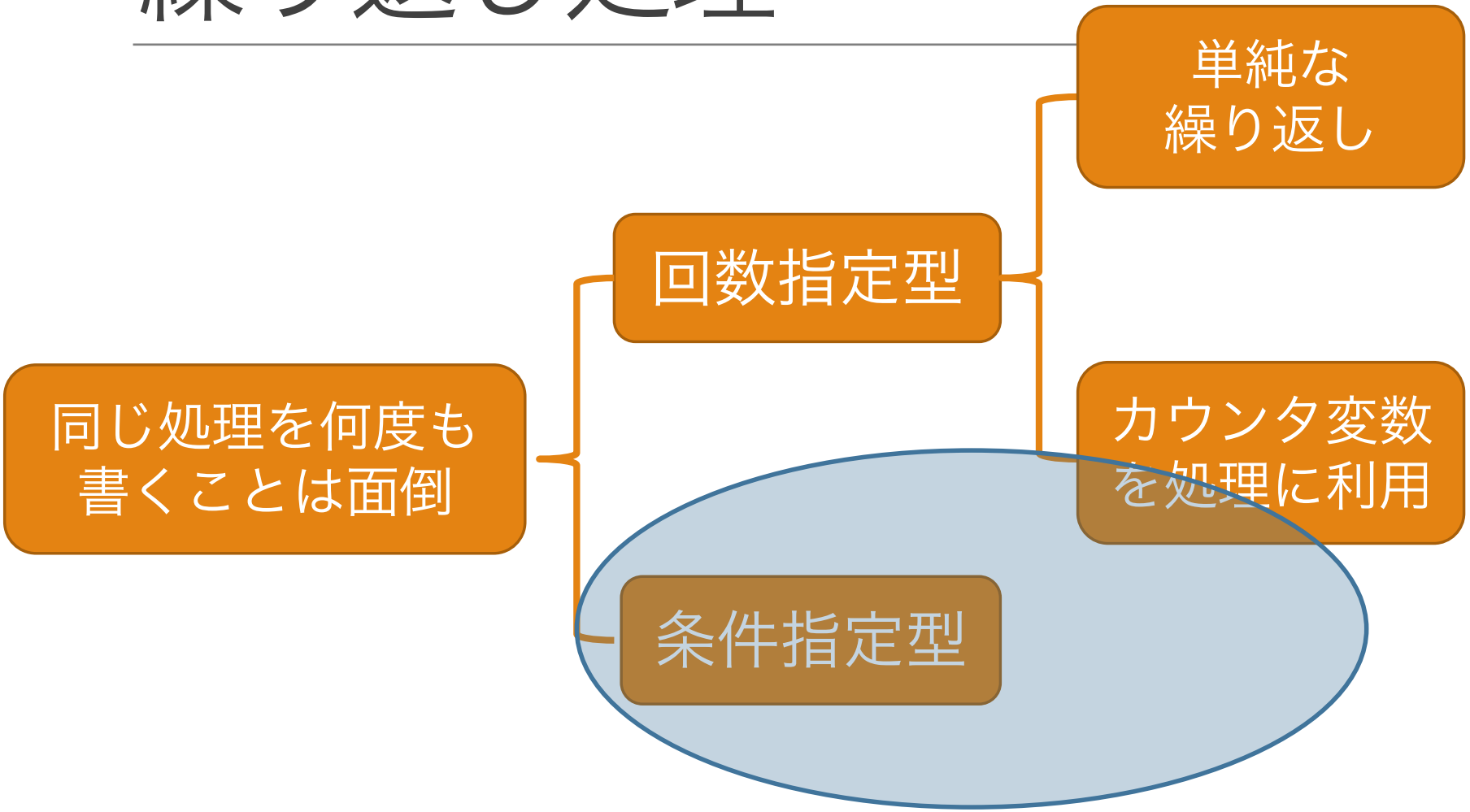

繰り返し処理
生物ユニット
PROCESSING FRIENDS

プログラムの構成要素

逐次処理	最初頃にやっていた、 上から順番に命令を実行
条件分岐処理	前々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
関数	処理の塊に名前をつけて、 その名前で処理を行う

前回の続き

繰り返し処理



繰り返し処理を使うためのこつ

繰り返しされているパターンを探し出す

```
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( 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);
```

Y=25X+25

少し複雑な回数指定処理

カウンタ変数を
利用して処理を行う

```
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);
```

```
for(int i=0;i<10;i++){
  line(i*25+25,20,
       i*25+25,180);
}
```

