社会情報科学概論（川嶋-3/4）
データとプログラム（実践編）

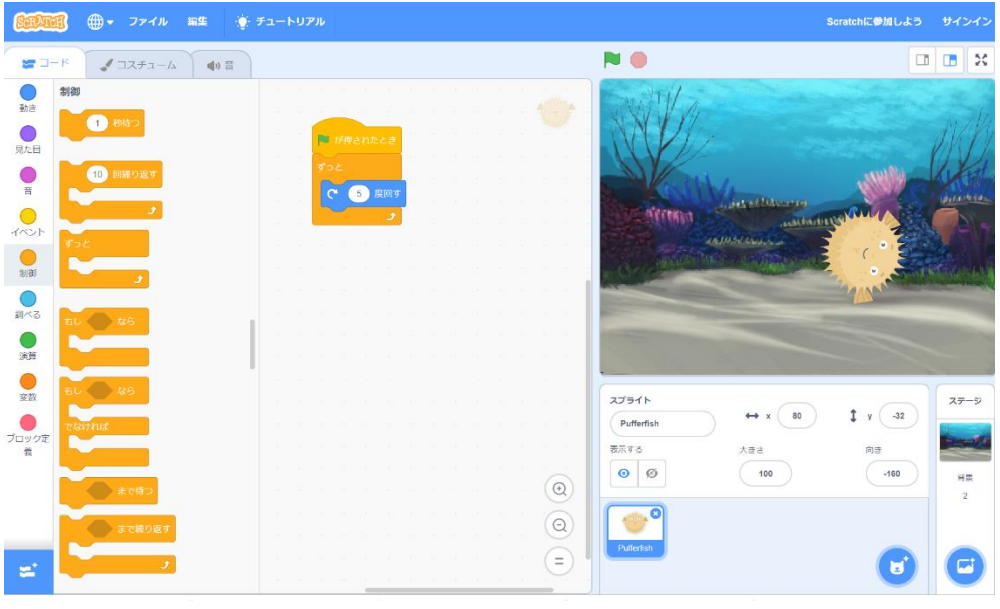
兵庫県立大学　社会情報科学部

川嶋宏彰

kawashima@sis.u-hyogo.ac.jp

Scratch

- ScratchはMITメディアラボによるプログラミング学習環境
  - 現在Version 3.0
  - オンライン版 (ブラウザ上)
  - オフライン版 (インストール版)
- ビジュアルプログラミング
  - ブロックを組み合わせる
  - Scratch の他にも Blockly などいろいろ
- 子供から大人まで
- 結構複雑なものを開発可能



Scratch

- Scratch は ゲームプログラミングに近い
  - スプライト：キャラクター（スプライトは複数あってよい）
  - ステージ：背景など（ステージは一つ）
  - 各「スプライト」もしくは「ステージ」にコードが紐づいている
    - それぞれのコードが並列に実行される
    - コード間ではメッセージや変数でやり取り・・・やや複雑
- 今日は一つのスプライトのコードだけを用いる
  - 並列処理なしが基本形（後期以降のプログラミング演習も）
  - いきなり中級クラス

行動分析？基礎

- 人の嗜好や行動には偏りがある
  - Amazonの推薦システム，行動ターゲティング広告
  - たとえじゃんけんであっても
- ランダムに手を出す相手にはじゃんけんの勝率は5割
  - 「グー」「チョキ」「パー」の確率がそれぞれ1/3
  - 勝率＝勝った回数／（勝った回数＋負けた回数）
- じゃんけんには癖がある
  - 人によって出す手に偏りがある
  - 状況にも依存する（過去の手が何だったか，過去の勝負結果）
- (例) 昭和～令和まで続いているアニメ
  - 1991年の秋に「んがぐぐ」から「じゃんけん」に変更

行動分析？基礎

- サザエさんじゃんけん研究所
  - http://park11.wakwak.com/~hkn/
  - 1991年 (平成3年)のじゃんけん開始時から記録

平成の記録といってもよい?! (約30年!)

- サザエさんじゃんけん白書（2019年春版 4/6）によれば
  - サザエさんじゃんけんにも中の人（担当者）がいると思われる
  - 各クールの初回には「チョキ」を出しやすい
  - 3回続けて同じ手がでる確率がランダムに比べ極端に低い
  - 傾向が過去に6通り（現在6人目の担当者？）
    - 過去2回に対し次に出た手の頻度を記録

社内人事の事情まで…

- 過去の傾向を分析して事前に予想（現在はTwitter利用）
  - 1996~2018年の勝率: 70%以上
  - 平成最後の勝負も勝利でかざっている

クラスタリング，変化点検出，マルコフモデル…
（機械学習・時系列分析でもおなじみの手法）

じゃんけんゲーム（機械学習版？）

1. じゃんけんをするゲームを作る
   - 人 vs コンピュータ
2. 人の傾向を記録
   - 「グー」「チョキ」「パー」の度数分布
   - 最頻手の計算（最大値の計算アルゴリズム）

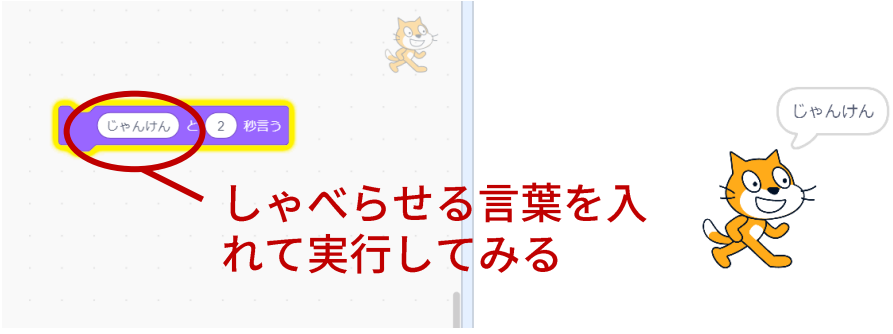
目標: 2 の途中

3. 統計分析: 偏り(癖)はあるのか？
   - 帰無仮説: 等確率(全ての手が確率1/3)である
   - 対立仮説: 等確率ではない … カイ二乗検定
4. 機械学習による予測とコンピュータの手の決定
   - 傾向に応じて出す手を変える (最頻手に勝つ手を一定割合入れる)
   - 過去1, 2回や勝負結果等に応じた偏りも考慮して出す手を変える

