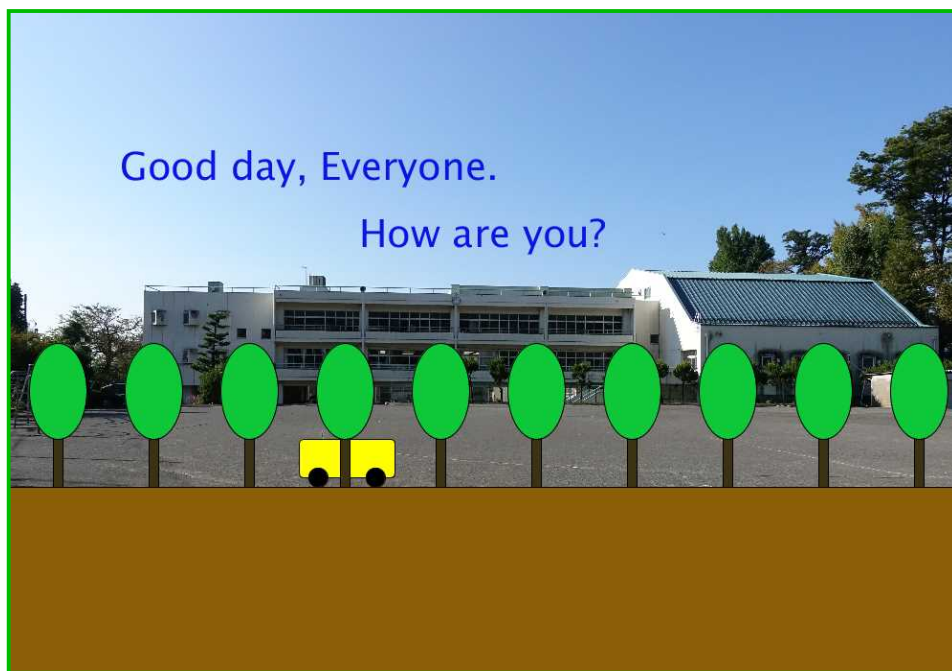


Processing

第4回



松田小学校／寄小学校

4 ステップ 0 : 前回の復習だよ

次のコードを打ってみよう。どんな図形になるかな？

```
1 size(600, 400);  
2 fill(0, 0, 255);  
3 rect(150, 100, 300, 200, 10);
```

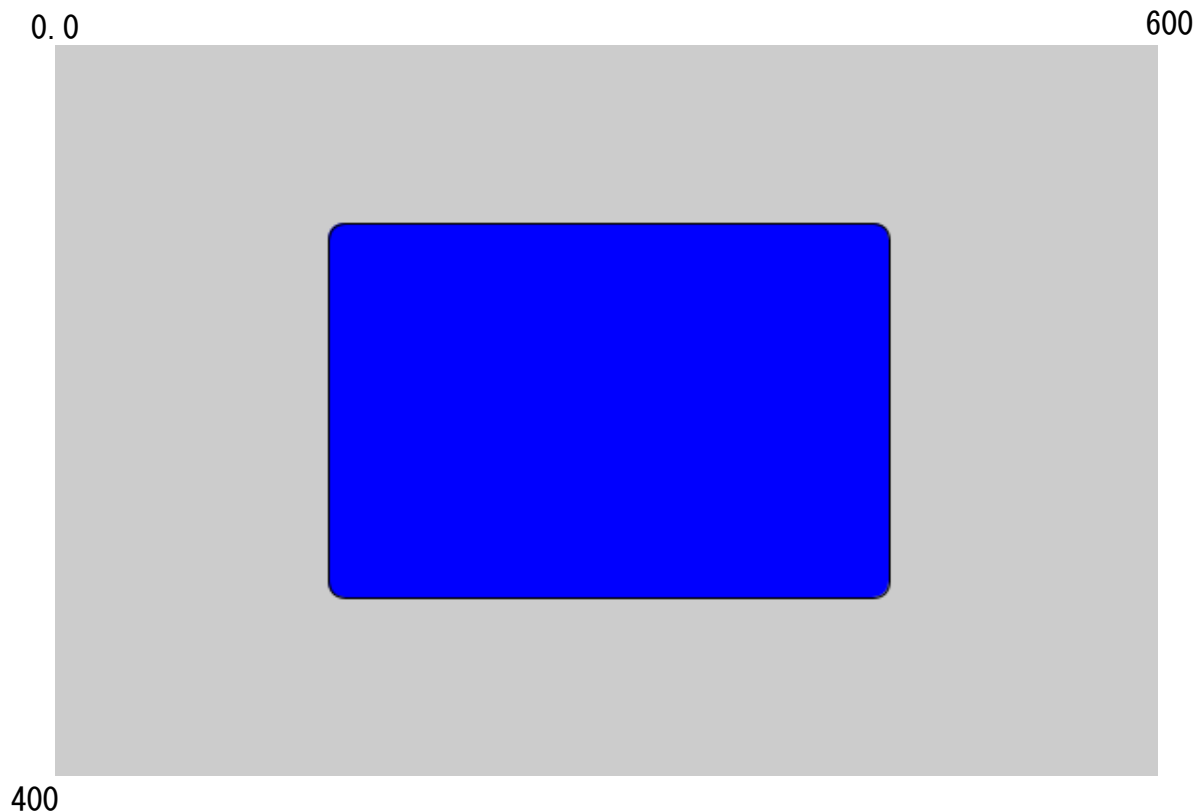
上のコードに下の2行目を加えて実行してみよう。

```
5 textSize(50);  
6 text("XXXXXX", 200, 200);
```

