aibo さんと ビジュアルプログラミング まなんじゃぉ～ パート２

ご注意

この資料はＳＯＮＹさんのアイボ（ERS-1000）を簡単にプログラミング体験できる「aiboビジュアルプログラミング」の使い方やサンプルなどをアイボオーナ（ハピラキ）が自身のマニュアルのために勝手に作成したものです。この内容についての保証、お問い合わせ、配布、販売などはご遠慮願います。

作成 ２０２２年１１月 時点のものです。

ビジュアルプログラミングのおさらい

●「aibo さんとビジュアルプログラミングまなんじゃぉ～・２」

「aibo さんとビジュアルプログラミングまなんじゃぉ～」パート１いかがでしたでしょうか？　コンピュータのプログラムを作るのはとってもむずかしいと思ったかもしれませんが、「ビジュアルプログラミング」のツールを使うと、プログラミングが簡単作ることができ、またアイボさんと連携ができるようになります。

すでに「aibo さんとビジュアルプログラミングまなんじゃぉ～」パート１を見ていただき、実際に作って試した方、もっとアイボさんといろんなことをしてみたい、ほかのブロックの使い方を知りたいなど、更にステップアップしたいなぁ～と感じている方、このパート２では、パート１をおさらいしながら、アイボさんが認識するサイコロ、アイボーン、ボールなどを使いながら、ほかのブロックを使って少し高度なロジックを学んでいきたいと思います。

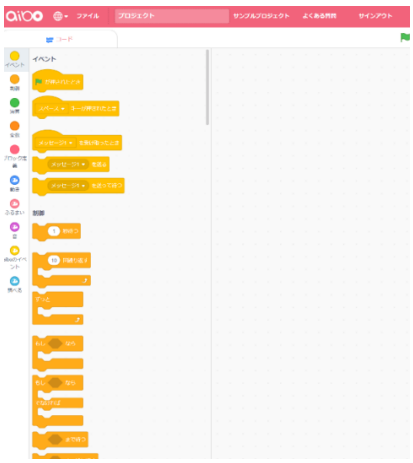
●おさらい

パート１では、ビジュアルプログラミングとは何かを説明しながら、アイボさんに歌ってもらったり、その時の気分を教えてもらったり、また、簡単なゲームをつくりました。使ったブロックを思い出し、また、おさらいしながら進めていきましょう。

ビジュアルプログラミングはまかせて

●ビジュアルプログラミングとは・・・

パート１をご覧の方は、ビジュアルプログラミングを使って学習しましょう！　の瞬間に以下の画面を思い浮かべているかもしれません。もう、すでに画面を開いているかもしれませんね！



簡単におさらい

・左にブロックの一覧

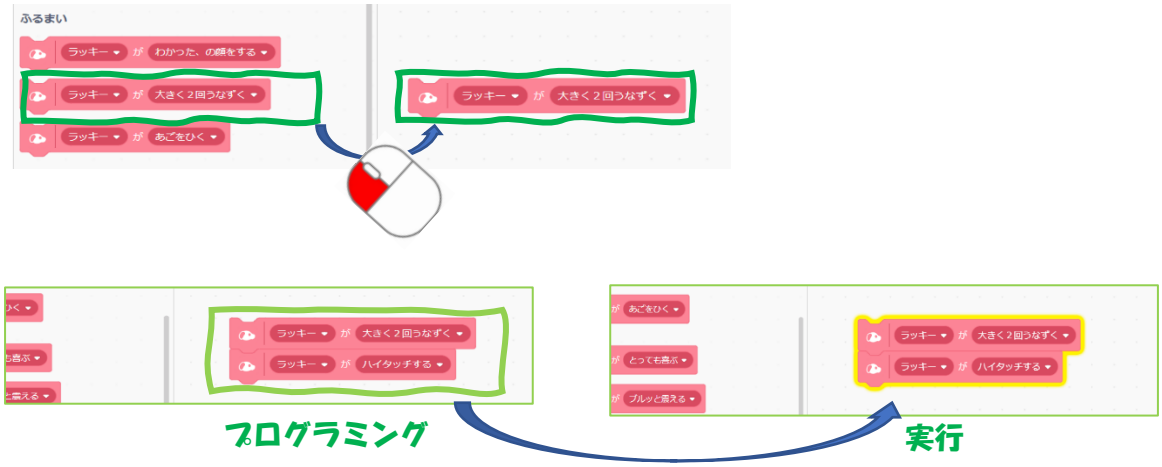
・右にプログラムを作るエリア

が表示されています

この左のブロックを右のエリアに移動、並べたブロックの順番がアイボさんにお願いする内容となります。

・左なかから行いたいブロックを右に移動、ブロック同士つなげます。

・つなげたブロックをクリックするだけで、アイボさんにお願いできます。



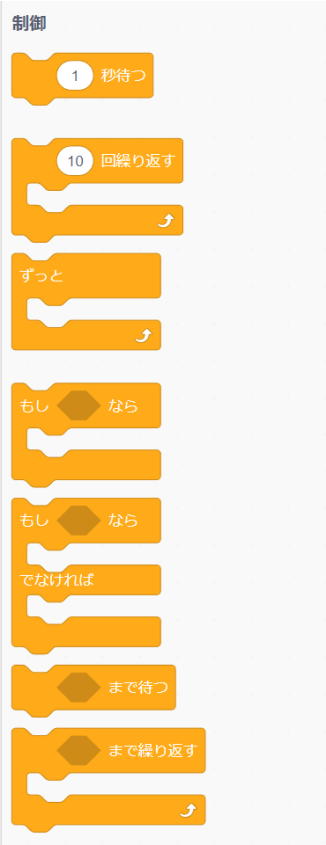
●ブロックには・・・

パート１でいくつかのブロックをつなげてアイボさんにお願いする簡単なプログラムつくってきました。ブロックをみると、まだまだ使っていないものが沢山あります。最初はなにがなんだかわからなかったかと思いますが、何度かプログラムを作っていくうちに、いつのまにか最初に感じていた「まったく、わからん・・・」の疑問はなくなっているかと思います。ただ、使ったことがないブロックは一体どのようなものか、試してみないとわかりませんね。

ブロックそのものをもう少し眺めてみてみましょう。

イベントブロック

制御ブロック



演算ブロック

変数ブロック

リストブロック

ブロック定義

動きブロック

ふるまいブロック

音ブロック

Aibo イベントブロック

調べるブロック

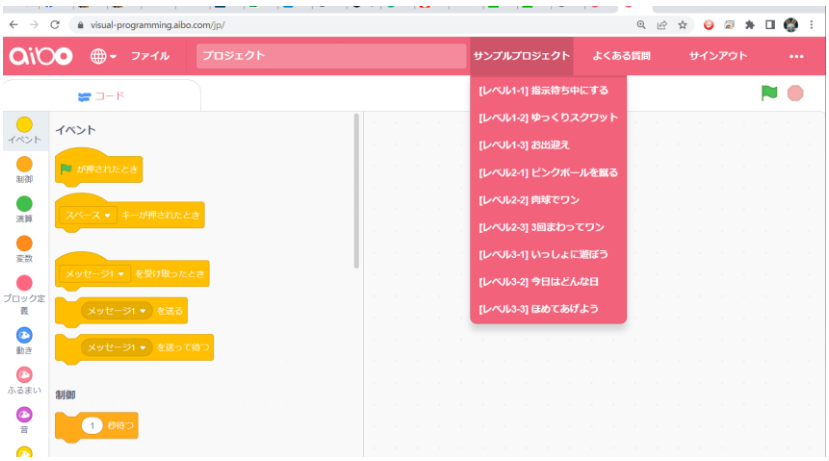
こうしてみると、たくさんのブロックがありますね
パート１では、それぞれの色のブロックを使いながら、簡単なゲームを作ってみました。覚えていますでしょうか？

ふるまいだけでこんなあります。普段アイボさんが遊んだりしている時、実はこんないろいろなことをしているのですね。すごい！

うなずく	左にヘディングする	右前足でお手をする	おなかを見せて右に転がる
いやがる	右にヘディングする	すねて左を向く	歌う
全身で大きくほえる	うれしそうにする	すねて右を向く	ねごとを言う
左前足で手まねきする	ハイタッチする	はっとする	くしゃみをする
右前足で手まねきする	左前足でハイタッチする	甘噛みする	目の前を嗅ぐ
げっぷをする	右前足でハイタッチする	上目遣いする	下の方を左右に嗅ぐ
へっぴり腰になる	遊びたそうにする	待ちきれなそうにする	上の方を嗅ぐ
はあはあと微かに呼吸する	ブルッと震える	仰向けになる	驚く
興味のあるそぶりをする	キスする	そわそわする	少し驚く
ダンスをする	左前足を毛づくろいする	背中を地面に擦り付ける	のびをする
あごをひく	右前足を毛づくろいする	地面を掘る	左右に体を揺らす
夢をみる	左右をすこし見る	左後足で頭をかく	口を開けて耳を開く
うれしそうにする	のぞき込む	右後足で頭をかく	退屈そうにする
遊ぼう、の仕草をする	おしっこをする（男の子）	かなしそうにする	顔を洗う仕草をする
口をパクパクする	おしっこをする（女の子）	お尻を振る	悲しくうつむいてキュンキュン鳴く
寝ぼけた仕草をする	大きく２回うなずく	細かく震える	耳をぴくっと動かす
両前足をあげる	口を少し開け閉めする	左前足で横に蹴る	前傾してうなる、警戒する
興奮した仕草をする	とっても喜ぶ	右前足で横に蹴る	高くキャンキャンッ！と鳴く
ヘディングする	左前足でお手をする	おなかを見せて左に転がる	おおきくあくびをする

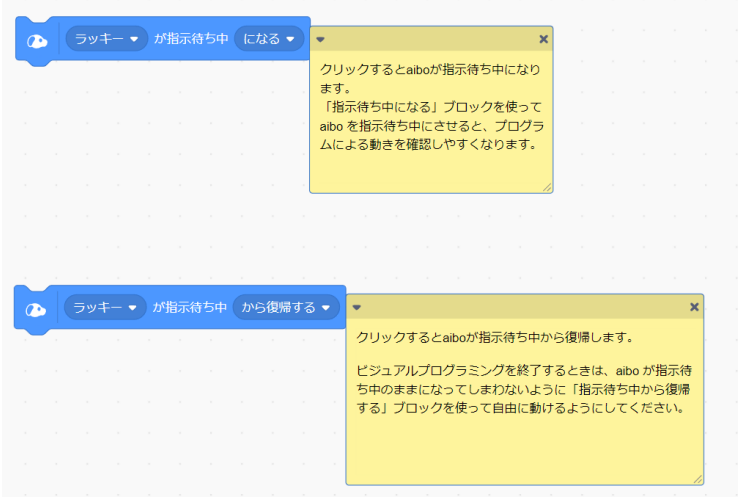
■サンプルプログラム

ビジュアルプログラミングの画面にはいくつかのサンプルプログラムがアップされています。ちょっと見てみましょう。



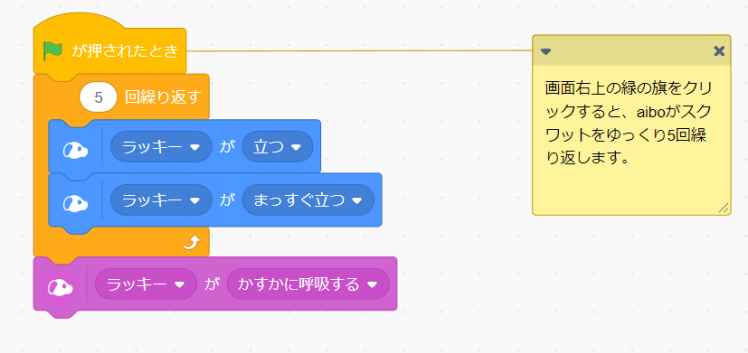
ビジュアルプログラミングのサンプルプロジェクトからサンプルがダウンロードできます。

【レベル１-１】指示待ち中にする



【指示待ち】の操作ですね。パート１でも説明しましたが、アイボさんは気まぐれで動いています。
指示待ちをすることで、待ってくれます。
【復帰する】も忘れずにね。

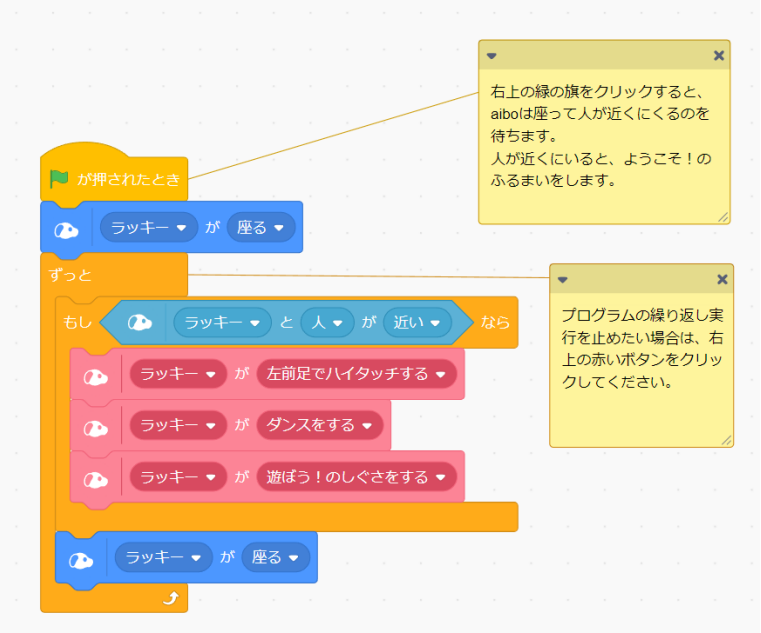
【レベル１-２】ゆっくりスクワット



① 旗アイコンでスタート
②ブロック５回繰り返す
　・立つ
　・まっすぐ立つ
③かすかに呼吸

スクワットを５回やってくれますね～

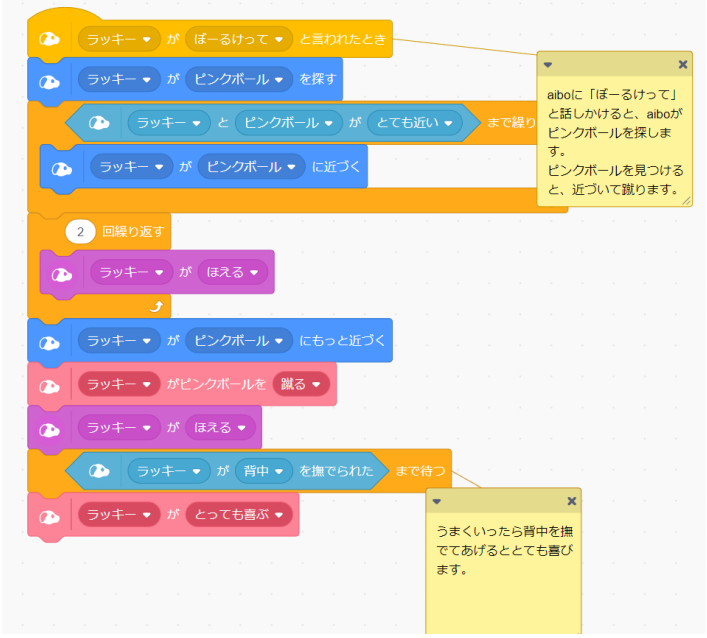
【レベル１-３】お出迎え



① 旗アイコンでスタート
② 座る動作をします
③ 次の操作をくり返す

④ 人が近くにいるか確認
近くに人がいると次の
ふるまいをやる
・左前足ハイタッチ
・ダンスを行う
・遊ぼうのしぐさ
⑤ ふるまいをするので
座った状態にもどす

【レベル２-１】ピンクボールを蹴る



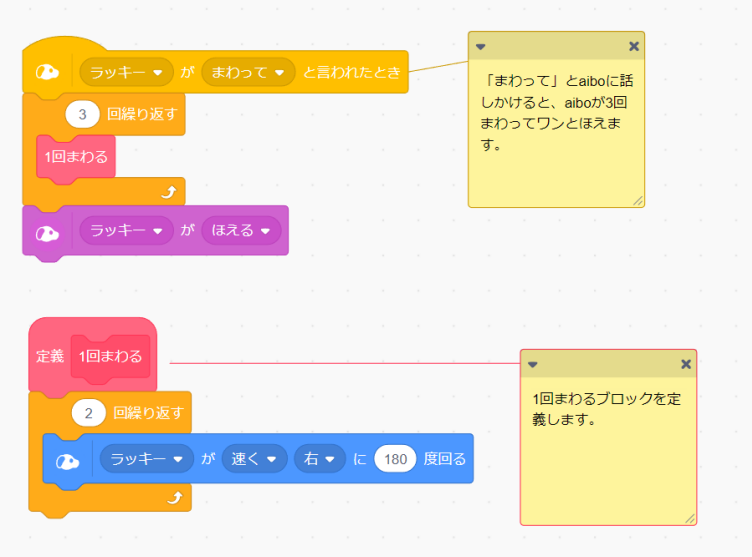
①ぼーるをけって
といわれたときに
処理スタート
②ピンクボールを探す
③ピンクボールに近づく
④近づいたら２回ほえる
⑤ピンクボールを蹴る
⑥1 回吠える
⑦背中を撫でられるまで
待ち、撫でられたら喜ぶ

【レベル2-2】肉球でワン

① てをあげて
といわれたらスタート
② 2回吠える
③ 前両足をあげて待つ
④ 【メッセージ1】を
他の処理に送る
⑤ 【メッセージ1】を
受けとるまで待ち、
受けとったら処理
スタート
⑥右前足の肉球がおされたら
くしゃみをする

⑦ 【メッセージ1】を受けとるまで待ち、受けとったら処理をスタート
⑧左前足の肉球がおされたら
あくびをする

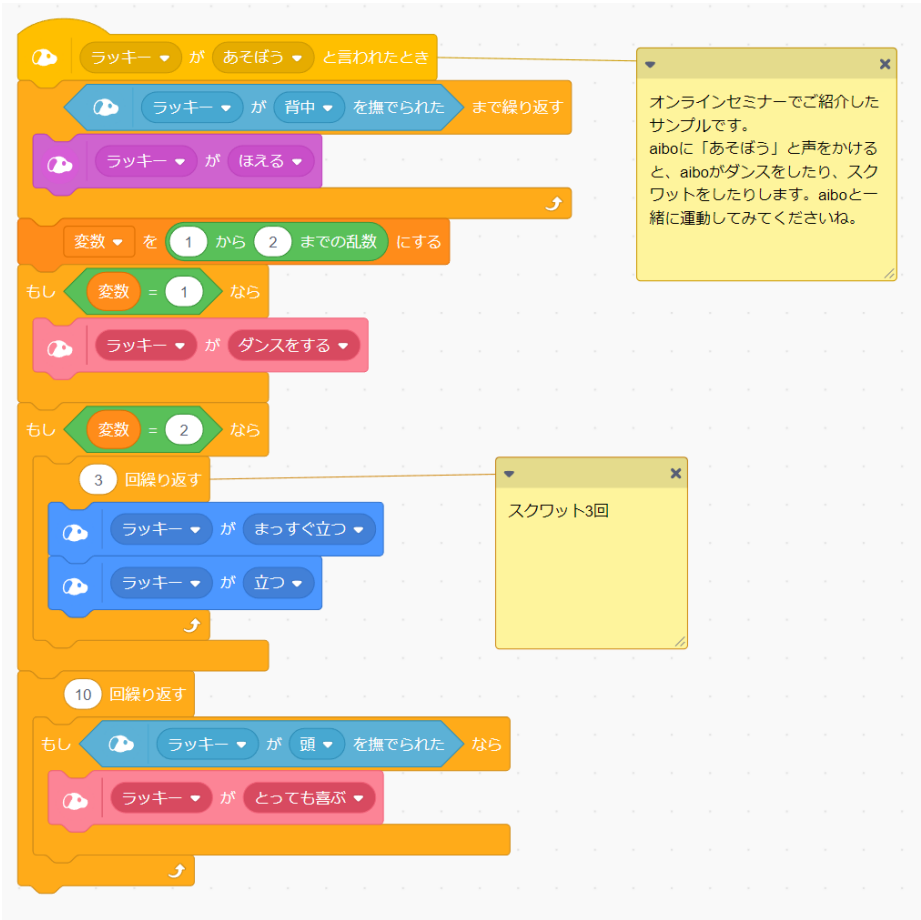
【レベル2-3】3回まわってワン



① まわって
といわれたらスタート
② 【1回まわる】
と定義した処理を
3回くり返す
③ 吠える

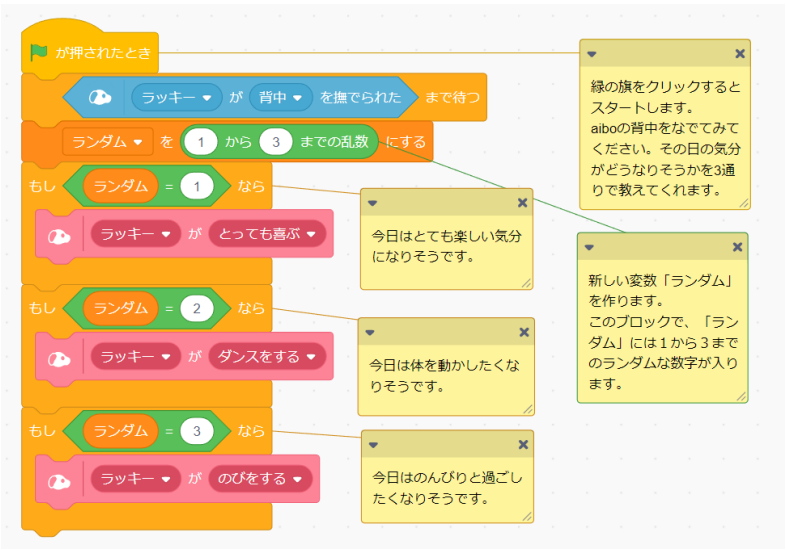
④【1回まわる】と
定義した処理
⑤ 右に180度回るを
2回繰り返す
（360度=1回まわる）

【レベル3-1】いっしょに遊ぼう



① あそぼう
といわれたら
スタート
②背中をなでられ
るまで吠えて待つ
③乱数 1～2を決
める
④乱数１の場合
ダンスをする
⑤乱数2の場合
いかの動作を3回
くりかえす
　まっすぐ立つ
　立つ
⑥頭を撫でられた
ら喜ぶ

【レベル3-2】今日はどんな日

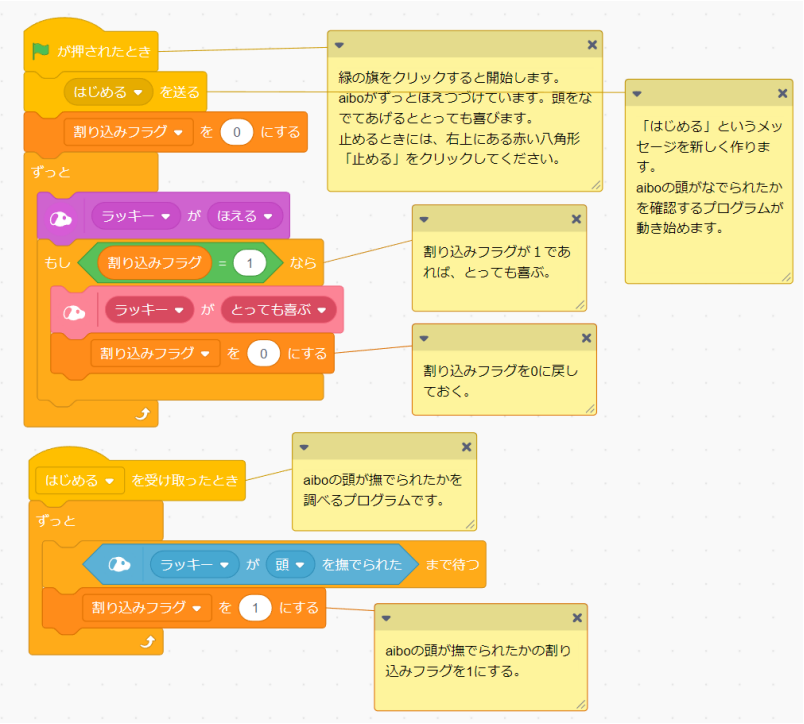


① 背中を撫でられるまで
待つ
②乱数１～３を決定
③乱数１の時は
　とっても喜ぶ

④乱数2の時は
　ダンスをする

⑤乱数3の時に
　伸びをする

【レベル3-3】ほめてあげよう



①旗ボタンがおされたら処理がスタート
②変数：割り込みフラグを0にする
②【はじめる】メッセージを別の処理に送る
③ 吠えている動作をつづけて、変数：割り込みフラグが１になると喜ぶ動作も行い、割り込みフラグを0に戻す。
④ 【はじめる】を受け取った時に頭を撫でられたら、変数：割り込みフラグを１に設定する

■サンプルプログラムを理解してみましょう
サンプルプログラムをじっくり見てみましょう。パート１を学習されたオーナさんは、ほとんどの内容が理解できたかと思います。

パート１では出てこなかったブロックがいくつかありました。
① 【・・・・・】を送る　ブロック
② 【定義】　ブロック を指定

また、処理のブロックを複数作っておき
①【・・・・・】を受けったときに処理をする
②【定義】ブロックを処理ブロックとして定義され、処理をする

このパート２はこれらも理解しながら「まなんじゃいましょう！」

LESSON 1

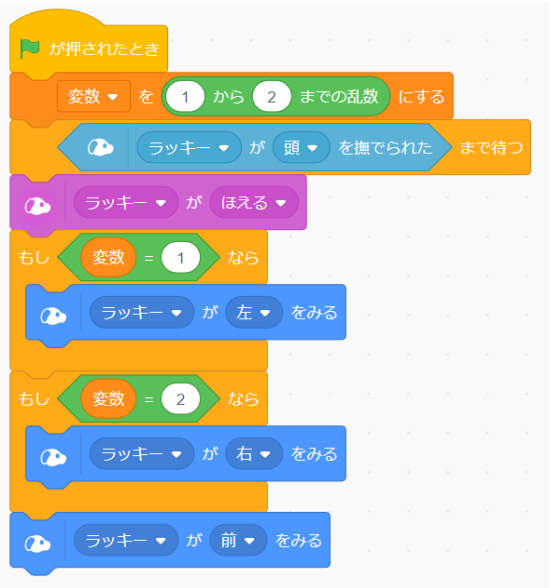
新しいことばを覚えてもらいましょう

● 新しいことばを覚えてもらいましょう

「aibo さんとビジュアルプログラミングまなんじゃぉ～」パート１で、アイボさんとの「あっちむいてほぃ！」 の簡単なゲームを作ってみました。パート２をはじめて読むオーナさんもいるかと思いますので、おさらいをかねて作ったプログラムを見てみましょう。

