



---

# プログラミング演習(3)

## 変数：計算とアニメーション

---

中村, 高橋  
小林, 橋本

# 目標



- Processing で計算してみよう
- Processing で変数を理解する
- Processing で繰り返しのアニメーションをしてみよう
- Processing で相対位置に描画してみよう

# 計算してみよう



- 地球の半径は6378.137km. では, 地球1周の距離はどれくらいになるでしょうか?

```
sketch_130406a | Processing 1.5.1
File Edit Sketch Tools Help
sketch_130406a$
println( 6378.137 * 2 * 3.14 );
```

40054.703

```
println(6378.137 * 2 * 3.14);
```

l は l でも I でもなく  
L の小文字

\* は x の意味



# 標準出力とは？



## println(表示したい内容);

- 数値や文字を表示して確認したい時に利用
- 数値を表示する場合は, そのまま数値や数式を括弧内に記述したら表示可能

```
println((10+8)*10/2);
```

- 文字列 (nakamuraなど) を表示する場合は, ダブルクォーテーション「"」で最初と最後を囲む

```
println("nakamura");
```

# 演算はどうするか？



$$5 + 3$$

$$10.5 + 3.8$$

$$5 * 3$$

$$5 * 3 + (6 - 4) / 2$$

$$(5 + 3) * 6 / 2$$

$5 + 3$	5 に 3 を加える
$5 - 3$	5 から 3 を引く
$5 * 3$	5 に 3 を掛ける
$5 / 3$	5 を 3 で割る
$5 \% 3$	5 を 3 で割った余り
$+ 5$	5 そのもの
$- 5$	5 の + と - を反転

処理順序は算数と一緒に

# 演習（色々計算）



- 半径が15cmの球体の体積を求めよ
  - 体積の求め方は？  $V = \frac{4}{3}\pi r^3$
- 楕円の面積を求めよ（長半径100, 短半径50）
  - 楕円の面積の求め方は？  $S = \pi ab$
- 上底が18, 下底が20, 高さが18の台形の面積
  - 台形の面積の求め方は？  $S = \frac{(jotei+katei) \times takasa}{2}$

コンピュータは計算間違いしない！  
計算はコンピュータに任せよう！

# 命令に計算式を入れてもOK



- 画面の中央に配置したい！
  - 横幅と縦幅を2で割った値が中央だ！

```
void setup()
{
  size(400, 300);
  background(255, 255, 255);
}

void draw()
{
  fill(255, 0, 0);
  ellipse(200, 150, 150, 150);
}
```

```
void setup()
{
  size(400, 300);
  background(255, 255, 255);
}

void draw()
{
  fill(255, 0, 0);
  ellipse(400 / 2, 300 / 2, 150, 150);
}
```

# 命令に計算式を入れてもOK



- 画面の中央に配置したい！
  - 横幅と縦幅を2で割った値が中央だ！
  - 横幅widthと縦幅heightを使っても良いよ！

```
void setup()
{
  size(400, 300);
  background(255, 255, 255);
}

void draw()
{
  fill(255, 0, 0);
  ellipse(width / 2, height / 2, 150, 150);
}
```



# 整数と実数と真偽と文字列



**【整数：int】**  $\dots, -3, -2, -1, 0, 1, 2, 3, 4, \dots$

- ただし, 扱える数字には上限と下限があります

**【実数：float/double】**  $3.145926, 1.414213, \dots$

- 有理数のみ, 無理数はありません

**【真偽値：boolean】** true: 真, false: 偽

- 真偽値は, 「～なら～」や「～の間～を繰り返す」のように条件分岐などで使われます

**【文字列：String】** 中村聡史, Satoshi, nkmr,  $\dots$

- 文字列はダブルクォーテーション「"」またはシングルクォーテーション「'」でくくって表現します



- Processingでは整数 ( 0, 1, 2, 3, 4, 5, 6, ... )  
と, 実数 ( 0.0, 1.0, 2.0, 3.5, ... ) を明確に区別
  - 整数同士の計算結果は整数になる！！
  - 整数と実数の計算結果は実数になる
  - 実数と実数の計算結果は実数になる
    - $50 + 100 = 150$
    - $50 * 100 = 5000$
    - $50 + 100.0 = 150.0$
    - $50.0 / 100.0 = 0.5$
    - $50 / 100.0 = 0.5$
    - $50 / 100 = 0$
    - $50 / 20 = 2$

# 実数を整数に変換



- 「実数を整数」または「整数を実数」として扱いたい場合は、キャスト（ある型から別の型へと変換すること）を行う
  - キャストを行う時は括弧で変換先の型を書く

```
(int)3.14159265
```



```
3
```

```
(float)3
```



```
3.0
```

```
3 / (float)10
```



```
0.3
```

```
3 / (int)10.0
```



```
0
```

# キャストの特徴



- 実数から整数への変換では切り捨てとなる

```
(int)3.14
```



```
3
```

```
(int)3.9
```



```
3
```

- 四捨五入したい場合は 0.5 を足してから

```
(int)(3.14 + 0.5)
```



```
3
```

```
(int)(3.9 + 0.5)
```



```
4
```

# 文字列の足し算は？



- "nakamura" + "satoshi"  
– "nakamurasatoshi" となります
- "60" + "20"  
– "6020" となります
- 60 + "20"  
– "6020" となります
- "60" + 20  
– "6020" となります
- 60 + 20  
– 80 となります

# サイコロを作ってみる



- 同じ確率で 1, 2, 3, 4, 5, 6 のいずれかが選ばれて, その数値を表示するプログラムを作る
- 考え方
  - 1~6の値をランダムに取り出して表示する!
  - ランダムな値を取り出す方法は random !