51～52ページ

繰り返しパターンの別の見方

```
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(  0+25,20,  0+25,180);
line( 25+25,20, 25+25,180);
line( 50+25,20, 50+25,180);
line( 75+25,20, 75+25,180);
line(100+25,20,100+25,180);
line(125+25,20,125+25,180);
line(150+25,20,150+25,180);
line(175+25,20,175+25,180);
line(200+25,20,200+25,180);
line(225+25,20,225+25,180);
```

1つ前のx座標の値に25を足している

繰り返しパターンの別の見方

```
line(  0+25,20,  0+25,180);
line( 25+25,20, 25+25,180);
line( 50+25,20, 50+25,180);
line( 75+25,20, 75+25,180);
line(100+25,20,100+25,180);
line(125+25,20,125+25,180);
line(150+25,20,150+25,180);
line(175+25,20,175+25,180);
line(200+25,20,200+25,180);
line(225+25,20,225+25,180);
```

```
x = 0;//線分のX座標値
x = x + 25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
以下繰り返す
line(x,20,x,180);
```

繰り返しパターンの別の見方

```
x = 0;
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
以下繰り返す
x = x+25;
line(x,20,y,180);
```

```
x = 0;//線分のX座標値
x = x+25;
line(x,20,x,180)
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
以下繰り返す
x = x+25;
line(x,20,x,180);
```

一番最後はxの値は250になっている。

繰り返しパターンの別の見方

```
x = 25;//線分のX座標値
line(x,20,x,180)
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
以下繰り返す
line(x,20,x,180);
x = x+25;
```

一番最後はxの値は275になっている。

繰り返しパターンの別の見方

```
x = 0;
for(int i=0;i<10;i++){
 x = x+25;
  line(x,20,x,180);

}
```

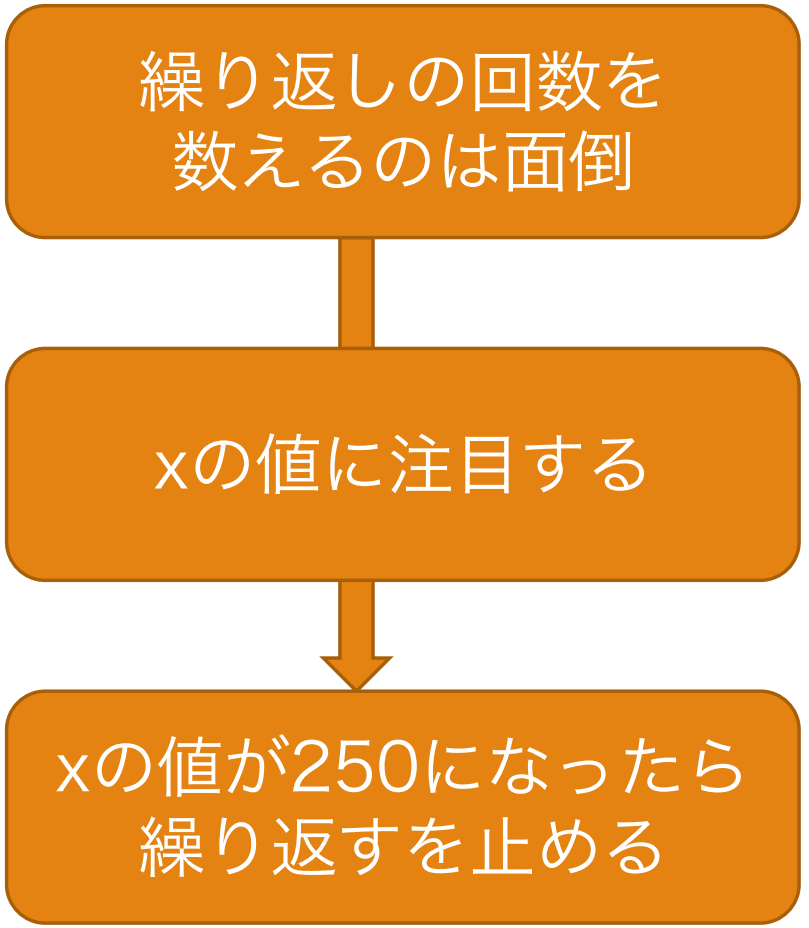
```
x = 25;
for(int i=0;i<10;i++){
  line(x,20,x,180);
  x = x+25;
}
```

```
x = 0;
for(int i=0;i<10;i++){
  line(x+25,20,x+25,180);
  x = x+25;
}
```

繰り返しパターンの別の見方

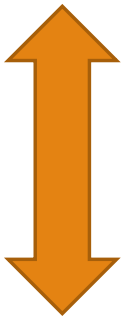
```
x = 0;//線分のX座標値
x = x+25;
line(x,20,x,180)
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
以下繰り返す
x = x+25;
line(x,20,x,180);
```