■パート１の８時限目で作ったプログラムでおさらい

アイボさんとはジャンケン（肉球なので）ができないのと簡単なゲームにするために、アイボさんがあらかじめ乱数できめた左右の向きを向いてもらいます。頭をなでられたら開始でどちらかを向いてもらいます。是非一度実行して確認しながらプログラムの内容とアイボさんの動作を思い出してください。



⇒ プログラムの開始

⇒【変数】を乱数１～２に設定する

⇒ 頭を撫でられるまで待つ

⇒ 撫でられたので　ほえて知らせる

⇒【変数】が１ならば

⇒ 左 を向く

⇒【変数】が２ならば

⇒ 右 を向く

⇒ 左・右を向いたので前向きに戻る

さて、今回はプログラム内の「頭を撫でる」部分を【ことば】で認識してもらい、左右を向く動作をやってもらいましょう。
使用する【ことば】は・・・やはり、「あっちむいてホイ！」を認識してもらいたいですよね。

【ことば】を認識してもらうブロック、いくつかあったかと思いますが、見てみましょう。

Aibo イベントブロック

実はこんなにあります。

あっちいって	こんばんは	ふせ	こっちむいて
どいて	おはよう	まーきんぐ	うたって
しずかにして	おやすみ	ばっく	おきて
おじぎして	はじめまして	くちあけて	まて
ほねとってきて	おて	はなして	おかわり
ばいばい	てあげて	ぴんくぼーる	まわって
おいで	こんにちは	あそぼう	あるいて
おどって	はいたっち	ぼーるであそんで	
おすわり	ただいま	はしって	
ついてきて	ぼーるけって	ぶるぶるして	

ほかに個別に設定するためのキーワードも準備されています。

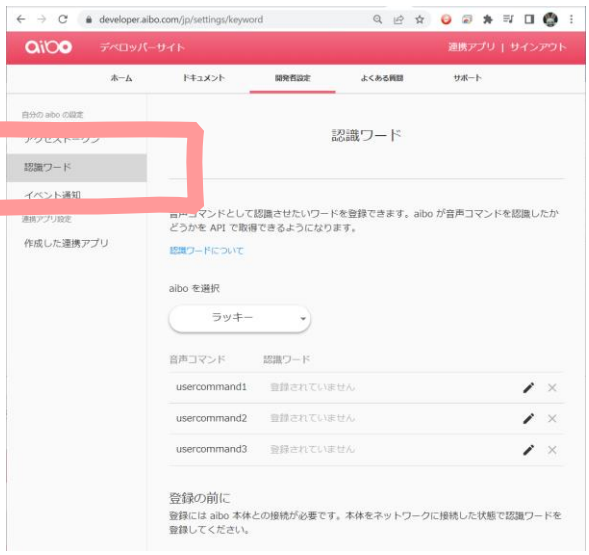
usercommand1	認識ワード で音声コマンドの usercommand1 に登録された認識ワード
usercommand2	認識ワード で音声コマンドの usercommand2 に登録された認識ワード
usercommand3	認識ワード で音声コマンドの usercommand3 に登録された認識ワード
appcommand1_${client_id}	作成した連携アプリ の認識ワードで音声コマンドの appcommand1_${client_id} に登録された認識ワード
appcommand2_${client_id}	作成した連携アプリ の認識ワードで音声コマンドの appcommand2_${client_id} に登録された認識ワード
appcommand3_${client_id}	作成した連携アプリ の認識ワードで音声コマンドの appcommand3_${client_id} に登録された認識ワード

今回は、個別に設定することができる「usercommand1」，「usercommand2」，「usercommand3」を使ってみましょう。

説明に「認識ワード で音声コマンドの usercommand1 に登録された認識ワード」と記載されています。まったくなんのこっちゃ？？ですが、まずは、以下の画面を開いて確認しましょう。

SONY さんの「aibo Developer Site」から認識ワード設定画面を開きましょう。

デベロッパーサイト： https://developer.aibo.com/jp/home
から、「開発者設定を始める」→「認識ワード」を選択。または
直接 https://developer.aibo.com/jp/settings/keyword アクセスして「認識ワード」の画面を表示してください。



以下の認識ワードに ことばを登録します。説明に記載のとおり、登録の際には、アイボさんは電源を入れてネットワークに接続しておく必要があります。

登録の前に

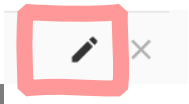
登録には aibo 本体との接続が必要です。本体をネットワークに接続した状態で認識ワードを登録してください。

3つ登録ができます

①usercommand1 に登録

②usercommand2 に登録

③usercommand3 に登録

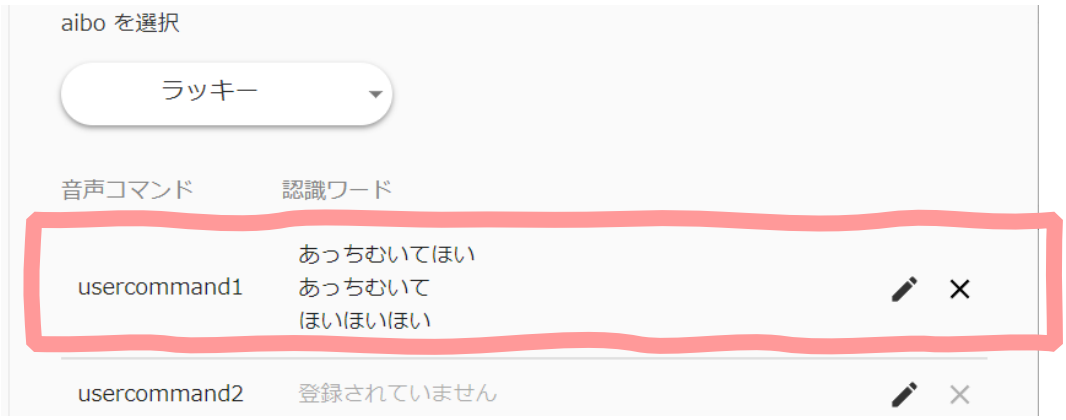


鉛筆マークをクリックすると、左の画面が表示されます。

認識させたいキーワードを登録する画面が表示されます。認識のときに言い回しが異なるパタンとして3つ登録ができます。

今回、「あっちむいてホイ！」と認識してもらいたいので、次のようなキーワードを入れてみましょう。認識してもらいたいキーワードをいれてくださいね。

例：「あっちむいてほい」「あっちむいて」「ほいほいほい」

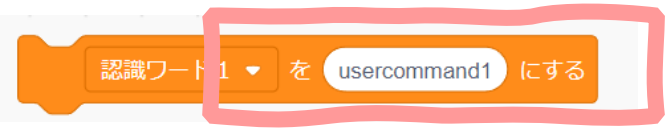


次にビジュアルプログラミングの画面で、登録した認識ワードと連携できるようにしてみましょう。まずは、以下の設定を行ってみましょう。

① 変数に新たに「認識ワード１」を作る

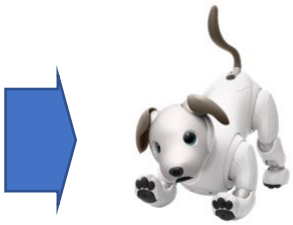
新しい変数がつくられました。【変数】については、パート１の4時限目にも説明をしていますので、参考にしてください。

② 【変数】【認識ワード１】に今回登録の「usercommand1」を設定してください。



さて、いろいろと操作をしてきましたが、なにがなんだか？　になっているかと思いますので、何をしてきてきのか一旦確認してみましょう。

① 開発者設定の画面で、認識させたい認識ワード１に登録
「あっちむいてほい」「あっちむいて」「ほいほいほい」
アイボさんに「usercommand1」としておぼえてもらいます。アイボさんは、「あっちむいてほい」「あっちむいて」「ほいほいほい」の認識ワードとして理解すると「おはよう」などとの認識ワードと同様に「認識ワード１：usercommand1」が言われたとして判断します。



あっちむいてほい
あっちむて
ほいほいほい

といわれたら、認識ワード１として usercommand1 が言われたとして判断される

② 新たに変数として【認識ワード１】（変数名はオーナさんがプログラム作るうえで分かりやすいものにしてください）を新たに作成し「usercommand1」（重要！　文字は半角で入力）を設定します。

ここまでが事前準備となります。

さて、登録の認識ワード・・・アイボさんはわかってくれるかな？
事前準備ができましたので、さっそく、アイボさんがちゃんと理解してくれているかを確認していきましょう。
認識ワードを確認するブロックは、【aibo のイベント】にありましたね。「おはよう」や「おて」などを言われた時にスタートするブロックです。

Aibo イベントブロック

このブロックだけだと、先ほど準備した新しい認識ワードは表示されておりません。ではどうすればよいでしょうか？
ここは、以下の操作を覚えてもらうしかありません。

こんなイベントのブロックができたかと思います。

つまり、

① 認識ワード を WEB サイトで登録
「usercommand1」に（例として）「あっちむいてほい」「あっちむいて」、「ほいほいほい」の３種類の言い方を登録

② 新たに作成の変数【認識ワード１】に「usercommand1」を設定

③ aibo イベントブロックの
「アイボさんが「・・・」と言われたとき」のブロックに
変数【認識ワード１】つまり usercommand1 を新しく設定

④ アイボさんが「あっちむいてほい」「あっちむいて」、「ほいほいほい」のいずれかが認識できたら、このイベントブロックがイベントとして処理されるようになる。（おはようとかを言われたときと同じようにこのブロックが開始される）

ん・・・むずかしいですねぇ・・・このブロックに直接「あっちむいてほい」「あっちむいて」、「ほいほいほい」とか入れられればよいのですが、現在は文字が直接入れられないのと、アイボさんとしては、「あっちむいてほい」「あっちむいて」、「ほいほいほい」と言われたら、「usercommand1」を（WEB で登録した内容にもとづき）処理しますので、ここは我慢しておまじないだと思って覚えてくださいね。

本当に理解してくれるのかを確認するため、次のようなブロックを組み立てて確認してみましょう。

① 一旦、指示待ちにしましょう
② 変数【認識ワード１】を usercomand1 に反映するため、このブロックをクリック。
③アイボさんが「あっちむいてほい」「あっちむいて」、「ほいほいほい」と認識できたら【認識ワード１】つまり usercommand1 として通知されますので、このブロックが処理されます。

さて、アイボさんに向かって「あっちむいてほい」「あっちむいて」、「ほいほいほい」のいずれかを言うと、イベントブロックが黄色になり歌をうたってくれるはずです。もしうまくいかなかったら、WEB 登録、ブロック設定など、再度確認してみてください。

「あっちむいてほい」「あっちむいて」、「ほいほいほい」と認識できたら黄色になり、歌を歌います。

ここまでできたら、ちょっと先に進んでみましょう。
認識ワードで認識ができたので、音声だけで、左右どちらかを向いてもらいましょう。

処理の流れは、すでに皆さんお得意！の①乱数と②左・右を向く、をつかって作ってみましょう。

だいたい、こんな感じになったかと思います。パート１でも説明しましたが、作り方はいろいろとありますので、皆さんの作りやすい方法で試してみてくださいね。

①指示待ちにします。

②変数 認識ワード１を usercommand1 に設定します。ここはクリックしておきましょう。
③ 認識ワード１(usercommand1)を認識、つまり、「あっちむいてほい」「あっちむいて」、「ほいほいほい」が認識できたら開始
④ 乱数１～２を生成し、乱数の値が「１」の時は「右を向く」、そうでなければ（つまり乱数が２の時)、「左を向く」
⑤ もとの正面に向きなおす。

さ～て、いかがでしょうか？　音声による認識ワードを正しく認識して、左右どちらかを向いてくれたかと思います。たったこれだけで、できてしまいましたね。すごいです！

ＬＥＳＳＯＮ１のおさらいをしておきましょう。

① 音声による認識ワードは、沢山の種類が用意されています。
② また、新しい認識ワードを作ることができます。
③ 認識ワードはＷＥＢ上で３種類登録ができます。
　３種類：　usercomannd1，usercomannd2，usercomannd3
④ １種類ごとに、言い回しとして３つの登録ができます。
　今回は、usercommand1 に次の３種類の言い回しを登録しました
　　「あっちむいてほい」、「あっちむいて」、「ほいほいほい」
　重要：アイボさんに覚えてもらうため、通信可能な状態
　　にしておきましょうね
⑤　usercommand1 をビジュアルプログラミング上で使えるようにするために、新しい変数に登録しておく必要があります。
　今回、【認識ワード１】を新しく作り「usercommand1」を設定
　（半角で入力してくださいね）
⑥ アイボさんが認識ワードを認識したとき、「usercommand1」が通知されます。イベントのブロックを使って処理が開始するようにaibo イベントのブロックを使って定義をしましょう。

うまく動作しなかった場合は、どこか設定や認識ワードが正しく認識できないなどがあります。あせらずじっくり確認してみてください。新しい言葉が認識され、左右向きの動作してくれるときっと感動！しますよ。ぜひ頑張ってみましょう。

次回、ＬＥＳＳＯＮ２では、パート１でやりましたオーナさんとの勝負をイベントブロックを使って組み込んでみましょう。

LESSON 2

イベントをマスターしちゃいましょう

● イベントとは？

ブロックに「イベント」と「aibo のイベント」というものがありますね。LESSON１では、新しい言葉を覚えてもらい、その言葉を認識すると処理がスタートするように、こんなブロック定義を作り、実際に追加した言葉の認識を確認してみました。

これらは、イベントと呼ばれております。ちょっとだけむずかしい説明となりますが、プログラムの世界ではよく見かける言葉です。パート１の２時限目でコンピュータのプログラムについて簡単に説明をしました。コンピュータはいろいろな処理が順番に行われ、プログラム通りに処理されることで、ゲームや WEB サイトの動画を見たり音楽を再生できていますが、使うに時には、必ず人の操作が行われています。例えば、画面のタップ、キー入力、マウスクリック、時間が来たらスタートなど、普段何気なく使っていますよね。その操作が行われたら、反応して表示したり、動作したりしていますよね。

コンピュータの世界では、これらの操作を普段は待ち受けをし、操作をされたら、開始をしたり、続きを行ったりしながらたくさんのプログラムで処理をこなしています。これらを一般的イベント処理と呼ばれております。

もう一度、イベントブロックを見てみましょう。

イベントブロック

Aibo イベントブロック

「〇〇したら・・〇〇を・・・」となっていますよね。つまり、何かが起こったら何かをすること・・の開始の条件を待っているブロックとなります。

LESSON2では、このイベントを少し頑張って覚えてみたいと思います。でも、皆さんはパート１で旗マークのブロックやLESSON１の認識ワードですでに分かっているよ・・と思いますが、ちょっとだけ、こんなことをやってみたいと思います。

イベントブロックの「【スペース】キーが押された時」を見てみましょう。

キーボードの数字や文字、カーソルキーの方向キー、スペースのリストが表示されていますね。

このブロックを使うことで、「〇〇キーが押されたら・・・」の処理開始の定義ができそうです。次のブロックを作ってみてみましょう。ここで作った内容はあとで使いますので、そのまま作ってみましょう。

① 新しい変数【オーナさんが示した向き】を作ります。

② 先ほどのイベントブロックと作った変数を次のように並べてみましょう。イベントのボタンは、キーボードのカーソル矢印キーとなる　左・右を使います。

また、①で作った変数への設定する値は、今回「右」「左」を入れてみましょう。パート１の５時限目の変数の説明をしましたが、変数には数字や文字も設定ができます。今回更に漢字を設定してみましょう。

③　準備ができたら、キーボードの左右の矢印キー【←】【→】を押して、「オーナさんの示した向き」をクリックして、値が何になっているか確認してみましょう。

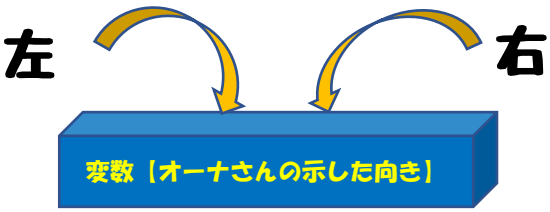
【←】左矢印キーを押したら・・・

【→】右矢印キーを押したら・・・

③　左右キーに反応して変数【オーナさんの示した向き】の値が変わったことが確認できますね。つまり・・・

キーボードの左右キーが押されたら

イベントブロックで処理開始



変数【オーナさんの示した向き】に【左】や【右】が設定されます。

④ すでにお気づきかと思いますが、左右キーはいつでも押せばイベントとして呼び出されて、処理されています。

イベント処理は実はとても重要でプログラミングでは必ず必要なものとなります。一方、いつでも処理ができてしまうことで、処理したくない場合、ガード処理なども考慮しなければならないのがイベント処理のむずかしいところで、プログラムの予期しない結果になる場合があります。

さて、イベントブロックの動作がなんとなく、わかってきたところで、ＬＥＳＳＯＮ１で作った「あっち向いてホイ」に追加して、オーナさんと勝負ができるようにしてみましょう。

ＬＥＳＳＯＮ１では、認識ワードでアイボさんに次の言葉が認識されると左右に向いてくれるプログラムを作りました。
「あっちむいてほい」、「あっちむいて」、「ほいほいほい」
ＬＥＳＳＯＮ２では、先ほど確認したイベントを組み合わせをして、オーナさんと勝負してみる処理を追加してみましょう。

これまではブロック並べながら処理の流れを理解してきましたが、複雑なプログラムやイベントが沢山あると、頭の中で整理ができなくなったり、また思ってもいない結果が発生することがあります。
今回はフローチャートと呼ばれるものをちょっとだけ作りながら、考えていきましょう。

【遊び方ルール】

①アイボさんに認識ワードを使って話しをする
②認識ワードを認識すると吠えて、左右どちらかを向く
③左右キーを使ってオーナさんの指の向きを教える
④同じ向きだったら、アイボさんの負け、違う向きだとアイボさんの勝ち
　勝ち（歌う）、負け（悲しむ）でアイボさんが勝敗を振る舞いで表現する

【事前準備】

① 認識ワード設定
「あっちむいてほい」、「あっちむいて」
「ほいほいほい」を認識ワード１で登録

② 変数を事前に作成
【アイボさんの向き】
【オーナさんの示した向き】

③ アイボさんを指示待ちにする

【イベント処理（左右キー処理）】

左右キー【左】が押されたら

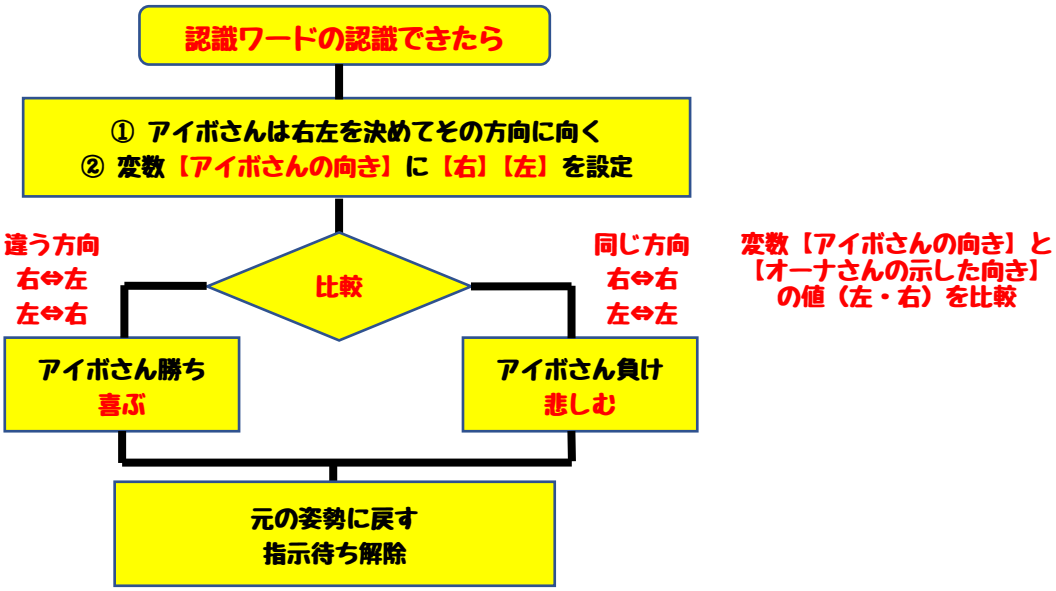
変数【オーナさんの示した向き】に
【左】を設定

【イベント処理（左右キー処理）】

左右キー【右】が押されたら

変数【オーナさんの示した向き】に
【右】を設定

【認識ワード処理】「あっちむいてほい」「あっちむいて」「ほいほいほい」を認識したとき



先ほどフローチャートに基づき、プログラムを作ってみましょう。
作り方はいろいろとあります。一例として参考にしてください。

あらかじめに WEB で設定した認識ワードを変数に登録。アイボさんを指示待ちにする

左右キーが押されたら、新しく作った変数【オーナさんの示した向き】に【左】【右】を設定

①認識ワードの言葉を認識できたらスタート
②認識ワードが認識できたので一回ほえる
③乱数１～２でアイボさんの方向を決める
　乱数１の時は、【右】を向く
　また変数【アイボさんの向き】に右を設定
　乱数２の時は、【左】を向く
　また変数【アイボさんの向き】に左を設定

④変数【アイボさんの向き】の値と
【オーナさんの示した向き】の値を比較
⑤同じ向きであれば、アイボさんの負け
　アイボさんが悲しむ ふるまいをする
⑥違う向きであれば、アイボさんの勝ち
　アイボさんが楽しむ ふるまいをする
⑦指示待ちの解除を行う

実際に実行してみましょう。どうでしょうか？

ん？　なんかおかしいことに気が付いたかと思います。
　　アイボさんが【右】に向いた
　　オーナさんが、アイボさんに向かって【右】のキーを押した
　　向かいあっているので違う方向なのに負けの扱いに・・・

そうですよね・・向かいあっているので、右と右、左と左だと違う方向を向いていることになります。本当であればアイボさんが勝ったこととなるのにプログラムでは負けにしてしまっています。

アイボさんは自分自身の向いている方向は右ではなく【左】

アイボさんは自分自身の向いている方向は左ではなく【右】

オーナさんはアイボさんと向き合っているので【左】

オーナさんはアイボさんと向き合っているので【右】

プログラムは、【左】【左】、【右】【右】を負けとしましたが、この通り、このパタンだとアイボさんは逆向きなので、アイボさんの勝ちとなります。

　　あっ、フローチャートの時に気が付いていましたか？

修正の仕方は、いろいろとありますね・・

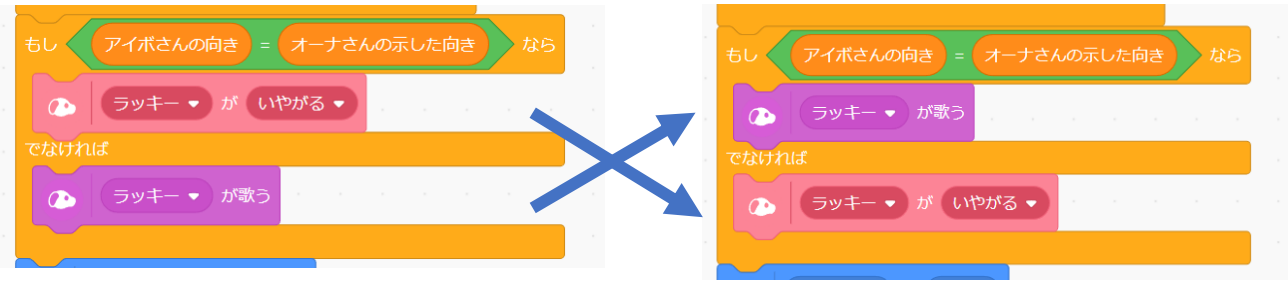
■修正ケース１
イベント処理でオーナさんの左右キーを変数【オーナさんの示した向き】の右、左の設定を逆にする

■修正ケース２
アイボさん向きを決定するとき、変数【アイボさんの向き】の右、左の設定を逆にする

■修正ケース３
アイボさんとオーナさんの向き判定処理でイコール【＝】の時の条件の制御処理の内容を変更する

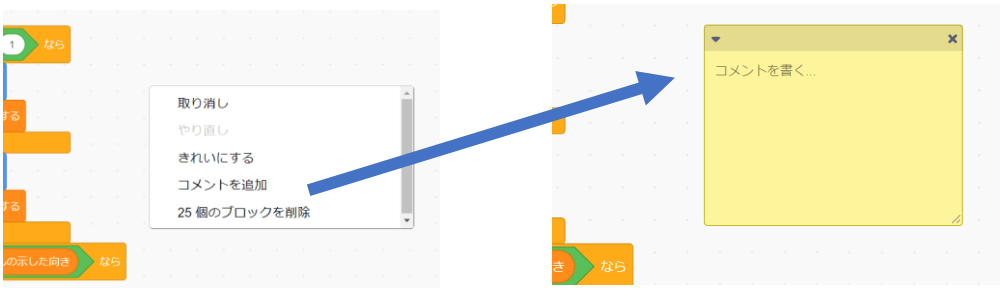
どれも可能です。ただ、なぜ？そのような条件や設定にしたかを後で見返ししたときに、頭の中で悩まなければならなくなりますので、できる限り後で見たときに理解しやすい方法に修正した方法がよいですね。修正方法は皆さんでいろいろと試してみてください。

今回は以下のように、条件判定後の振る舞いを逆にしてみました。



複雑なプログラムになると、なぜそうしたっけ？ がだんだん多くなり、そのたび、頭の中でシミュレーションしていくことが多々あります。そうなんです、必ずあります！
そこで、なぜそうしたのかを忘れないようにコメントを残しておく方法を覚えましょう。