← XXXXXX には、あなたの名前をローマ字で入れてね。

ミッション

あれ！ あなたの名前が見えないね。どうしたら見えるようになるだろうか。
4行目に何かコードを加えて、見えるようにしてみよう。



4 ステップ 1 : 変数 (=アルファベット) を使ってみよう

ファイルから新規を開いて、下のコードを打って実行してみよう。

```
1 size (600, 600);
2 rect (100, 100, 80, 50);
3 rect (100, 200, 80, 50);
4 rect (100, 300, 80, 50);
```

AをBへ移動するには、空欄には何を？

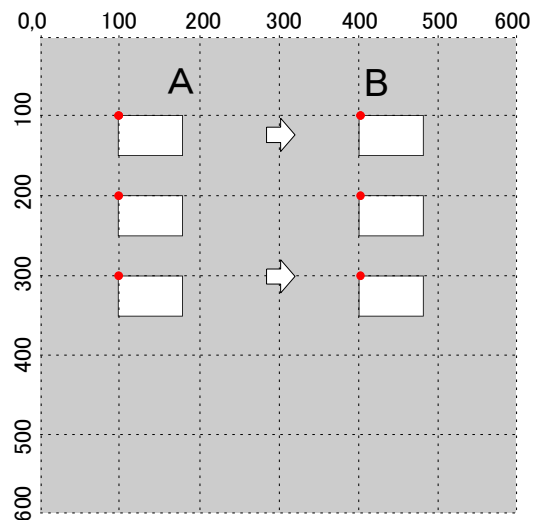
```
1 size (600, 600);
2 rect ( , 100, 80, 50);
3 rect ( , 200, 80, 50);
4 rect ( , 300, 80, 50);
```

もし、四角が100コあったら！！

```
rect (100, 100, 80, 50);
rect (100, 200, 80, 50);
rect (100, 300, 80, 50);
rect (100, 400, 80, 50);
//
//
rect (100, 10000, 80, 50);
```

下のようにX座標の値を全部変えなきゃ！！

```
rect (400, 100, 80, 50);
rect (400, 200, 80, 50);
rect (400, 300, 80, 50);
rect (400, 400, 80, 50);
//
//
rect (400, 10000, 80, 50);
```



100個
移動させるのは
大変だ...



