
プログラムの作り方と繰り返し処理
基盤ユニット
PROCESSING FRIENDS

今週のクイズ

今週のクイズは紙での配付はありません。

http://www.sato-lab.jp/imfuからダウンロードして下さい。

先週やったこと

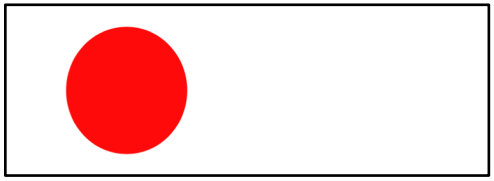
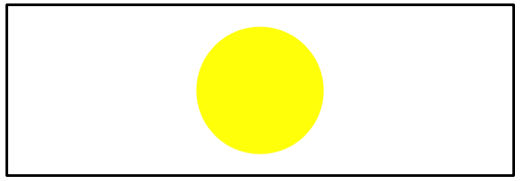
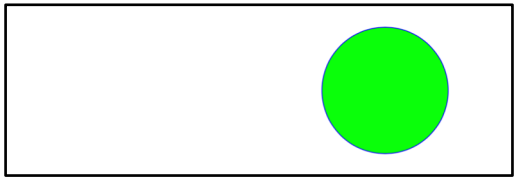
分岐処理

条件文

mouseX <= width/2
mouseX == 100
&&,||,!