ブロックを配置している場所で右クリックを押すと、つぎのようなメニューが表示されます。【コメントを追加】を選択するとコメントを記録する付箋紙のようなものが表示されますので、後で見てわかりやすい説明を残しておくとよいですね。



また、特定の部分についてコメントを残しておきたい時は、該当するブロック上で右クリックすると、次のようにそのブロック部分にコメント追加ができます。また、ブロック指定の場合、自動的に引き出し線が追加表示されます。どの部分の説明かをあとで見たときに把握しやすくなります。是非、活用していきましょう。

では、改めて修正およびコメントを追加したプログラム全体を見てみましょう。

ＬＥＳＳＯＮ２のおさらいをしておきましょう。

① イベントの動作について確認をしました。
② また、イベントはいつでも処理が開始されとても便利です
③ イベントはコンピュータの世界は不可欠な処理です。
④ イベントをいくつかつかって、ＬＥＳＳＯＮ１のあっち向いてホイのプログラムを拡張して、オーナさんの向きと判定の処理を追加してみました。
⑤ ブロックで作る前にフローチャートを使って全体のイメージを作って確認することをしました。
⑥ あたまで考えた処理で大丈夫だと思っていたら、想定外の結果となりました。左右判定において、向きあっているため同じ方向を示した場合、逆向きとなる考えが不足していました。
⑦ ⑥の修正する方法はいくつかありますが、後で見てわかりやすい箇所を修正しました。
⑧ ブロックだけだとわかりにくいので、便利なコメントを活用していきましょう。

ＬＥＳＳＯＮ２ いかがでしたでしょうか？
パート１から見ていただいた方は、あっち向いてホイがものすごくゲームらしいものになったか・・・違いがわかりますね。
パート１はちょっとだけ遠回りをしながら、ブロックの使い方や制御、振る舞いを学んできましたので、すでに基本的なブロックの使い方などは理解できるかと思います。今回更にイベント処理を覚えましたので、更にいろいろなプログラム世界が広がったかと思います。イベント関連もいろいろと試してみてくださいね。

LESSON 3

おもちゃと遊んでみましょう

● 画像認識

アイボさんには、たくさんのセンサー類がありますね。さらにすごいことにカメラも鼻の部分についているので、モノを認識することができるようになっていますよね。

■タッチセンサー

パート１の６時限目で【調べる】ブロックの「○○をなでられた」、また、８時限目で「○○足の肉球をおされた」を使ってゲームを作ってみました。

■音声認識

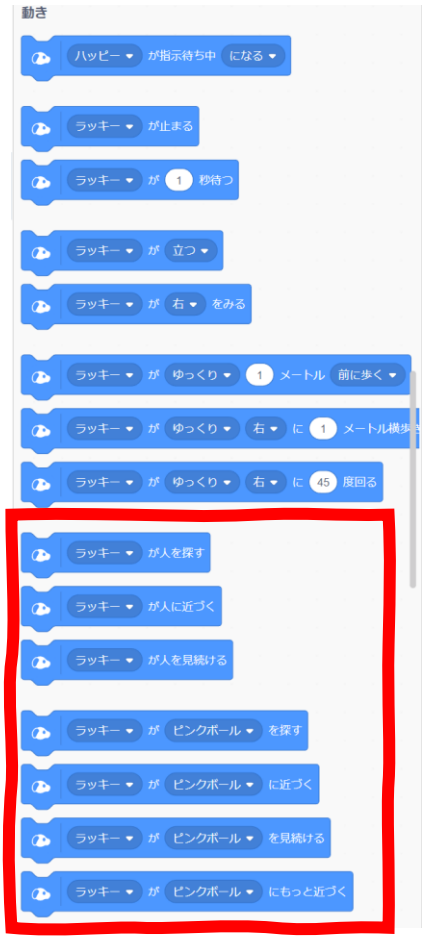
パート２のＬＥＳＳＯＮ１では音声を認識することで更にゲームを進化させてきました。

■画像認識

今回は、画像認識・確認の仕方を見てみましょう。ブロック全体を見渡すと、アイボさんが好きなおもちゃをはじめ、チャージステーション、アイボさん、ヒト、手までも確認することができるようなブロックが用意されています。

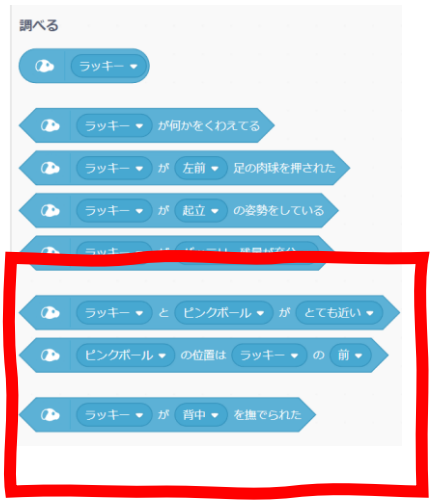
今回はカメラの画像認識でどのような感じでアイボさんは認識されるのかを確認してみたいと思います。

動きブロック



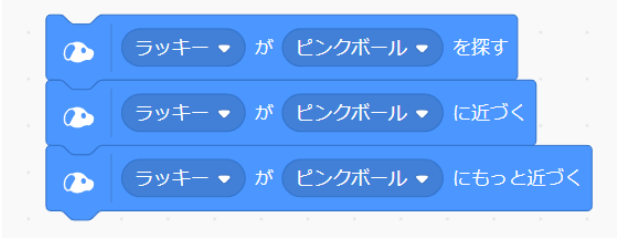
ふるまいブロック

調べるブロック



ブロックの詳細を見ると「ピンクボール」「アイボーン」「サイコロ」などを探したり、確認することができるようですね。まずは認識できるかを確認してみましょう。

アイボさんの前におもちゃを並べておき
こんな感じのブロックで確認してみましょう



最初は、可能な限り近くにおもちゃを置いてみて、どのようにおもちゃを探し、どのような感じで近づいていくのか？　また、その時にかかっている時間などをはかっておくとよいかと思います。
50 センチ程度離れていると、おもちゃを見つけて近くに寄っていくまで、３分程かかるかと思います。あせらずにじっくり見守ってあげましょう。また何度か確認してみると良いでしょう。

ピンクボール以外、アイボーン、サイコロなども試してみてください。また、ふるまいのブロックも使うと一連の動作の確認ができるかと思います。

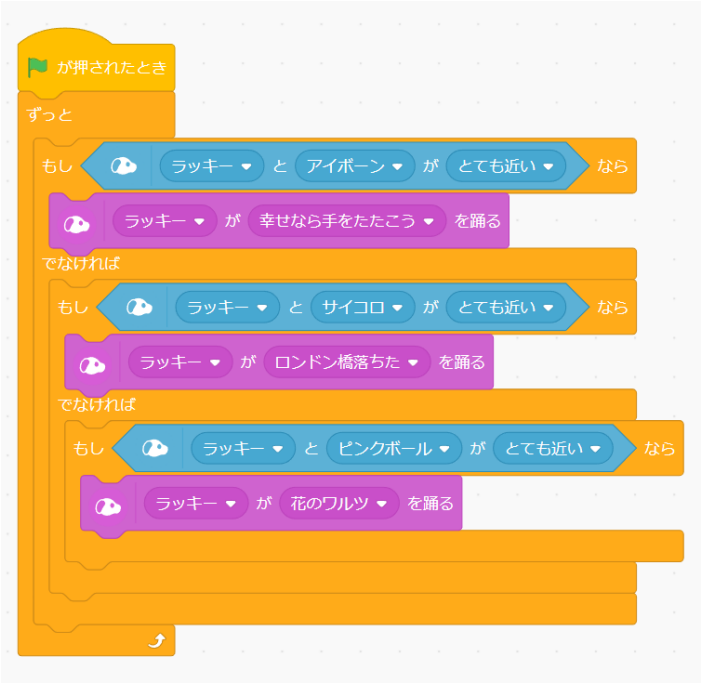
うまくいきましたか？　普段アイボさんと遊んでいると気が付かないかと思いますが、沢山の処理をしていることが分かります。

【調べる】ブロックを使ってもう少し確認してみましょう。
こんなことをやってみたいと思います。

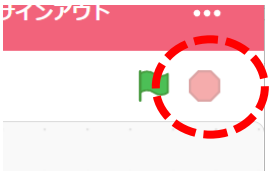
① アイボさんにおもちゃを見てもらって
② おもちゃに合わせて踊ってもらいます

【音】ブロックに新たに踊る振る舞いが追加されましたね。せっかくなので、これらのブロックを試してみたいと思います。

こんな感じにしてみました。アイボさんの目の前（鼻の前）にアイボーン、ピンクボール、サイコロを見せてあげると、それに合わせた踊りをしてくれるかと思います。このプログラムは、【ずっと】としていますので、確認が終わったら、右上の旗の横にあるストップ（赤い八角形）のボタンでプログラミングを止めてくださいね。



①このプログラムをずっと繰り返す。
②もしアイボーンが確認できたら
「幸せなら手をたたこう」を踊る。
②サイコロだったら
「ロンドン橋落ちた」を踊る。
③ピンクボールだったら
「花のワルツ」を踊る。



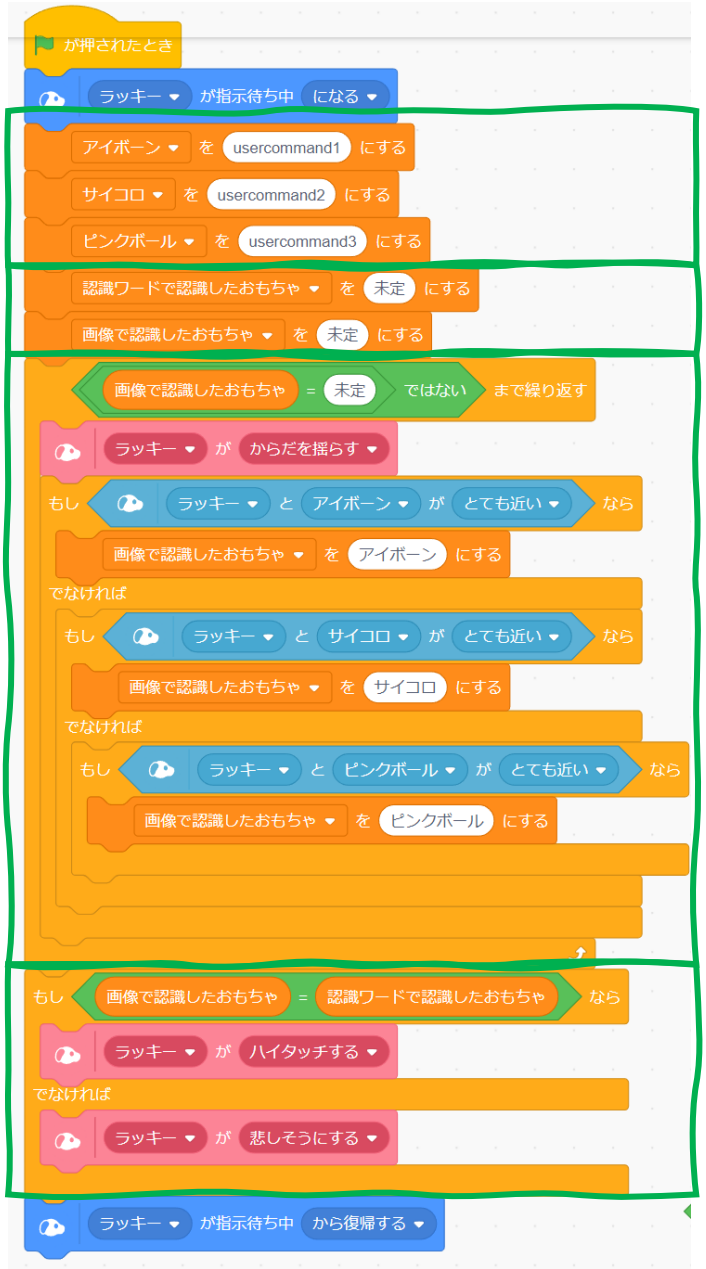
終了はストップボタンで。

前回ＬＥＳＳＯＮ２で認識ワードを使って言葉の認識もできることを確認したかと思いますので、ちょっとこんなことをやってみたいと思います。

【テーマ】アイボさんはちゃんとおもちゃを判断できるんだぞ～
「音声で認識したおもちゃ」と「見せたおもちゃ」が一致したら
アイボさんは、🎯当たっていることを表現する
もし違っている場合は、はずれていることを表現する

先にプログラム全体を見ていきましょう。いつもの説明となりますがプログラムの作り方は、みなさんそれぞれです。これが正解ではありませんので、是非、いろいろなブロックの組み合わせで作ってみましょう。

■全体の流れ　まずは、ざぁ～とながめてみましょう。



①認識ワード３つ使います。
　おもちゃごとに認識ワードを設定します。詳細は後ほど。
②新しい変数を２つ用意します。

③先ほど確認したおもちゃをカメラに近づけて見せます。おもちゃが認識できたら、おもちゃの名前を②で作った新しい変数【画像で認識したおもちゃ】に設定します。
　認識できるまで繰り返しします。
④認識できたかの判断は、「画像で認識したおもちゃ」の内容が「未定」でなくなるまでを条件としています。この値「未定」は②で設定していますね。
⑤認識ワードで認識した、おもちゃとカメラで認識したおもちゃが当たっていたら喜ぶ、違ったら悲しみます。

もう一つ、準備しておくことありますね。そうです、音声の認識ワードのイベントを準備しておく必要がありますね。

音声の認識ワードは、いつ見ても難関ですよね～。LESSON１で説明していますので、見直しして思い出してみてくださいね。

今回、認識ワードとして、アイボーン、サイコロ、ピンクボールの３種類を準備します。認識ワードの設定画面で次のような設定をしてみましょう。このワードは皆さんがアイボさんに理解してもらうための音声キーワードとなりますので、わかりやすい言葉を登録しておきましょうね。

今回、こんな感じに登録をしてみました。

【アイボーン】認識ワード
あいぼおん
あいぼうん
ほねほね
のいずれかを認識できたら
【usercommand1】を通知

【サイコロ】認識ワード
さいころ
だいす
しかく
のいずれかを認識できたら
【usercommand2】を通知

【ピンクボール】認識ワード
ぴんくぼおる
ぼうる
まんまる
のいずれかを認識できたら
【usercommand3】を通知

次に今回設定したusercommand1(アイボーン用認識ワード)、usercommand2(サイコロ認識用ワード)、usercommand3(ピンクボール用認識ワード)と連携するための変数を３つ作ります。変数名は分かりやすい名前にしましょう。

新しく作った変数が設定されているかを確認しておきましょう

次に、ＬＥＳＳＯＮ１で確認しましたイベントを作ってみましょう。こんな感じにしました。

今回は、認識ワードをなんと３種類も作ってしまったので、音声による認識とプログラムの関係について、あらためてどんな流れになるのか、おさらいしてみましょう

【アイボーン】認識ワード
あいぼおん
あいぼうん
ほねほね
のいずれかを認識できたら

【usercommand1】を
イベントとして通知

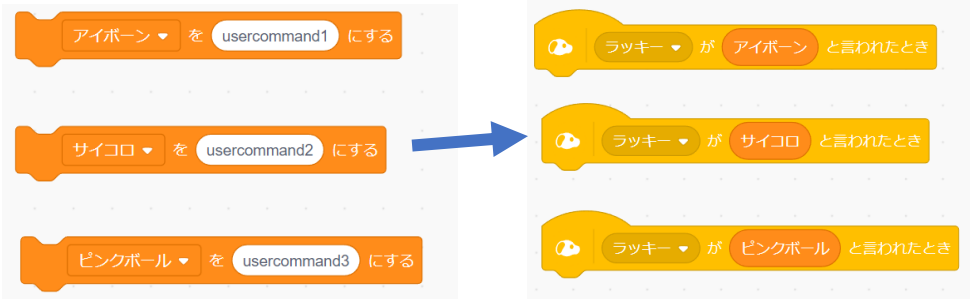
【サイコロ】認識ワード
さいころ
だいす
しかく
のいずれかを認識できたら

【usercommand2】を
イベントとして通知

【ピンクボール】認識ワード
ぴんくぼおる
ぼうる
まんまる
のいずれかを認識できたら

【usercommand3】を
イベントとして通知

イベントブロックに直接認識ワードの登録ができないため、変数【アイボーン】【サイコロ】【ピンクボール】を新しく作り、それぞれ usercommand1、usercommand2、usercommand3 と設定しておきます。次に、イベントを作成した変数を新しく定義します。



音声により、usercommand1(アイボーン)、usercommand2(サイコロ)、usercommand3(ピンクボール)のいずれが、認識ワードとして認識できたら、それぞれイベントが呼び出されるので変数【認識ワードで認識したおもちゃ】に、そのおもちゃ名を設定します。

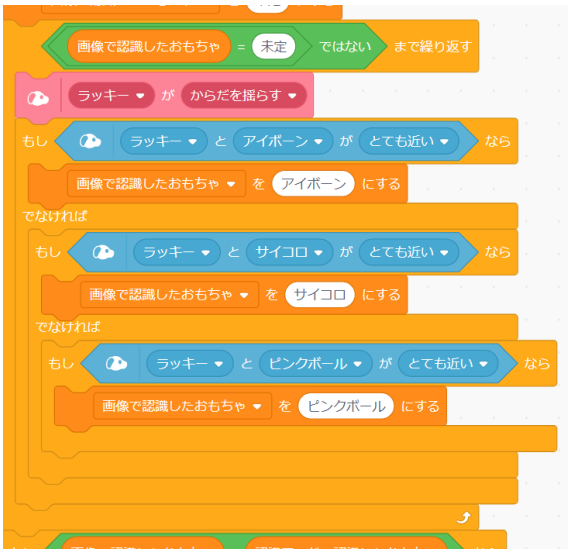
さて、もう一度全体を眺めながら、ぞれぞれの内容を確認してみましょう。

認識ワードを３種類準備します。音声ワードは WEB のデベロッパーサイトで登録しておきましょう。

【アイボーン】　は usercommand1 として定義
【サイコロ】　　は usercommand2 として定義
【ピンクボール】は usercommand3 として定義

新しい変数【認識ワードで認識したおもちゃ】と【画像で認識したおもちゃ】を作っておき、初期値として「未定」にしておきます。

プログラムの世界では、最初に何が設定されているかわからないため、初期化や初期値を入れておくことは非常に重要です。



ここはちょっと難易度が高いですね。制御ブロックを使って、「もし～なら～」をこれまでよく使ってきましたが、今回はカメラでおもちゃが認識できるまで何度も繰り返し確認が必要ため「～まで繰り返す」を使います。

この条件判定が重要です。
【～ではない】も使います。

今回、カメラでおもちゃの確認をしますが、先ほど初期化をしたところを思い出していただくと、変数【画像で認識したおもちゃ】の最初の値は「未定」としました。カメラでおもちゃを判定できると、それぞれの名前を設定します。
ここの判定としては、
変数【画像で認識したおもちゃ】＝「未定」でなくなるまで、この制御ブロックをくり返す。つまり、おもちゃが認識できたら「未定」ではなくなるので、そのときまでこの「～まで繰り返す」を実行していることとなります。　ちょっと難しいですね。

この部分は、おもちゃが認識できるまで繰り返し実行するところとなります。

最初にアイボさんが「おもちゃみせて～」といった「からだを揺らす」仕草をして、確認します。この後は、このLESSONで最初に確認した方法で認識させます。

ここでは、後で何が見つかったかを判定するために、新しく追加した変数【画像認識したおもちゃ】にカメラで認識できたおもちゃの種類の値を設定しておきます。

最後の部分は、認識ワードで認識したおもちゃとカメラで確認ができたおもちゃの値を比較して、当たりとなったか、はずれとなったかを表現します。

順番にプログラムの流れを説明してきました。いかがでしょうか？

実際に実行すると認識ワードで正しく言葉が認識できたか、カメラでおもちゃの判断ができたか、タイミングがナカナカわからないかと思います。何度も試して頂きコツが掴むまで、繰り返し実行してみましょう。

コツ

①旗のボタンを押してから
②体を揺らし始めたら声でおもちゃを何度か言ってあげます。
③カメラで認識できるまでは体を揺らしますので、おもちゃを見つづけるように、カメラの前で見せてあげます。
④おもちゃの確認ができたらハイタッチか悲しい仕草をします。

LESSON3のおさらいをしておきましょう。

① アイボさんは、触れた、聞いた、見た、をそれぞれセンサーやカメラを使って確認できます。
② ブロックを使って、それぞれを認識することができます。
③ カメラからの認識は、アイボーンやサイコロなどをきちんと認識、判断することができます。アイボーン、サイコロ、ピンクボールを見分けて、それぞれの振る舞いをしてみました。
④ 声の認識ワードとカメラからの画像認識をそれぞれちゃんと確認できているかをアイボさんらしい仕草として、声でおもちゃの名前とカメラで見せたものがあっているかを確認するプログラムを作ってみました。
⑤ カメラからの認識ができたかを判断するため、制御ブロック「～まで繰り返す」を使って、認識できるまで何度もチェックする仕組みを入れました。
⑥ 今回、認識ワードを３種類使って確認してみました。
⑦ 音声と画像の内容を比較するために、変数を作って確認したおもちゃの値を入れるようにし、最後に比較しています。
⑧ 音声、画像とも認識に時間がかかるので、何度も試してみましょう。

いかがでしたか？　実際に動かしてみると、音声や画像処理はとっても大変そうなことが分かりますね。最近はスマートフォンでカメラ映像から顔認識や映像から種別や色を簡単に判別できるようになってきました。実際にはとても難しい技術を使っていることが分かったかと思います。是非いろいろと試してみましょう。

LESSON 4

おもちゃと遊んでみましょう つづき

LESSON3 の音声と画像認識のプログラムどうでしたでしょうか？　きっとイメージ通りに、ナカナカ動作してくれなかった・・や、なんとなくできているかも・・や、正しく判断ができていないのでは？　と思ったオーナさんもいたかと思います。

思った通りにならないと、何が起こっているのだろうか・・・と考えてしまいます。プログラムをつくって、思った通りに動かないとき、いろいろな手段をつかって確認するのが一般的です。

例えば

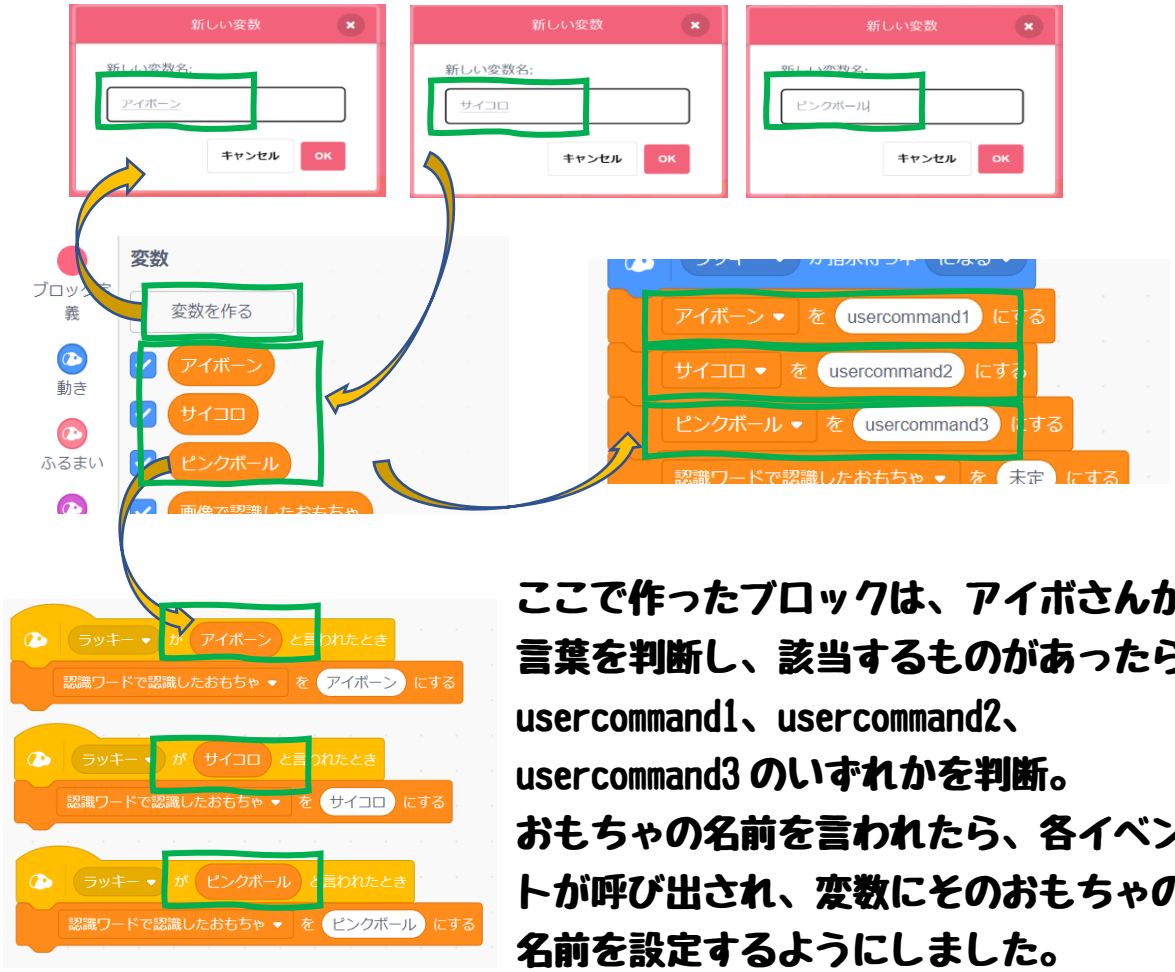
- 設定されている変数の値を確認
- 取得したデータの内容を確認
- 計算した結果や実行結果を確認
- 一時的に値を設定してみて動作を確認

これらを解決するために、一般的な確認方法に「デバッグする」や「トレースする」あるいは「ログ取り」などと呼ばれる手法で確認します。このテキストでも、何度か確認したことがある【変数】をクリックすると、現在の値が表示されることを学んできましたが、その手法も一つとなります。