一番最後はxの値は250になっている。



停止条件

xの値が250になったら
繰り返すを止める



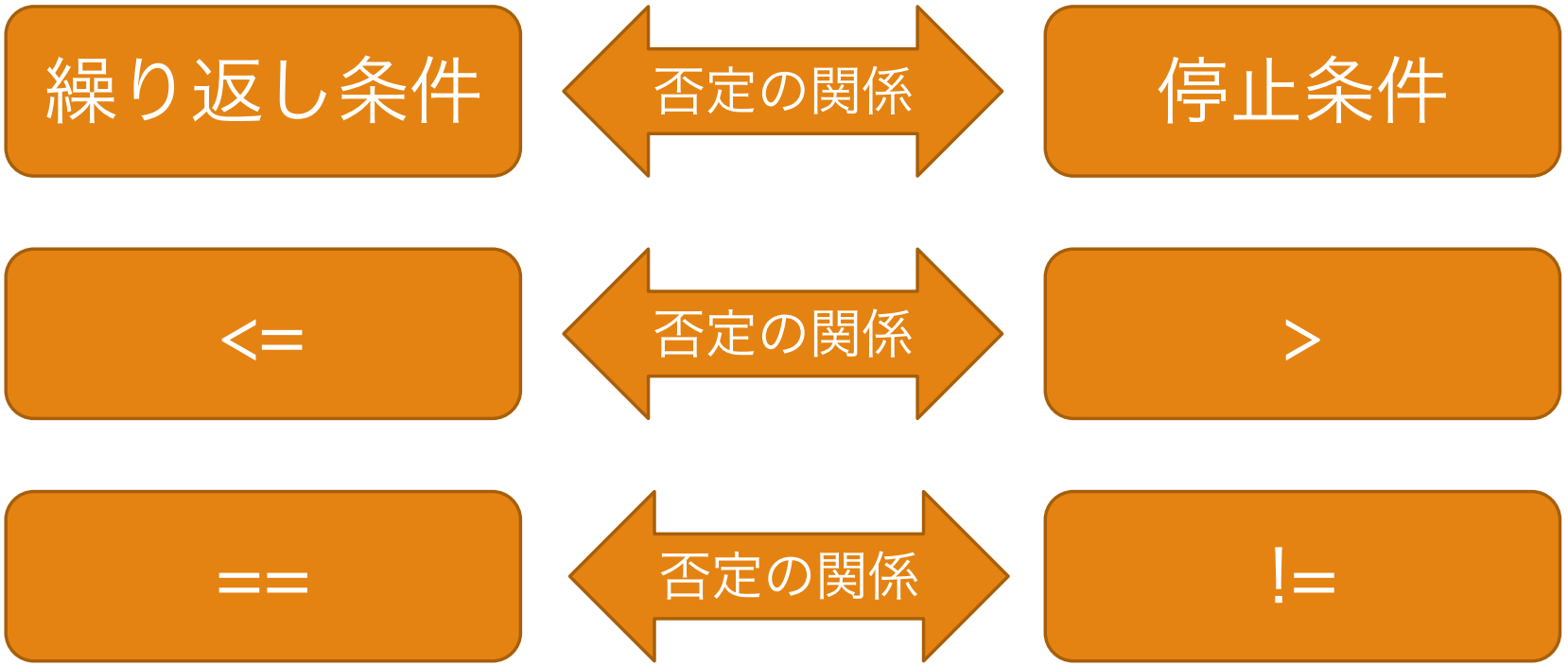
否定の関係

繰り返し条件

xの値が250でない時は
繰り返す

Processingでは繰り返し条件しか使えない

繰り返し条件と終了条件



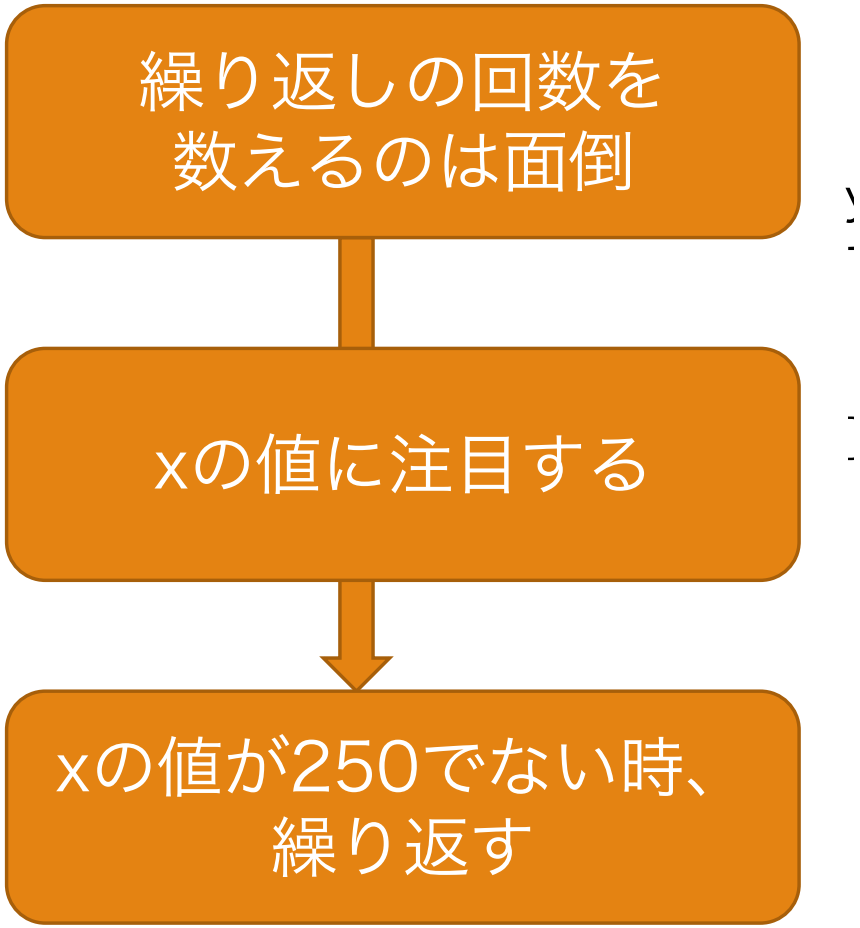
などなど

繰り返しパターンの別の見方

```
x = 0;//線分のX座標値
x = x+25;
line(x,20,x,180)