if命令

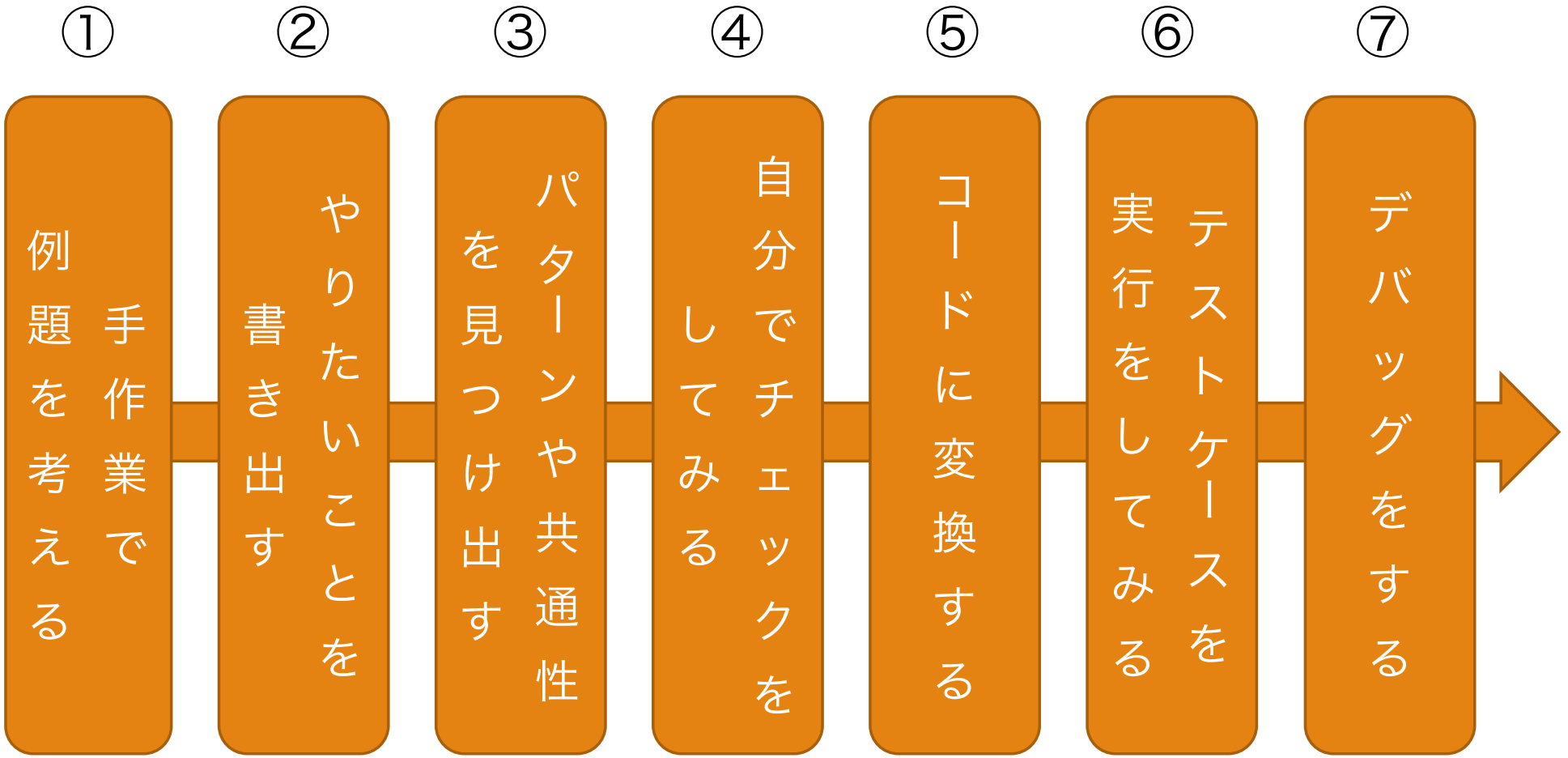
if(条件式){・・・}
if(条件式){・・・}else{・・・}

プログラムの構成要素

逐次処理	前々回までやっていた、 上から順番に命令を実行
条件分岐処理	前回にやった条件に応じて 実行する命令を選ぶ
繰り返し処理	同じ命令（の塊）を 繰り返し実行
処理関数	処理の塊に名前をつけて、 その名前で処理を行う