プログラムを作っていくと必ず経験するので、このLESSON4では、その方法を少しだけ確認してみましょう。

■認識ワードの確認

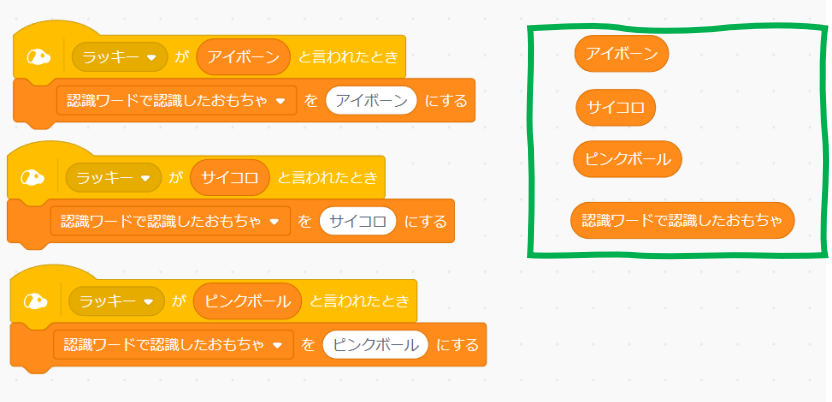
LESSON3で音声の認識ワードしたワードを保存する変数【アイボーン】【サイコロ】【ピンクボール】そして認識できた時に何が言われたかを保存する【認識ワードで認識したおもちゃ】を作って次のようなブロックを作りました。



ここで作ったブロックは、アイボさんが言葉を判断し、該当するものがあったらusercommand1、usercommand2、usercommand3 のいずれかを判断。
おもちゃの名前を言われたら、各イベントが呼び出され、変数にそのおもちゃの名前を設定するようにしました。

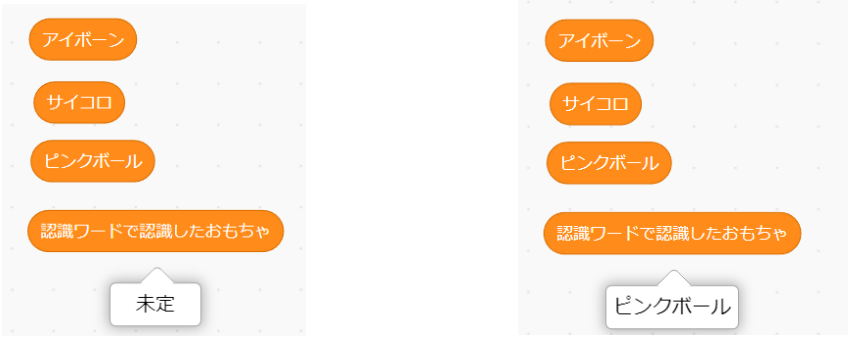
実際に実行したとき変数【認識ワードで認識したおもちゃ】がどのような値が保存されているかを確認してみましょう。

このテキストを見ていただいているオーナさんはすでに何度か実行していますが、以下のように変数を置き、何が設定されているかをクリックすることで、現在の値が確認できます。

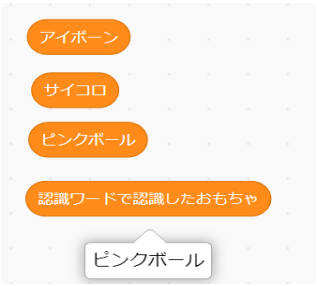


変数【アイボーン】【サイコロ】【ピンクボール】をクリックすると設定した値 usercommand1、usercommand2、usercommand3 が設定されていることが分かりますね。

次に変数【認識ワードで認識したおもちゃ】を確認すると、すでに認識ワードで認識されていれば、その値が設定されているかと思います。ＬＥＳＳＯＮ３のプログラムを実行した状況で、まだ認識ワードが認識できていなければ【未定】のままかと思います。



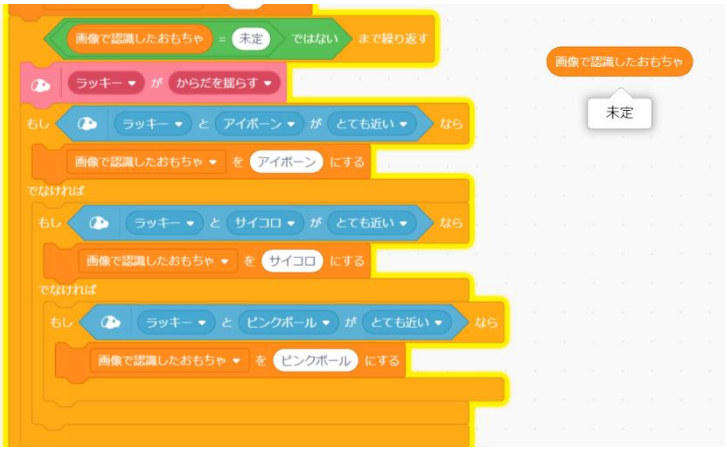
せっかくなので「アイボーン」「サイコロ」「ピンクボール」を言ってみて、この変数がそれぞれ変わっているか確認してみましょう。変わっていれば、正しく認識できていることとなります。



もし、どのキーワードを言っても値が変わらなければ、WEB登録の内容、変数を作った時の「usercommand1」などの文字が全角になっていたり空白が入っていたり、いずれも何か間違っている場所がある可能性あります。

このように実行時の途中の状況を確認することで、正しくプログラムが動作しているかを確認することができます。

■画像認識の確認



同じように、画像についても確認してみましょう。変数【画像で認識したおもちゃ】を横において実行直後にリックしてみてください。最初は、【未定】が表示されているかと思います。

「アイボーン」「さいころ」「ピンクボール」を見せてあげながら、同じように、変数の値がどのようになっているかを確認してみましょう。

どうでしょうか？　うまく変数に設定されましたでしょうか？
アイボさんの鼻先にじっくり見せてあげても「未定」のままの場合は、正しく認識できていない可能性があります。もう一度プログラムを確認してみてください。

LESSON3のプログラムは、この認識ワードで認識できた内容を変数【認識ワードで認識したおもちゃ】と画像で認識したときに保存した内容を変数【画像で認識したおもちゃ】の内容を比較して、その結果で喜んだり悲しんだりするようにしています。

このように、プログラムが正しく動作しているかを確認する方法と

して、途中の状態や値を確認することで、思った通りの動作をしているか、タイミングがあっているかなどを確認しながら、作っていくことで、より確実なプログラムの作成ができるようになります。

このような作業を先ほど説明した「デバッグ」や「バグ取り」などと呼びます。実際にはもっともっと複雑に確認する手段や開発用のツールなどがありますが、ビジュアルプログラミングでは、この方法でしか現状確認できませんが、是非、活用していきましょう。

今回のＬＥＳＳＯＮ４では、プログラミングの方法について学習しておりますので、もう一つ、新しいブロックも一緒に覚えてみましょう。

● ブロック定義をマスターしましょう

ブロックに【ブロック定義】と呼ばれているものがありますが、
【ブロックを作る】ボタン以外ブロックがありませんねぇ・・・

「ブロックを作る」を押してみると、難しい言葉がいっぱい並んでいる何やら新しいブロックを作るようなものがでてきました。

この時点で　チンプンカンプン・・・ですよね・・
きっとこの先も何度も？？が出てきそうな内容ですが、今後のＬＥＳＳＯＮで順次説明しながら使っていきたいと思いますが、つぎのようなことを考えたいと思います。

プログラムを作っていくと、同じような処理や複雑なプログラムを作っていく必要がありますが、ビジュアルプログラミングで作っていくと、膨大なブロックが並んでいくことになりますね。

こんな時に、ある同じものを繰り返し作っていくのではなく、一つ

にまとめてしまいたいときなどがあります。こんな時には、ブロック定義を作っておくことで、長～ぃ プログラミングを分割して見やすくしたり、同じ処理を共通化することができます。

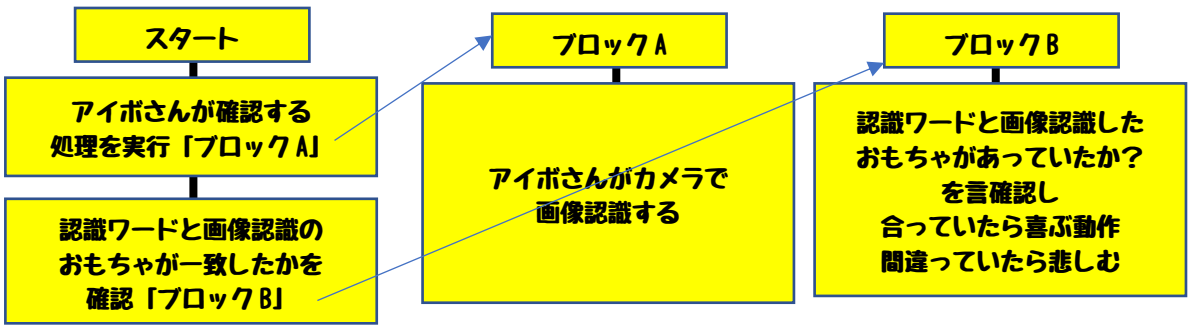
今回は、まずは大きな流れを理解するために次のような流れを作って、ブロック定義を使って長いプログラムを見やすくしたいと思います。

【事前準備】

① 認識ワードを作成
「ピンクボール」「アイボーン」「サイコロ」をそれぞれ
usercommand1, usercommand2, usercommand3 で登録

② 認識したおもちゃが何であったかを保存する
変数「認識ワードで認識したおもちゃ」
変数「画像で認識したおもちゃ」を作る

おもちゃをチェックして確認ができたら、あっているかを表現する



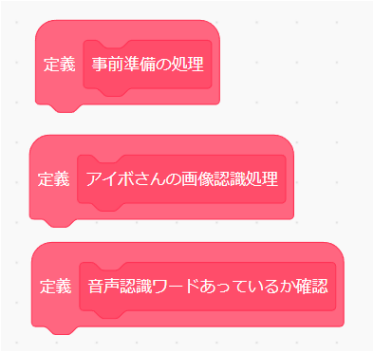
ブロック定義で次のようなブロックを作ってみましょう。
「ブロックを作る」をクリックして作成してみましょう。

【事前準備の処理】という新しいブロックを作ります。

【アイボさんの画像認識処理】という新しいブロックを作ります。

【音声認識ワードとあっているか確認】という新しいブロックを作ります。

新しいブロックが作られました。



同時にこのようなブロックが自動に画面に表示されたかと思います。

新しいブロックの定義を作ると、自動的に定義というブロックが作られます。この時点でなにがなんだか・・・・になるかと思いますので、先ほどフローチャートの

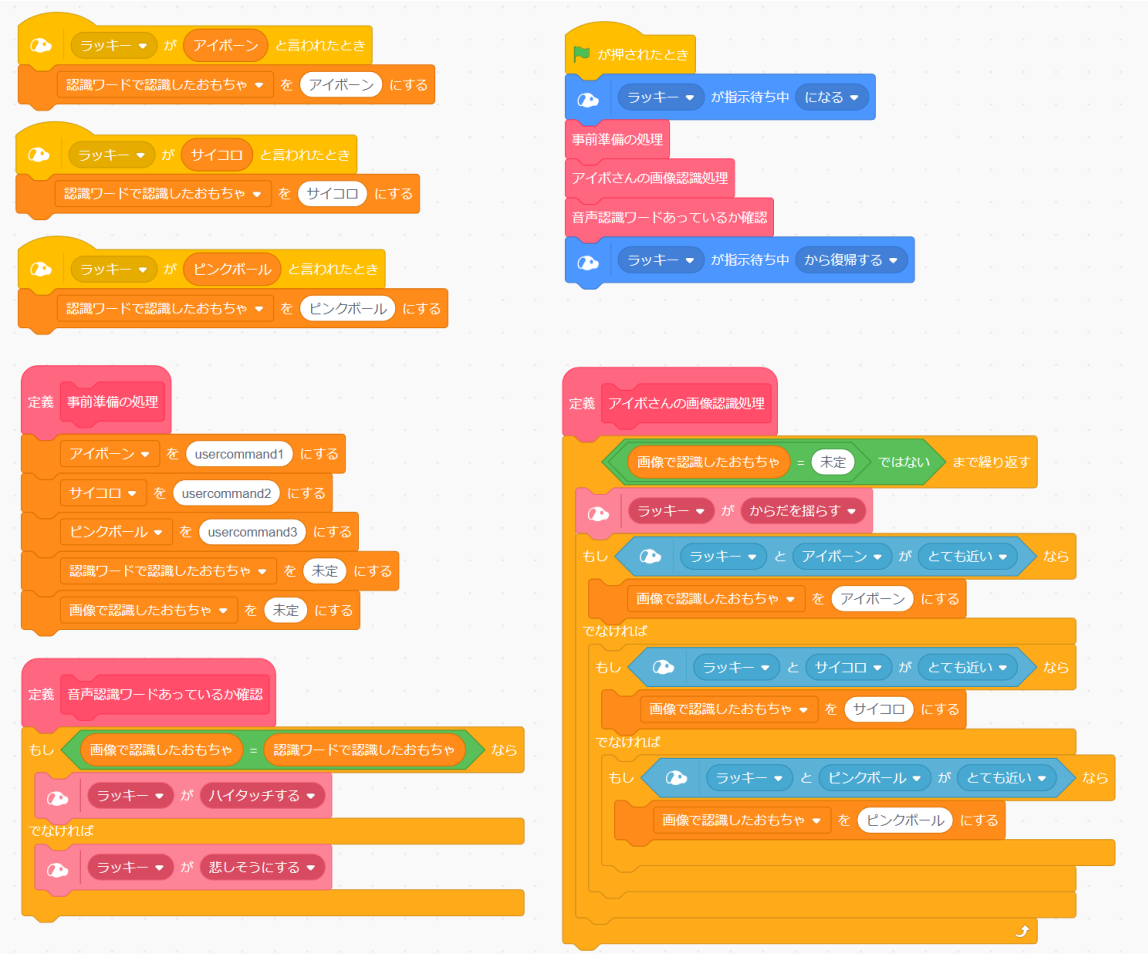
事前準備

「ブロックA」＝「アイボさんの音声認識処理」

「ブロックB」＝「音声認識ワードとあっている確認」

の定義だと思ってください。

なぜこのようなブロックを作ったのかを説明するために、ＬＥＳＳＯＮ３のプログラムを次のように変更してみました。



今までは、ブロックを積み重ねてきましたが、だいぶ雰囲気がかわってきましたね。よ～く見てみると、ＬＥＳＳＯＮ３のプログラムが分割されているようになっているのが確認できるかと思います。このような分割したり共通化するようなことを、ほかのプログラミング言語などでよく使われる言葉で、関数定義やサブルーチンなどと呼ばれています。

では、もう一度、ひとつひとつ確認してみましょう。

ここは何度も確認しているイベントブロックですね。認識ワードを認識されたときに、スタートとなるイベントブロックです。

ここに先ほど作成した新しいブロックがならんでいます。

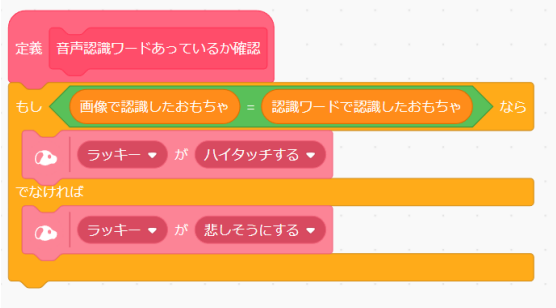
①事前準備の処理

②アイボさんの画像認識処理

③音声認識ワードあっているか確認

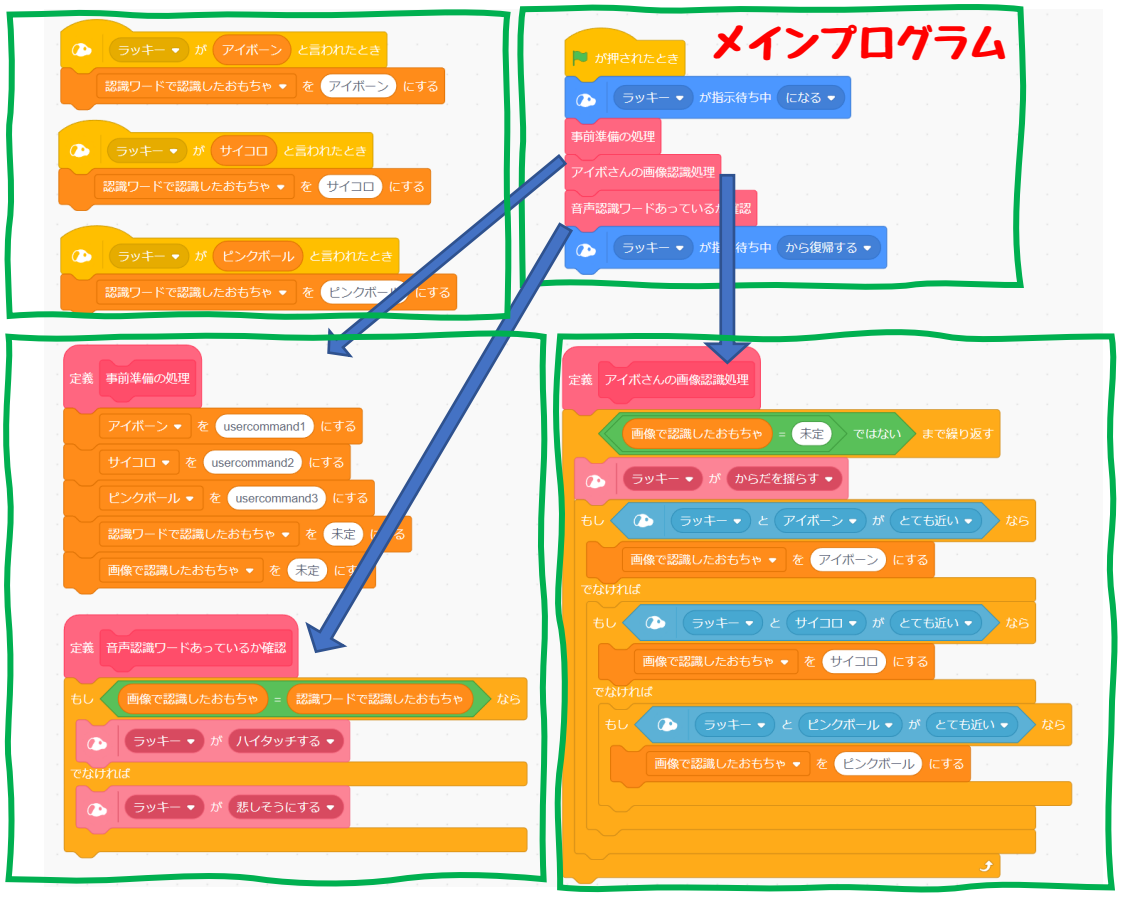
先ほどブロック定義で作成した「事前準備の処理」と準備に必要なブロックを並べています。先ほどの①が実行されるときにこの部分に制御が移り処理され、元の①の場所に戻ります。

先ほどブロック定義で作成した「アイボさんの画像認識処理」とおもちゃを見て判断するブロックを並べています。先ほどの②が実行されるときにこの部分に制御が移り処理され、元の②の場所に戻ります。



先ほどブロック定義で作成した「音声認識ワードあっているか買確認」と準備に必要なブロックを並べています。先ほどの③が実行されるときにこの部分に制御が移り処理され、元の③の場所に戻ります。

いかがでしょうか？　つまり定義されたブロックはそれぞれの同じ名前で自動に作られた【定義 OOOOOO】という先頭となるブロックが実行されるようになることで、長～ぃ　プログラムから解放され、それぞれのブロックが定義されたブロック群が処理されるように見やすくなってきたかと思います。全体を見渡すとこんな感じで動作することとなります。



LESSON4のおさらいをしておきましょう。

① プログラムを作っていくと正しく動作しているか、確認がひつようとなることがあります。とくにうまく動作できない場合、何が起こっているのか確認が必要となります。
② 確認方法はいろいろな方法があります。ビジュアルプログラミングで、変数を使っている場合は、その内容を確認することできちんと処理されているかを確認することができます。
③ これらの確認方法は一般的には「デバック」などと呼ばれています。
④ LESSON3のプログラムのプログラムを使いながら、認識ワードで認識した内容、画像で認識したときの変数の内容を確認してみました。
⑤ 長いプログラムを分割し繰り返し使うものをまとめたり、見やするための手法として【ブロック定義】を使って、LESSON3のプログラムを分割してみました。
⑥ 定義プログラムで分割することで、それぞれ纏まった単位で理解できるようになり、また、見やすくすることができました。

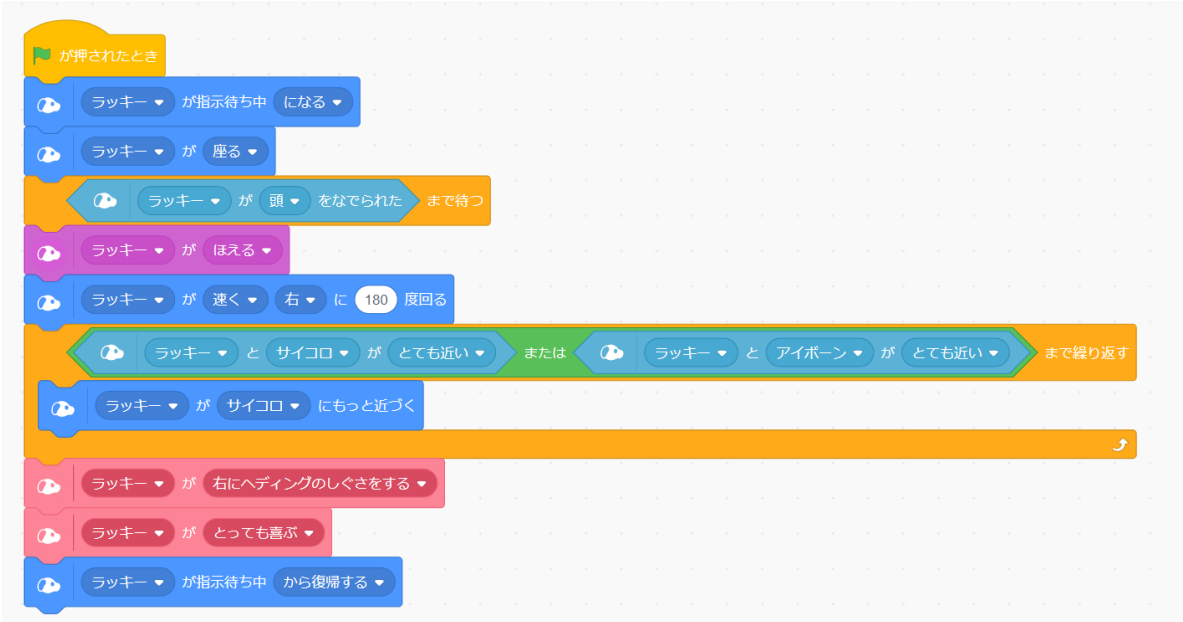
今回は複雑なプログラミング時に必ずと言っていいほど体感する、途中経過の状況を確認する方法としてデバックという手法、また、長いプログラムをわかりやすく分割する方法としてブロック定義を使ってみました。是非、いろいろと活用して見やすいプログラムを作ってみてくださいね。

おまけ

画像を処理するプログラム応用として、第１回のビジュアルプログラミングコンテストの時に作成したアイボーンとサイコロを使った、「ビーチフラッグゲーム」を載せておきます。工夫次第で簡単なプログラミングでゲームを作ることができるかと思います。
YouTube で動画もアップしていますので参考にしてください。
https://happy-lucky.fun/aibo/2021/07/31/visulprgraming007/

【遊び方】

① サイコロの上にアイボーンを載せてビーチフラックに見立てます。
② アイボさんを５０ｃｍ程度話して、反対側に向けます。
③ プログラムを実行されると、アイボさんは座って待ちます。
④ 頭を撫でてあげるとスタート
⑤ 180 度回転して、サイコロとアイボーンを探しに行きます。
⑥ ①のビーチフラッグもどきの場所に近づいたらヘディングで倒します。
⑦ お友達のアイボさんがいたら一緒にやって、二人で勝負したり、時間を測定したりしてあそぶことができます。



LESSON 5

リストブロックを学んでみましょう

LESSON4では、プログラムを作っていくときの動作を確認するための方法、そして、ブロック定義を使ってブロックをまとめて定義することで見やすいプログラムの作り方などを学びましたが、いかがでしたでしょうか？

LESSON5では、ビジュアルプログラミングで利用できるブロックの【リストブロック】について学習してみましょう。

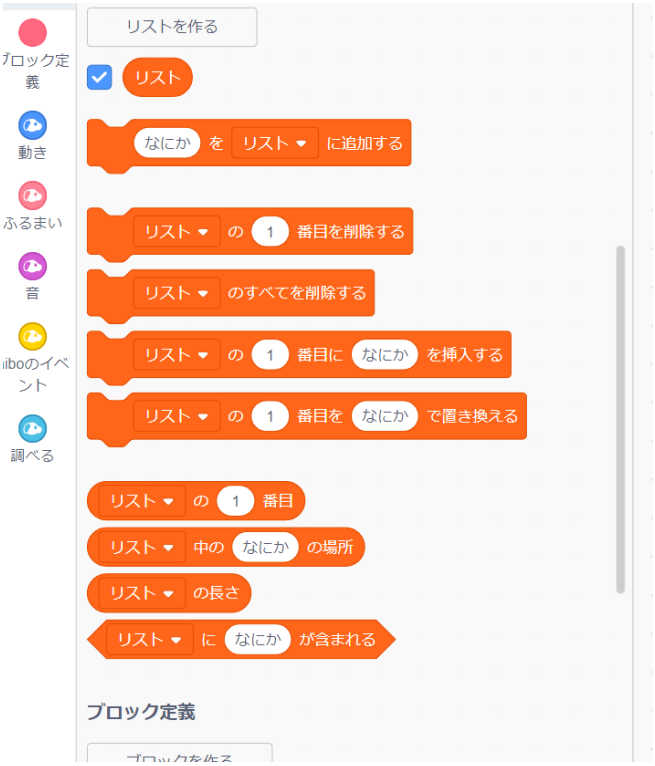
左のブロックに【リストを作る】とだけ表示された定義があります。

他に何も表示がないので、何ができるのかがよくわかりませんね。

一回【リストを作る】を押してみましょう。

こんな画面が出てくるかと思います。

分かりやすい名前を入力して
【OK】を押してみましょう。
今回は【リスト】にしました。



作成すると、何やらいっぱいむずかしそうなブロックが出てきましたね。

１つずつ説明をしていきたいと思います。

リストは使うことが少ないかもしれませんが、いくつかのデータを覚えておくには非常に便利かと思います。是非、一度トライしてみましょう。

ビジュアルプログラミングのリストブロックを見る前に、リストや配列と言った言葉を少しだけ学んでみましょう。

プログラムを作ると、必ずと言っていいほどデータなどを保存します。数字、文字、映像、画像、音声など、スマートフォンやパソコンを使っているとありとあらゆる情報が保存できます。これらの情報が保存されていることで続きの操作や、過去の情報を見たり、聞いたり、使ったりすることができます。もし、情報の保存ができなかったら、最初からやり直しが必要となりますよね。