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
line(x,20,x,180);
x = x+25;
```

以下繰り返す

```
x = x+25;
line(x,20,x,180);
```



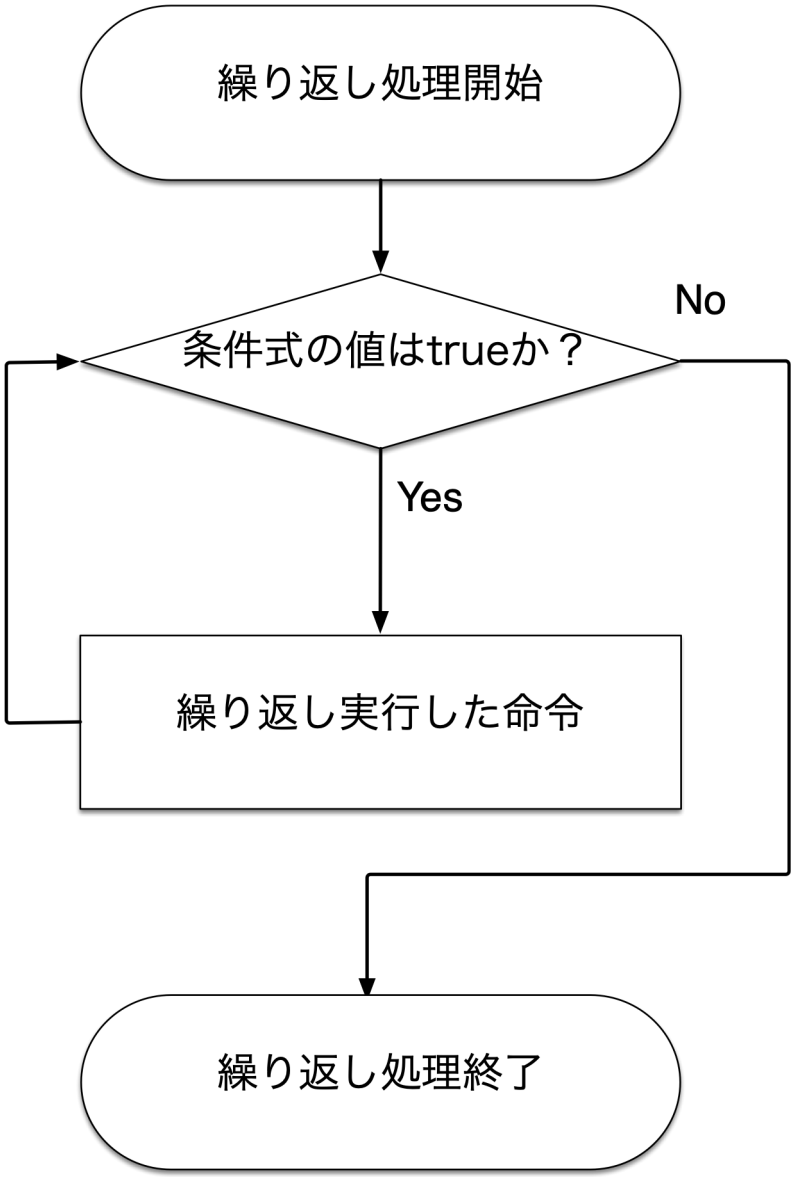
条件指定の繰り返し処理

while命令を使う

```
while(繰り返し条件){
    繰り返し処理内容
}
```

```
x = 0;
while(x != 250){
   x = x+25;
   line(x,20,x,180);
}
```