プログラムを作るための7つのステップ



①手作業で例題を考える

手作業で例題を考える

小さな（簡単な）例題を
手で解いてみる

状況を表す図（絵）を
描いてみる

問題や状況は
明確になったか？

関連知識が必要かも？

②やりたいことを書き出す

やりたいことを書き出す

やりたいことの手順を
きちんと書き出す

③パターンや共通性を見つけ出す

パターンや共通性を見つけ出す

それぞれの場合のアルゴリズムを考える

パターンや共通性を見つけ出す

条件分岐処理、繰り返し処理、変数などが利用出来ないかを考える

このステップが難しければ、①や②に戻って考える

別な入力や状況ではどうようになるか？

④自分でチェックをしてみる

自分でチェックをしてみる

考えもらしたパターンや状況がないかを考える

他の入力や状況でのチェックをする

⑤コードに変換する

コードに変換する

考えたアルゴリズムを
コードに変換する

この授業では
Processingを使う

①〜④までの処理は他の
言語を利用する場合でも
同じ

⑥テストケースを実行をしてみる

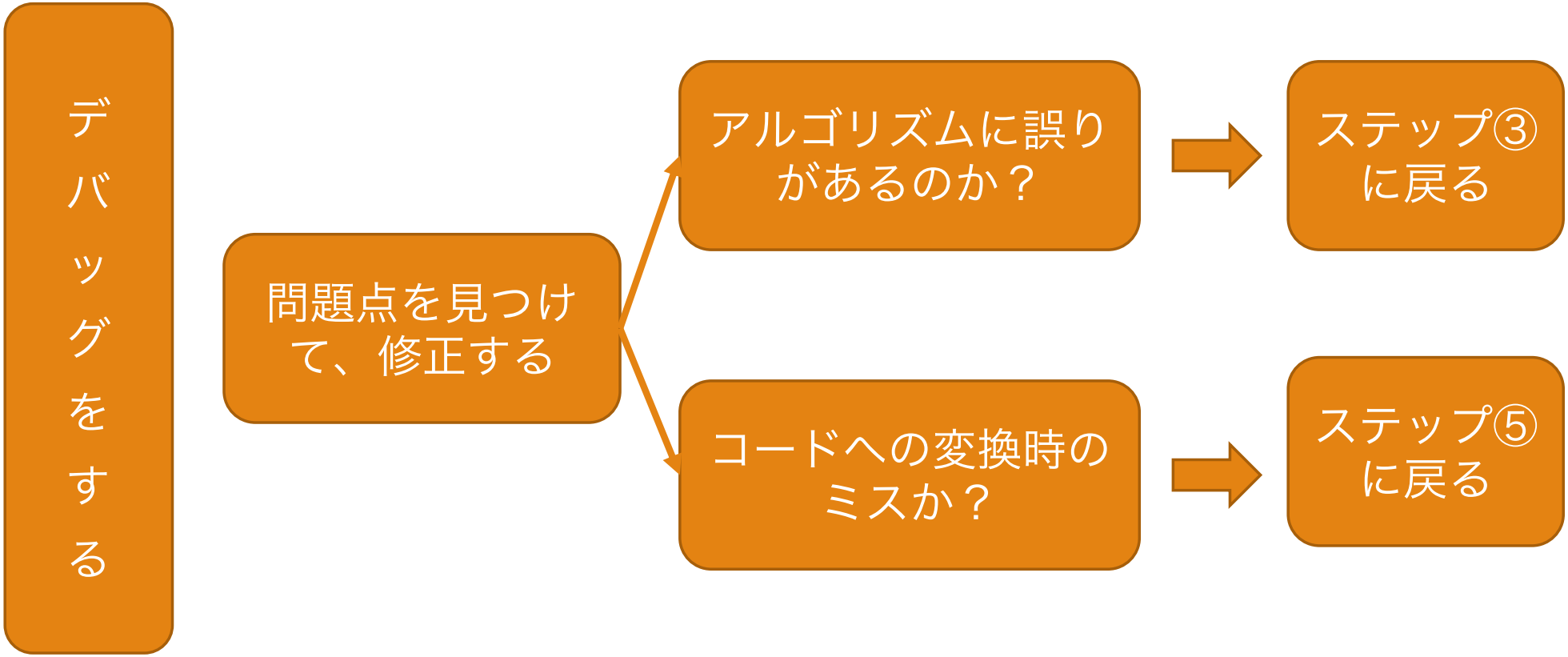
テストケースを実行をしてみる

プログラムを実行してみる

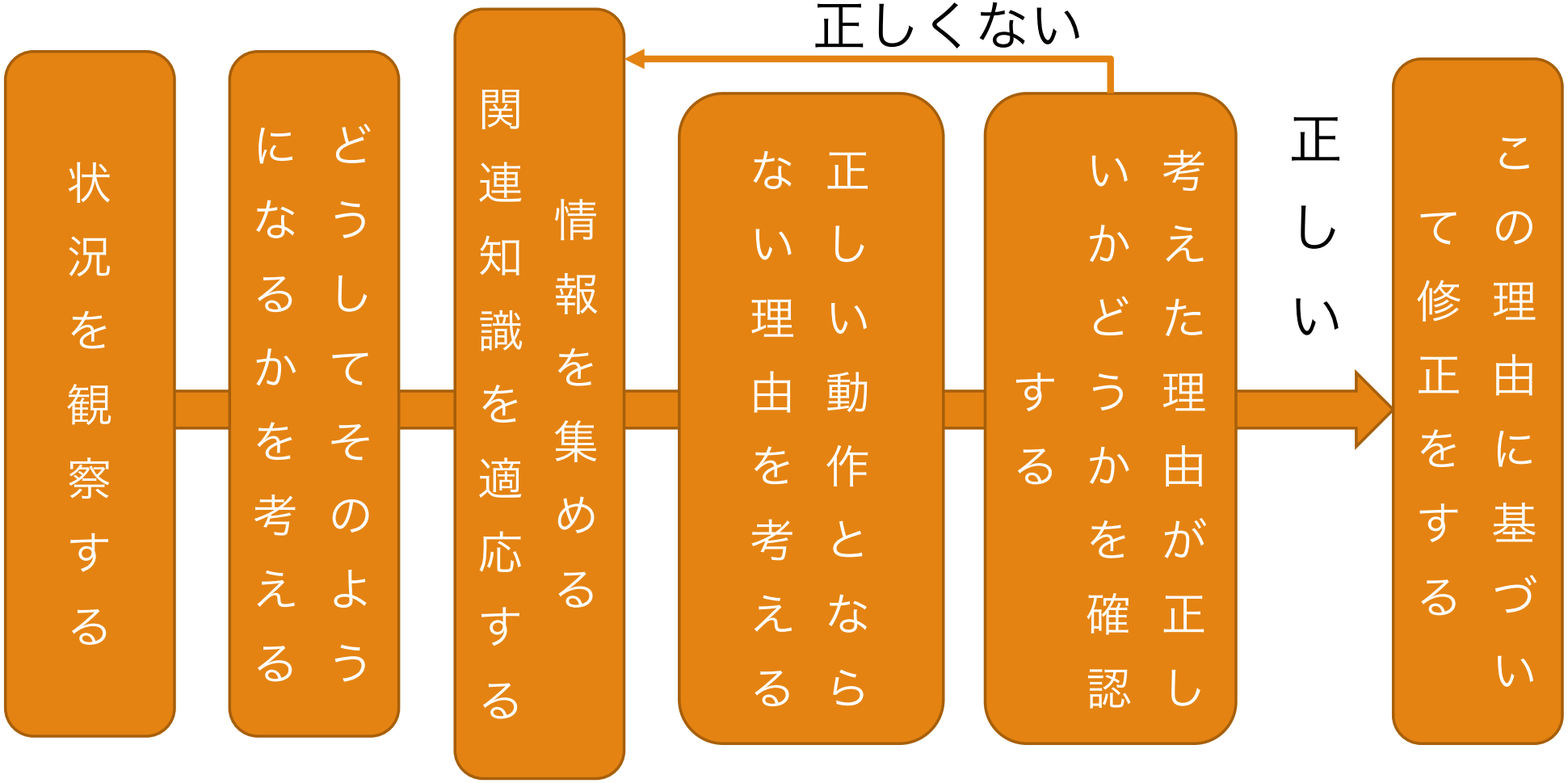
実行結果が正しいかを確認する

境界条件のテストを忘れない

デバッグをする


デバッグをする
問題点を見つけて、修正する
アルゴリズムに誤りがあるのか？
ステップ③に戻る
コードへの変換時のミスか？
ステップ⑤に戻る

デバッグの手順


状況を観察する
どうしてそのようになるかを考える
情報を集める
関連知識を適応する
正しい動作とならない理由を考える
考えた理由が正しいかどうかを確認する
正しくない
正しい
この理由に基づいて修正をする