こんな時は、変数を使うと簡単になるよ。

最初に変数の宣言をすれば、プログラムの中で使うデータや値を保存するための**変数**(=箱)を使えるんだ。変数にはアルファベットなら何でも何文字でもOK！

宣言 `int a;` とか `int matsuda;` でもOK
イントって読むよ

`int` は `integer` の省略形で、整数という意味だ。
インテジャーって読むよ

```
1 int a;
2 a=100;
3 size (600, 600);
4 rect (a, 100, 80, 50);
5 rect (a, 200, 80, 50);
6 rect (a, 300, 80, 50);
```

変数 a を使う宣言

X 座標の値(= a の値)

```
1 int a=100;
```

1行に合体してもOKだ。

ミッション

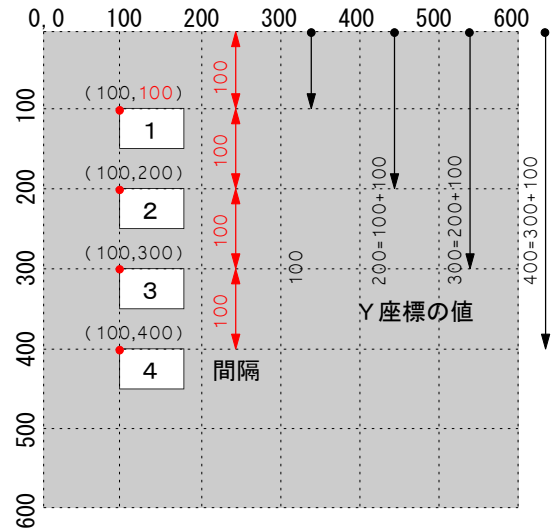
- ① 2行目の `a = 100;` を `a = 300;` や `a = 500;` にして実行してみよう。
- ② 四角形の横幅に変数 `b` を使って、`int b=50;` 宣言で実行してみよう。 `b=80;` では？

4-ステップ 2 : 変数の応用だ=繰り返し

```

1 int a; ← 変数 a を使う宣言
2   a=100; ← 右図 1 の X 座標
3 size (600, 600);
4 rect (a, 100, 80, 50);
5 rect (a, 200, 80, 50);
6 rect (a, 300, 80, 50);
7 rect (a, 400, 80, 50);

```



上のコードのY座標は、初めが 100 で 100 ずつ大きくなる。間隔が同じだね。間隔が同じという規則性がある場合は、繰り返しのコード `for(~)` を使うと簡単になるよ。

新規を開いて、下のコードを打って実行してみよう。コンソールにでたかな？

```

1 int i; ← 変数 i を使う宣言
2 for (i=100; i<=400; i=i+100) { ← for(~) の意味は、i を
3   println(i); ← 変数 i の値をコンソール 100 から 400 まで、
4 } ← =(黒い部分)に打ち出せ 100 を加えながら繰り返せ。

```



for(~) の各意味は、下の通りだ。

for	繰り返しの開始記号
(i = 100;	変数 i を 100 (=初期値)から
i <= 400;	変数 i が 400 以下なら { }内を実行する
i = i+100)	{ }内の i に 100 を足して更新し続ける
{println(i);}	{ }内の i の変化をコンソールに書く命令

ミッション

1. 上のコードの、`i = 100` を `i = 200` に変えて実行するとどう変わる？
2. 次に、`i <= 600` に変えて実行すると、コンソールはどう変わる？
3. 次に、`i = i+200` と変えて実行すると、コンソールはどう変わる？
4. 最初の値が 300 で、1100 以下まで、400 ずつ大きくなるように、下のコードの空欄を埋めてみよう。実行するとコンソールの数字はどうなったかな？

```

1 int i;
2 for (i=□; i<=□; i=i+□) {
3   println(i);
4 }

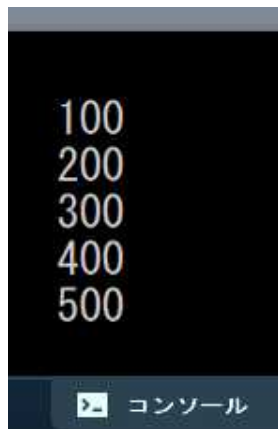
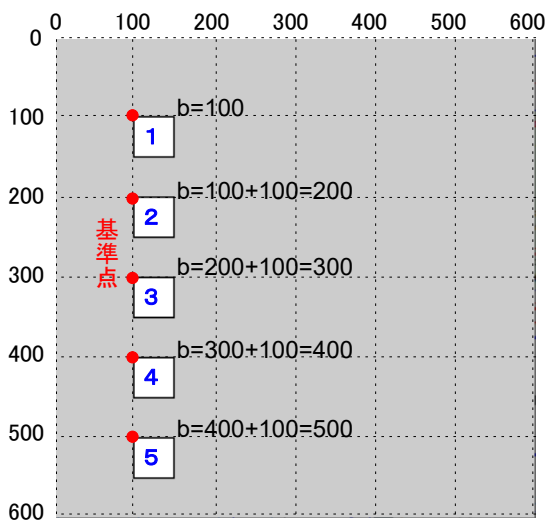
```