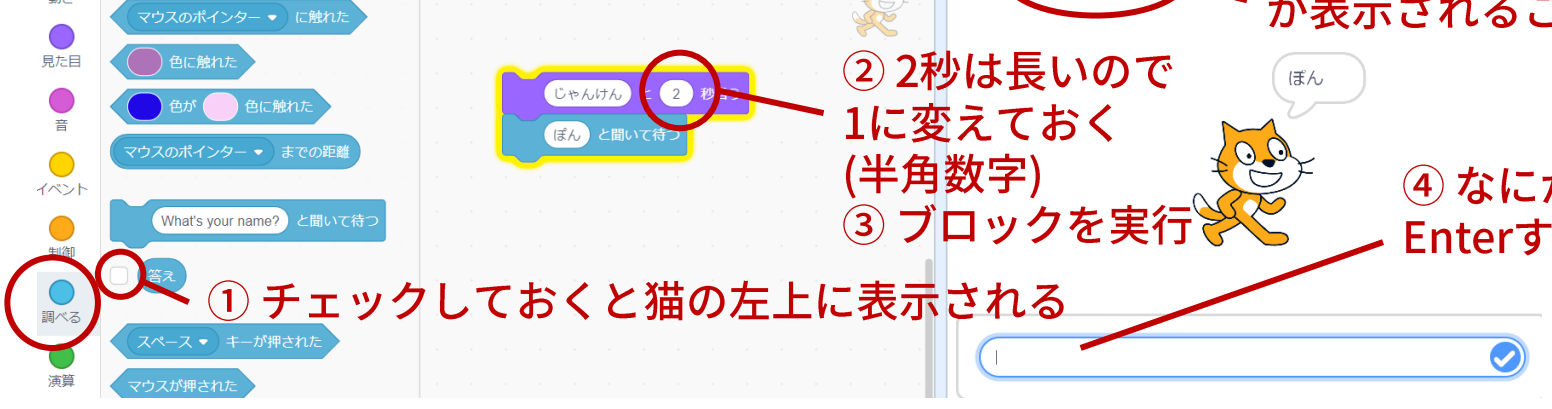
じゃんけんゲーム

- 「じゃんけん」と言う

- ブロックの色の部分をクリックすると実行される
- つながったブロックの場合はまとめて実行される



- 「ぽん」と聞いて待つ



変数とリスト

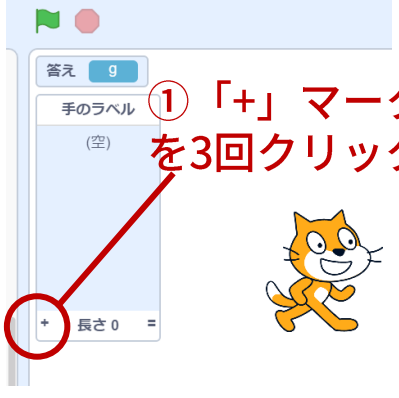
- 仕様: 半角 g, c, p (グー，チョキ，パー) どれかを入力してもらう
  - g →1, c → 2, p → 3 に変換し，内部では数値で扱う
  - 変換にはリストを使う．変換後は「人の手」という名前の変数に入れる
- 変数: １つの値 (数値もしくは文字列)を記録しておける箱
- リスト: 複数の値を記録しておける引き出し (はじめ0段, 追加削除可)

変数とリスト

- g→1, c→2, p→3 に変換
  - リストの添え字 (index) を返すブロック  を利用

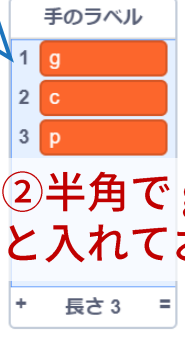
リスト名

ここに入れた中身が何番目に入っているか(index)を教えてくれる
見つからないときは 0 になる



① 「+」マークを3回クリック

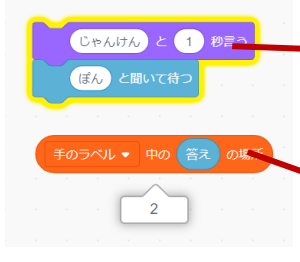
リストのindex



②半角で g, c, p と入れておく

③ g, c, p いずれかを入れてブロックをクリックするとindexが表示されるのを確認

④ 「答え」のブロックを入れる



⑤ ブロックを実行し入力欄に文字を入力

⑥ 変換後の数値が表示されるかクリックで確認

変数とリスト

- 変換後の数値を変数に入れる
  - 人 (プレイヤー)の出した手を入れるので「人の手」という名前にしておく

①「変数を作る」をクリックして「人の手」という名前の変数を作成

変数に関係するブロックを試しておこう

変数に値を入れるブロック

変数に値を加算するブロック（変数の中身が文字列のときは置きかえられる）

② リストで変換した数値を変数に入れる (代入)