情報を集める

関連知識を適応する

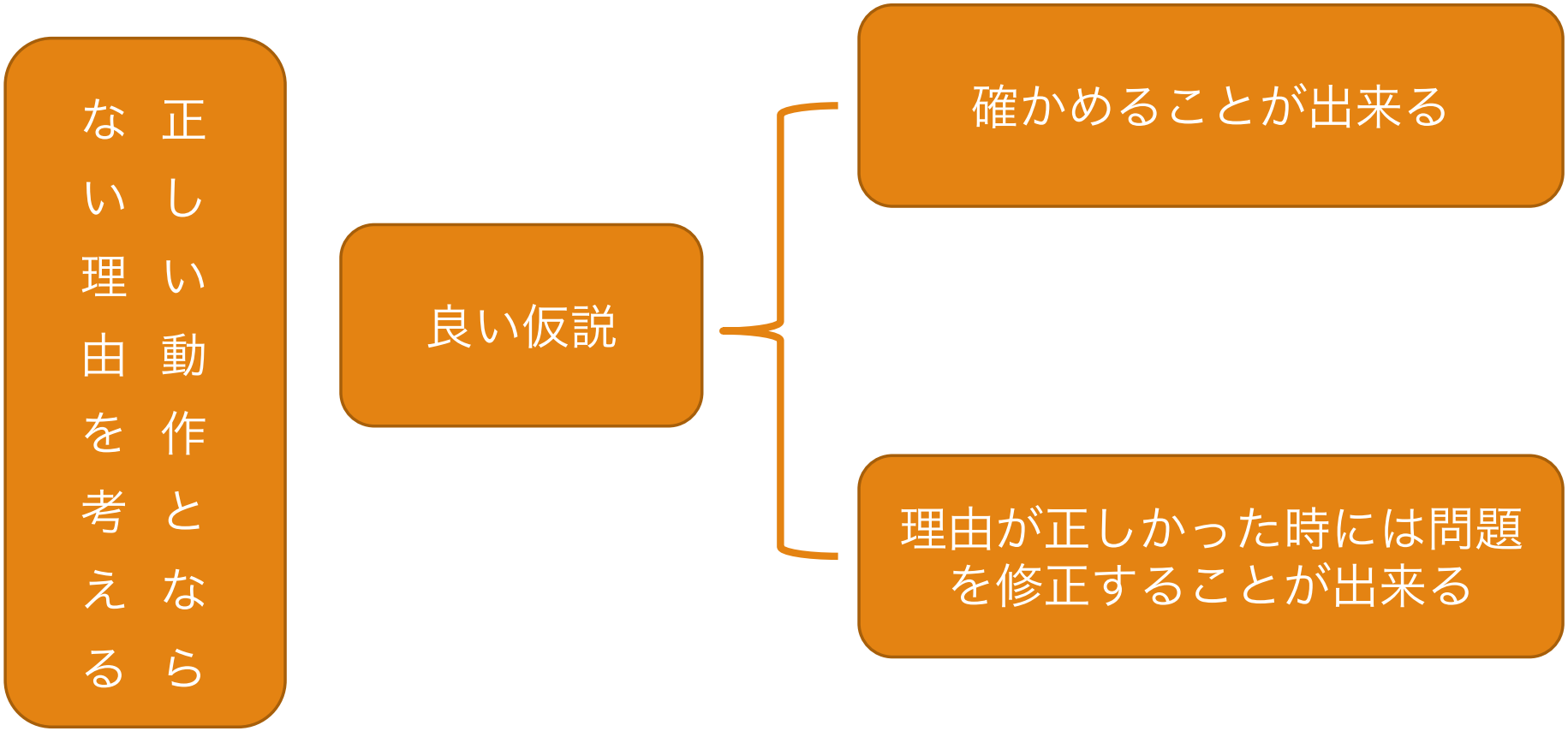
情報を集める

- println関数
- デバッグツールを使う
- コードの命令を自分で実行してみる

別の機会に紹介する予定

宿題のようにやってみる

正しい動作とならない理由（仮説）を考える



良い仮説、悪い仮説

悪い仮説

プログラムは間違っている

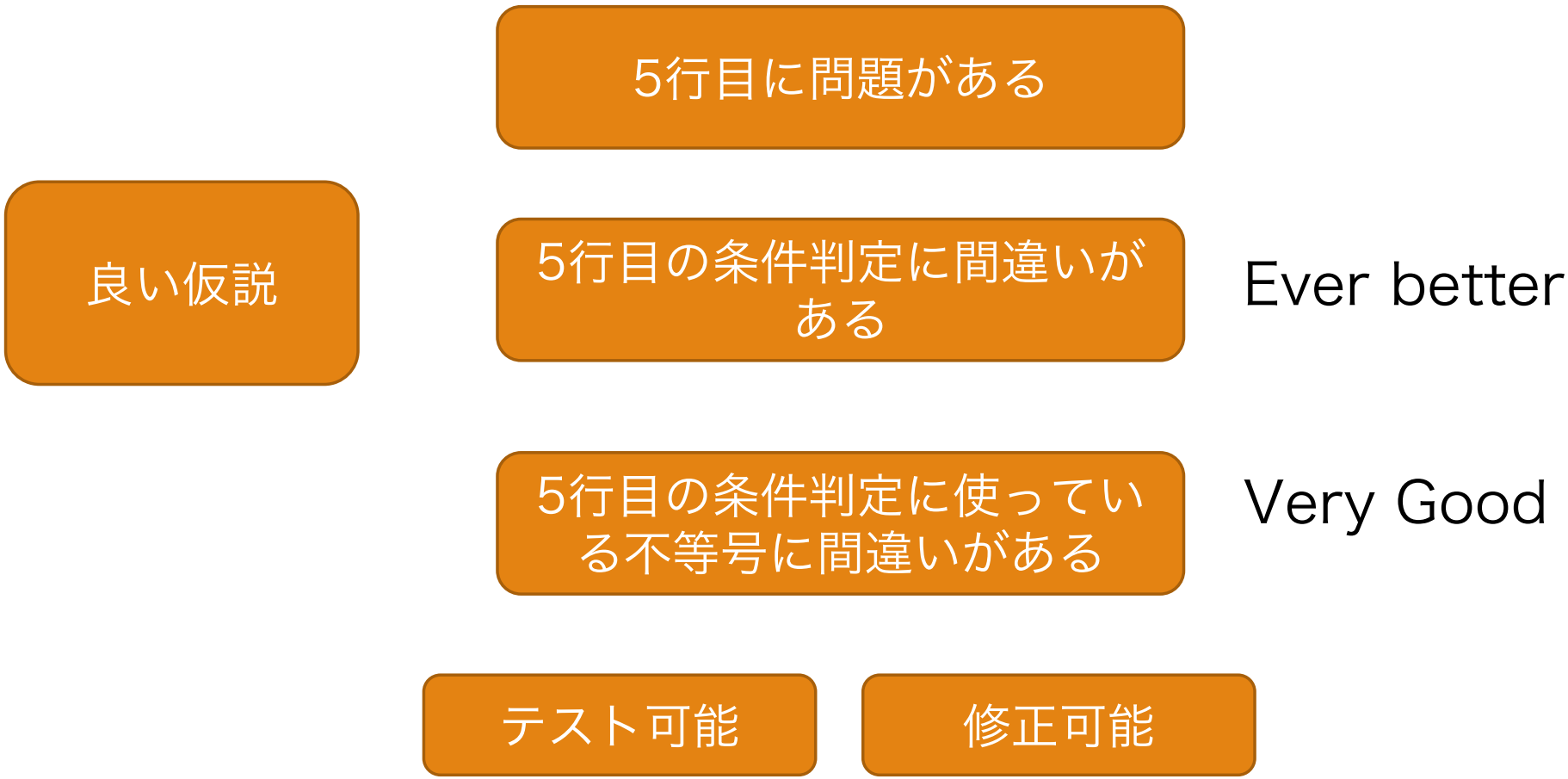
良い仮説

5行目に問題がある