`random(この値以上, この値未満)`

- 1以上で7未満の数字を取り出すのは…

`random(1, 7)`

# サイコロを作ってみる



- 1以上7未満の数字を出力！

```
println(random(1, 7));
```

- 残念ながら実数が表示されてしまう！
- そこでこの値を整数にキャストする

```
println((int)random(1, 7));
```

- できた！ ちなみに下記でもOK

```
println((int)random(0, 6) + 1);
```

```
println((int)random(6) + 1);
```

# 変数



- 変数とはコンピュータ（プログラム）が何かしらの情報・データ（数字やテキストなど）を記憶する場所（セルのようなもの）
  - コンピュータは記憶の場所を沢山もっており，そこに値を代入（保存）したり，取り出したりできる
  - 最後に代入された情報のみを記憶している

変数名	値
i	10
r	
:	:

← 11

← 3.14





- 変数とは値を入れるスペース（どのようなスペースかを定義する必要あり）
  - int（整数）： 0, 1, 2, 3, 4, 5, ...
  - float（実数）： 1.00, 3.14, 1.414, 1.732, ...
  - boolean（真偽）： true, false
  - String（文字列）： nakamura, 中村聡史

整数の変数 `x` を作成

```
int x;
```

実数の変数 `humidity` を作成

```
float humidity;
```

真偽の変数 `flag` を作成

```
boolean flag;
```

文字列の変数 `name` を作成

```
String name;
```

# 変数



```
int x;  
float humidity;  
boolean flag;  
String name;
```



変数の型	変数名	値
int	x	
float	humidity	
boolean	flag	
String	name	

# 変数について



- 名前は英字1文字以上であれば何文字でもOK
  - 2文字目以降であれば数字もOK
- 最初にどんな型かを宣言する
  - 整数なのか, 実数か, 文字列か...
- 別のものに対して同じ名前は使えません
  - 変数は名前で区別されています
  - 大文字小文字が違えば別のものになります
- 変数に値を代入する場合は「=」を使う
  - 左側に代入する対象を, 右側に代入する内容を

# 変数



```
int x;  
x = 5;  
float humidity;  
humidity = 30.5  
boolean flag = true;
```



変数の型	変数名	値
int	x	5
float	humidity	30.5
boolean	flag	true



- 変数  $r$  に  $10 * 2$  の値を格納する方法

```
int r;  
r = 20;
```

```
int r;  
r = 10 * 2;
```

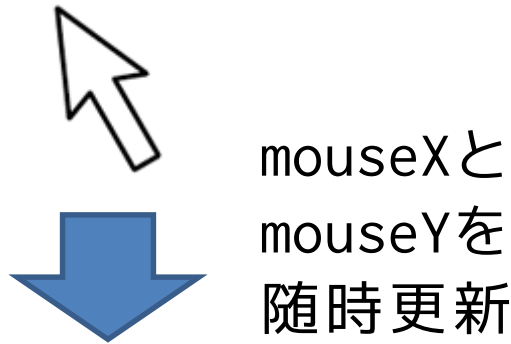
```
int r = 20;
```

```
int r = 10 * 2;
```

どれでもOK!



- mouseX や mouseY も変数
  - コンピュータが勝手に値を更新してくれている



型	変数名	値
int	mouseX	????
float	mouseY	????



```
void setup()
{
  size(400, 300);
}

void draw()
{
  point(mouseX, mouseY);
}
```

# 変数への値の代入



- 値の代入

```
x = 1;
```

```
y = 5;
```

```
z = x + y;
```

= を1つ使うことで代入

- 数を増やす

```
count++;
```

```
y--;
```

`count = count + 1;`

`y = y - 1;`

という意味

# 変数の演算はどうするか？

明治大学総合数理学部  
先端メディアサイエンス学科  
中村研究室



$x + y$

$\text{width} * \text{height}$

$r * r * 3.14$

$(10+x)/(6-y)$

$10 \% a$

$x + y$	x に y を加える
$x - y$	x から y を引く
$x * y$	x に y を掛ける
$x / y$	x を y で割る
$x \% y$	x を y で割った余り
$+ x$	x そのもの
$- x$	x の + と - を反転

処理順序は算数と一緒に



# カーソルからずらすには？

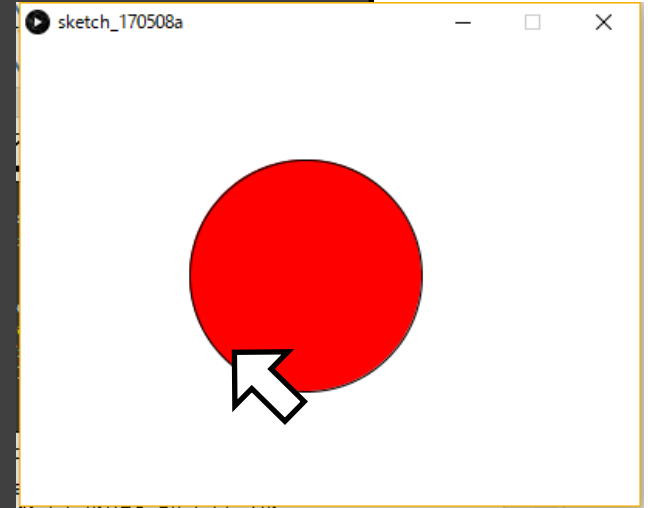
明治大学総合数理学部  
先端メディアサイエンス学科  
中村研究室



```
void setup()
{
  size(400, 300);
}

void draw()
{
  background(255, 255, 255);
  fill(255, 0, 0);
  ellipse(mouseX + 50, mouseY - 50, 150, 150);
}
```