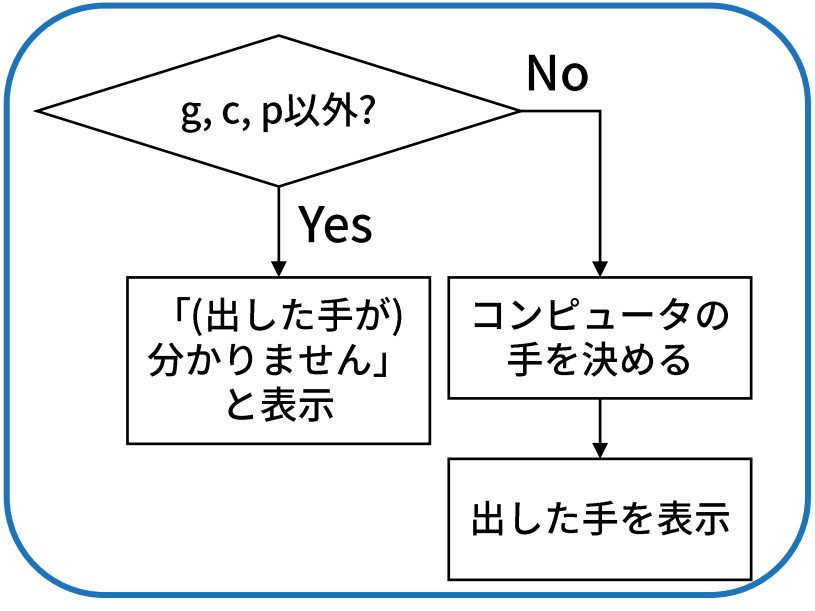
リスト関係のレポーターブロックとブーリアンブロックも試してみよう

ブロックの形状で区別できる

boolean: 真偽

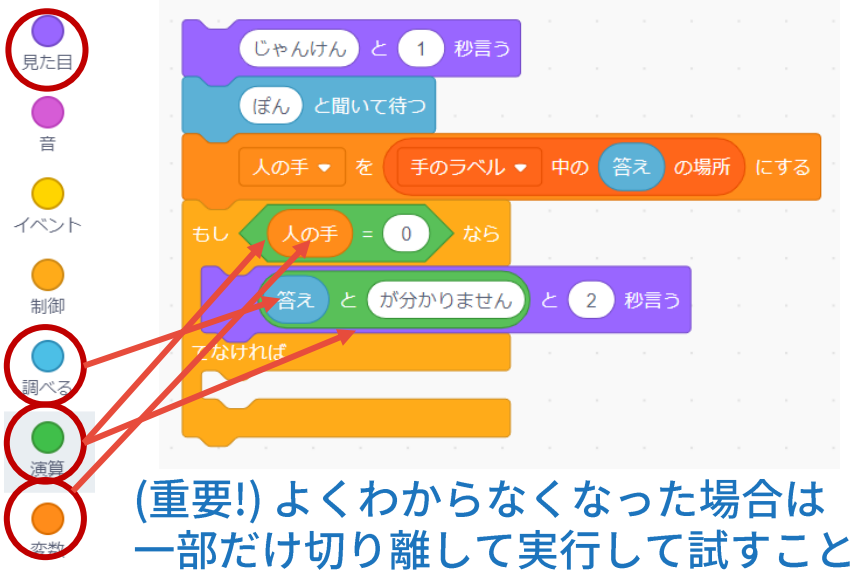
条件分岐

- プレイヤーの手が有効か否かを判断して条件分岐する
  - 無効な入力 (g, c, p 以外) ならメッセージを出そう
  - 有効な入力 (g, c, p いずれか) なら「コンピュータの手」を決めよう

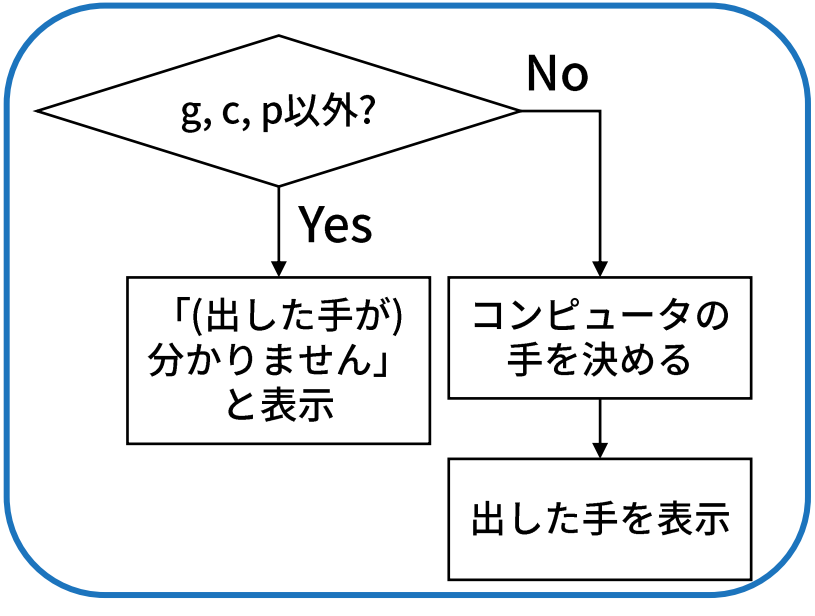


条件分岐

- プレイヤーの手が有効か否かを判断して条件分岐する
  - 無効な入力 (g, c, p 以外) ならメッセージを出そう
  - 有効な入力 (g, c, p いずれか) なら「コンピュータの手」を決めよう



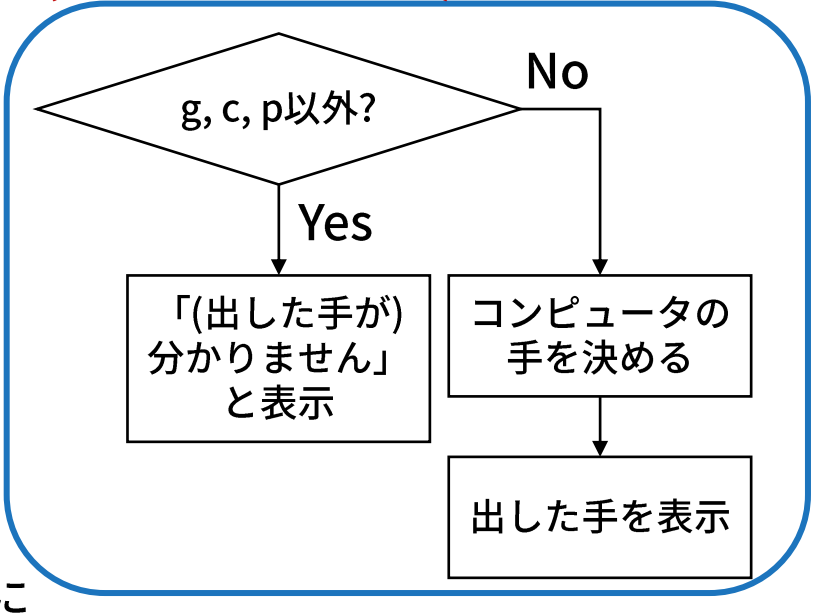
(重要!) よくわからなくなった場合は
一部だけ切り離して実行して試すこと



条件分岐

- プレイヤーの手が有効か否かを判断して条件分岐する
  - 無効な入力 (g, c, p 以外) ならメッセージを出そう
  - 有効な入力 (g, c, p いずれか) なら「コンピュータの手」を決めよう