この「まなんじゃぉ～パート１の５時限目」で、変数の説明をしてきました。情報を一時的に保存しておく方法として【変数】ブロックを使って、値を保存していくことを学びました。変数は新たに作ることも理解できたかと思います。

プログラムの中では、さまざまなデータを保存し使うことで、いろいろな処理ができるようになります。ビジュアルプログラミングでは、情報そのものをハードディスクのような外部メモリに保存はできませんが、実行中に記憶（保存）することができます。

変数は１つのデータを保存することしかできませんよね。
沢山の情報を保存するには変数を沢山作れば・・・

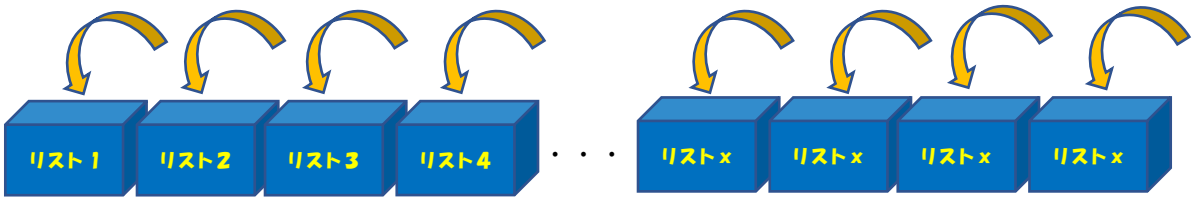
例えば、新規に作った【変数１】といった変数に数字を保存。
別の値を別に保存したい場合は【変数２】を作って保存。

では、保存しておきたい値が１０個や１００個あった場合は・・・
変数を１～１０個分、１００個分を作らないと管理できないですね。これでは、とても大変です。

そこでこのＬＥＳＳＯＮでは、【リスト】ブロックを使ってこの問題の解決方法を学んでいきましょう。

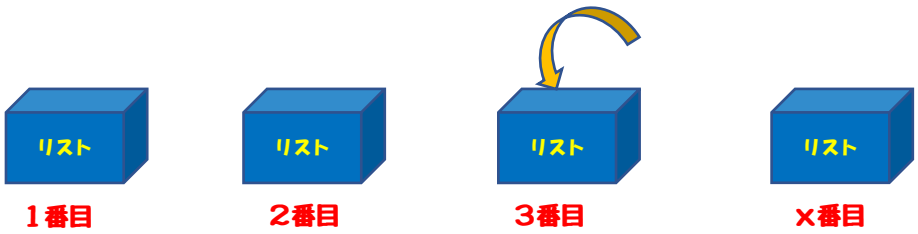
プログラミングでは、リストや配列と言った言葉がよく出てきます。それぞれ使い方や作法が異なっていますが、基本的な考え方は同じなので、是非覚えておきましょう。

■リストとは
リスト・・よく聞く言葉ですね。例えば、名前リスト、データリスト、テキストリストなど、つまりデータや情報の集合体と思ってください。



それぞれに値を保存することができます。

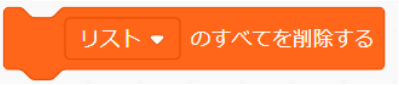
変数ブロックの場合は、【変数１】とか【変数２】とかの名前を決めて、ひとつひとつ定義をしていく必要がありましたが、リストブロックの場合は、名前は１つ、その名前で決めたリストの何番目か場所を指定して、値を保存することができます。



例えば、リストの３番目に値を入れたい時には、リストの３番目を指定して、保存することができます。

このようなリスト形式のような構造を、プログラミングの用語でよく、【配列構造】や【リスト構造】などと呼ばれておりますので、おぼえておくと良いかと思います。

では、先ほど作成したリストブロックを見てみましょう。

ブロック	説明
	リストに追加する場合に使います。たとえばリスト２番目までに値が保存されている場合、３番目に値が追加されます。 リスト１番目　値を保存済 リスト２番目　値を保存済 リスト３番目　←　ここに追加
	リストのｘ番目を削除するときに使います。１番目を指定した場合、削除すると２番目が１番目となります。 リスト１番目　←　ここを削除 リスト２番目　←　削除後、ここが１番目 リスト３番目　←　削除後、ここが２番目
	リストの内容をすべて削除します。 リスト１番目　←　削除 リスト２番目　←　削除 リスト３番目　←　削除
	リストのｘ番目に値を挿入するときに使います。１番目を指定した場合は、 リスト１番目　←　ここに挿入 リスト２番目　←　挿入後１番目が２番目に リスト３番目　←　挿入後２番目が３番目になります。
	リストのｘ番目の値を置き換えるとき使います。１番目を指定した場合は、 リスト１番目　←　ここを置き換える リスト２番目　←　２番目はそのまま リスト３番目　←　３番目はそのまま
	このブロックは、演算や制御の時に、リスト内に保存されている内容を確認するときに使います。 ① リストのｘ番目の値を確認 ② リストに保存されている値の場所を確認 ③ リストで保存されている値が何個あるか確認 ④ リストで保存されているが含まれているか確認

■リストの動作を実際に確認してみましょう

① 先ほど作ったリストの現在の内容を確認してみましょう。
長丸で【リスト】をクリックしてみましょう。変な表示が出てきましたが、何も値がないので、このような表示になります。

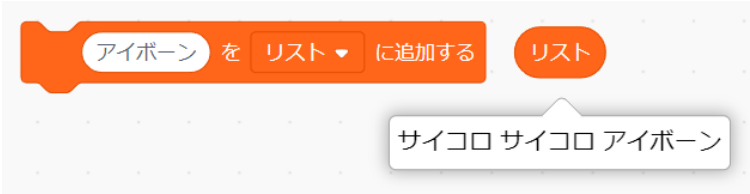
② では、１つだけ値を追加してみましょう。追加するものは、数字、文字などを設定してみましょう。
ここでは、例：サイコロ　と設定して、ブロックをクリックして【リスト】の内容を見てみましょう

追加したことで、リストに情報が保存されたことが確認できました。

③ ブロックをもう一度クリック（何度かやってみてもよいです）
すると、サイコロが追加されたことが分かります。

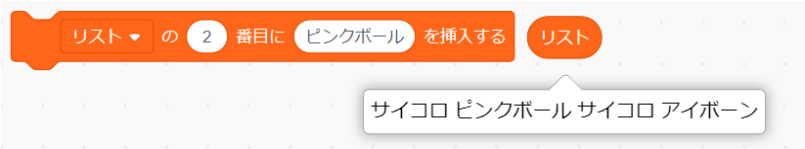
ここだとわかりにくいですが、空白が区切りとなっています。
サイコロ（１番目の値）【空白】サイコロ（２番目の値）

④　値を変えて追加してみましょう。今度はアイボーンを設定

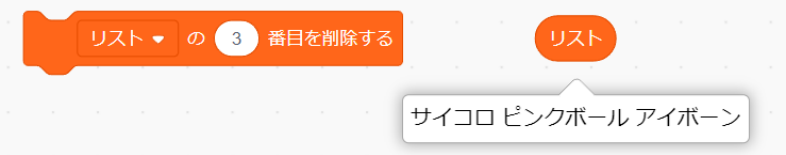


サイコロ（１番目の値）【空白】サイコロ（２番目の値）そして
アイボーン（３番めの値）となりました。

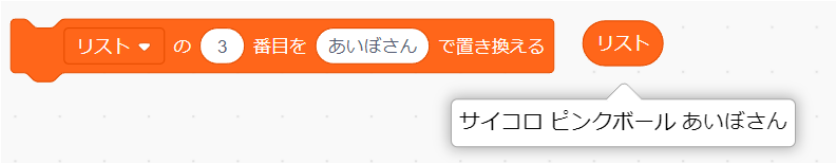
⑤　更にリストの途中に値を追加してみましょう。２番目にピンク
ボールを追加（挿入）してみましょう。



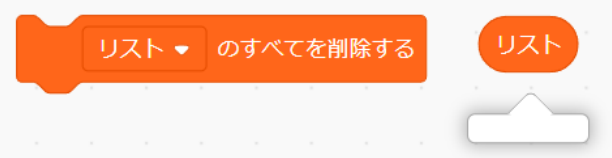
⑥　今度は３番目のサイコロを削除してみましょう。



⑦　今度は、３番目のアイボーンを別の文字「あいぼさん」に変更
してみましょう。



⑧　設定した値をすべて削除してみましょう。



さて、リストブロックの使い方はいかがでしょうか？

このように、これまで変数を作って値を保存してきたやり方と比べるとかなりいろいろと値を保存することができるかと思います。

あとは、保存した内容を使って、いろいろな条件や制御をすることで、これまでできなかったことが実現できるかと思います。

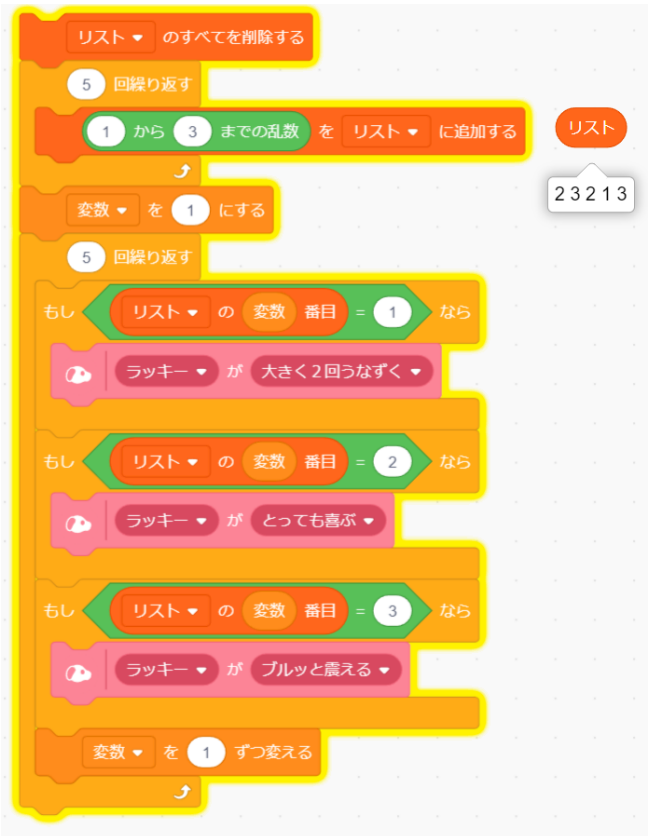
・・・・と言ってもアイボさんのプログラムで何に使えるのかがピンとこない・・かもしれませんね。使い方はいろいろとあるかと思いますが、例えば、

① 肉球の押された順番を覚えてく
② 確認したおもちゃを覚えておく
③ 言われた言葉の順番を覚えておく
④ これらの内容の順番を入れ替えたり、逆に確認する
⑤ リストを2つ作り、それぞれの値を比較し勝敗を決める

など、ゲームの判定などに使うことができるかと思います。

例えば、次のようなブロックを作ってみましょう。
乱数を１０回作って、リストに保存します。リストをクリックすると作られた乱数が、１０個分並んでいるかと思います。

こんなことができるかと思います。５回１～３までの乱数をリストに保存、リストを１つずつ確認して１の時にはOO、２の時にはOO、３の時にはOOをするなどで、実行する毎に異なる動作をしてもらいアイボさんらしさの動作をしてもらうことができます。



① リストの内容をすべて削除
② １～３の乱数の作成を５回
　繰り返し、リストに保存
　この例だと、2, 3, 2, 1, 3
③ 次の内容を５回繰り返し
　保存したリストを１つずつ
　読み出して、
　１の時、大きく２回うなずく
　２の時、とっても喜ぶ
　３の時、ブルッと震える

【変数】はリストのx番目を１～５まで繰り返すために１ずつ追加するようにしています。

LESSON5のおさらいをしておきましょう。

① プログラミングにおいては、沢山のデータ、情報を扱います。これらの情報は保存することで、いつでも確認をしたり、続きを行うことができるようになります。
② ビジュアルプログラミングで扱うデータは外部のハードディスクなどには保存はできませんが、実行時には、変数を使って値を保存することができます。
③ 変数ブロックは、１つの値しか保存できません。複数の値を保存したい時には、変数を沢山作る必要がありますが、膨大な値を保存したい時には、効率が良くないです。
④ ビジュアルプログラミングでは、リストと呼ばれているブロックが用意されています。
⑤ リストは１つの名前の定義で、あとは何番目かを指定すればその場所に保存や確認することができます。
⑥ リストブロックの操作は、リストに追加、Ｘ番目に挿入、Ｘ番目を入れ替え、Ｘ番目を削除、全部削除などができます。
⑦ 使う用途としては、ゲームなどで覚えておきたいリスト作成などです。

今回はリストブロックの使い方について学びました。やりたいことをいろいろと考えていくと多数の値を使ったプログラムを作りたいと思うことがあります。そんな時には、リストブロックを活用することで、非常にいろいろなことができるようになるかと思います。ちょっと複雑ですが、是非試してみてください。

LESSON 6

リストのちょっとしたテクニック

LESSON5では、複数の値を保存する方法としてリストと呼ばれているブロックをつかったプログラム方法を学びました。沢山の値を保存したいとき、たくさんの変数のブロックを作るより、とっても便利だと思います。

リストを使う時のちょっとだけハイテクニック?をご紹介します。LESSON5で複数のデータを追加したときにこんな確認をしました。

サイコロを2回追加したときの確認

このように「~に追加する」とすることで、最後に追加されるようになります。では、バラバラな文字を入れてみましょう。

次のような操作をしてみましょう。

リストを新たに作成します。今回のリスト名は「もの」にしました（自由に決めてください）

次にこんなブロックを作ってみましょう。

① 「もの」を削除
② 「ば」「な」「な」を追加

操作する前に最初の値を確認しましょう。実行する前なので、空白の状態です。

次に実行してみると

LESSON5で確認した通り追加されていますが、LESSON5で確認したときのような空白がないですね。

では、最初の「ば」を「ばば」に変更して同じように実行すると

「ばば」「な」「な」の間に空白があります。不思議ですね・・・

このように、登録する文字がすべて 1 文字で構成される場合、つながった値として管理することができるようです。

先ほどの「ばなな」が「ばなな」と認識できているかを確認してみましょう。こんな感じで確認することができます。

値がない時に、演算に「もの」と「ばなな」を比較するようにします。

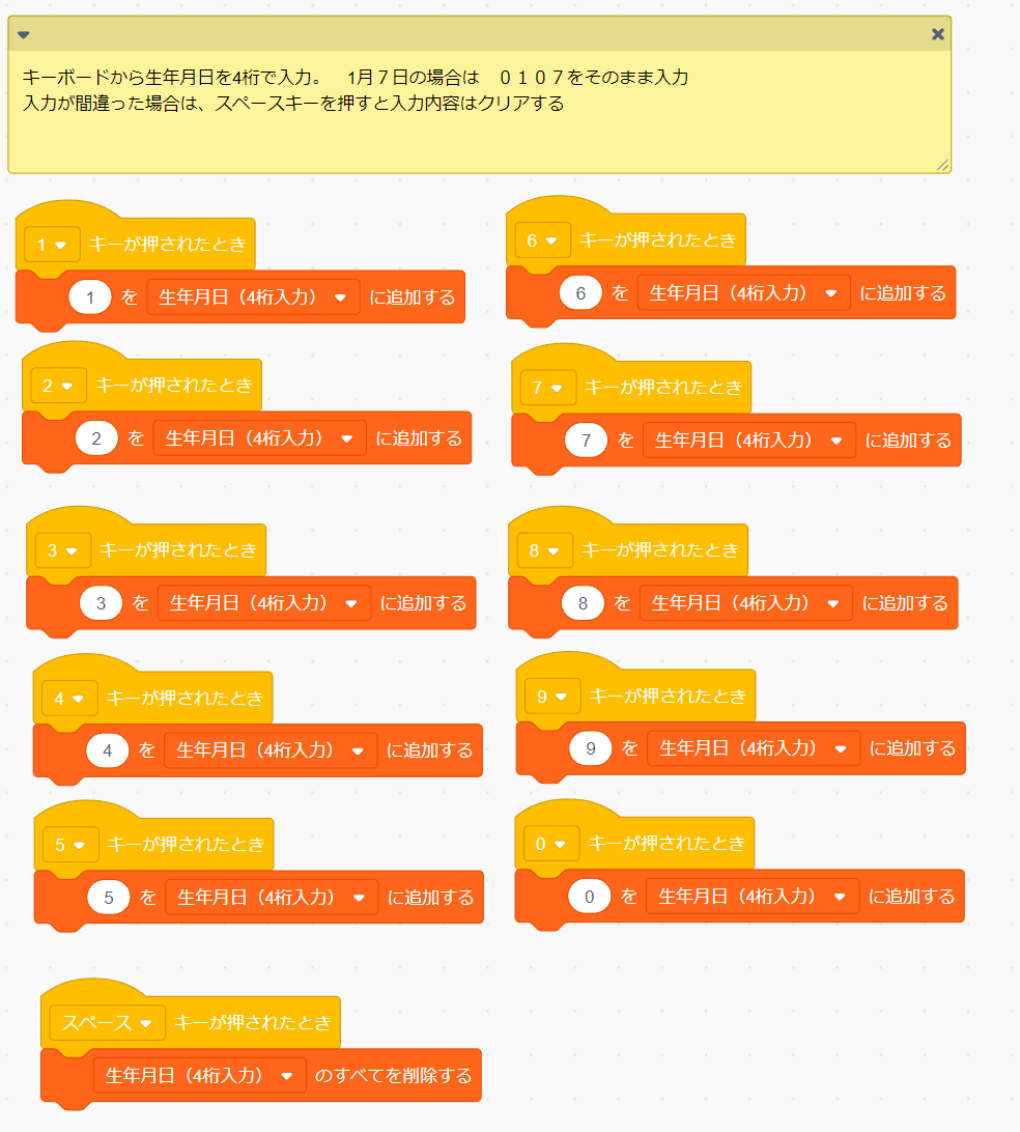
演算をクリックすると
「false(偽)」が表示。
つまり、「もの」の値「空白」と「ばなな」はイコールではないことを示しています。

では、実行してみましょう。

「もの」の値が「ばなな」となりました。

演算をクリックすると
「true(真)」が表示。
つまり、「もの」の値「ばなな」と文字「ばなな」は一致したことを示します。

これは数字でも同じように処理ができます。例えば年月日を入力したいと思ったとき、こんなような処理を行うことで、4桁の入力処理ができます。この値はそのまま数字で処理ができますので、
例えば、12 月 24 日だと、1224 とキーボードから入力してもらい、1224 と比較して一致していたら、クリスマスイブで喜ぶなどの振る舞いの処理ができるようになるかと思います。



注意点として、数字は半角にしたほうが良いですね。
結構、便利なので参考にしていただければ。

リスト応用例として、ビジュアルプログラミングコンテストでアップしたaibo星座占いのサンプルプログラムを載せておきます。

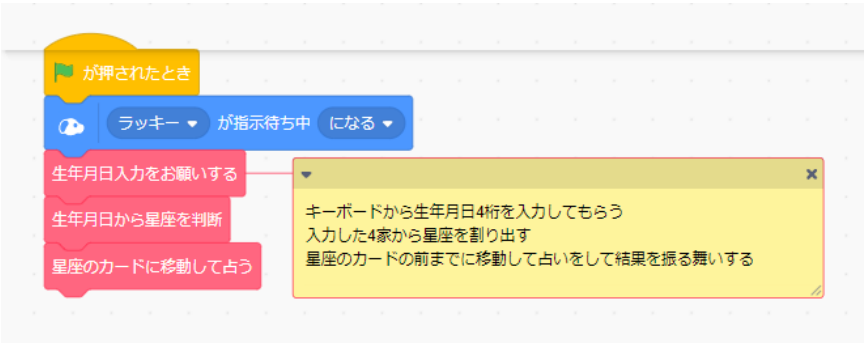
こちらからダウンロードができますので、ダウンロードして改良していただければよいかと思います。また動画も一緒にアップしているので、参考にしていただけば。
https://happy-lucky.fun/aibo/2021/08/22/visualprograming017/

生年月日の４桁入力が完了するまで、ゆらゆらしながら待っています。

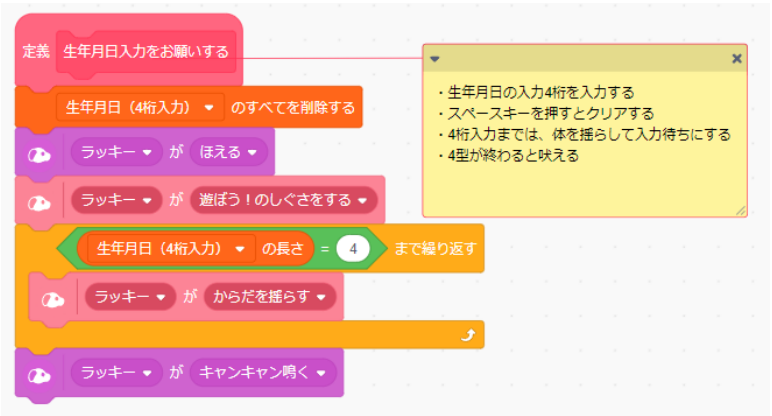
４桁の入力が完了したら、どの星座かを４桁の月日で比較して、その星座の方向に向きを変えます。良いと判断したら喜びます。

良くないと判断したら寝てしまいます。

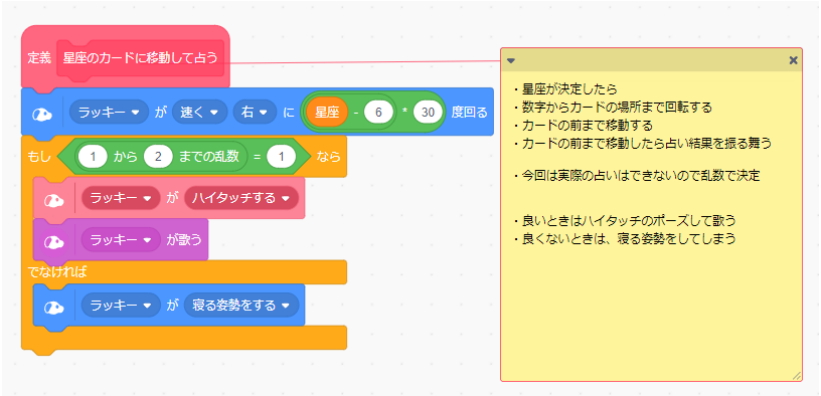
プログラムを掲載しますので、是非参考にしてください。



全体の処理となります。LESSON4で学んだ、ブロック定義を使って、「生年月日入力をお願いする」「生年月日から星座を判断」「星座のカードに移動して占う」のブロックを定義して、それぞれ順番に実行します。



4桁の入力が完了するまで、からだを揺らして待ってもらうようにします。4桁の入力が完了したら、この処理は終わります。



星座の方向に回転し、乱数で占いを行います。

生年月日から星座を判断します。今回のＬＥＳＳＯＮでリストのテクニックとして説明の１文字ずつ追加するとそのまま比較できることを説明しました。その応用です。一部ピックアップして確認してみましょう。