**マウスのXY座標からの差分で指定**



- mouseX と mouseY はカーソルの位置
- 「+」や「-」で場所を動かす

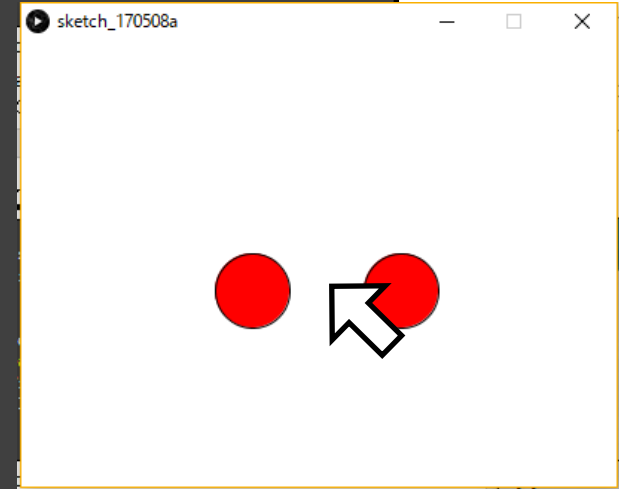
# 2つ描くには…



```
void setup()
{
  size(400, 300);
}

void draw()
{
  background(255, 255, 255);
  fill(255, 0, 0);
  ellipse(mouseX - 50, mouseY, 50, 50);
  ellipse(mouseX + 50, mouseY, 50, 50);
}
```

**マウスのXY座標からの差分で指定**

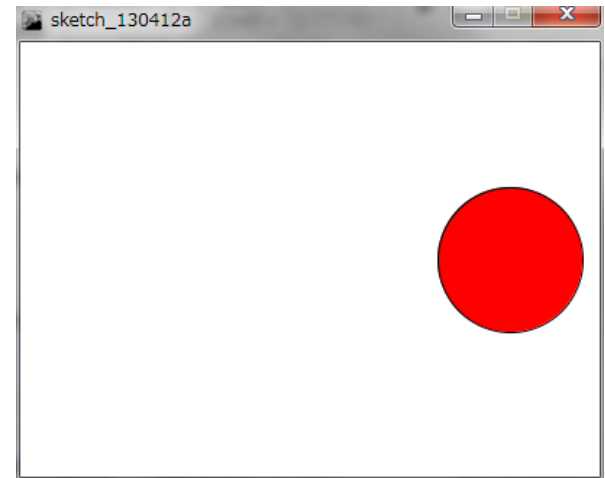
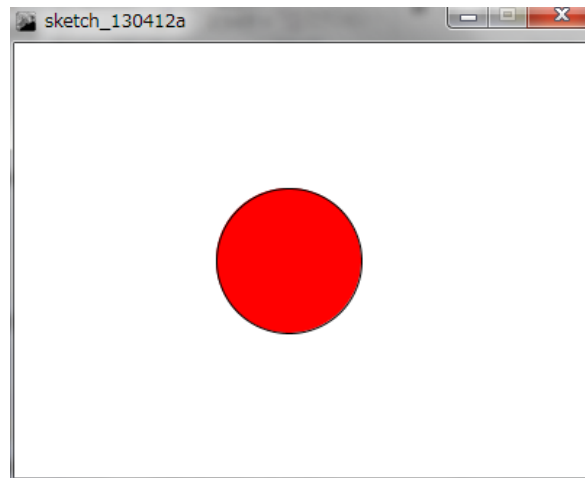
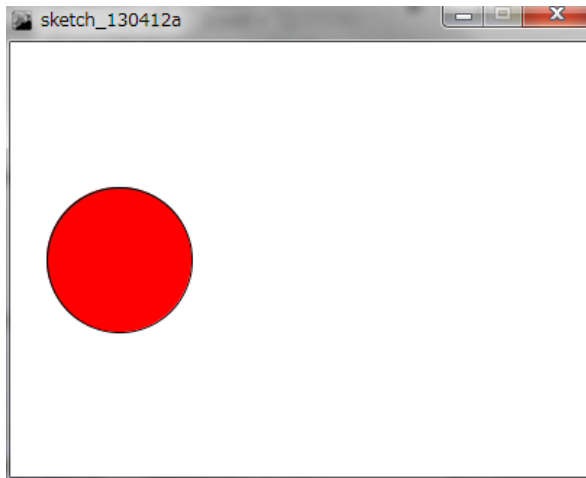