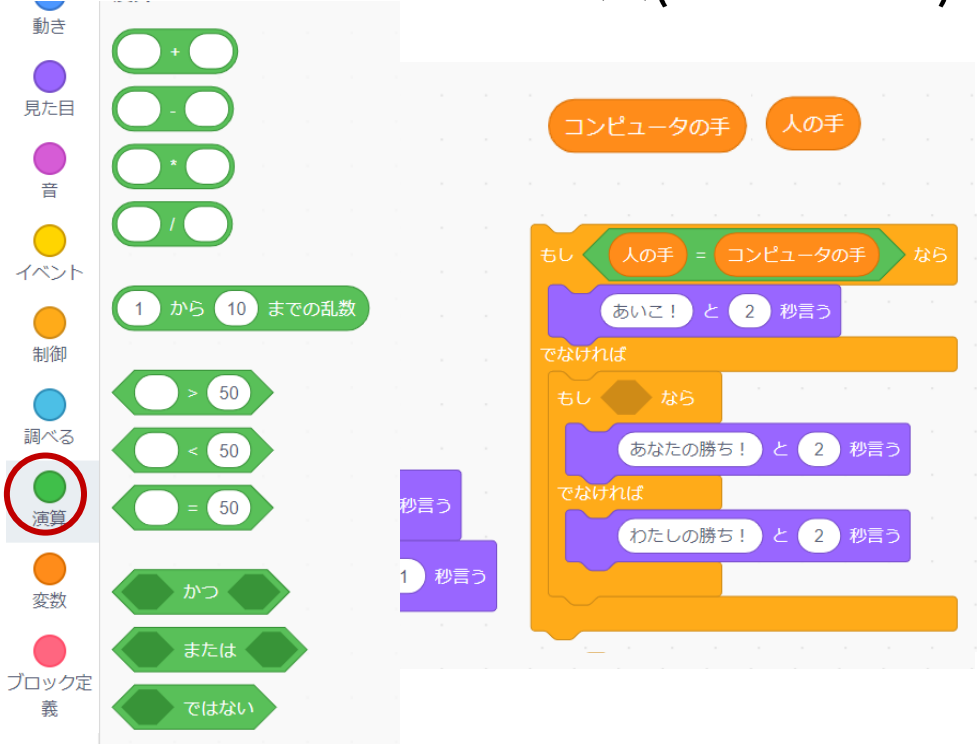
「コンピュータの手」(もしくは「猫の手」)という名前の変数を作成しておく

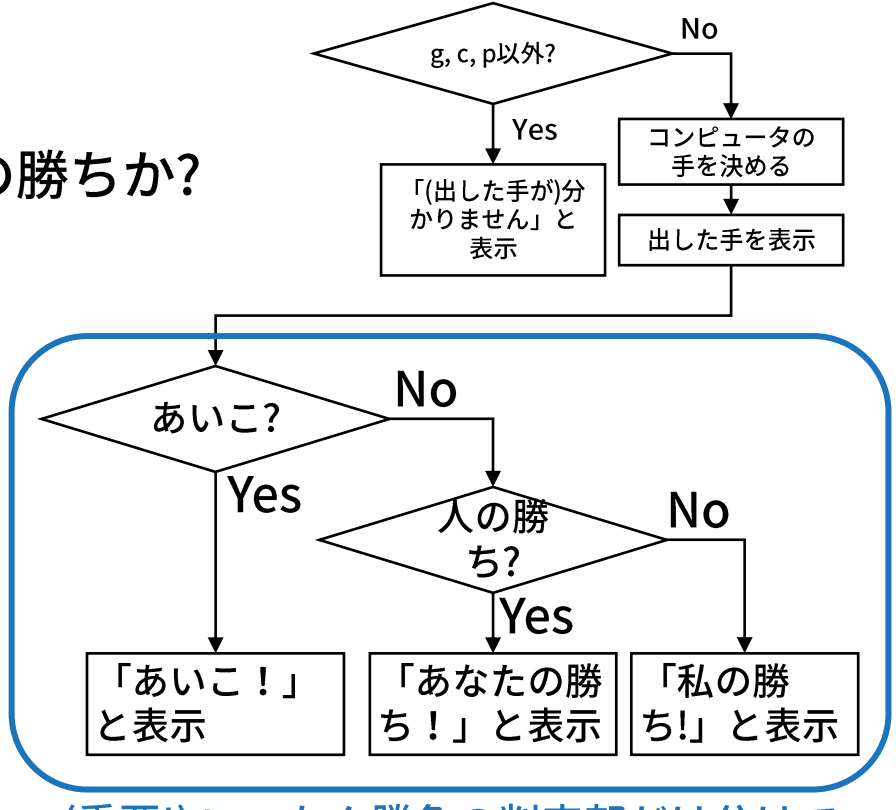


1→g, 2→c, 3→p へ変換されて表示 (日本語にしたければ別途リストを作成しておけばよい)

勝負の判定

- 勝負の判定も条件分岐
  - あいこか?
  - あいこでないなら人(プレイヤー)の勝ちか?



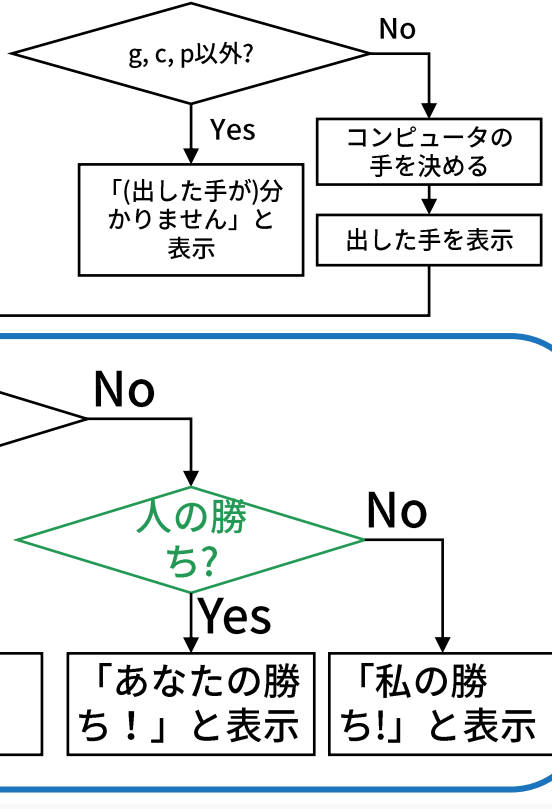


(重要!) いったん勝負の判定部だけ分けて作り，動作をテストする (単体テスト)

勝負の判定

- 人(プレイヤー)の勝ちはどう判定できる?

人の手	コの手
g (1)	c (2)
c (2)	p (3)
p (3)	g (1)