(初期値) (繰り返す上限) (更新式)

4-ステップ 3 : 繰り返しの式の意味…for だよ

新規を開いて、四角形を描くコードと for() を合体させてみよう。

```
1 int a=100; ← 四角形のX座標に変数 a を使う宣言
2 int i; ← 四角形のY座標に変数 i を使う宣言
3 size(600, 600);
4 for (i=100; i<=500; i=i+100) {
5     rect(a, i, 50, 50);
6     println(i);
7 }
```



b = 100 のとき、b = b + 100 は

- 1 回目 $b = 100$ 代入
- 2 回目 $b = 100 + 100 = 200$
- 3 回目 $b = 200 + 100 = 300$
- 4 回目 $b = 300 + 100 = 400$
- 5 回目 $b = 400 + 100 = 500$

b <= 500; だからここまで
もし、**b < 500;** なら、
500を含まないから、4回目まで。

プロセッシングで使う比較記号

- =** 等しい(等号)ではなく、代入記号だよ
- ==** 等しい(等号)
- !=** 等しくない
- A < B** AはBより小さい
- A > B** AはBより大きい
- A <= B** AはB以下(等しいか小さい)
- A >= B** AはB以上(等しいか大きい)

プログラミングでは **二** は等号ではなく代入記号で、右辺の数字や記号を左辺に代入するのに使うんだよ。

< や > の後ろに = が付くと、付かないでは意味が違うから気をつけよう。

たとえば、**a < 5** は5未満で、5は含まないけど、**a <= 5** は5以下で、5を含むんだ。

for(~) の数字を変えれば、いくら図形が増えても対応できる

4-ステップ 4 : 繰り返しをもっと深く…for だよ

前のコードを手直して実行しよう。変数宣言は `for()` のカッコの中に入れてもOKだ。

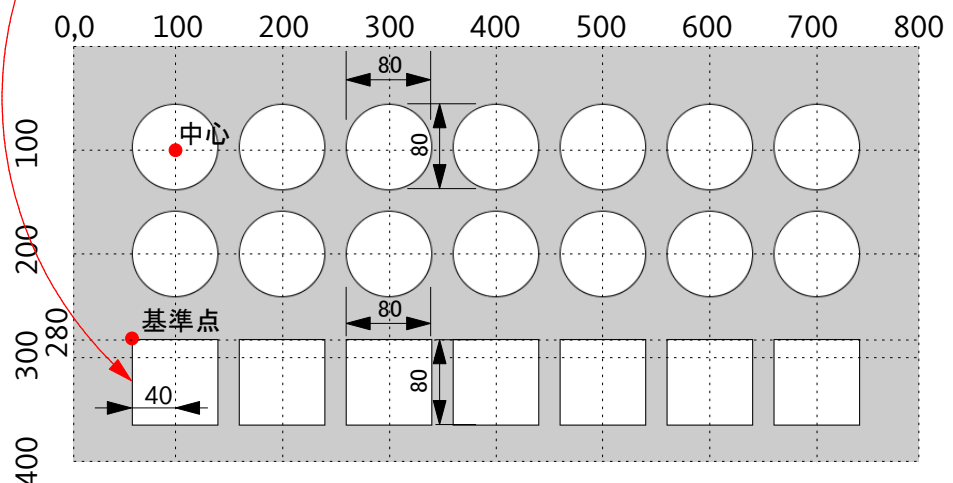
```
1 int a=80;
2 size(800, 200);
3 for(int i=100; i<=700; i=i+100) {
4     circle(i, 100, a);
5 }
```