- mouseX と mouseY はカーソルの位置
- 「+」や「-」で場所を動かす

# 円を繰り返し動かしたい



(Q) 400x300のウィンドウで円を画面の左端から右端まで移動し，右端まで来たら左端に登場させたい



# 円を繰り返し動かしたい



## • 考え方

- frameCount は  $0, 1, 2, \dots, 299, 300, 301, 302, 303, \dots$  と増えていくが、「300」になったときに「0」になるようにしたい！
- frameCount は  $0, 1, 2, \dots, 299, 0, 1, 2, 3, \dots$  と増えていくようにすると解決するはず！
- $300 = 0, 301 = 1, 302 = 2, 303 = 3, 304 = 4$  とするにはどうしたらよい？
  - 「300 を引いたらいい？」 → 「0 の場合に  $-300$  になる」
  - 答えは「300で割った余りを計算する！」
  - つまり「 $\text{frameCount} \% 300$ 」となる

# 円を繰り返し動かしたい



## (A1) 円を左端から右端まで移動し繰り返す

1. 円の中心のY座標を150で固定しX座標を変更
2. X座標を増やすと円は右に, 減らすと左へ移動
3. draw() で描画する度にX座標を増やせばOK
4. draw() の度に frameCount という変数が1増加
5. X座標の値を  $\text{frameCount} \% 300$  とする!

```
void setup()
{
  size(300, 200);
}
void draw()
{
  background(255, 255, 255);
  println(frameCount);
  ellipse(frameCount % 300, 150, 100, 100);
}
```

# 円を繰り返し動かしたい



## (A1) 円を左端から右端まで移動し繰り返す

1. 円の中心のY座標を150で固定しX座標を変更
2. X座標を増やすと円は右に, 減らすと左へ移動
3. draw() で描画する度にX座標を増やせばOK
4. draw() の度に frameCount という変数が1増加
5. X座標の値を  $\text{frameCount} \% \text{width}$  とする! でもOK

```
void setup()
{
  size(300, 200);
}
void draw()
{
  background(255, 255, 255);
  println(frameCount);
  ellipse(frameCount % width, 150, 100, 100);
}
```

# 円を繰り返し動かしたい



## (A2)円を左端から右端まで移動し繰り返す

1. 円の中心のY座標を150で固定しX座標を変更
2. X座標を増やすと円は右に, 減らすと左へ移動
3. draw() で描画する度にX座標を増やせばOK
4. X座標の値をxとし,  
描画する度に増やす
5.  $x \% 300$  に描画する  
( $x \% \text{width}$ でもいいよ)

```
int x;

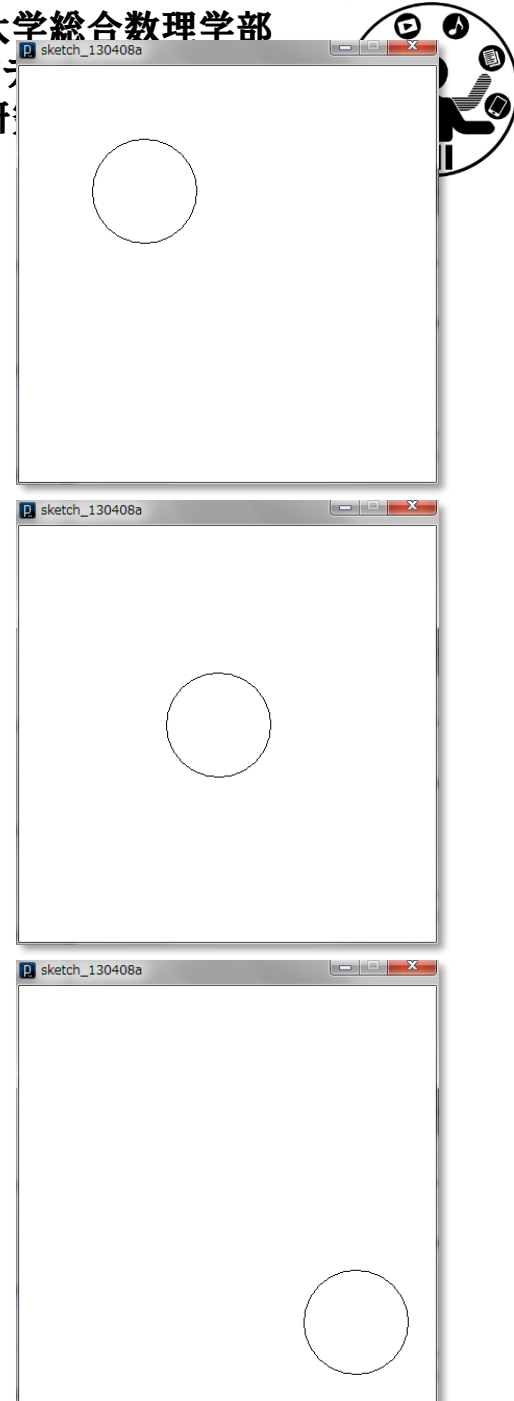
void setup()
{
  size(300, 200);
  x = 0;
}

void draw()
{
  background(255, 255, 255);
  ellipse(x % 300, 150, 100, 100);
  x = x + 1;
}
```

# アニメーションしてみよう

明治大学総合数理学部  
先端メディア  
中村研

```
int i;  
  
void setup()  
{  
  size(400, 400);  
  i = 0;  
}  
  
void draw()  
{  
  background(255, 255, 255);  
  ellipse(i, i, 100, 100);  
  i = i + 1;  
}
```





# アニメーションしてみよう

変数  $i$  を定義

```
int i;
```

```
void setup()
```

```
{
```

```
  size(400, 400);
```

```
  i = 0;
```

```
}
```