その1

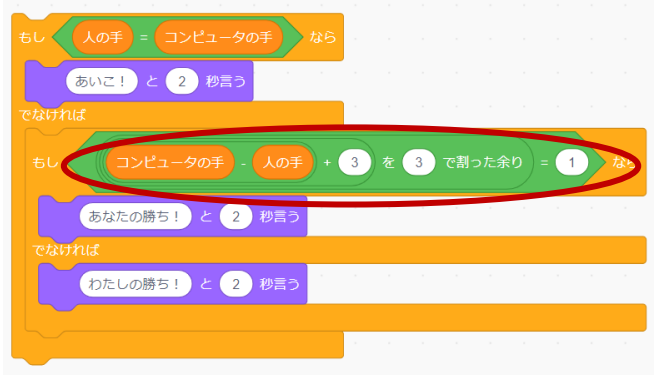
勝負の判定

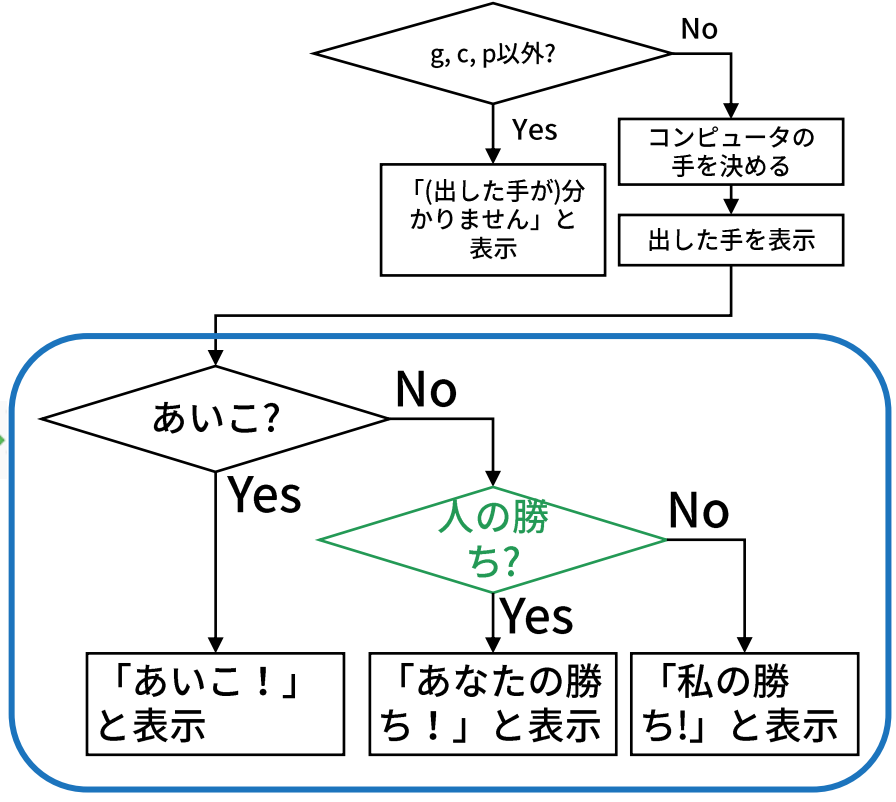
- もっと短く判定できる?

人の手	コの手
g (1)	c (2)
c (2)	p (3)
p (3)	g (1)

その2

その3



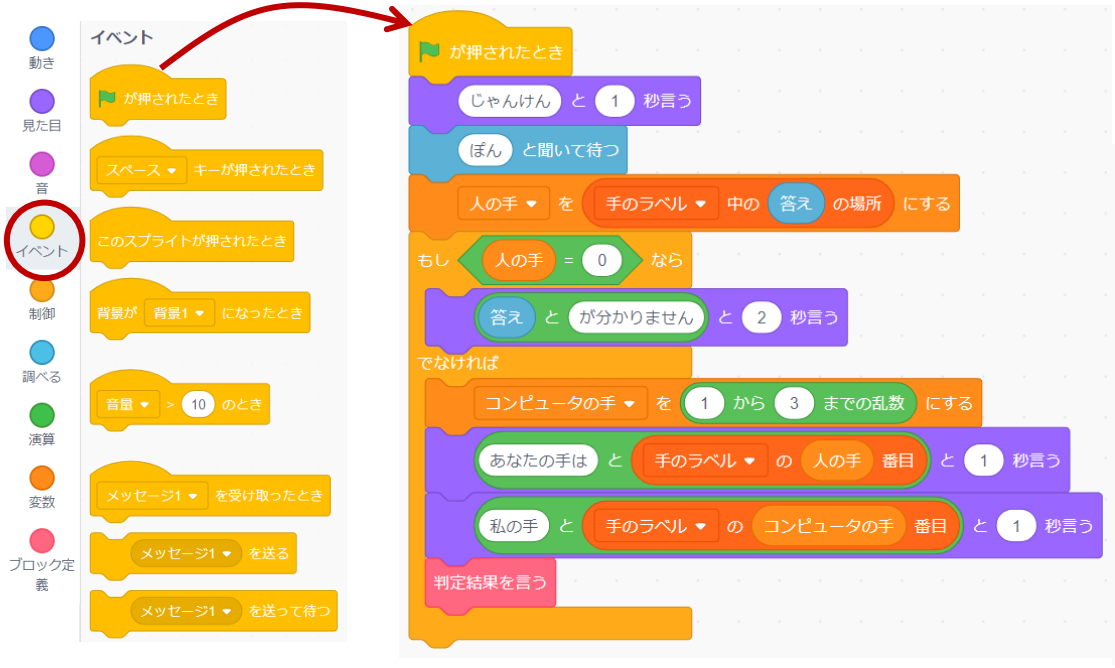


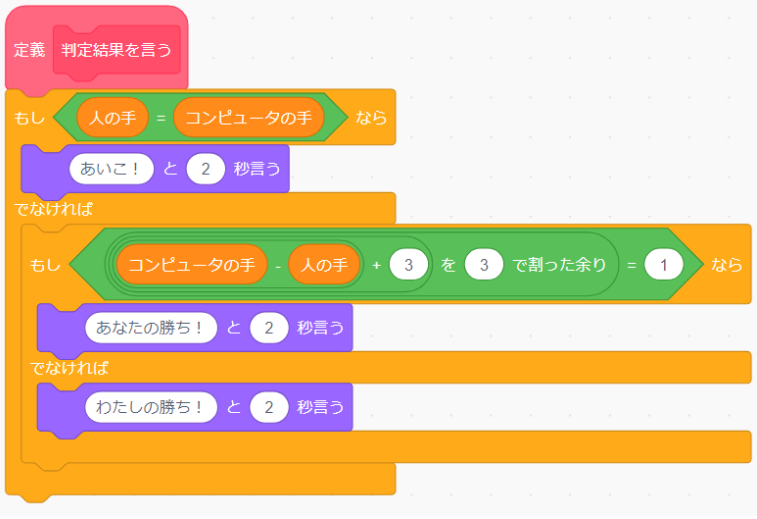
定義 (独自ブロック, 関数)