上のコードの意味は、
右を読んでね。

```
for          //繰り返しの開始記号
(int i=100;   //変数 i を 100 (=初期値)から
i<= 700;    //変数 i が 700 になるまで { } 内を実行する
i = i+100)   // i を 100 ずつ大きくして繰り返す
{circle(i, 100, a);} //変数 i を変化させながら描く
```

`for()` の `{ }` の中には色々なコードを入れることができるんだ。加えてみよう。

```
1 int a=80;
2 size(800, 400);
3 for(int i=100; i<=700; i=i+100) {
4     circle(i, 100, a);
5     circle(i, , a);
6     rect(i-40, 280, a, a);
7 }
```



ミッション

1. `size(800, 800);` で、最初の円の座標が $X=100, Y=100$ で始まる直径 80 の円を Y 座標に平行に 7 個描いてみよう。
2. `size(800, 800);` で、最初の円の座標が $X=100, Y=100$ で始まる直径 80 の円を斜め下 45° に向けて 7 個描いてみよう。
3. `for-1` と名前を付けて保存しよう。

4-ステップ 5 : 頭の体操をしてみよう

下のプログラミングをした時に、計算結果にどんな数字が入るか考えてみよう。

a=100;

↓ 代入すると
aはいくつ?



a=1;

↓ 代入すると
100*aはいくつ?

*は掛ける(×と同じ)
/は割る(÷と同じ)

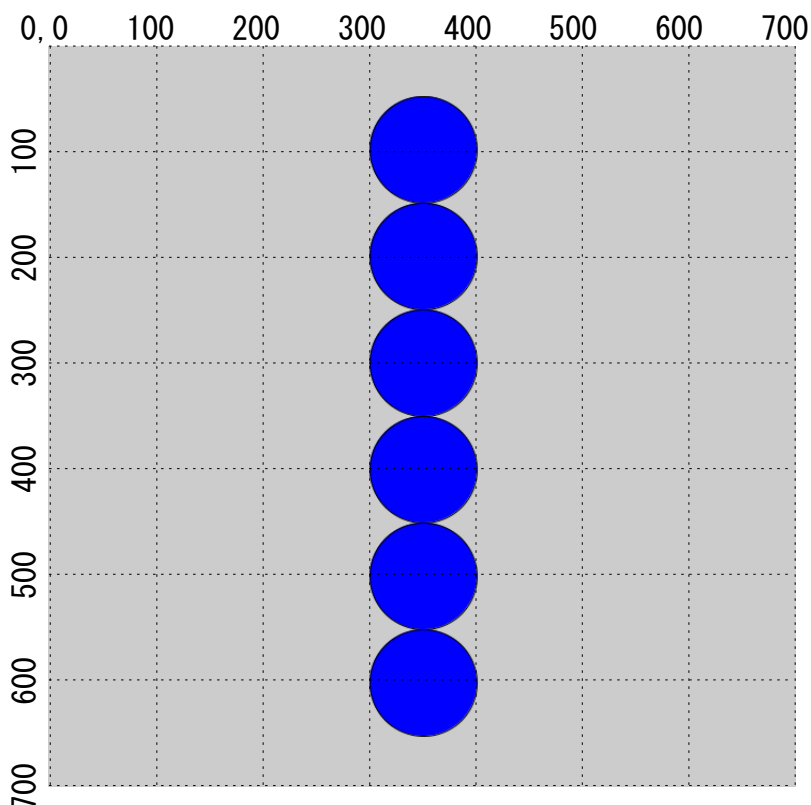
ファイルから新規で、下のコードを打ってみよう。

```
1 size(600, 200);  
2  
3 for(int a=100; a<=500; a=a+100) {  
4   circle(a, 100, 50);  
5 }
```

上のコードを手直して、同じ結果になるように空欄を埋めてみよう。

```
1 size(600, 200);  
2  
3 for(int a=1; a<=5; a=a+1) {  
4   circle(□*a, 100, 50);  
5 }
```

