えると

```
倍率　０.３
代入　：ｄｘ　１
代入　：ｄｙ　０
繰り返し　１０００
移動待ち　：ｄｘ　：ｄｙ　０.０２
代入　：ｄｙ「：ｄｙ　＋　０.０５」
もし「ｙ（）　＞　２００」
代入　：ｄｙ「－１　＊　：ｄｙ」
終わり
終わり
もし「絶対値（ｘ（））＞　３００」
代入　：ｄｘ「－１　＊　：ｄｘ」
終わり
終わり
```



実行結果のイメージ (途中まで)

3 行追加したことで少し変化が起こります。

ここまでに紹介した命令の一覧

ピクトアニメーション命令一覧

命令の様式	処理
回転 引数1 引数2 引数3 引数4	引数4 秒後に 引数1 で指定される体の部位を反時計回りに 引数2 度だけ 引数3 秒かけて支点を中心に等速回転する。引数4 が省略された時は、引数4 に 0 が、引数3、引数4 の両方が省略された時はいずれも 0 が入力されているものとして取り扱う。
回転待ち 引数1 引数2 引数3	引数1 で指定される体の部位を反時計回りに 引数2 度だけ 引数3 秒かけて支点を中心に等速回転する。回転が終了する まで次の命令は実行されない。
移動 引数1 引数2 引数3 引数4	引数4 秒後に 引数3 秒かけて x 軸正方向に 引数1 ピクセル，y 軸正方向に 引数2 ピクセルだけ全体を等速直線移動する。引数4 が省略された時は、引数4 に 0 が、引数3，引数4 の両方が省略された時はいずれも 0 が入力されているものとして取り扱う。
移動待ち 引数1 引数2 引数3	引数3 秒かけて x 軸正方向に 引数2 ピクセル，y 軸正方向に 引数3 ピクセルだけ全体を等速直線移動する。直線移動が終了するまで次の命令は実行されない。
正面	ピクさんを正面向き（初期状態）にする。
側面	ピクさんを側面向きにする。
クリア	ピクさんの状態を直立状態（初期状態）にする。

算術演算子

算術演算子の様式	評価
A ＋ B	A と B を足す
A － B	A から B を引く
A ＊ B または A × B	A と B を掛ける
A / B または A ÷ B	A を B で割る
A ％ B	A を B で割ったあまり

比較演算子

比較演算子の様式	評価	
A ＞ B	A が B より大きい	
A ＞＝ B	A が B より大きいか等しい	≧と同じ意味
A ＜ B	A が B より小さい	
A ＜＝ B	A が B より小さいか等しい	≦と同じ意味
A ＝＝ B	A と B が等しい	＝と同じ意味
A ！＝ B	A と B が等しくない	≠と同じ意味

◰　ピクトアニメーション、ピクトグラフィクス共通の命令一覧

命令の様式	処理
代入：引数 1　引数 2	変数　引数 1 に　引数 2　を代入する。引数 1 の前には ： をつける。
もし　式 1	条件式 式 1 が真ならば対応する　*他でもし*　または　*他*　または　*終わり*　までの命令を実行する。
他でもし　式 1	もし対応する先述の　*もし*　または　*他でもし*　の条件が全て満たされなくて、かつ条件式 式 1 が真ならば対応する　*他でもし*　または　*他*　または　*終わり*　までの命令を実行する。
他	もし対応する先述の　*もし*　または　*他でもし*　の条件が全て満たされない場合、対応する　*終わり*　までの命令を実行する。
繰り返し　引数 1	対応する　*終わり*　までの命令を 引数 1 回繰返す。
終わり	繰返しの終了。
待ち　引数 1	引数 1 秒間、何もせずに待つ。待ちが終了するまで次の命令は実行されない。
倍率　引数 1	ピクさんの拡大率を 引数 1 にする。（標準は 1 ）

◰ 関数一覧

関数の表記	機能	戻り値
乱数（最小，最大）	整数 最小 以上 整数 最大 以下のランダムな値を返す。	整数 最小 以上 整数 最大 以下のランダムな値。
角度（部位 ）	体の部位を示す文字列 部位 の向きを右向き（X 軸正方向）を 0 度として反時計回りの角度を返す。また値は、0 から 359 の整数値をとる。体の部位を示す文字列は二重引用符 (") で囲うこと。例えば左上腕の角度を知りたいときは「角度 (" 左上腕 ")」となる。	体の部位を示す文字列 部位 の向き (0 から 359 の整数値をとる)。
x（）	ピクさんの x 座標を整数値で返す。	ピクさんの x 座標
y（）	ピクさんの y 座標を整数値で返す。	ピクさんの y 座標
余弦（角度）	角度 度の余弦を返す	角度 度の余弦
正弦（角度）	角度 度の正弦を返す	角度 度の正弦
正接（角度）	角度 度の正接を返す	角度 度の正接
絶対値（値）	値 の絶対値を返す	値 の絶対値

ピクさんからの挑戦状

5回目のピクさんからの挑戦状です。ピクさんからの挑戦状は、あなたがやりたいと思うところからやってください。ピクチャレ 1 から順に取り組む必要はありません。また、全てのピクチャレに取り組む必要もありません。あなたがやりたいピクチャレに取り組んでください。

さあ、ピクさんからの挑戦状「ピクチャレ」に挑戦してみましょう。

ピクチャレ 1 腕をあげさげさせよう

次の (a) から (c) の条件を満たすプログラムを作成してみよう。

(a) 5 分の 4 の確率で 1 秒かけて左腕を 120 度時計回りに回転する。
(b) 7 分の 3 の確率で 1 秒かけて右腕を 120 度反時計回りに回転する。
(c) (a) の動作と (b) の動作は同時に行われる。
　左腕を上げる行為と、右腕を上げる行為はそれぞれ別々に判断される。

ピクチャレ 2 ピクさんダンサー

30 拍からなるダンスをします。1 拍は 0.6 秒とします。最初は 1 拍目、最後は 30 拍目です。

(1) 3 の倍数でもあり、5 の倍数であるとき、つまり 15 の倍数の時は、最初の 0.3 秒で左腕を −120 度、右腕を 120 度それぞれ同時に回転し、その後 0.3 秒で左腕を 120 度、右腕を −120 度それぞれ同時に回転します。

(2) 15 の倍数ではなく、3 の倍数の拍目の時は、最初の 0.3 秒で左腕を −120 度回転し、その後 0.3 秒で左腕を 120 度回転します。

(3) 15 の倍数ではなく、5 の倍数の拍目の時は、最初の 0.3 秒で右腕を 120 度回転し、その後 0.3 秒で右腕を 120 度回転します。

(4) 3 の倍数でもなく、かつ 5 の倍数でもないときは、最初の 0.3 秒で左足を −10 度回転し、その後 0.3 秒で左足を 10 度回転します。

ピクチャレ 3 オリジナルピクさんを作ろう

条件分岐（もし）を使ったオリジナル作品を作ってみよう。

MEMO