- 一言で言い表せる処理のまとまりは新たなブロックとする
  - (例) 「判定結果を言う」の定義を作る

イベントでゲームを開始

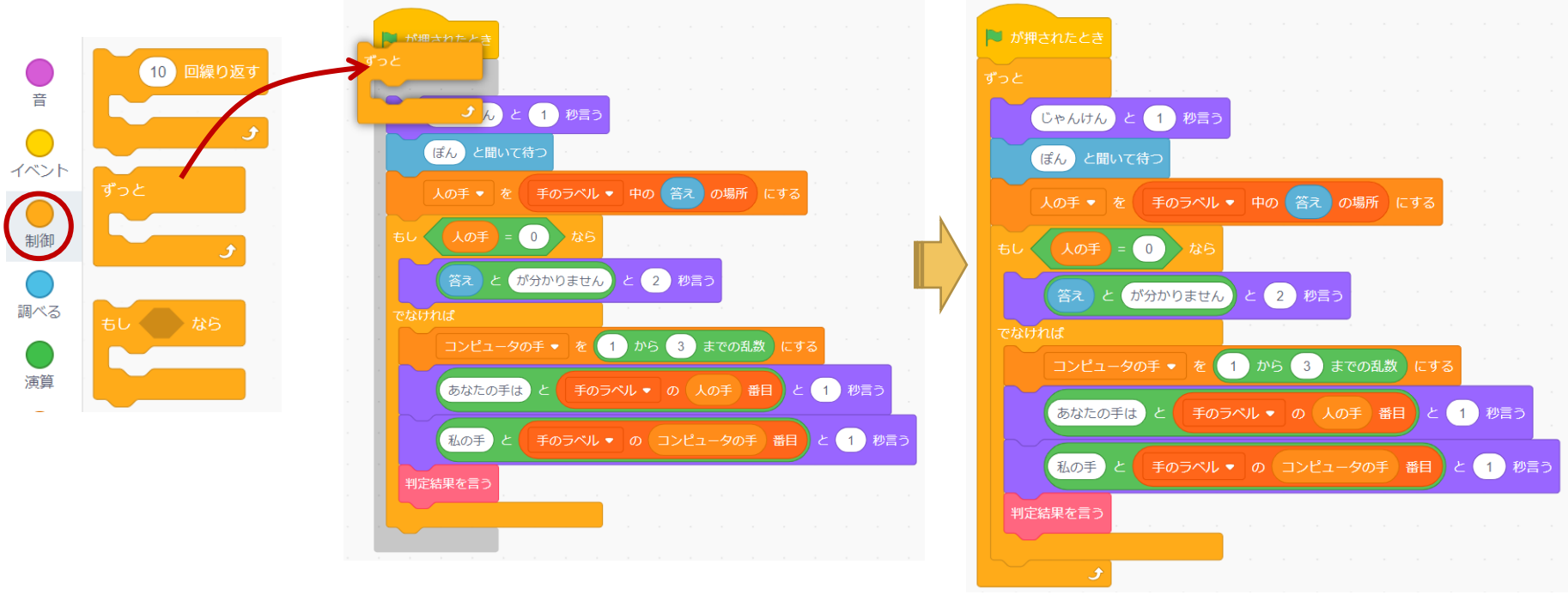
- 緑の旗クリック時にゲームを開始させる
  - ちなみに赤丸で実行終了





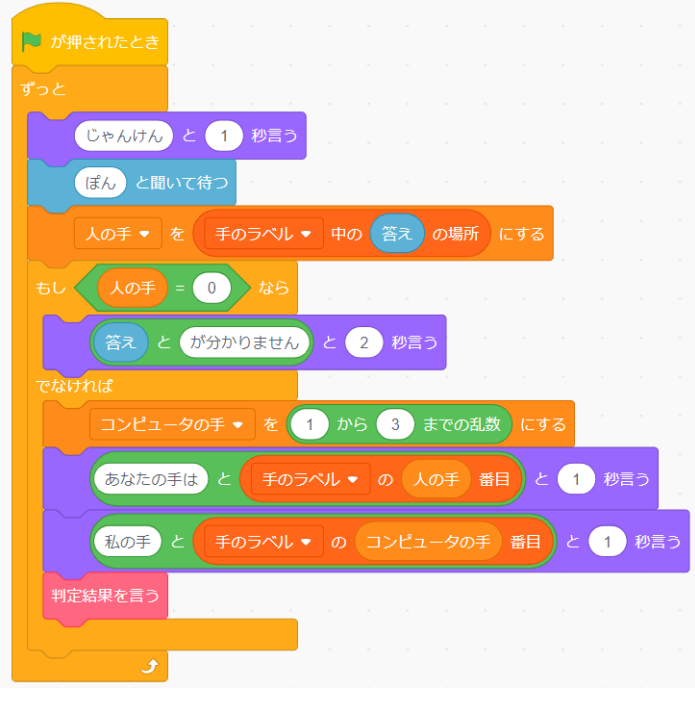
ループ

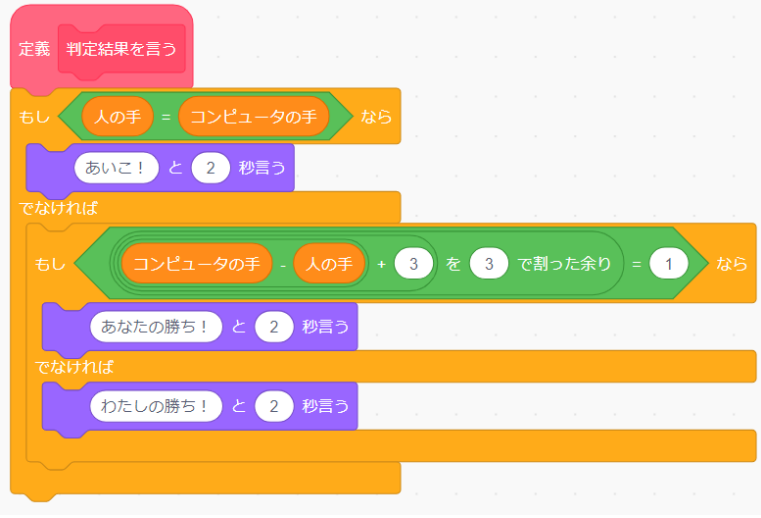
- 緑の旗を毎回押さずともじゃんけんを繰り返すにはループで
  - 無限ループを使ってみる（回数指定のループもある）



じゃんけんプログラム完成

- 条件分岐，ループ (繰り返し)，変数，リスト，乱数，四則・剰余演算，定義，入出力など，重要な要素を含むプログラム
  - 「グー」「チョキ」「パー」の手の絵を表示するなどの拡張も容易