$i$  に  $0$  を代入

```
void draw()
```

```
{
```

```
  background(255, 255, 255);
```

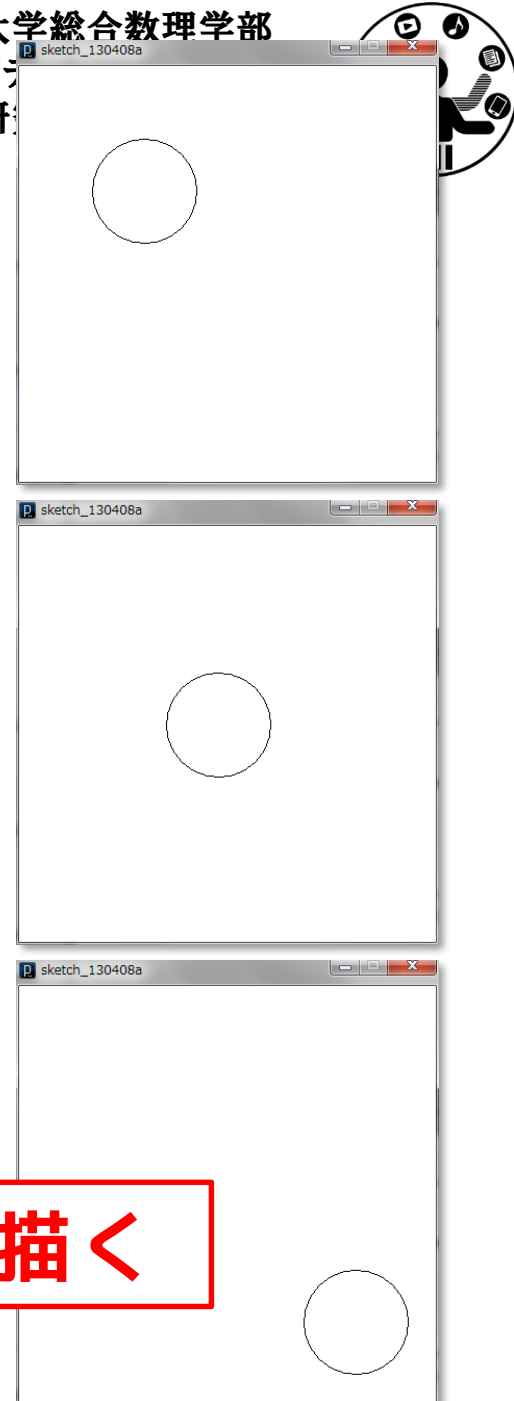
```
  ellipse(i, i, 100, 100);
```

```
  i = i + 1;
```

```
}
```

座標  $(i, i)$  に円を描く

$i$  を 1 増やす



# アニメーションしてみよう



変数  $x$  を定義

```
int x;
```

変数  $y$  を定義

```
int y;
```

```
void setup()
```

```
{
```

```
  size(400, 300);
```

```
  x = 0;
```

```
  y = 0;
```

```
}
```

変数  $x$  に  $0$  を代入

変数  $y$  に  $0$  を代入

```
void draw()
```

```
{
```

```
  background(255, 255, 255);
```

```
  ellipse(x, y, 100, 100);
```

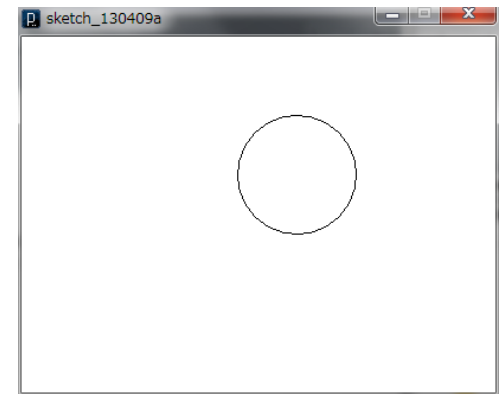
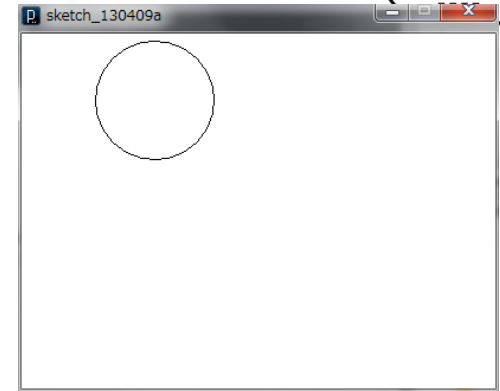
```
  x = x + 2;
```

```
  y = y + 1;
```

```
}
```

$x, y$  に円を描画

$x$ は1回あたり2増やす  
 $y$ は1回あたり1増やす



# アニメーションしてみよう



```
int x = 0;
int y = 0;
void setup()
{
  size(400, 300);
}

void draw()
{
  background(255, 255, 255);
  ellipse(x, y, 100, 100);
  x = x + 2;
  y = y + 1;
}
```

変数  $x$  を定義して 0 に

変数  $y$  を定義して 0 に

変数を定義するときに  
値を代入してもOK!