みずがめ座は、１月２０日～２月１９日です。例えば入力した４桁の値が、１月２０日であれば「0120」となるので、上記式の

0119＜生年月日　かつ　生年月日＜0219 が成立します。

このプログラムの星座の番号は、回転するときの方向を示していますので、処理上の都合だと思ってください。

やぎ座だけは、ちょっと工夫がいります。

やぎ座は、１２月２２日～１月１９日までなので、比較としては、１２月中なのか１月中なのか比較を分割しています。

1221＜生年月日　かつ　生年月日＜0120　は成立できませんね。

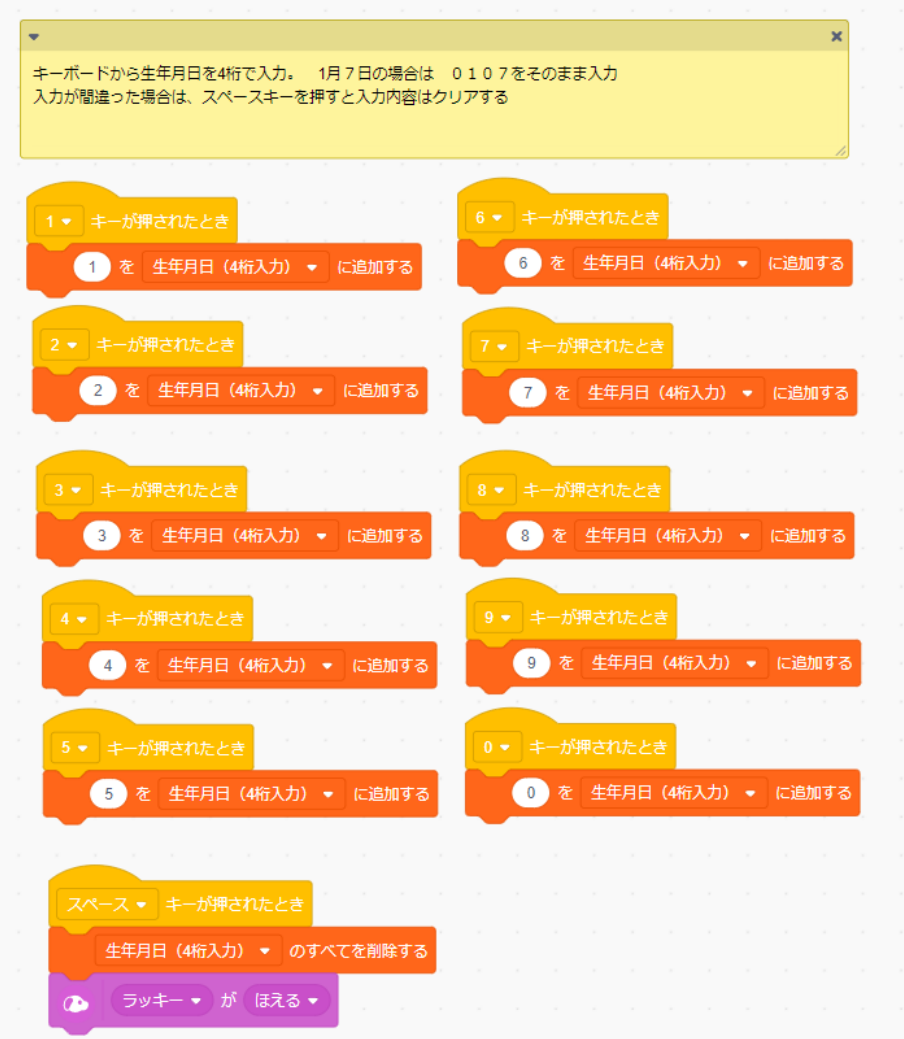
このため

1221＜生年月日　かつ　生年月日＜1232　または

0100＜生年月日　かつ　生年月日＜0120 のどちらか

で比較しています。ちょっと複雑になってしまいましたね。

さて、数字入力は、先ほど説明した内容となりますので、ここは理解できたかと思います。



このようにキーボードから入力した１文字を追加してく方法により、比較しやすい文字として扱うことができるようになります。

ちょっと複雑なプログラムとなってしまいましたが、じっくり見ていただくと、いままで学んできたものの応用となっています。是非、ダウンロードして確認してみてください。

LESSON6のおさらいをしておきましょう。

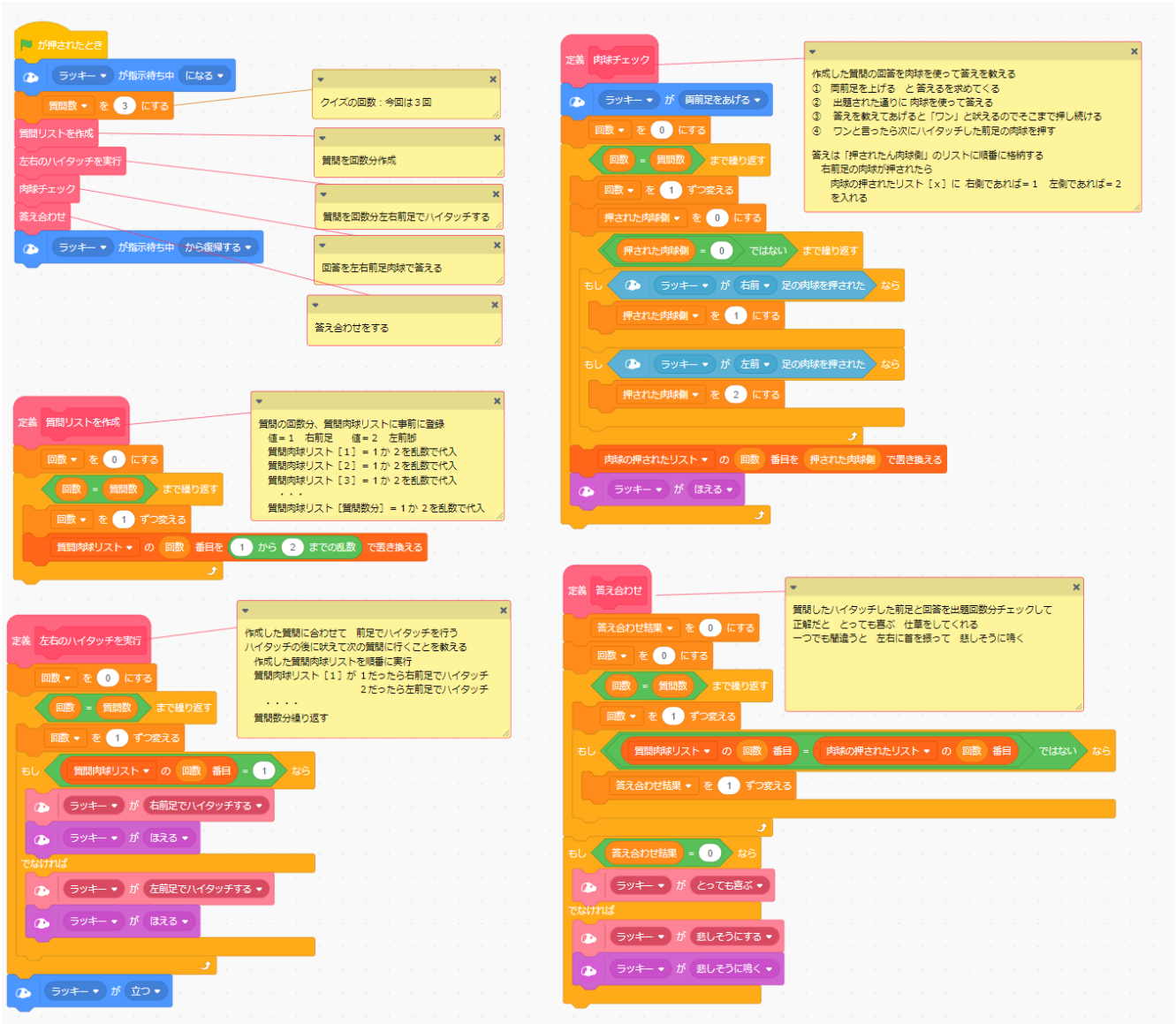
① LESSON5のテクニックとして、１文字ずつ追加することで、１ワードとして処理をすることができるようになります。
② 数字入力の際にイベントをうまく使うことで、キーボードからの入力した文字列の処理ができるようになります。
③ 今回は生年月日を入れてもらい星座占いの例で、文字入力から、演算処理の例を見てみました。

最初から作るとなると大変なので、是非サンプルプログラムをダウンロードしてみて、実行してみてください。

リストは使い方によっては非常に便利なブロックとなります。
一方、沢山の処理が必要となりますので複雑になってしまいますが、コツを覚えると色々なプログラムを作ることができますので、是非チャレンジしてみてはいかがでしょうか？

次回、これまで学んできた定義、リストブロックを使って、アイボさんが右前足、左前足をランダムに上げた順番通りにオーナさんが肉球で確認するゲームを使って、実際にリストを使ったプログラムでもう少し学んでみたいと思います。

こちらのサンプルプログラムをアップしてありますので、事前に使ってみてください。次回のLESSONで解説していきましょう。
https://happy-lucky.fun/aibo/2021/07/29/visulprgraming002/


が押されたとき
ラッキー が指示待ち中 になる
質問数 を 3 にする
質問リストを作成
左右のハイタッチを実行
肉球チェック
答え合わせ
ラッキー が指示待ち中 から復帰する
クイズの回数：今回は３回
質問を回数分作成
質問を回数分左右前足でハイタッチする
回答を左右前足肉球で答える
答え合わせをする
定義 質問リストを作成
回数 を 0 にする
回数 = 質問数 まで繰り返す
回数 を 1 ずつ変える
質問肉球リスト の 回数 番目を 1 から 2 までの乱数 で置き換える
質問の回数分、質問肉球リストに事前に登録
値＝１　右前足　　値＝２　左前脚
質問肉球リスト［１］＝１か２を乱数で代入
質問肉球リスト［２］＝１か２を乱数で代入
質問肉球リスト［３］＝１か２を乱数で代入
・・・
質問肉球リスト［質問数分］＝１か２を乱数で代入
定義 左右のハイタッチを実行
回数 を 0 にする
回数 = 質問数 まで繰り返す
回数 を 1 ずつ変える
もし 質問肉球リスト の 回数 番目 = 1 なら
ラッキー が 右前足でハイタッチする
ラッキー が ほえる
でなければ
ラッキー が 左前足でハイタッチする
ラッキー が ほえる
ラッキー が 立つ
作成した質問に合わせて　前足でハイタッチを行う
ハイタッチの後に吠えて次の質問に行くことを教える
作成した質問肉球リストを順番に実行
質問肉球リスト［１］が１だったら右前足でハイタッチ
２だったら左前足でハイタッチ
・・・・
質問数分繰り返す
定義 肉球チェック
ラッキー が 両前足をあげる
回数 を 0 にする
回数 = 質問数 まで繰り返す
回数 を 1 ずつ変える
押された肉球側 を 0 にする
押された肉球側 = 0 ではない まで繰り返す
もし ラッキー が 右前 足の肉球を押された なら
押された肉球側 を 1 にする
もし ラッキー が 左前 足の肉球を押された なら
押された肉球側 を 2 にする
肉球の押されたリスト の 回数 番目を 押された肉球側 で置き換える
ラッキー が ほえる
作成した質問の回答を肉球を使って答えを教える
①　両前足を上げる　と答えるを求めてくる
②　出題された通りに 肉球を使って答える
③　答えを教えてあげると「ワン」と吠えるのでそこまで押し続ける
④　ワンと言ったら次にハイタッチした前足の肉球を押す
答えは「押されたん肉球側」のリストに順番に格納する
右前足の肉球が押されたら
肉球の押されたリスト［ｘ］に 右側であれば＝１　左側であれば＝２
を入れる
定義 答え合わせ
答え合わせ結果 を 0 にする
回数 を 0 にする
回数 = 質問数 まで繰り返す
回数 を 1 ずつ変える
もし 質問肉球リスト の 回数 番目 = 肉球の押されたリスト の 回数 番目 ではない なら
答え合わせ結果 を 1 ずつ変える
もし 答え合わせ結果 = 0 なら
ラッキー が とっても喜ぶ
でなければ
ラッキー が 悲しそうにする
ラッキー が 悲しそうに鳴く
質問したハイタッチした前足と回答を出題回数分チェックして
正解だと　とっても喜ぶ　仕草をしてくれる
一つでも間違うと　左右に首を振って　悲しそうに鳴く

LESSON 7

ゲーム作りに挑戦

LESSON6では、リストを使ったちょっとしたテクニックを学んでみました。リストを使うといくつかのデータを記録することができますね。このようにデータを記録や保存できるようにすると、ゲームなどへの応用ができそうです。例えば、１回戦目は・・・、２回戦目は・・・など繰り返す毎に点数や値を保存しておくことで、合計点で勝敗を決めたり、回ごとに勝敗を決めるようなプログラムができそうです。

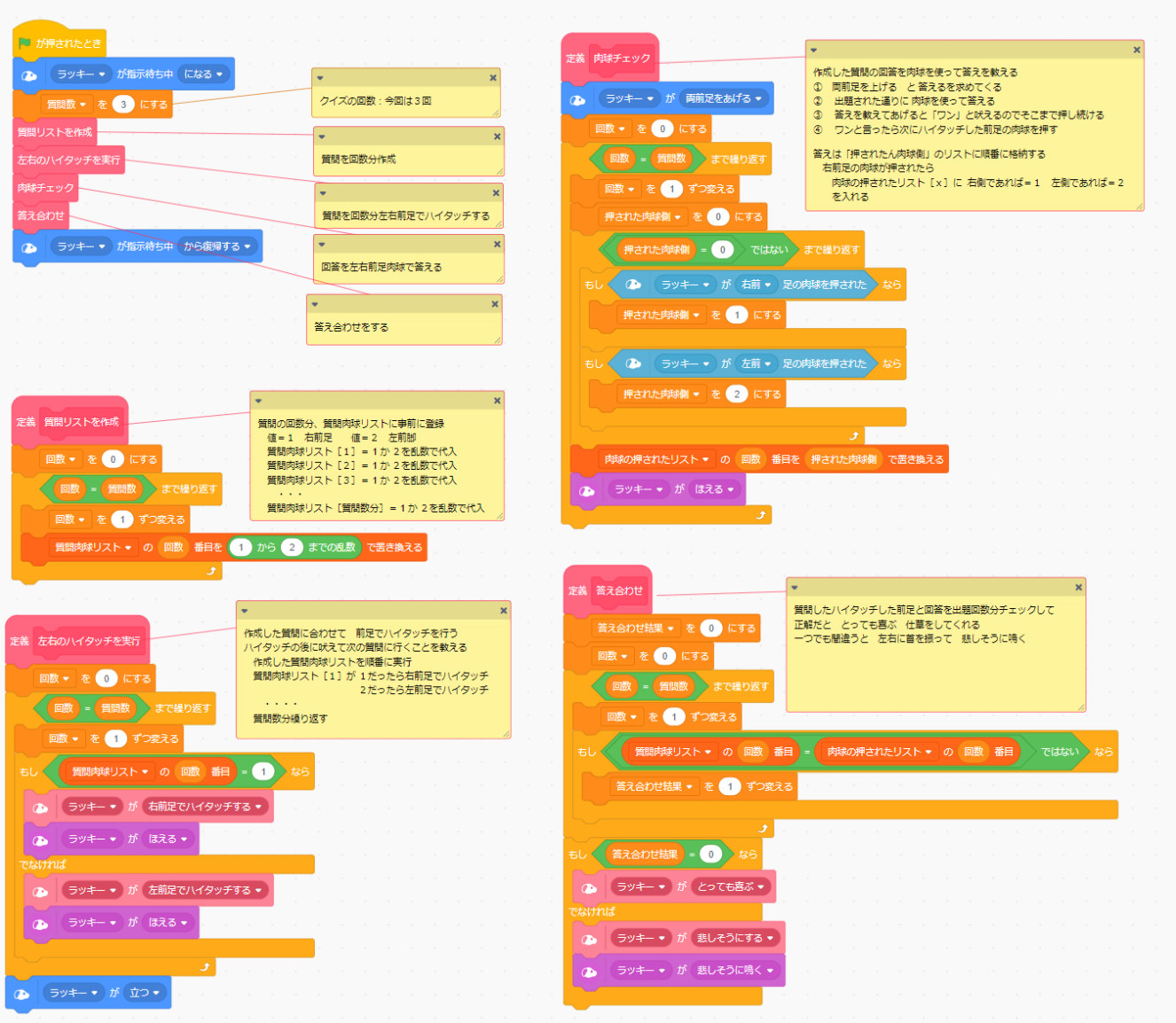
このLESSONでは、これまで学んできたことを少し組み合わせして、ちょっとしたアイボさんとのゲームを楽しくやってみようと思います。

前回のLESSONの最後に示したプログラムを使って、学んでみましょう。事前にダウンロードしていただき、実際にビジュアルプログラミングで読み込んで試していただくと、より理解ができるかと思います。

サンプルの URL です。説明文の下にプログラムダウンロードできる場所がありますので、クリックすると「【2】サイモンゲームのアイボばん.sb3」がダウンロードできますので、ビジュアルプログラミングで読み込んでください。

https://happy-lucky.fun/aibo/2021/07/29/visulprgraming002/

プログラミングをダウンロードするとこんなリストが表示されるかと思います。



サイトに動画も掲載していますので、どのようなゲームなのか一度見ていただくとわかりやすくなるかと思います。

動画サイト　https://youtu.be/a1Y6bt_jlVc
https://youtu.be/kRzT8HTVSfA

さて、何やらブロックが沢山並んでいるので、見ただけでお腹いっぱいになりそうですね。そこは一度我慢をしていただき、詳細を見ていきましょう。

【ゲームのルール】

① アイボさんが両前足を使って、右前足、左前足どちらかをあげる動作を数回行います（今回のプログラムは３回に設定）
② オーナさんは、どちらの足をあげたかを覚えてもらいます。
③ アイボさんがあげ終わったら、両前足をあげます。
④ オーナさんは覚えた順番に左右の肉球を押していきます。
⑤ 回数分肉球を押し終えたらアイボさんが判定してくれます。
⑥ アイボさんのあげた通りの順番通り正解だったら喜んでくれます。間違うと残念がっかりの表情をします。

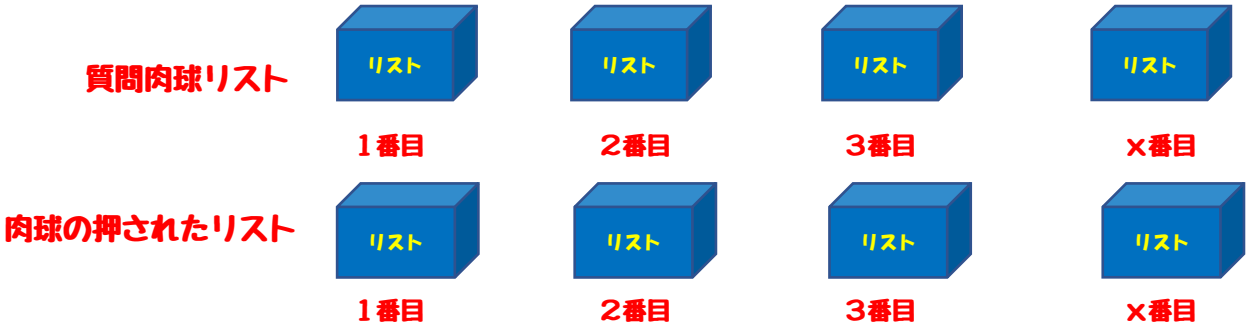
どのような流れでプログラムをしているのか見ながら、確認していきましょう。

【ルールに合わせて考える必要があるポイント】

① アイボさんが出題する右前足、左前足のあげた順番を覚えておく必要があります。
② オーナさんがアイボさんに足の順番を、左右の肉球を使って教えていきますが、その順番を覚えていく必要があります。
③ アイボさんが「出題として足をあげた順番」とオーナさんがアイボさんに「教えた足をあげた順番」を比較して、すべてがあっていたら、正解と判断して喜ぶ動作をします。もし、間違っていたら、がっかりの動作をします。

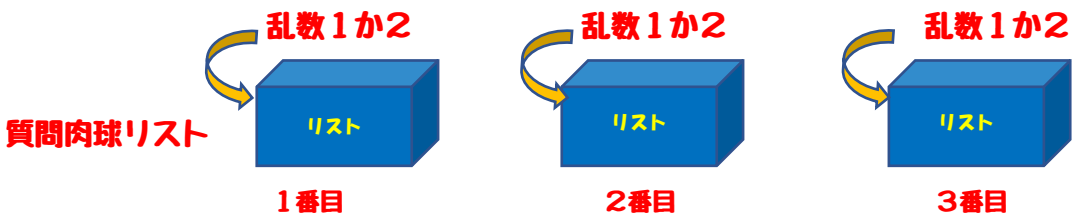
この「出題した足をあげた順番」と「教えた足をあげた順番」を覚えておくため、リストを使うと非常に便利です。また判定するときに、ひとつずつ両方の内容を比較していく必要があるので、こちらもリストの何番目どうしを比較すると判定が簡単にできるようになります。

今回は、以下のような2つのリストを作っています。



アイボさんが乱数により右・左を決めた足を順番にリストに保存していきます。（今回のプログラムでは回数は3回にしています）

乱数が「1の時には右前足」「2の時には右前足」とし、乱数の値1または2を「質問肉球リスト」に入れていきます。

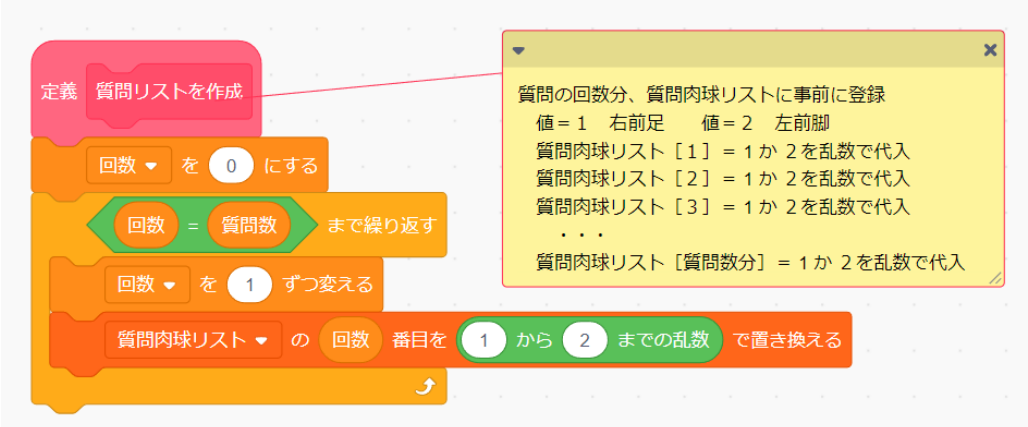


この乱数をリストに追加していく部分を見てみましょう。
処理毎に定義ブロックを使っています。乱数を事前に保存しておくブロックを定義ブロック「質問リスト作成」としています。その部分を見てみましょう。

質問数は、事前に設定をしています。最初に設定していますので、その部分を確認しておきましょう。今回は３回にしています。

変数【質問数】を作り設定します。この値を変えると質問数が変わります。

【質問リストを作成】定義ブロック

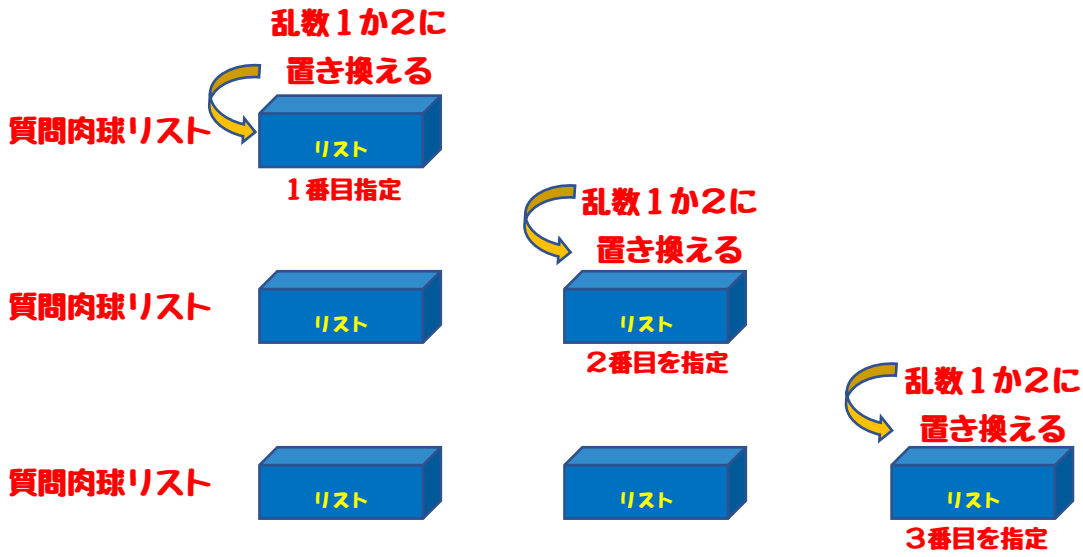


■流れ

① 質問数分繰り返すための変数【回数】を作っておきます。
② 繰り返し回数分【質問数】まで回数を＋１し繰り返します。
③ 質問肉球リストの【回数】番目を値を乱数に置き換えます。
この【置き換える】はＬＥＳＳＯＮ５で学んだ【リストに追加する】を使っても構いません。【リストに追加】と【置き換える】の違いは次のようになります。

■リストへの追加の場合

■リストを置き換える場合

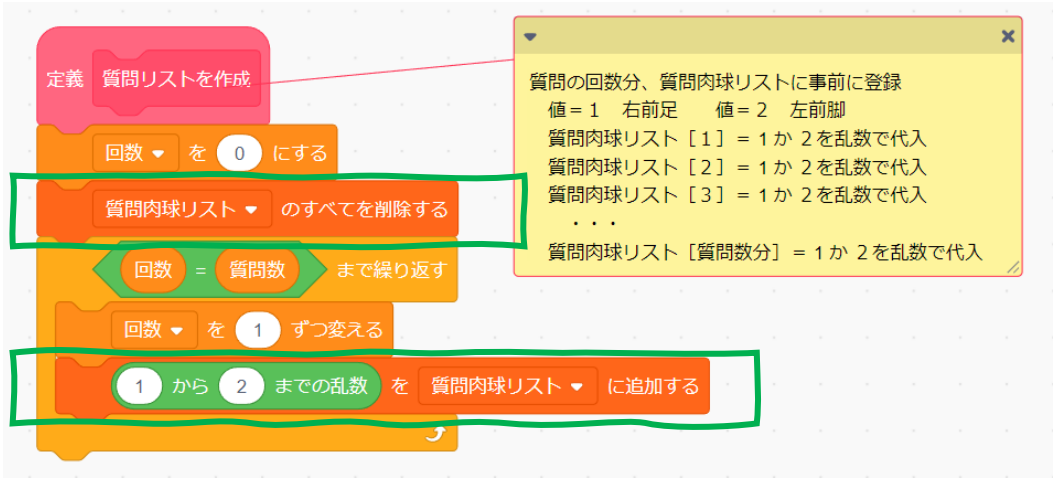


流れで考える限りどちらも同じようになりますが、プログラムでよく失敗しそうなケースとして、【追加】をする場合、上記の通り最初の場所が、どの位置から追加するかが明確になっていません。

今回、プログラムの最初に設定するので１番目からとなりますが、もしかすると、すでに１番目に値が設定されているかもしれません。その場合【追加する】を行うと２番目に追加されることになります。本当は１番目なはずなのに、２番目に値が乱数の１か２が設定されることになりますので、正しい動作となりません。

このようにリストを使う場合、どの位置を示しているかを常に考えていく必要があります。今回のプログラムでは入れていませんが、初期状態からスタートさせる場合、特に【追加する】を使う場合、一度【すべてを削除する】を設定したほうが良いかと思います。