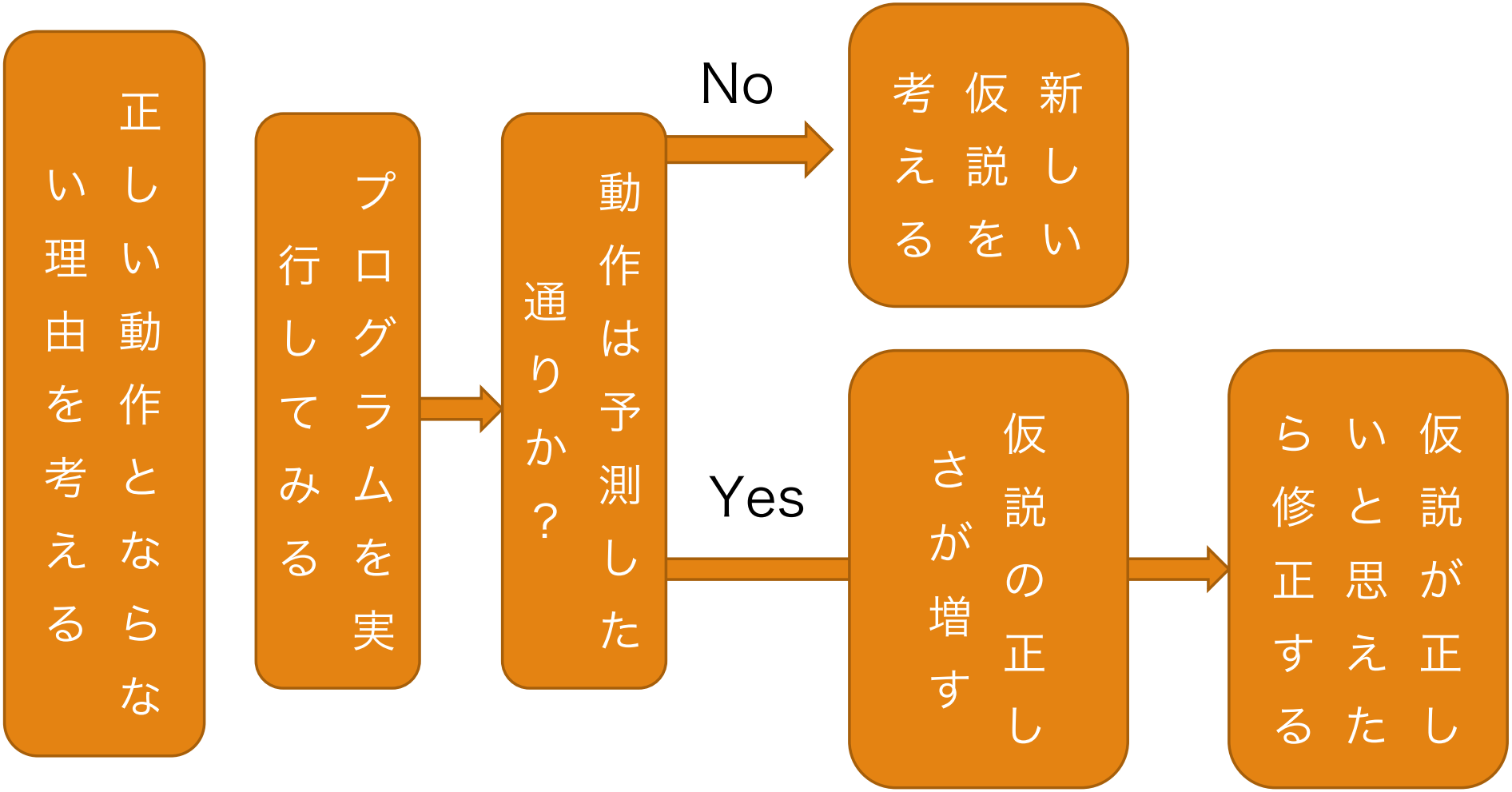
5行目の条件判定に間違いがある

Ever better

良い仮説、悪い仮説


良い仮説
5行目に問題がある
5行目の条件判定に間違いがある
Ever better
5行目の条件判定に使っている不等号に間違いがある
Very Good
テスト可能
修正可能

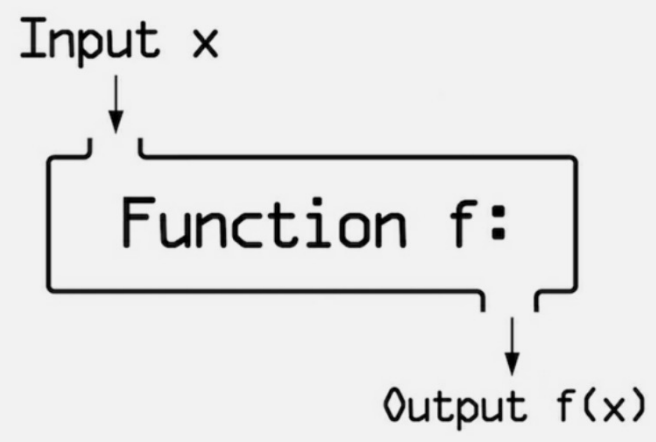
正しい動作とならない理由（仮説）を考える



関数

関数(function)　　メソッド(method)

まとまった処理の塊に名前をつける機能
つけた名前を利用して、処理を行うことができる



引数の数	関数の呼び出し方	例
引数が無い場合	関数名()	noStroke()
引数が1 つの場合	関数名(引数$_1$)	strokeWeight(3)
引数が2 つの場合	関数名(引数$_1$,引数$_2$)	size(600,400)
引数が3 つの場合	関数名(引数$_1$,引数$_2$,引数$_3$)	fill(255,10,10)
引数が4 つの場合	関数名(引数$_1$,引数$_2$,引数$_3$,引数$_4$)	ellipse(0,0.100,200)
以下続く		