$x, y$  に円を描画

$x$ は1回あたり2増やす  
 $y$ は1回あたり1増やす



# 値の動きを見てみよう！



```
int x=0;
int y=0;

void setup()
{
  size(400, 300);
}

void draw()
{
  background(255, 255, 255);
  ellipse(x, y, 100, 100);
  x = x + 2;
  y = y + 1;
  println("x = " + x);
  println("y = " + y);
}
```

The screenshot shows the Processing 2.0.3 IDE window titled "sketch\_140513a". The code editor contains the same code as shown in the previous block. The console window at the bottom displays the output of the program, showing the values of x and y increasing by 2 and 1 respectively in each iteration of the draw loop.

```
sketch_140513a | Processing 2.0.3
File Edit Sketch Tools Help
sketch_140513a
}
void draw(){
  background( 255, 255, 255 );
  ellipse( x, y, 100, 100 );
  x = x + 2;
  y = y + 1;
  println( "x = " + x );
  println( "y = " + y );
}
y = 304
x = 610
y = 305
x = 612
y = 306
x = 614
y = 307
x = 616
y = 308
x = 618
y = 309
14
```

# 計算結果を表示してみよう



- 入力して実行してみましよう  
– 半径から面積を求める

```
float r = 10.0;
println(r * r * 3.14);
r = 5.31;
println(r * r * 3.14);
r = 15.3 / 2;
println(r * r * 3.14);
r = r * 4;
println(r * r * 3.14);
```

```
314.0
88.53576
183.76067
2940.1707
```

# text による表示



- 通常の文字列表示方法：「"」で文字列を囲む  
text("出力する文字", X座標, Y座標);  
変数と文字列をくっつけるときは「+」を使う！
- 現在のマウスカーソル座標を表示するには？
  - 座標は mouseX, mouseY という変数に格納
  - 文字列と変数を表示するにはどうするか？
  - 文字列と変数をつなげるときには「+」を使う
  - 下記のコードを前頁のellipseの後に入れましょう

```
text(mouseX + ", " + mouseY, 100, 100);
```

# 四角形と面積を計算



- 左上の原点からマウスカーソル位置まで四角形を表示し，その面積を計算して表示する！

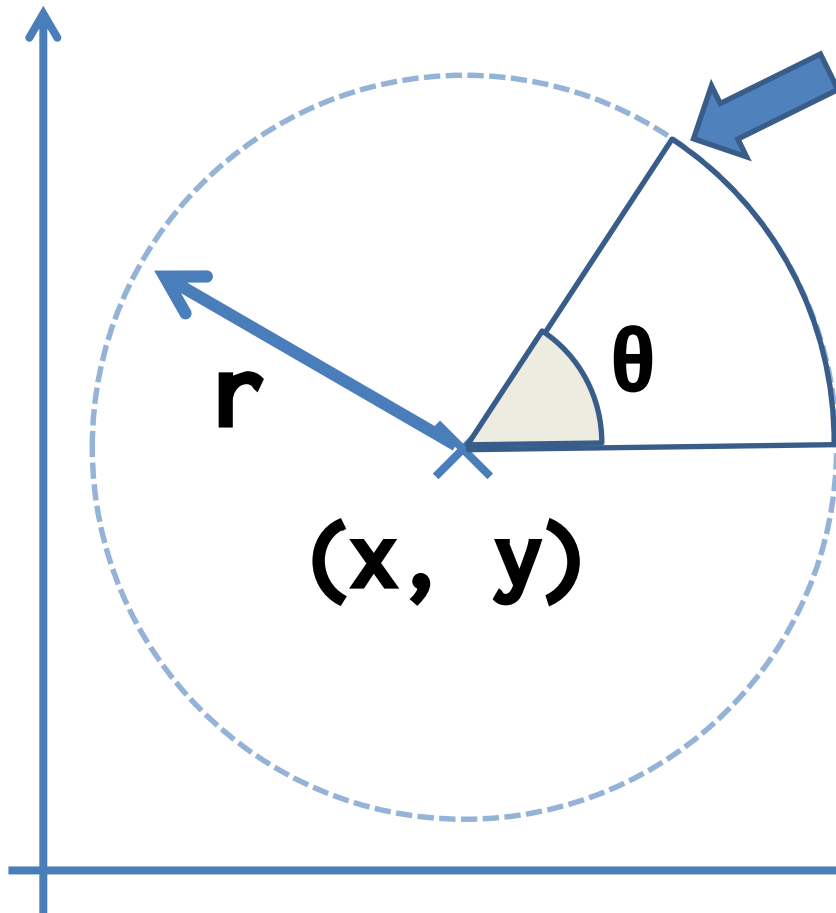
```
void setup()
{
  size(400, 300);
}

void draw()
{
  background(255, 255, 255);
  fill(255, 0, 0);
  rect(0, 0, mouseX, mouseY);
  text("menseki " + mouseX * mouseY, 0, 280);
}
```

# 角度を指定した描画



(Q) 画面中央 (200, 150) から, 半径100, 角度が60度の点まで線を引くにはどうするか?



この点の座標は?