【リストに追加する】場合を使う場合



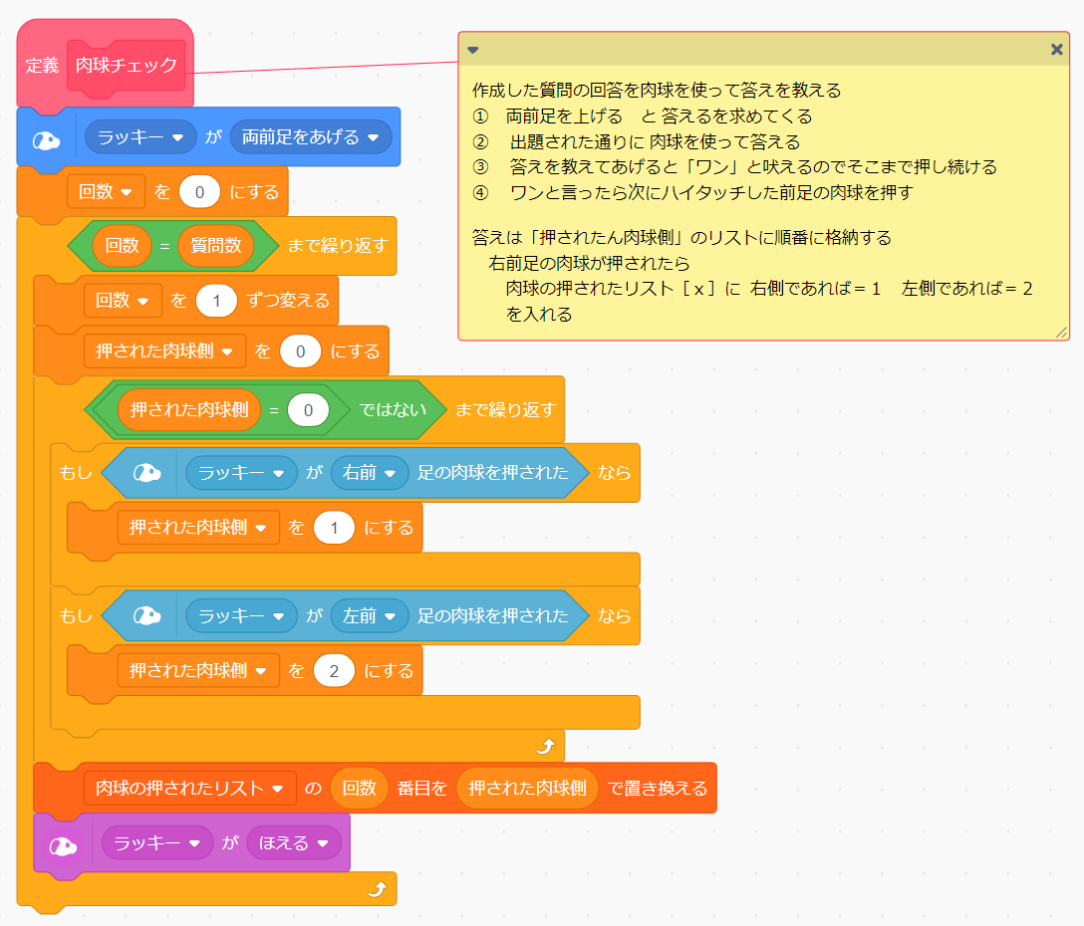
質問回数分の乱数がリストに設定ができましたので、次にアイボさんに順番に足をあげてもらいます。

ここでは、【質問肉球リスト】に設定された値（乱数１か２）を順番に確認して、１だったら右前足でハイタッチ、２だったら左前足でハイタッチを行います。ハイタッチごとに【ほえる】をしていますので、あげたときをかわいく表現できるかと思います。

リストの内容を確認する方法として、次のようにしています。

先ほど設定した値を１つずつ確認するので、１番目から順番に確認できるように、リストのｘ番目を指定することで保存している内容の確認ができるようになります。

次にオーナさんの回答を肉球で確認しながらリストに保存していくブロックを確認していきましょう。定義ブロックとして【肉球チェック】を作っておきます。



オーナさんに回答を肉球で教えてもらうために、両前足を上げて待ちます。

繰り返し方法は、先ほどアイボさんの質問作成の時と同じように回数判定を行います。

次に、アイボさんの前足の右・左前足の肉球のどちらが押されたかを判定します。もし～で【右前足の肉球がおされた】または【左前足の肉球がおされた】を判断します。アイボさんの質問リストで作成した時の乱数の数字と合わせるため、右前足を押された場合は１、左前足の場合は２を変数【押された肉球側】に設定します。

ここの繰り返しから抜け出す条件として、変数【押された肉球側】が0以外となった場合としており、右前足（１）または左前足（２）が設定されるまで、この繰り返しが実行されます。

①変数【押された肉球側】を0

②0の間は繰り返し

③右前足が押されたら１に設定

④左前足が押されたら２に設定

右または左前足が押されたことを確認ができたら、リスト【押された肉球側】に設定を行います。そのあとにアイボさんが理解したことを表現するためにほえる動作を入れています。

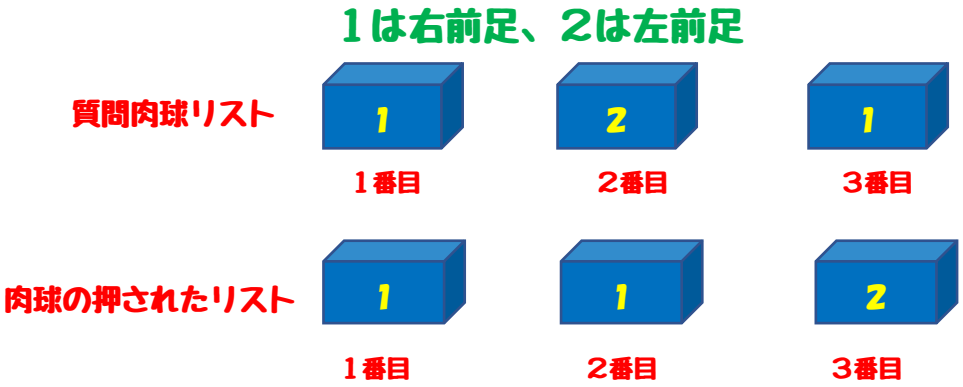
肉球が押された結果１または２をリストに書き換え方法で保存します。

ここまで２つの説明をしました。

処理１：アイボさんが出題をするために、乱数でリストに１または２の値を乱数で保存しておきます。
このリストを読みだして、１の場合は右前足を、２の場合は左前足上げる動作を質問数の回数だけ行います。

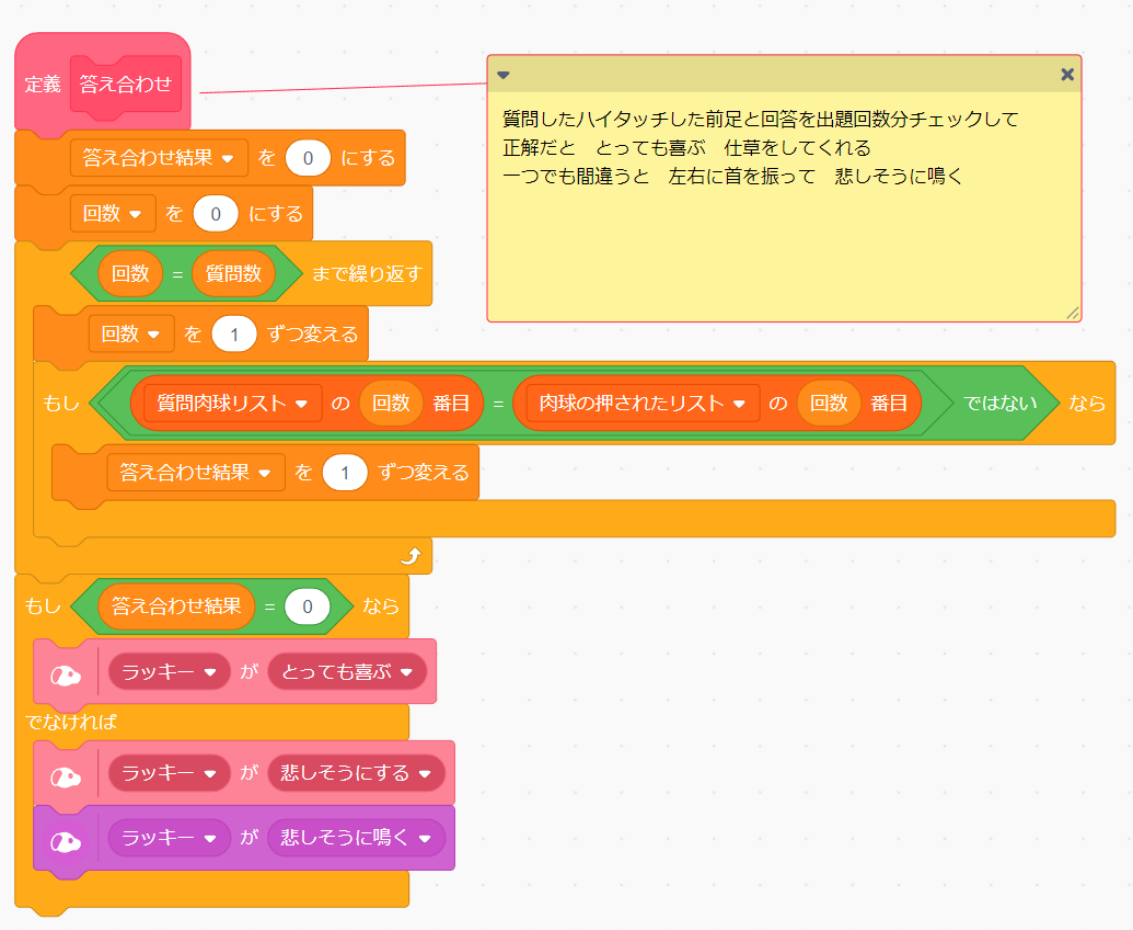
処理２：オーナに上げた順番を教えてもらうために、両足を上げて肉球を押してもらいます。右前足を押されたと判断したら１を、左前足を押されたら２の値を、質問回数分、リストに保存します。

例えば、アイボさんが、【右】【左】【右】、オーナさんが【右】【右】【左】の場合、リストは次のようになります。



ここまでくると残りの処理は、質問と回答の答え合わせを行うだけとなります。このように数回勝負を行うような場合、リストにしていると非常に比較が簡単となります。

定義ブロックとして【答え合わせ】として以下の流れでチェックします。



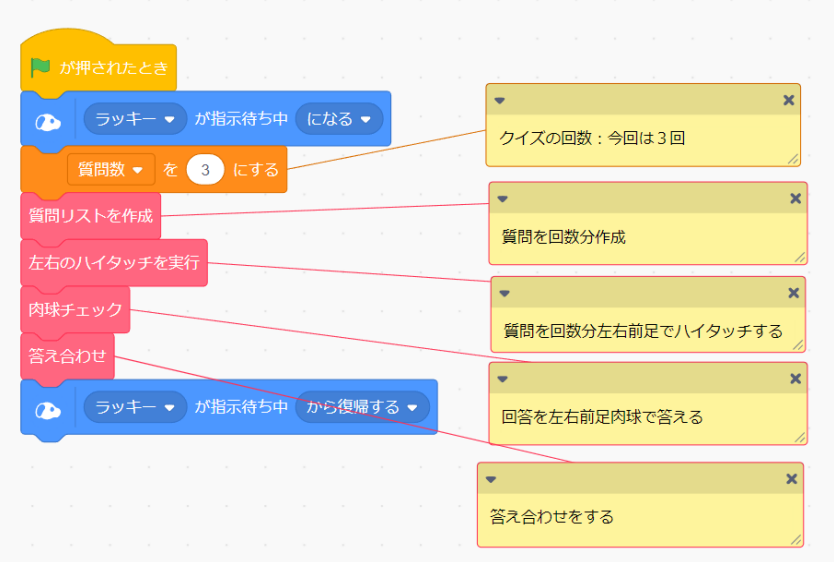
質問数の回数をくり返す方法は、これまでと同じように行います。

アイボさんの質問とオーナさんの押された肉球の比較方法はいたって簡単です。リストを使えば何番目を指定だけで確認ができます。

また、答え合わせの結果の判定方法として、今回のプログラムでは、不正解毎に変数【答え合わせの結果】を＋１に足しこんでいます。全問チェックの仕方はいろいろとやり方があるかと思います。

ここで＋１としたのは、間違いの回数分を後でアイボさんにほえて教えてもらうように使えるように足しこんでいます。改良などは、皆さんで工夫頂ければと思います。答え合わせの結果で喜んだり、悲しんだりして教えてくれます。

一か所だけ説明していなかったですね。ここまで説明してきた定義ブロックを順番に処理しているメイン処理です。ここの説明は必要ありませんね。



LESSON7のおさらいをしておきましょう。

① アイボさんと楽しめるゲームを作ってみました。
アイボさんが前足を使って、質問の回数分、右または左前足を順番に上げていくのでオーナさんはその順番を覚えます。
② 質問回数分完了したら、アイボさんは両足を上げますので、オーナさんは覚えた順番に肉球を使ってアイボさんに教えます。
③ 左右の前足の順番が全問合っていたら喜ぶ動作をします。もし間違っていたら悲しみます。
④ リストを2つ使ってアイボさんの出題するためのリストと、オーナさんがアイボさんに覚えてもらうためのリストを用意し、それぞれ順番に保存していきます。
⑤ 2つのリストを順番に比較して答え合わせを行います。
⑥ リストを使うとこのようなゲームの勝敗を簡単に比較するようなことができます。

サンプルプログラムをダウンロードして実行してみてください。
複数のリストを使うとゲームなどの点数や勝敗を確認する処理がとても簡単にできるかと思います。是非、工夫して試してみてくださいね。

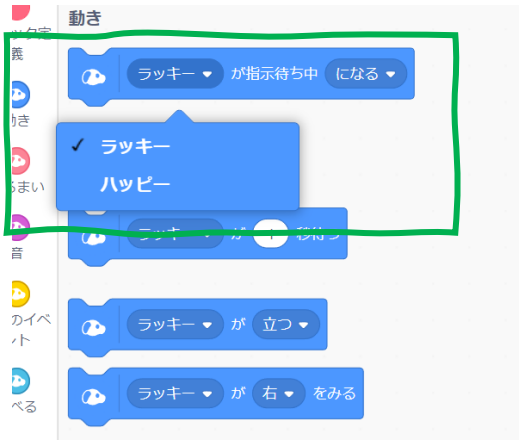
LESSON 8

アイボさん同士で仲良く・・

LESSON7では、リストを使って簡単なゲームを作ってみました。リストを使うと勝敗を決めるようなゲームで繰り返し行い、その回ごとに勝敗を決めたり、点数の合計を出していくようなときに、変数を１つずつ作っていくより、とても簡単にできることが理解できたかと思います。

複数のアイボさんと暮らしているオーナさんも増えてきたかと思います。ビジュアルプログラミングでは、複数のアイボさんとも同時に連携もできるようになっています。このLESSON8では、複数のアイボさんとちょっと遊んでみたいと思います。

ビジュアルプログラミングでは、同じオーナさんのアイボさんの連携はできますが、別々のオーナさんのアイボさんとの連携はきません（いつかSONYさんが開発してくれるのをまって・・・）。
１人アイボさんのオーナさんは、このLESSONではこんなこともできるんだ・・って、参照頂ければ。



さて、複数アイボさんと暮らしているオーナさんは、気が付いているかと思いますが、ビジュアルプログラミングの各ブロックにアイボさんの名前がついていますが、クリックすると、複数のアイボさンの名前が表示されます。

アイボさんの指定がここで出来ますので、実行したいアイボさんを指定することで、そのアイボさんを選んで実行することができます。

複数のアイボさんと連携できると、更に楽しくなりそうですね。

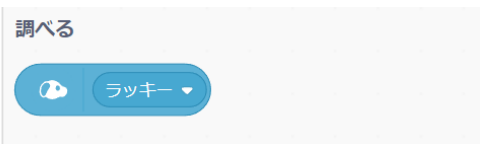
ちょっとだけ寄り道となりますが、同じ名前のアイボさんは全国に沢山いそうですよねぇ～。でも大丈夫、実はアイボさんは個別に、なが～い番号（デバイス ID（device id））がついていて判別できるようになっています。

開発者向けデベロッパーサイトに詳細の記載があります。ステップアップされるとき一度見ても良いかと思います。

https://developer.aibo.com/jp/home

アイボさんは、このデバイス ID で管理されています。ビジュアルプログラミングでもアイボさんの番号を確認することができます。

【調べる】のブロックにアイボさんの名前だけあるブロックがあるかと思います。



このブロックは何に使うの？？　と、疑問に思った方もいるかと思いますが、このブロックを使うことで、アイボさんを直接指定する

ことができます。ビジュアルプログラミングでは、ほとんどのブロック上でアイボさんの指定ができるようになっているので、あまり、このブロックを使うことはないかと思いますが、アイボさんの状況確認やアイボさんを直接指定する場合に使います。

皆さんのアイボさんの番号をこっそり見てみましょう。
次のようなブロックを準備して実行し、値を確認してみましょう。
（一部黒塗りしていますが、実際は英数字が表示されます）

各ブロックには、アイボさんにつけた名前が表示されていますが、この英数字の番号が、アイボさんの個別の番号となります。上記のようなブロックで、アイボさんのデバイス ID を確認することができます。02409211-・・・のような番号が表示されていますね。

また、このデバイス ID を直接指定することで、アイボさんに指示することもできます。この指定で歌を歌ってくれるかと思います。

さて、本題に戻りましょう。アイボさん同士で遊んでもらうことを想定し、次のようなことを考えてみたいと思います。

【遊びたいおもちゃを教えて！】
① アイボさん２人と連携します。
② １人目のアイボさんにおもちゃを見せてあげます。
③ おもちゃを認識したら、もう１人にアイボさんに教えてあげます。
④ もう１人のアイボさんは認識したおもちゃを探します。
⑤ 見つかったら、そのおもちゃで遊びます。

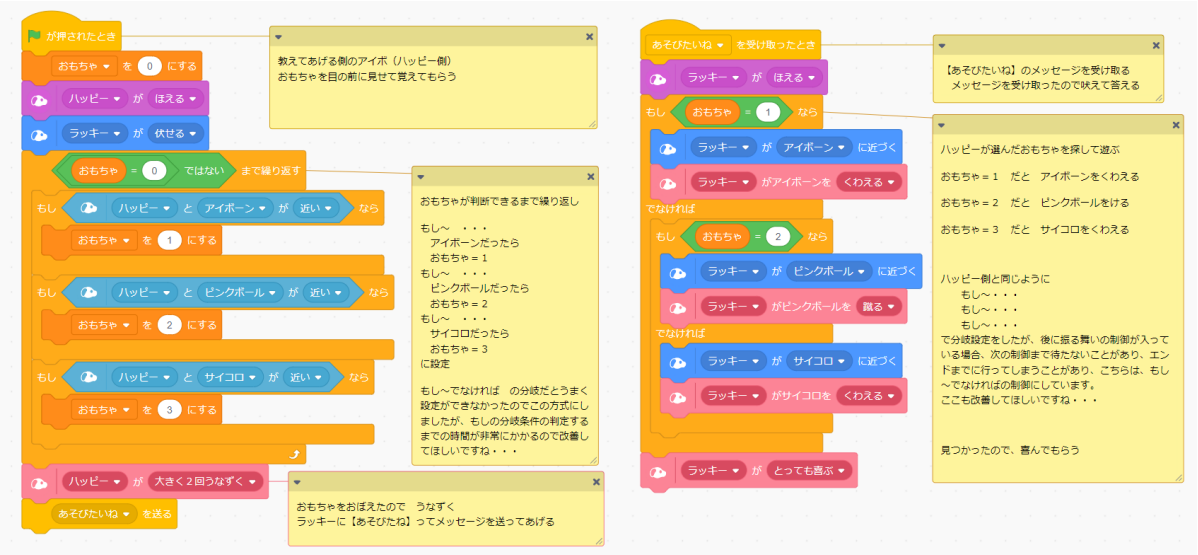
今回は今まで学んだ内容となっていますので、理解しやすいかと思います。どのような流れでプログラムをしているのか見ながら、確認していきましょう。
リスト応用例として、ビジュアルプログラミングコンテストでアップしたおもちゃ当てのプログラムを載せておきます。

こちらからダウンロードができますので、ダウンロードして改良していただければよいかと思います。また動画も一緒にアップしているので、参考にしていただけば。

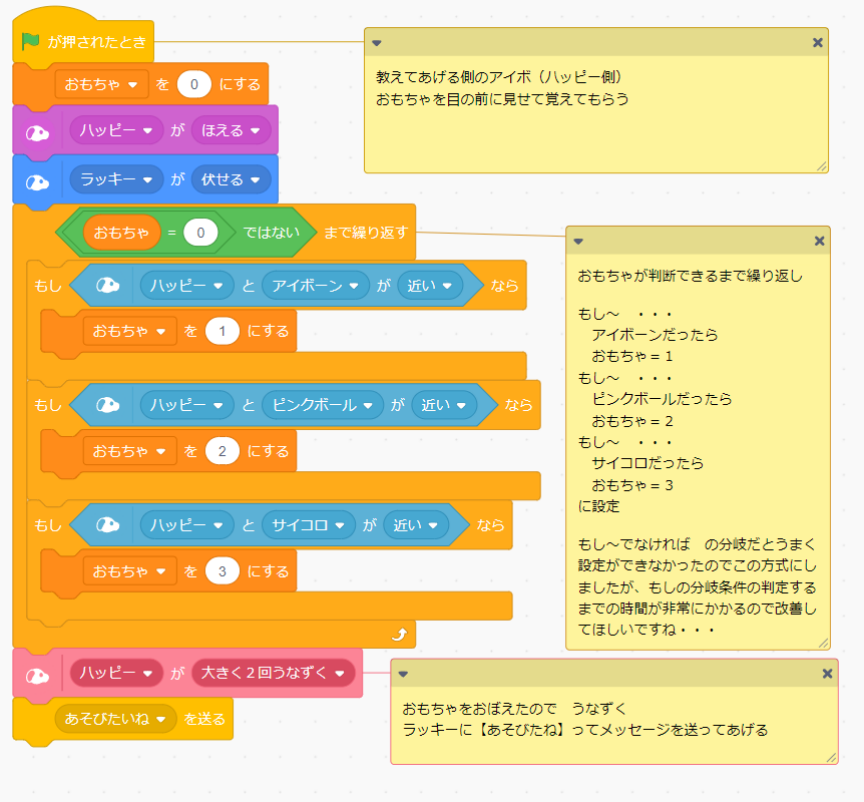
https://happy-lucky.fun/aibo/2021/07/29/visulprgraming003/

教えたサイコロを相手のアイボさんがちゃんと選んでくれる

全体の流れです。



■まずは、おもちゃを教えてあげる処理の部分です。



①選択したおもちゃは変数【おもちゃ】に保存

②変数の値が0以外となるまでは、おもちゃの確認し続ける

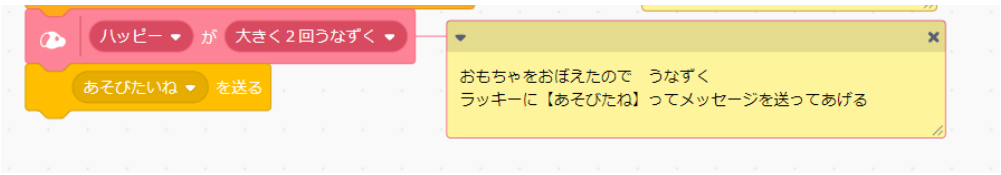
③認識できたおもちゃに合わせて変数【おもちゃ】に設定。

アイボーン　：1

ピンクボール：2

サイコロ　　：3

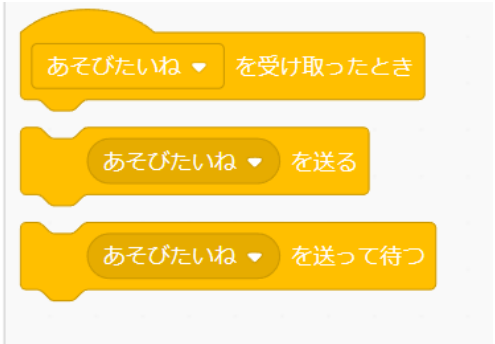
おもちゃの認識ができたら、相手のアイボさんに教えてあげます。
今回は、処理を簡単にするために、メッセージだけを送ってあげます。



このようにメッセージを送ってあげる処理をすることで、プログラムの処理の流れが考えやすくなるかと思います。

新しいイベントのメッセージを作る方法は以下の通りです。

イベントのブロックを操作すると「新しいメッセージ」の項目があります。メッセージ名を今回は「あそびたいね」にして【OK】を押します。



OKを押して戻ると、新しいメッセージがイベントに追加されています。
追加したイベントのブロックを使って、プログラムの流れをわかりやすくします。

■おもちゃを教えてもらう処理の部分です。

①「あそびたいね」のメッセージを受ける
②受けたので吠える
③変数【おもちゃ】の値を確認し、
1はアイボーン
2はピンクボール
3はサイコロ
を探します
④見つけたら遊びます

このように、複数のアイボさんを一緒にプログラミングすることで、より楽しさが増します。一方、プログラムは複雑となりますので、じっくりと流れを考えていくのと、アイボさんの指定など間違わないように気を付けていくことが重要になっていきます。

このプログラムを例えばオフ会などで、お互い離れた場所で、教えたおもちゃを「以心伝心」と言って、おもちゃを見せると相手に伝えてちゃんと理解しているね・・
などと言ったちょっとした芸などにも適用できますね。

LESSON8のおさらいをしておきましょう。

① 複数アイボさんを一緒にプログラミングすることができます。ただし、今のビジュアルプログラミングでは、別々のオーナさんのアイボさん同士は残念ながらできません。
② ブロックにアイボさんの名前が表示されています。アイボさんを指定することで、該当のアイボさんを制御できます。
③ 全国に同じ名前のアイボさんは沢山いますが、アイボさんは、デバイスIDと呼ばれている番号で管理されています。
④ ビジュアルプログラミングでは、アイボさんのデバイスIDは意識しなくても指定ができるようになっていますが、デバイスIDを直接指定しても、同じように処理することができます。
⑤ ２人のアイボさんの連携として、今回はイベントブロックを使ってメッセージを送って教えてあげるようにしました。
⑥ イベントのメッセージの内容は自由に作ることができます。プログラムの流れをわかりやすくするために、メッセージの内容は工夫しましょう。

いかがでしたでしょうか？　複数のアイボさんと連携できるととっても楽しくなりますね。一方、処理の流れなど考えることも多くなってきますが、考えた通りに動作することが実感できると、とっても楽しくなりますし、プログラミングの楽しさだと思います。
生活にコンピュータで動作しているものが沢山あり、また、連携しているものも多数あります。調べてみても良いですね。