乱数

乱数
でたらめな数を作り出す仕組み

random：関数

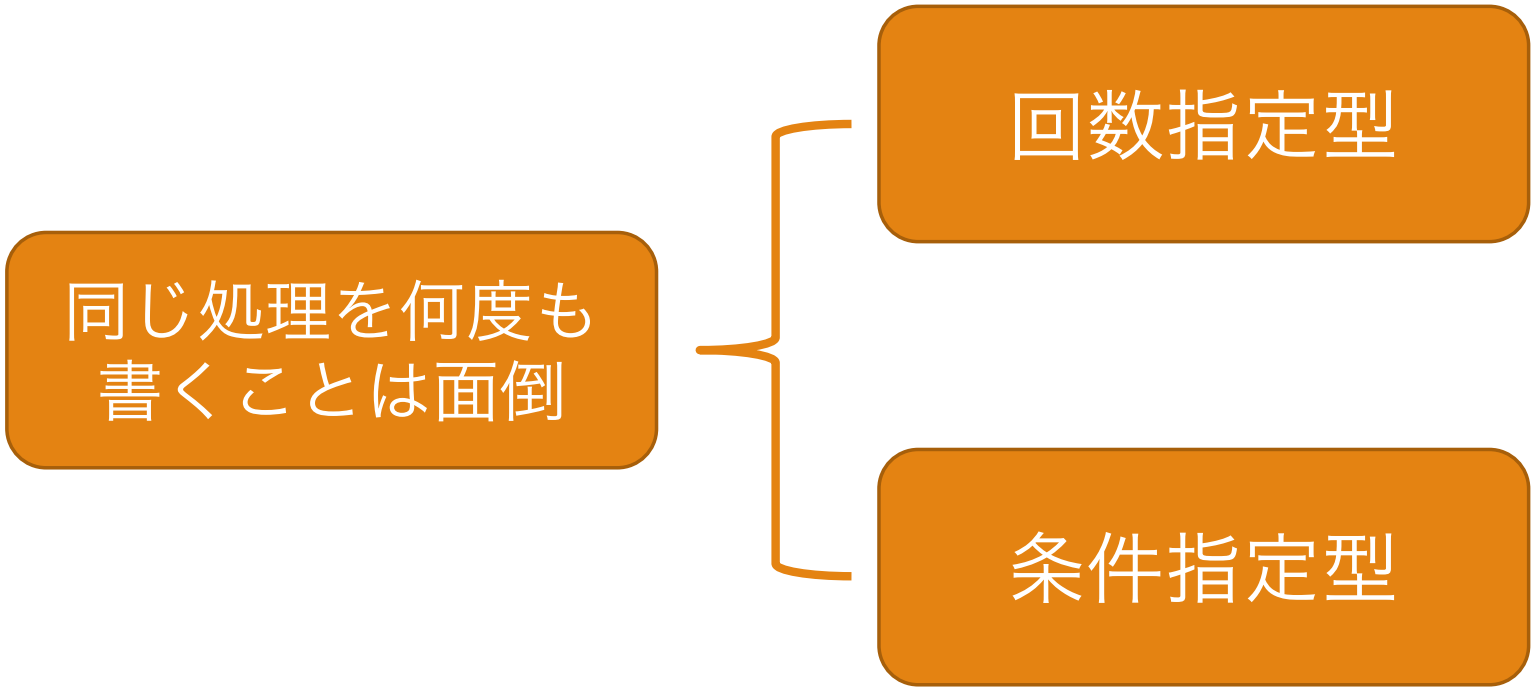
テキスト：45ページ

random（上限）：0以上上限未満の乱数
random(下限,上限)：下限以上上限未満の乱数

プログラムの構成要素

逐次処理	前々回までやっていた、 上から順番に命令を実行
条件分岐処理	前回にやった条件に応じて 実行する命令を選ぶ
繰り返し処理	同じ命令（の塊）を 繰り返し実行
処理関数	処理の塊に名前をつけて、 その名前で処理を行う

繰り返し処理



繰り返しパターンを見つ出す

```
float x,y;
size(400,200);
background(150);
fill(255);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
 :
 :
```

```
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
```

テキスト:48ページ、サンプル4-5

繰り返しパターンを見つ出す

```
float x,y;
size(400,200);
background(150);
fill(255);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
 :
 :
```

```
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
```

繰り返し回数は何回か？

回数指定型

48ページ目から

```
for(int i=0;i<10;i++){
  x = random(width);
  y = random(height);
  ellipse(x,y,20,20);
}
```