63〜ページ


繰り返し処理開始
条件式の値はtrueか？
No
Yes
繰り返し実行した命令
繰り返し処理終了

同じ動作をする プログラム3種

```
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);
```

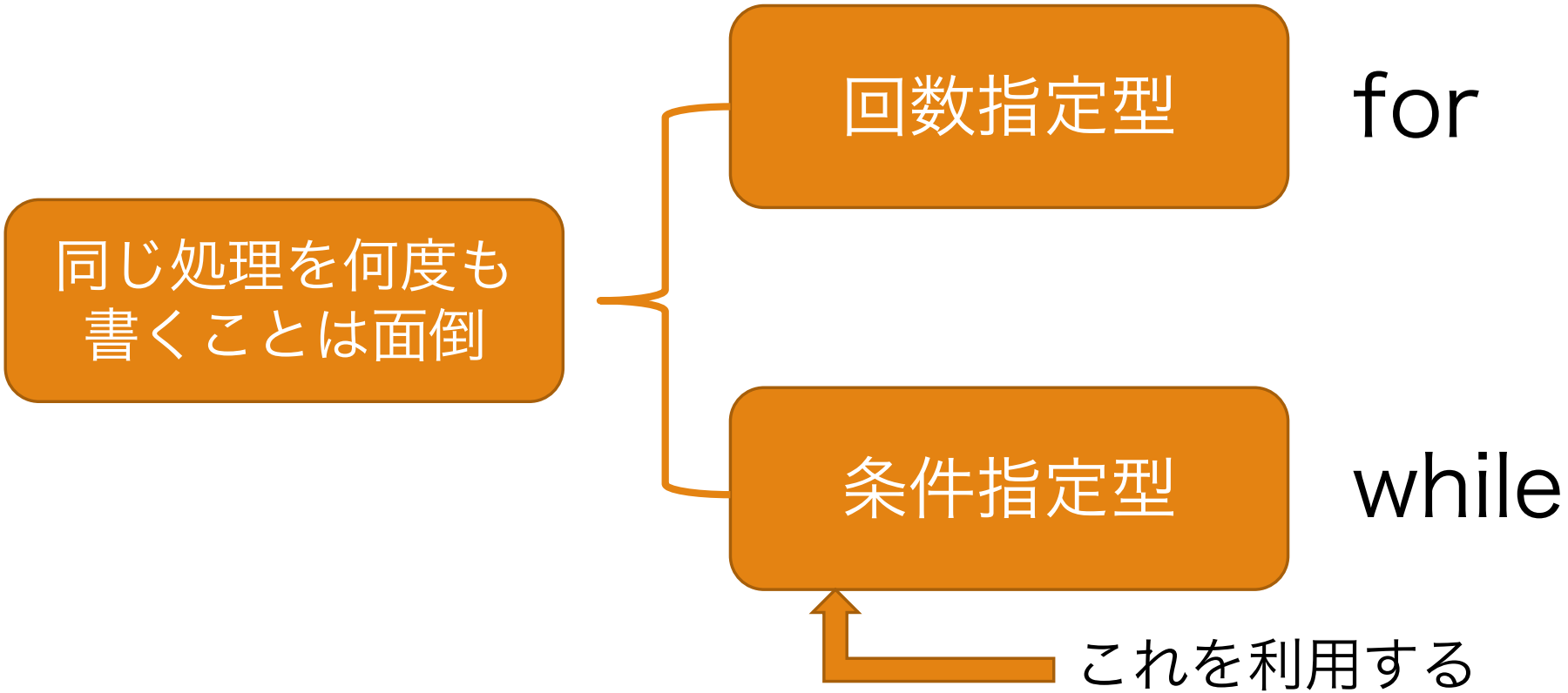
```
size(300,200);
background(255);
stroke(0);

for(int i=0;i<10;i++){
  line(25*i+25,20,
       25*i+25,180);
}
```