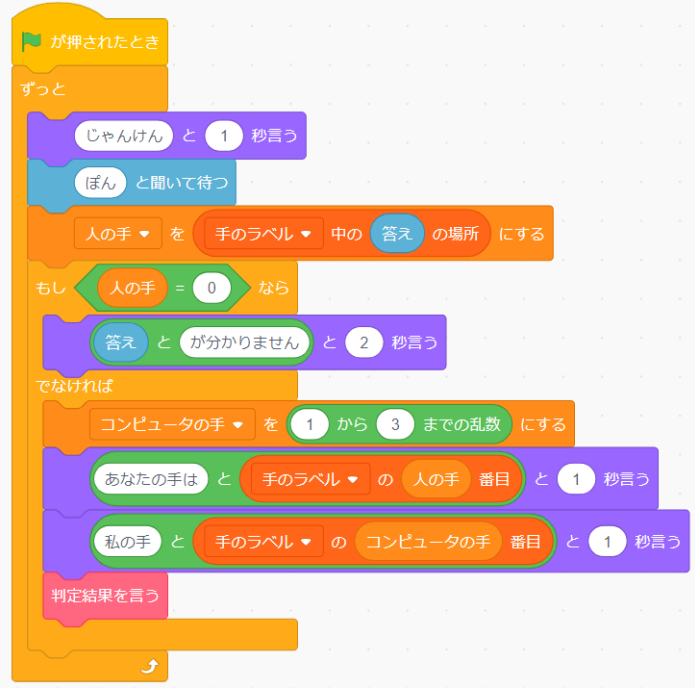


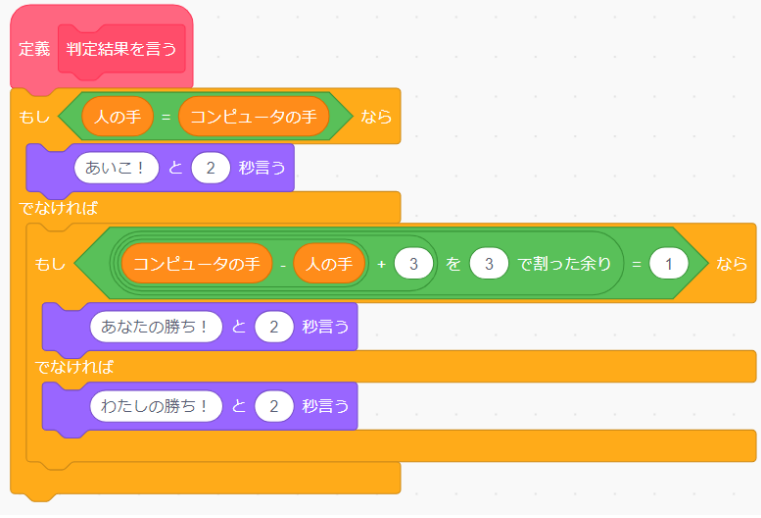
行動ログ

- 人(プレイヤー)の手の傾向を見るために，過去に出した手の記録 (ログ) を取る

どの箇所でログを取ればよいだろう?
どうやって取ればよいだろう?

リストを利用可能





リストの操作

- Scratchのリストは連結リストなので追加/挿入/削除が簡単

「テスト」という名前のリストを作りブロックの動作を確認しておこう

1. 100, 200, 300, 400 までブロックを使って順に追加
2. 1番目を123に置き換え
3. 2番目を削除
4. 2番目に789を挿入
5. いったん全て削除
6. 1から3の乱数を200個追加
7. いったん全て削除
8. 2から200までの偶数を追加
9. リストの要素の総和を計算

リストの操作

- リストはループ (繰り返し) と相性がいい

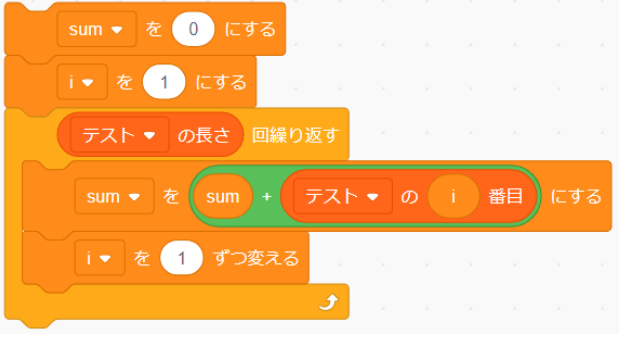
練習続き…

5. いったん全て削除
6. 1から3の乱数を200個追加
7. いったん全て削除
8. 2から200までの偶数を追加
9. リストの要素の総和を計算

5, 6

7, 8
変換「x」を
作成しておく

9
変換「i」「sum」
を作成しておく

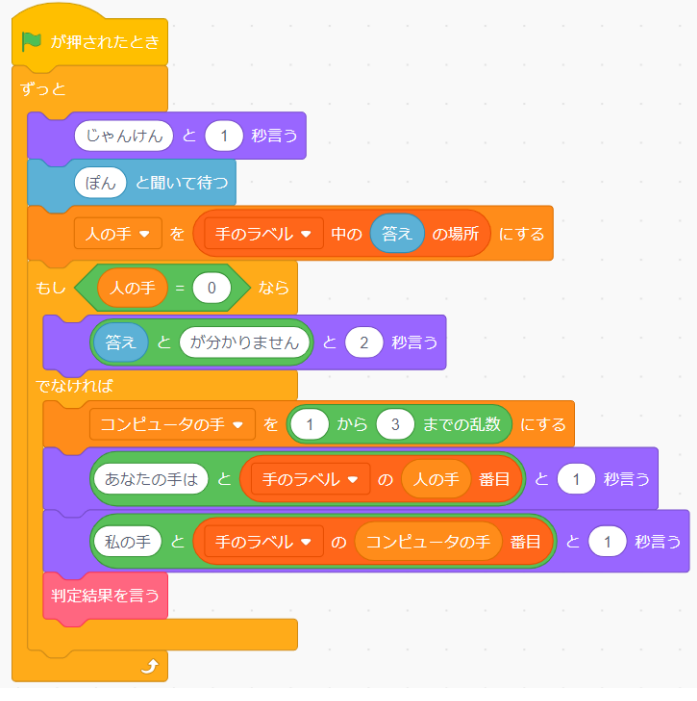


7, 8の別解

行動ログ

- 人(プレイヤー)の手の傾向を見るために，過去に出した手の記録 (ログ) を取る

どの箇所でログを取ればよいだろう?
どうやって取ればよいだろう?