繰り返し回数の指定

カウンタ変数
繰り返し回数を
おぼえている変数

少し複雑な回数指定処理

カウンタ変数の利用を
利用して処理を行う

51ページ目から

繰り返しパターンを見つけ出す

```
size(300,200);
background(255);
stroke(0);

line(25,20,25,180);
line(50,20,50,180);
line(75,20,75,180);
line(100,20,100,180);
line(125,20,125,180);
line(150,20,150,180);
line(175,20,175,180);
line(200,20,200,180);
line(225,20,225,180);
line(250,20,250,180);
line(275,20,275,180);
```

```
size(300,200);
background(255);
stroke(0);

line( 0*25+25,20, 0*25+25,180);
line( 1*25+25,20, 1*25+25,180);
line( 2*25+25,20, 2*25+25,180);
line( 3*25+25,20, 3*25+25,180);
line( 4*25+25,20, 4*25+25,180);
line( 5*25+25,20, 5*25+25,180);
line( 6*25+25,20, 6*25+25,180);
line( 7*25+25,20, 7*25+25,180);
line( 8*25+25,20, 8*25+25,180);
line( 9*25+25,20, 9*25+25,180);
line(10*25+25,20, 10*25+25,180);
```

テキスト:51～52ページ

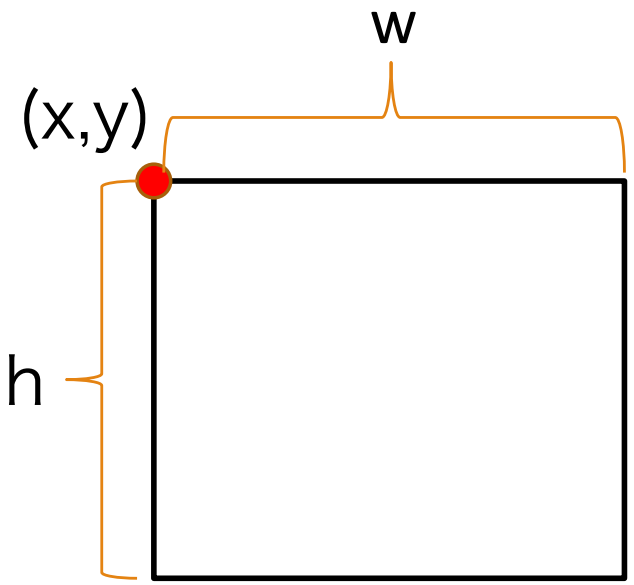
繰り返し処理の入れ子

forによる繰り返し処理の中に、
再びforによる繰り返し処理が来る

テキスト55ページから

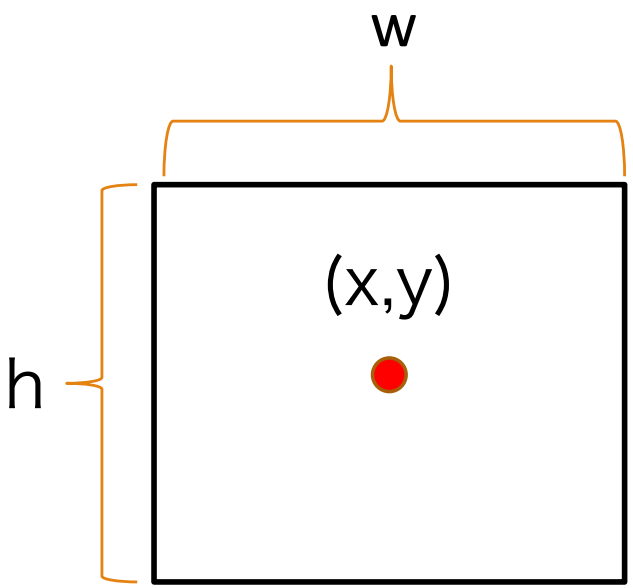
rect(x,y,w,h)

rectMode(CORNER)

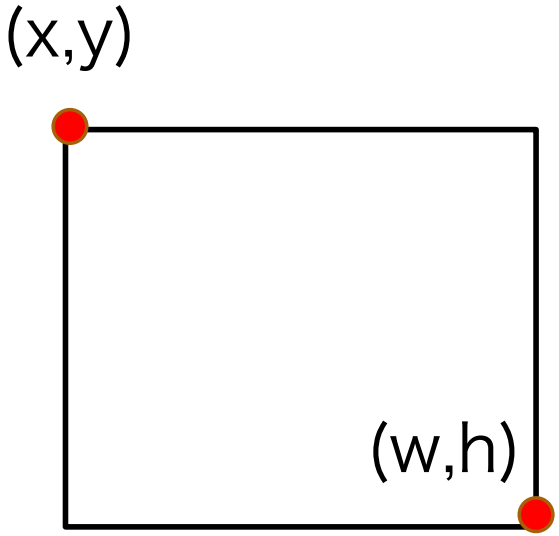


これがデフォルトの動作

rectMode(CENTER)



rectMode(CORNERS)



今日やった事

乱数:random関数

回数指定の繰り返し処理

rectMode関数

今週のクイズ

今週のクイズは紙での配付はありません。

http://www.sato-lab.jp/imfuからダウンロードして下さい。

授業時に配布した資料

http://www.sato-lab.jp/imfu/index.html

においてあります。