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

x = 0;
while(x != 250){
  x = x+25;
  line(x,20,x,180);
}
```

プログラム作成時には繰り返し回数がわからない場合



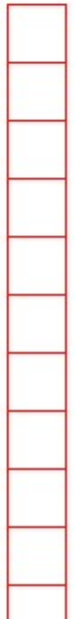
例えば、

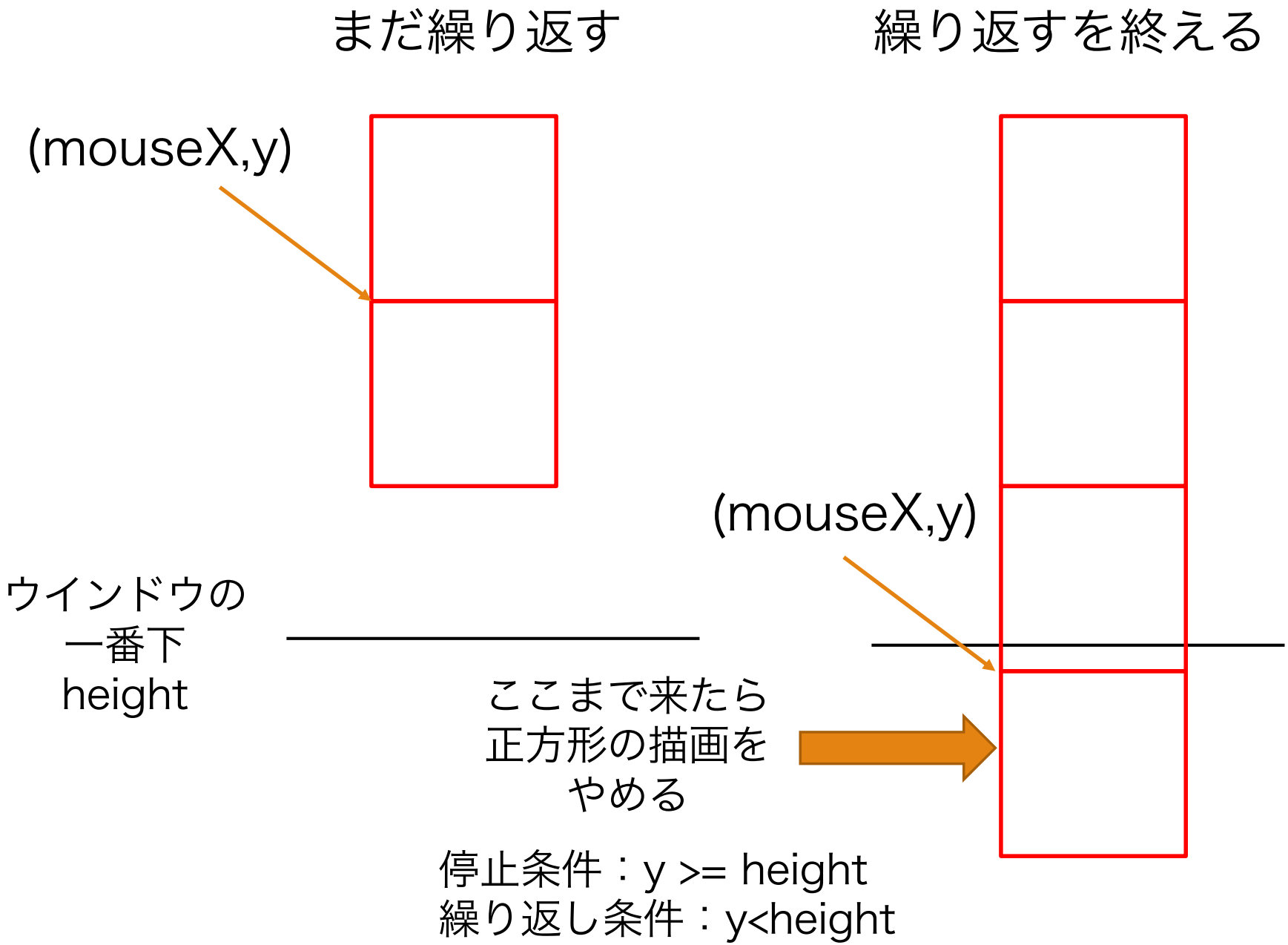
現在のマウスの位置から、下の方に向けて正方形を描く。ただし、塗りつぶしをしません。一辺の長さは30とする。