下の図になるように、上のコードを手直してみよう。



balls-6 で保存しよう。

4ステップ 6 : 繰り返しを徹底的に理解する

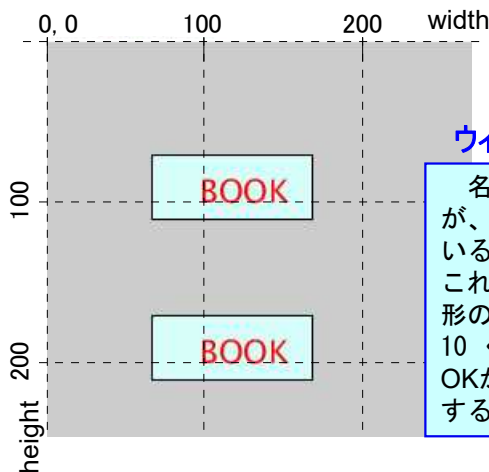
ファイルから新規を開いて、X=100,Y=100 を基準点として
あなたの名前を好きな色で、Y方向に5回繰り返して描いてみよう。

ヒント

1. size(700,600); にしよう。
2. 繰り返しには for を使うんだね。 !! これだけ大文字だよ
3. アルファベットの大きさは、textSize(); 命令を使うよ。
4. アルファベットは " " で囲まないと文字がでないぞ。
5. 文字の基準点はどこだ?
6. Y方向に繰り返すんだから、Y座標に変数を使えば良いんだ。
7. 右の図のBOOKのように、あなたの名前が並べば成功だ。

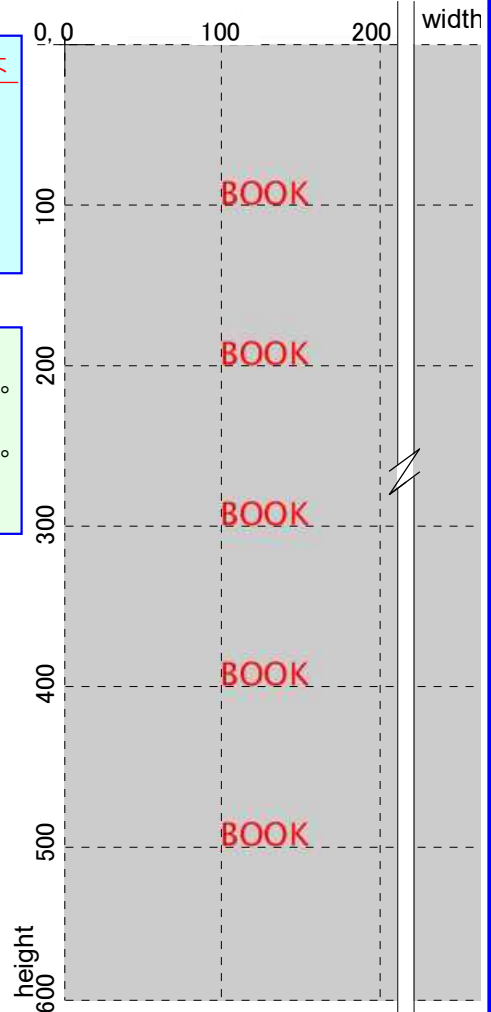
ミッション: 難問だよ

1. あなたの名前を好きな色の四角形で囲ってみよう。
2. for 文から最後の行までの間に2行のコードを入れるんだよ。
3. 四角形の基準点はどこだ?
4. 名前より四角形の方が短かったら、横に長くすれば良いんだ。
5. Y方向に繰り返すんだから、Y座標に変数を使うんだよね。
6. できたら、my-name で保存しよう。



ウインドウの一部だ

名前と四角形の中心が、ちょっとズれているね。
これを直すのは、四角形の基準点をY方向に10くらい移動すればOKかな。ここを調整するのが厄介だね。



次回の予告

今回は、繰り返した図形に別々の色をつけてみよう。
if という命令をつかうんだけど、ちょっと難しくなるよ。
団子を食べる夢を見ながら、次回を楽しみしててね。

Bye-bye!