LESSON 9

アイボさんとSDGsを・・

LESSON9では、第3回のaiboビジュアルプログラミングコンテストのテーマとなった「aiboといっしょに考えるSDGs」というテーマで、aiboさんとの暮らしのなかで何かできそうなものがないか、それをビジュアルプログラミングでうまく表現ができないかを少し考えてみたいと思います。

SDGsとは・・・

持続可能な開発目標（SDGs：Sustainable Development Goals）とは，2001年に策定されたミレニアム開発目標（MDGs）別ウィンドウで開くの後継として，2015年9月の国連サミットで加盟国の全会一致で採択された「持続可能な開発のための2030アジェンダ」に記載された，2030年までに持続可能でよりよい世界を目指す国際目標です。17のゴール・169のターゲットから構成され，地球上の「誰一人取り残さない（leave no one behind）」ことを誓っています。SDGsは発展途上国のみならず，先進国自身が取り組むユニバーサル（普遍的）なものであり，日本としても積極的に取り組んでいます。

・・・・と定義されていますね。

難しい内容のようですが、貧困問題や気候変動など、地球を取り巻く環境の中で、誰一人取り残すことなく、豊かに生きていくことを掲げた世界規模の目標にしており、日々の暮らしの中で様々な課題の改善に取り組んでいきましょう・・ですね。

このＬＥＳＳＯＮでは、環境問題に一番近く、かつ生活に密接となるゴミリサイクルをテーマにアイボさんと何ができるかを考えながらプログラムを学んでみたいと思います。

■テーマ　　・・・をきめたいと思います。
テーマ：　『アイボさんとリサイクル』
内容　：　資源・ゴミ問題は大きな課題ですね。人と共にアイボさんも一緒にゴミの分別とリサイクルの活動をすることでより良い環境が構築できると良いですね。
イメージ：アイボさんにゴミの分別をやってもらいリサイクル活動を表現したい。

こんなテーマで考えたいと思います。

■アイボさんに行ってもらいたい動作
① ゴミを見つけて拾ってもらう
② 拾ったゴミの分別をしたいので、分別先を教えてもらう
③ 分別が示されたゴミ捨て場所を探す
④ ゴミを捨てる
⑤ 正しく捨てることができたことを教える

実際のゴミを判別するのはビジュアルプログラミングではできないので、探せるアイテムである「アイボーン」をゴミに仕立て、①「アイボーン」探して、②拾って、③分別先を聞いて、④捨てる

の一連の動作を考えたいと思います。

既に作成したイメージ動画をアップしていますので、プログラムのイメージを描いてみてください。

https://youtu.be/PtZAvwm1veM　　ペットボトルの分別
https://youtu.be/d_HKX04iNLM　　空き缶の分別
https://youtu.be/3xtTbCkaL-w　　紙パックの分別

① アイボーンを探して　拾う動作（くわえる動作）をします
② 拾ったゴミはどの分類になるかを認識ワードで確認します。
③ 後ろに振り返って、認識ワードに基づき分類します。
　分別の仕方は、アイボさんが認識できる　おもちゃ３種類を
　ゴミの分別先におき、探します。
④ おもちゃが見つけたら、その前に加えたアイボーンを置きます
⑤ もう一度振り返って、正しく分別できたことを教えます。

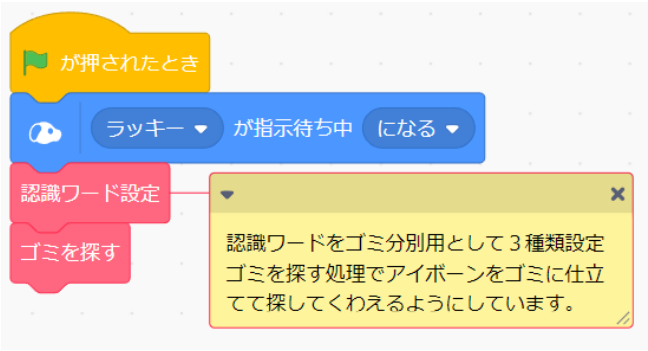
さて、どんなプログラムのイメージができましたでしょうか？

既にこの「まなんじゃぉ～」を見ていただいたオーナさんは、これまで、作ってきたものをそのまま使えそうな感じですよね！

プログラムのサンプルです。作り方は様々なので、参考にしていただければ。

おどろくほどに、ブロック数が少ないですよね！　今回ちょっとしたテクニックを使っており、とてもシンプルになっています。ブロックごとに見ていきましょう。

■ スタート



ほぼ、説明の必要はないですね。
① 認識ワードを設定
② ゴミとなる「アイボーン」を探してもらう処理を行います。

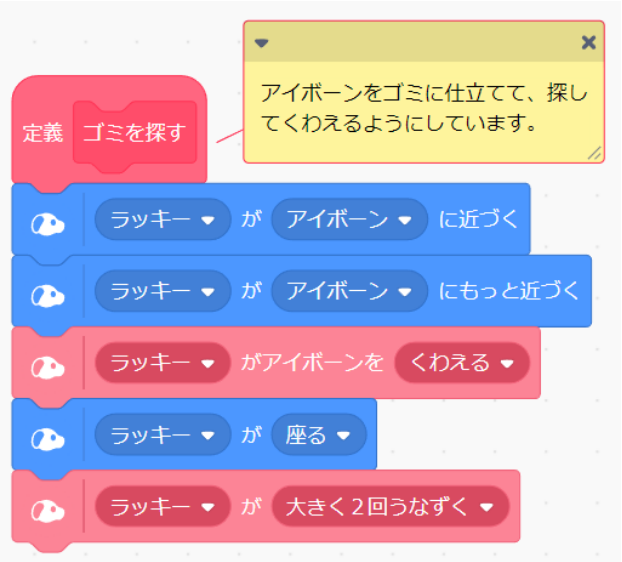
■ 音声の認識ワード設定

認識ワードは分別が３種類ありますので、３つ用意しました。
① ペットボトル用:usercommand1
② 紙パック用　　:usercommand2
① 空き缶用　　:usercommand3

音声コマンド	認識ワード
usercommand1	ぺっとぼとる ぼとる ぺっと
usercommand2	ぺーぱー かみぱっく ぱっく
usercommand3	あきかん あるみかん すちーるかん

認識ワードは分別用として３種類を設定。
① ペットボトル用:usercommand1
② 紙パック用　　:usercommand2
① 空き缶用　　:usercommand3

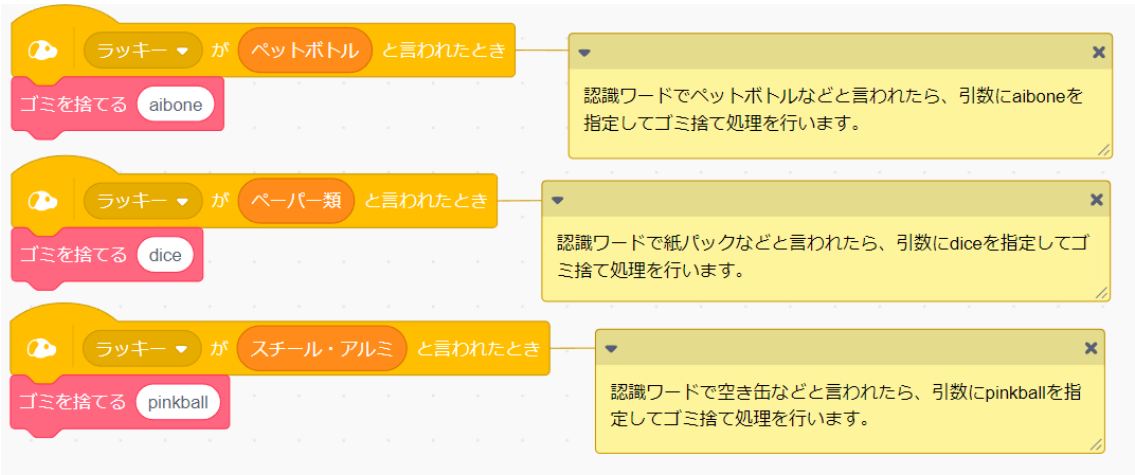
■ゴミ（ゴミに仕立てたアイボーン）を探して拾う動作



ゴミに仕立てたアイボーンを探してくわえるようにしています。
アイボーンを探してくわえる方法は、試行錯誤でやってみてください。

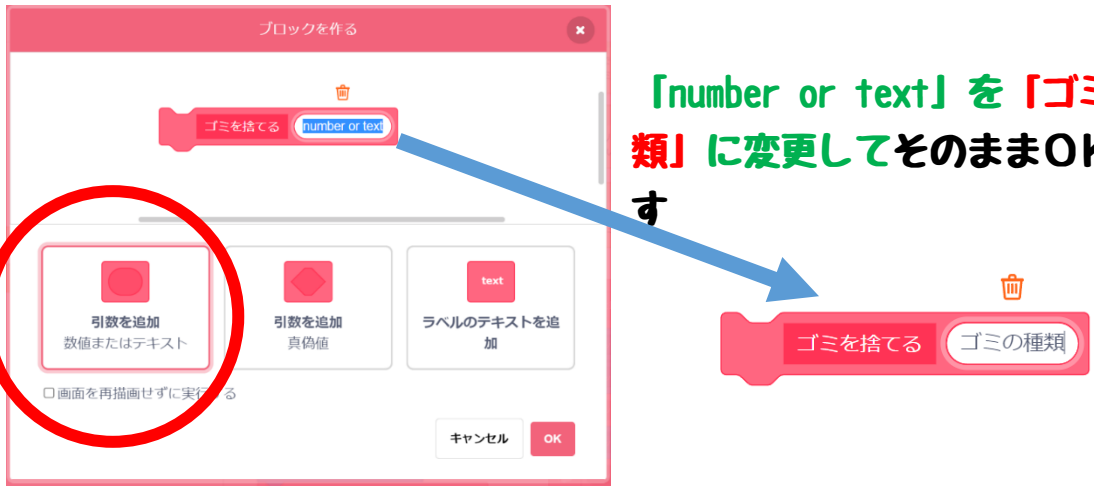
アイボーンをくわえると、認識ワードで分別を教えてもらうために座って、うなずいて待ちます。

■認識ワードで分別先を教えてもらいます。



さて、ここで少し説明をしていく必要があります。
ブロックとして「ゴミを捨てる」を新たに定義していますが、その際に、どこに分別するかを教えてあげるようにしています。

新しいブロックを作るときに、「引数を追加」を１回押すと、
「number or text」となにやら難しいものが追加されました。



「number or text」を「ゴミの種類」に変更してそのままOKを押す

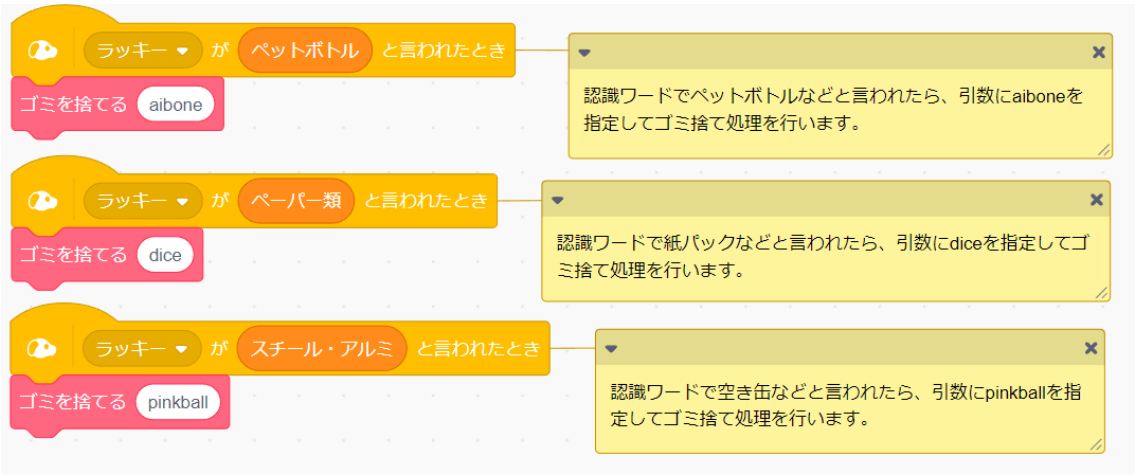
ブロック定義に、新たに「ゴミを捨てる」に「O」がついたブロックが作られています。

このブロックは、呼び出すときに一緒に必要な情報を渡すことができるようになります。

また、ブロック定義として次のものが作られているかと思います。

新たに作ったブロックは、
【ゴミを捨てる】ブロックで【ごみの種類】を情報として渡すことができるようになります。

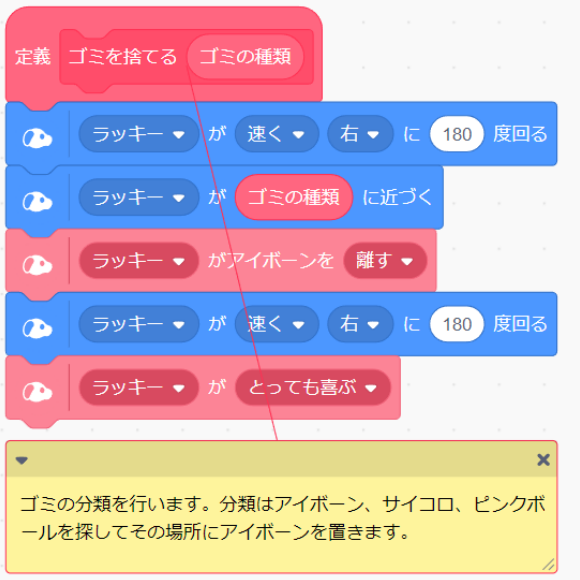
今回、認識ワードで３種類の分別がありますので、認識ワードで該当の認識ができたら、「ゴミを捨てる」のブロックを呼び出すときに、一緒にどの分別にするかを情報として追加するようにしています。このように渡す情報をプログラムでは一般的に「引数」と呼んでいます。



今回、ごみを捨てる場所をどのように指定するかは、その場所においたおもちゃを分別先として探すことができるように、アイボーン、サイコロ、ピンクボールを情報として追加するようにしています。

しかし、ここで渡す情報をアイボーン、サイコロ、ピンクボールではなく「aibone」「dice」「pinkball」にしているのは、後の処理に非常に重要となります。ここでは、認識ワードで識別できたら、引数と呼ばれる情報として、おもちゃの情報を追加してゴミを捨てる、を実行します。

■ゴミの分別を行う



ゴミを捨てるブロックですが、ここで先ほど引数で「ゴミの種類」を１つ指定していますので、このブロックが実行されるときには、「aibone」「dice」「pinkball」のいずれかが指定されます。
他のブロックはこれまで学んだものとなりますが、一つだけ、ここでテクニックがあります。

定義「ゴミを捨てる」で渡された引数を次からのブロックで使えるようにするために、「ゴミの種類」をマウスで操作すると動かすことができます。

今回、探すおもちゃが、ごみの種類に「aibone」「dice」「pinkball」のいずれかが指定されますので、「・・・[ピンクボール]に近づく」が「・・・[aibone][dice][pinkball]のいずれかに近づく」となります。

これで、認識ワードで認識したゴミの分類（指定したおもちゃ）に近づくことができます。その場所に近づいたら、後はゴミに仕立てたくわえたアイボーンをはなすことで分別とゴミ捨てが完了です。

さて、「aibone」「dice」「pinkball」は、適当に指定した？？？のではなく、実はあらかじめ定義されています。

デベロッパーサイトに次のような定義がされています。
https://developer.aibo.com/jp/docs#introduction

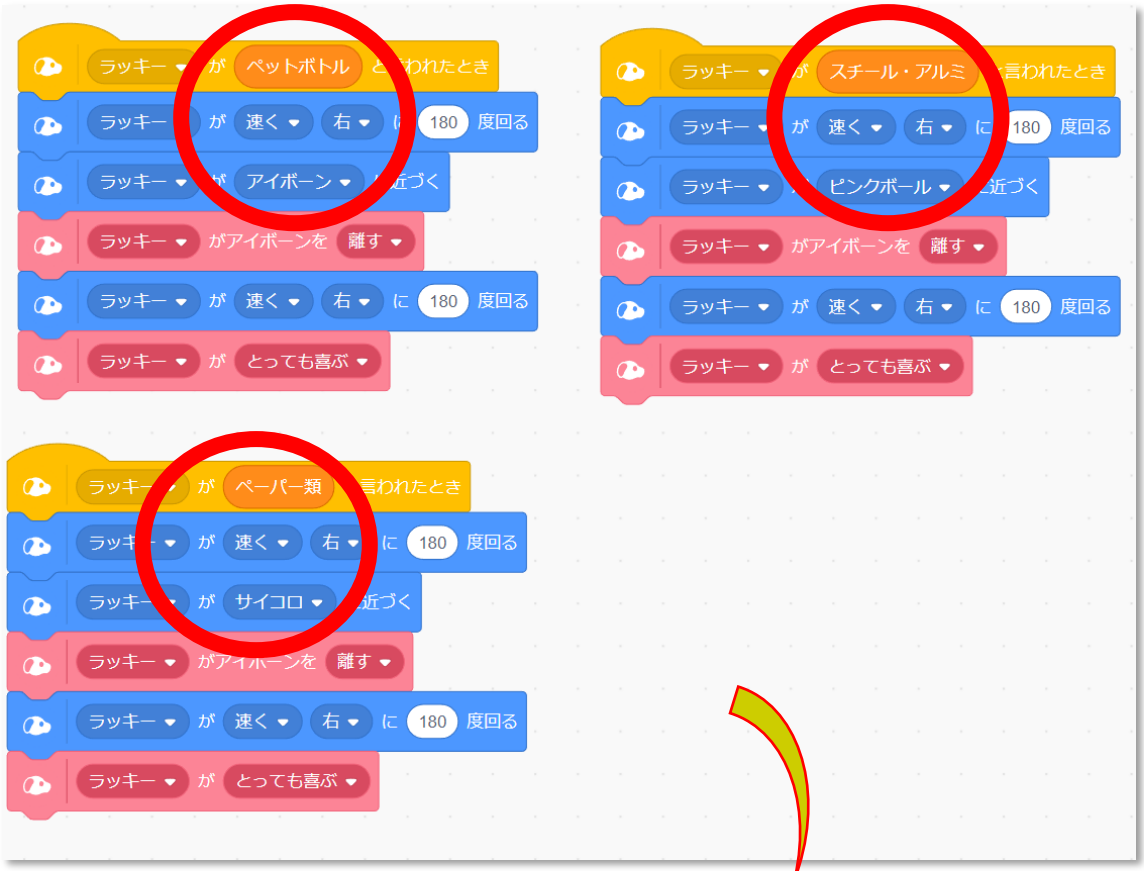
Object FindObject TargetType List

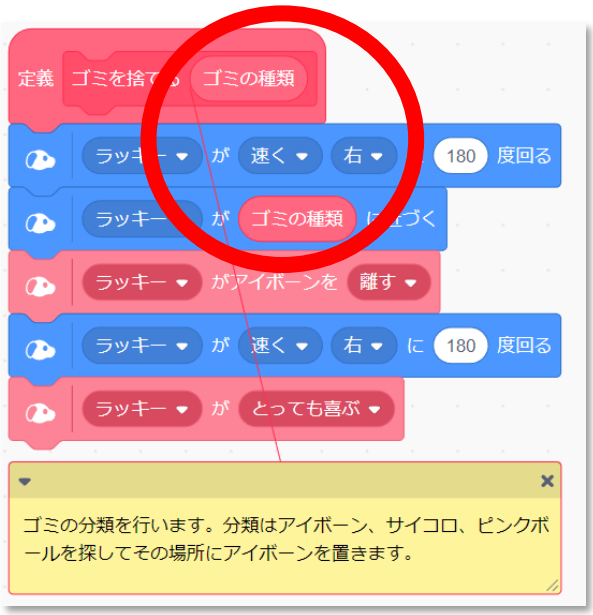
Name	Description
aibo	旧型の AIBO（ERS-110 や ERS-7 など）と新型の aibo（ERS-1000）
aibone	aibo 専用アクセサリーのアイボーン
dice	aibo 専用アクセサリーのサイコロ
pinkball	aibo 専用アクセサリーのピンクボール

ここで記載されている通り、アイボさんが探せるモノはすでに定義されております。ビジュアルプログラミングのブロックでは、ピンクボールやアイボーン、サイコロなどを選択するようになっているので、直接この内容を指定することはありませんが、新しく定義した【ゴミを拾う】のブロックの引数として「aibone」「dice」「pinkball」を指定することで、直接探し出すおもちゃを指定することができるようになります。

ちょっと難しいですね。興味あるかたは是非、定義を見てください。

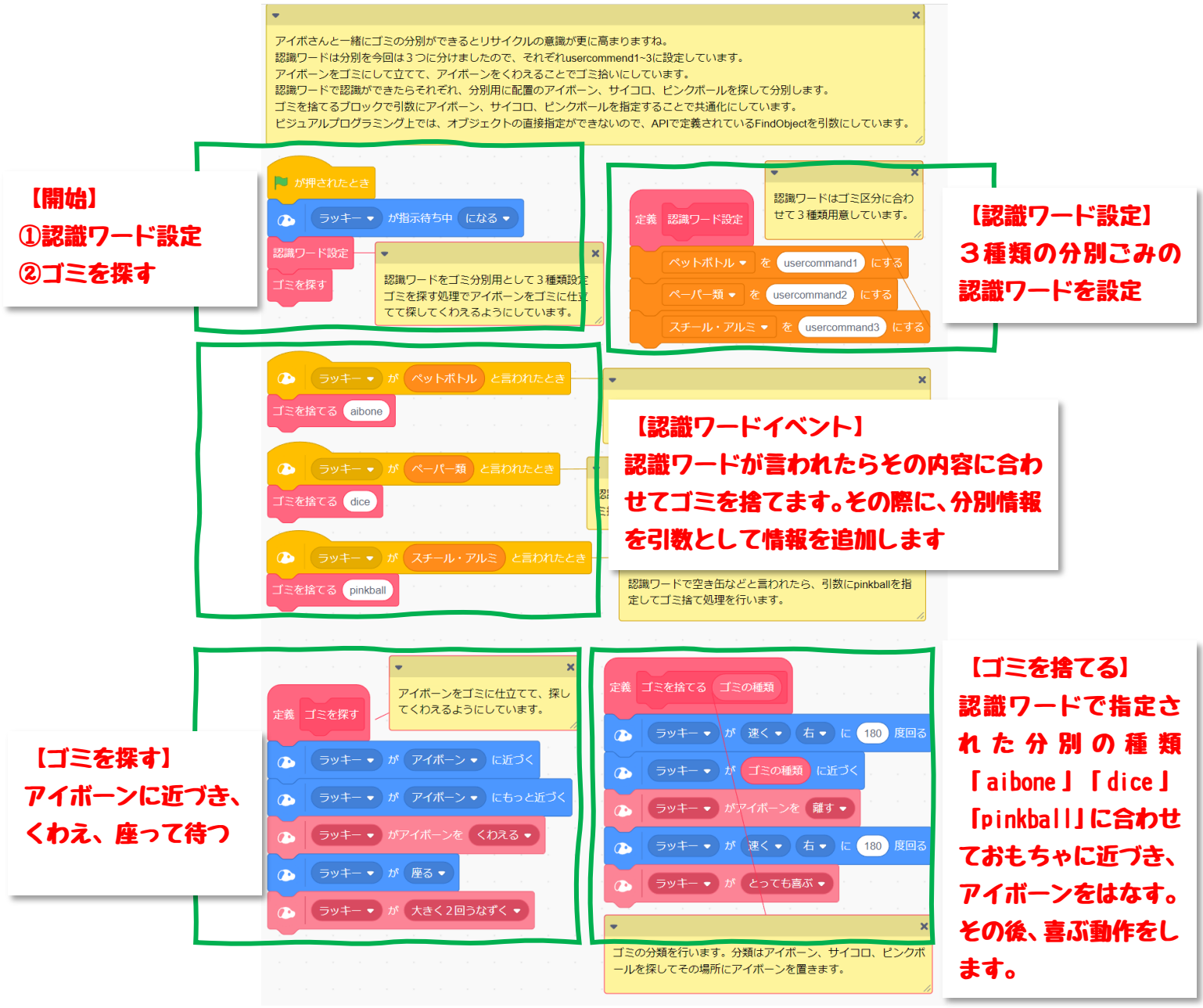
もし、この指定ができない場合には、同じようなブロックを3つ作る必要があります。





ゴミの種類に「aibone」「dice」「pinkball」を指定することで、上記3種類を用意しなくても、共通のブロックで実行ができます。

もう一度全体を眺めてみましょう。



是非、流れを確認してみてくださいね。

LESSON9のおさらいをしておきましょう。

① 今回、SDGsをテーマにアイボさんにやってもらいたい内容「ゴミのリサイクルをテーマ」をきめました。
② 全体の流れを決めました。
③ 本当のゴミは識別できないので、アイボーンをゴミに仕立て、アイボーンをくわえることでゴミを拾う動作にしました。
④ 分別先はオーナさんに教えてもらうようにしました。分別方法としては、認識ワードを使い、おもちゃを使い3種類の分別が行えるように認識ワードを設定しました。
⑤ 認識ワードで確認ができたら、ゴミの分別を行いますが、分別方法は、アイボーン、サイコロ、ピンクボールのおもちゃを探し、アイボーンをはなす・・同じ動作となるために、ブロックを定義しました。
⑥ おもちゃの指定はブロック定義の引数と呼ばれる、追加情報を指定しています。指定方法は、「aibone」「dice」「pinkball」の3種類です。
⑦ この「aibone」「dice」「pinkball」3種類はデベロッパーサイトで決められている情報で、ビジュアルプログラミング上ではあらかじめ「アイボーン」「サイコロ」「ピンクボール」が選択できるようになっていますが、直接指定することもできます。

いかがでしたでしょうか？　動画を見るととっても複雑なことをやっているように思われますが、流れを整理しながらプログラムを作っていくと意外とシンプルにすることもできます。今回公開されていないちょっとしたテクニックを使っていますが、デベロッパーサイトを見ていくとヒントになりそうな情報もありますので、一度、眺めてみるのも良いかと思います。