mouseYの値から正方形を描き始めて、描く**正方形**のY座標の値を30増やすことを繰り返しながら、**正方形**を描いていく。

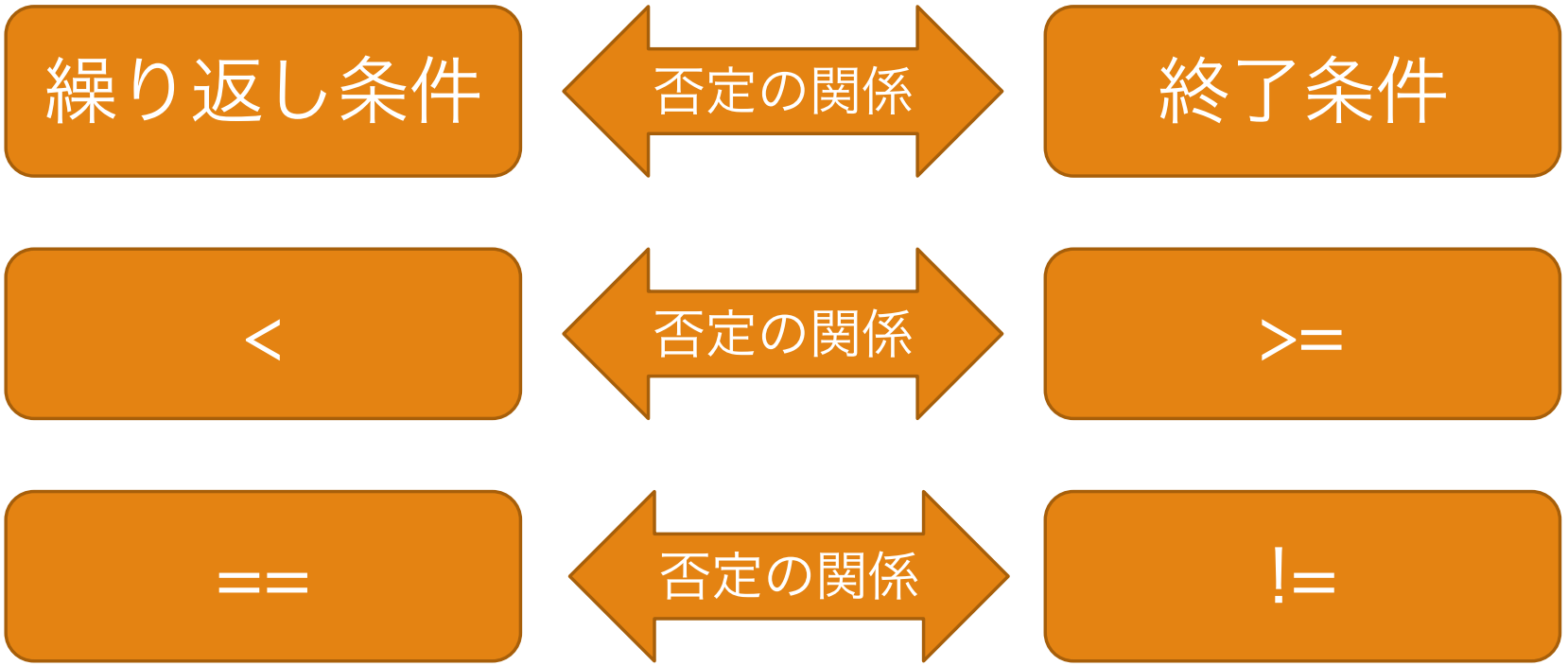
1. 描く正方形のX座標の値はmouseX
2. 描く正方形のY座標の値を変数yに保存する
3. 変数yの値は最初はmouseY
4. 正方形を描く
5. yの値を30増やす
6. 4と5を繰り返す

何回繰り返すの？




まだ繰り返す
繰り返すを終える
(mouseX,y)
(mouseX,y)
ウインドウの
一番下
height
ここまで来たら
正方形の描画を
やめる
停止条件：y >= height
繰り返し条件：y<height

繰り返し条件と終了条件



などなど