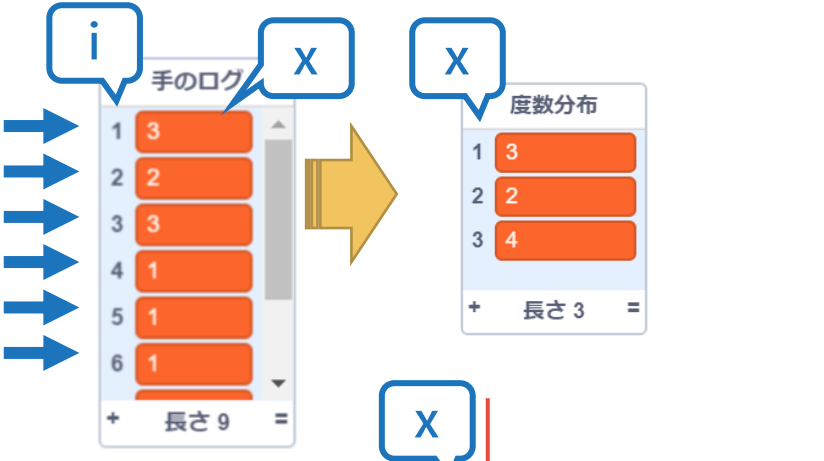
① 「手のログ」という名前のリストを作成

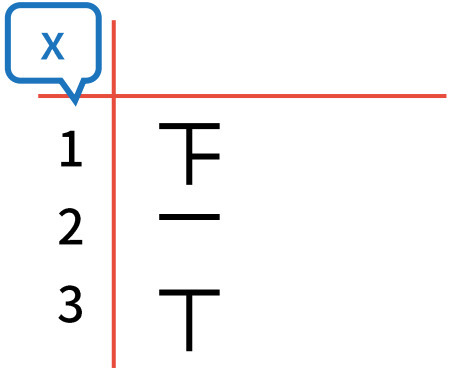
最初にログの初期化をするならこれを入れる(今は入れなくても可)

② 人の出した手をログに追加(リスト名注意)

度数分布

- 人(プレイヤー)の各手の出現回数を度数分布にまとめる
  - ログを順にたどって各手の回数を1ずつ増やしていく





初期化

最頻手 (最大値) を求める

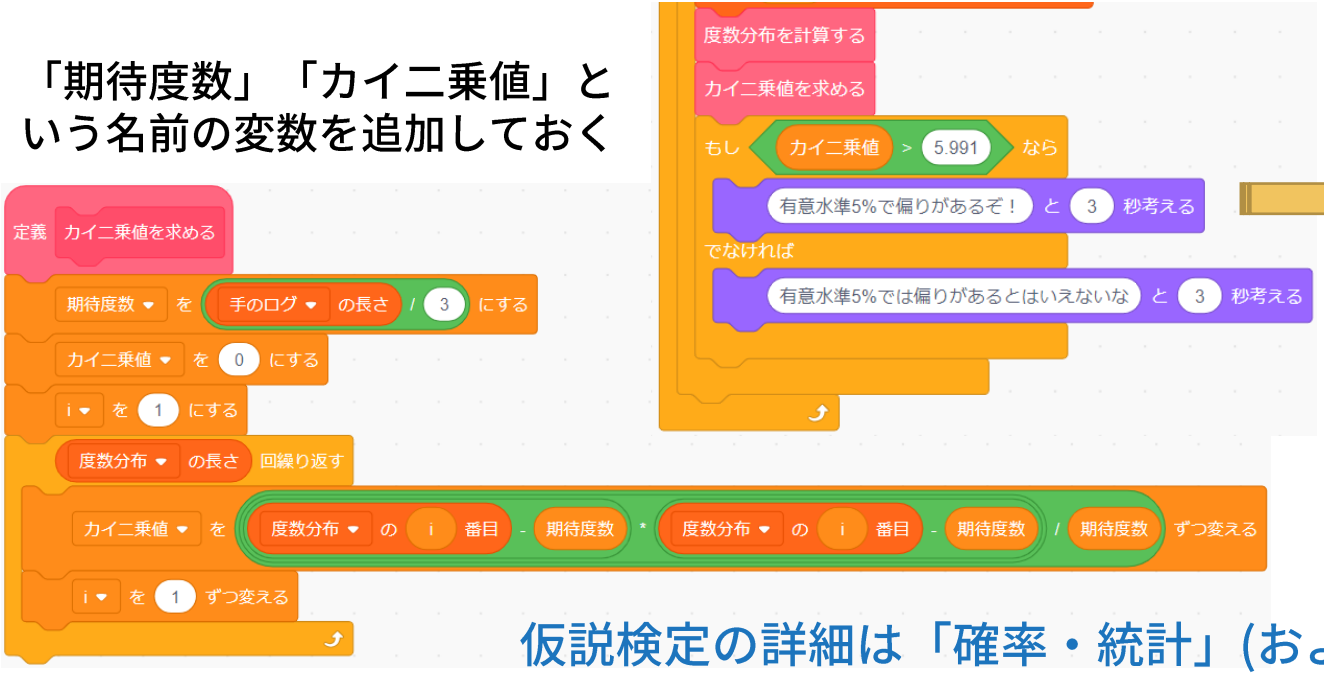
- 度数分布の最大値をとる手が最頻手
- 最大値を見つけるアルゴリズム (手順)
  - 変数max を(リストの値より小さい)-1 とする
  - リスト (度数分布)の要素を順にたどり
    もし要素の値が max よりも大きければ
    max にその値を入れる (その時点での最大値)
  - リストのすべての要素をたどれば終了

様々なアルゴリズムが「アルゴリズムとデータ構造」で紹介

仮説検定を行う

- 適合度検定で想定と異なる分布を見分ける
  - (例) 「グー: 1/3, チョキ: 1/3, パー: 1/3」
  - 帰無仮説: 全て同じ確率 1/3，対立仮説: 偏りがある

「期待度数」「カイ二乗値」という名前の変数を追加しておく



この時だけ偏りを考慮して手を決める?

仮説検定の詳細は「確率・統計」(および「統計学」)にて

様々な拡張

- 簡単な拡張
  - じゃんけんの手をアイコンや絵で表示
  - 勝ち/あいこ/負けの回数を変数で記録
  - 勝率を表示
- やや高度な拡張 (このあたりから Scratch だとやや面倒)
  - 可視化
    - 度数分布をヒストグラムで表示
    - 1つ前の手と現在の手の相関を散布図で表示
  - 機械学習
    - 過去2〜3回の手に対して次の手がどう決まるか傾向を学習
    - 勝負結果によって出す手がどう変わるかを学習

プログラムの保存方法

- ファイル > コンピュータに保存する
  - 「Scratchのプロジェクト.sb3」というファイルに保存されます
- 次回続けるとき：ファイル > コンピュータから読み込む
  - 保存したファイルを選択してください

- もし，オンライン版を「サインイン」して使っている場合は，オンライン上に保存することも可能です