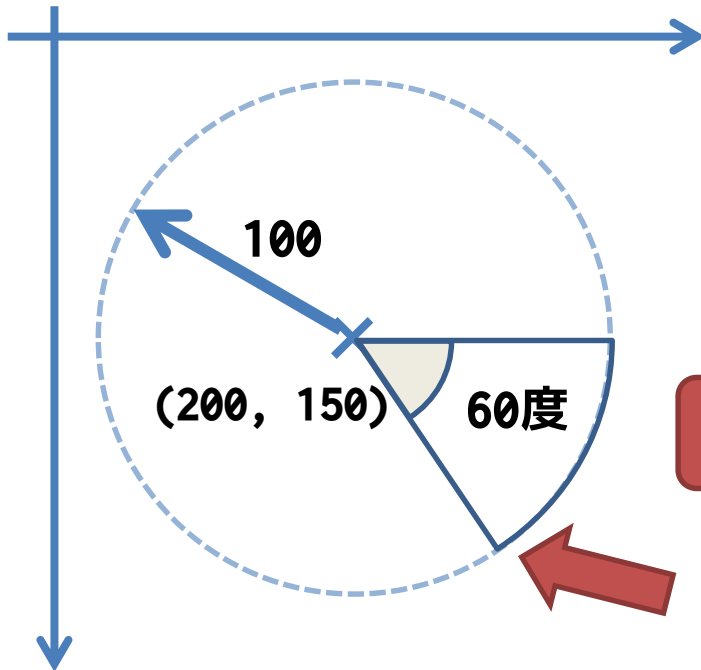
$$(x+r*\cos\theta, y+r*\sin\theta)$$



# 角度をどう扱うか



- $\cos(\text{角度})$ ; は角度のコサイン値
- $\sin(\text{角度})$ ; は角度のサイン値
  - ただし, 角度はラジアン単位 ( $0 \sim 2\pi$ )
  - ちなみに,  $\pi$ はPI (大文字のピーとアイ) と表現



y軸方向が上から下へなので  
回転方向が逆になる

60度はラジアンで  $(60/360) * 2\pi$

# 角度をどう扱うか

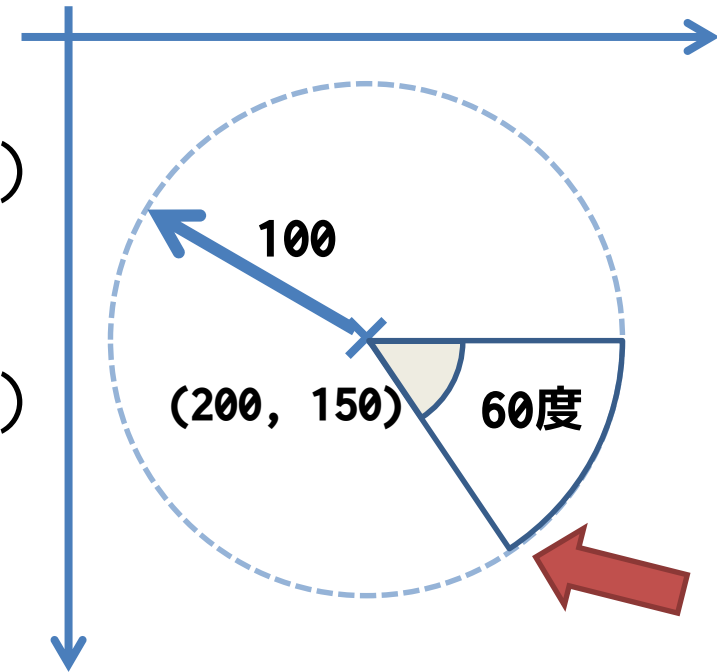


x 座標は

$$200 + 100 * \cos((60/360)*2*PI)$$

y 座標は

$$150 + 100 * \sin((60/360)*2*PI)$$



- ただ、うまくいかず、線が横に描画される  
- なぜ??



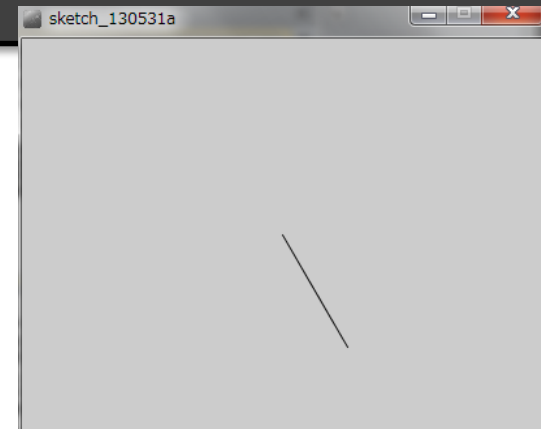
- $200+100*\cos((60/360)*2*PI)$  の計算は  
整数+整数\*cos((整数/整数)\*整数\*実数)
- **整数/整数=整数** となるので,  $60/360$  は本来  $0.1666\dots$  なのに **小数点以下切り捨てで0** に  
 $200+100*\cos(0*2*PI) = 200+100*\cos(0) = 300$

# 角度を指定した描画



- $x$ の座標:  $200+100*\cos((60.0/360.0)*2*PI)$
- $y$ の座標:  $150+100*\sin((60.0/360.0)*2*PI)$
- 線は `line(x1, y1, x2, y2);` で描画できる

```
size(400, 300);  
float x = 200+100*cos((60.0/360.0)*2*PI);  
float y = 150+100*sin((60.0/360.0)*2*PI);  
line(200, 150, x, y);
```