```
int y;

void setup(){
  size(400,400);
  background(255);
  stroke(255,10,10);
}

void draw(){
  noFill();
  y = mouseY;
  while(y < height){
   rect(mouseX,y,30,30);
   y = y + 30;
  }
}
```

```
int y;

void setup(){
  size(400,400);
  background(255);
  stroke(255,10,10);
}

void draw(){
  noFill();
  y = mouseY;
  while(y < height){
   rect(mouseX,y,30,30);
   y += 30;//yの値を30増やす
  }
}
```

複合代入演算子とインクリメント・デクリメント演算子

ある変数の値を増やしたり、減らしたりする処理を書くことが多いので簡易的な書き方が用意されている。

```
x = x + 25; //意味は変数xの値を25増やす
x += 25; //意味は変数xの値を25増やす
x -= 25; //意味は変数xの値を25減らす
x --; //意味は変数xの値を1だけ減らす
--x;
x++; //意味は変数xの値を1だけ増やす
++x;
```

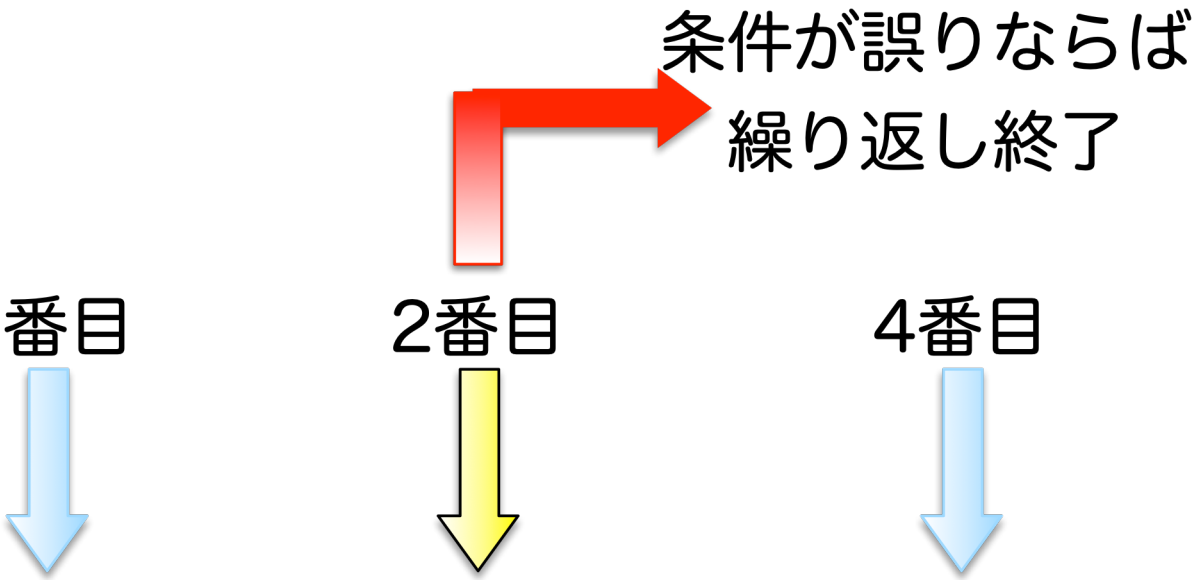
Processingのforは強力

```
int y;
void setup(){
  size(200,400);
}
void draw(){
  background(255);
  stroke(255,10,10);
  noFill();
  y = mouseY;
  while(y < height){
   rect(mouseX,y,30,30);
   y = y + 30;
  }
}
```

whileバージョン

```
int y;
void setup(){
  size(200,400);
}
void draw(){
  background(255);
  stroke(255,10,10);
  noFill();
  for(y = mouseY; y<height; y +=30){
   rect(mouseX,y,30,30);
  }
}
```

forバージョン



for(初期化;条件チェック;更新){

3番目：ここに書かれている命令を実行

その後、4番目を実行し、2番目に戻る

}

割り算/に注意

float：実数
- 1.0/2.0 -> 0.5
- 9.0/2.0 -> 4.5
- 9.0/2 -> 4.5
- 9/2.0 -> 4.5

int：整数
- 1/2 -> 0
- 9/2 -> 4
- 商を求める

A	B	A/Bの結果
int	int	int
int	float	float
float	Int	float
float	float	float

余りを使うとちょっと便利

%で余りが求まる	1 % 2 -> 1 4 % 2 -> 0
偶奇判定	2で割った余りが 0なら偶数、1なら奇数
倍数（約数）判定	kで割った余りが 0ならkの倍数
巡回する数	idx = (idx+1) % 6; 0->1->2->3->4->5->0->1……

演算子の優先順位

演算子の優先順位 → 数式のどの部分から先に計算すべきかの規則

1. 括弧の中
2. かけ算と割り算
3. 足し算と引き算

1+2*3

(1+2)*3

優先順位	演算子	説明
1	[] . ++ --	配列要素、メンバへのアクセスなど
2	! ~ - +	前置の単項演算子など
3	* / %	乗法、除法、剰余
4	+ -	加法、減法
5	<< >>	ビット単位のシフト
6	< <= > >=	大小比較
7	== !=	等価、非等価比較
8	&	ビット単位の論理積
9	^	ビット単位の排他的論理和
10	|	ビット単位の論理和
11	&&	かつ
12	||	または
13	= += -= *= /= %= &= |= ^= <<= >>=	代入、複合代入演算子

演算子の優先順位を考慮すると

mouseX + 10 < width/2

(mouseX + 10) < (width/2)

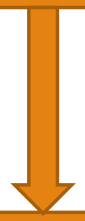
width/3 < mouseX && mouseX < 2*width/3

((width/3)< mouseX) && (mouseX < (2*width/3))

演算の順番に自信が無いときは

積極的に括弧（）を使う

width/3 < mouseX && mouseX < 2*width/3



(width/3 < mouseX) && (mouseX < 2*width/3)

今日の授業でやった事

条件指定型の
繰り返し処理

繰り返し処理が最初の難関です。
頑張って、理解して下さい。

whileとforに関して、
きちんと理解すること！！

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。