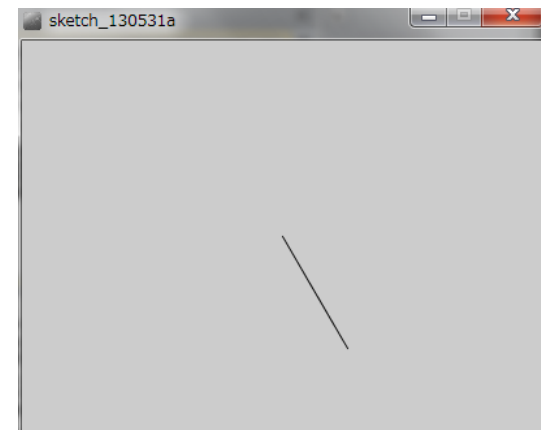


# 角度を指定した描画



- xの座標:  $200 + 100 * \cos(\text{radians}(60))$
- yの座標:  $150 + 100 * \sin(\text{radians}(60))$  でもOK!
  - $\text{radians}(\text{角度})$  で, ラジアン の値に変換できる!

```
size(400, 300);  
float x = 200 + 100 * cos(radians(60));  
float y = 150 + 100 * sin(radians(60));  
line(200, 150, x, y);
```





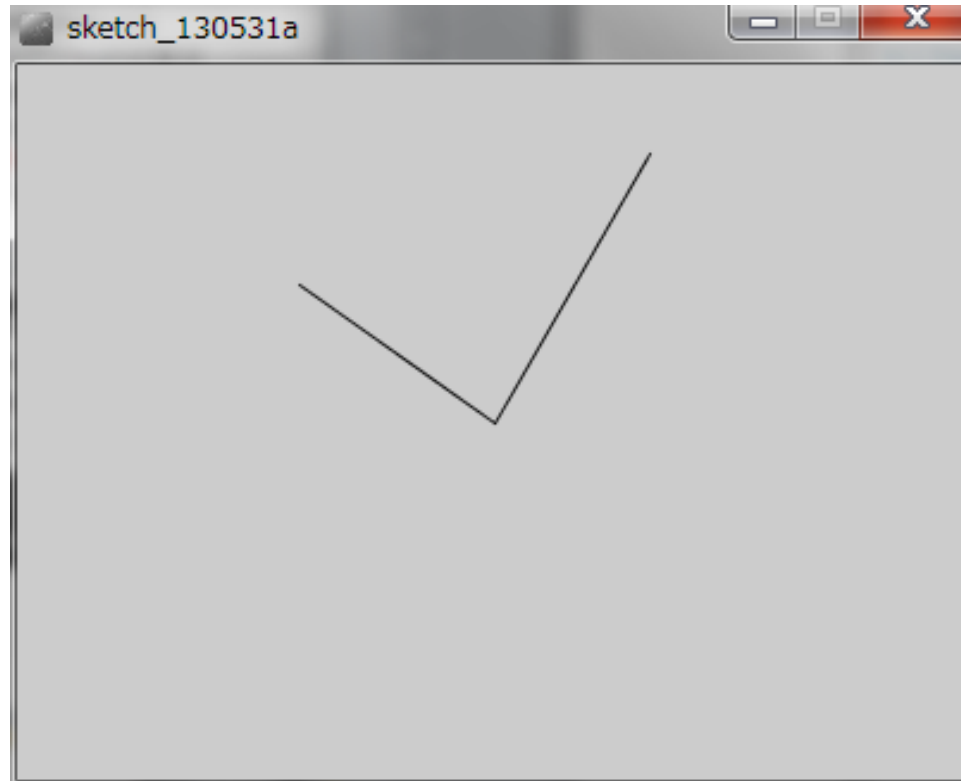
- println で出力してみよう！
  - 180度は $\pi$ なので3.1415…となるはず

```
println(PI);  
println(radians(180));  
println(2 * PI);  
println(radians(360));  
println(0.5 * PI);  
println(radians(90));
```

# 予習問題



- 10時10分の時の，アナログ時計の長針と短針を描画してみましょう



# 予習問題



## • ヒント

– 中心の座標は ( , )

– 短針の長さ =

– 短針が指し示す10時の角度 =

– 短針の端の座標は ( , )

– 長針の長さ =

– 長針が指し示す10分の角度 =

– 長針の端の座標は ( , )



# 計算して描画

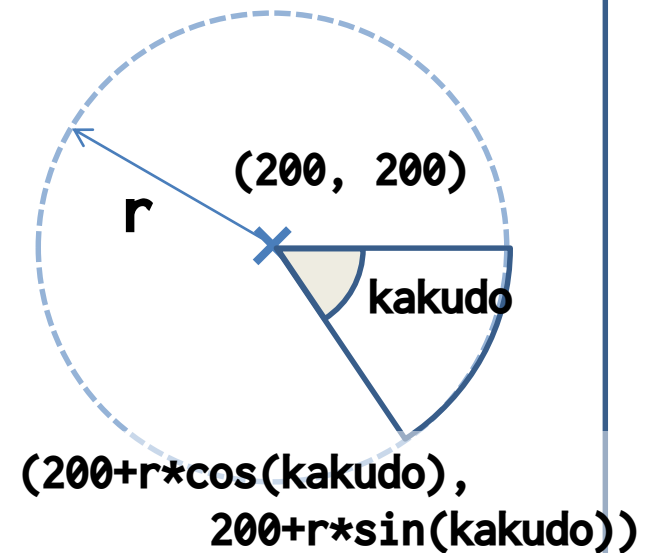


- 円をぐるぐる回転させてみる

```
float kakudo = 0;

void setup()
{
  size(400, 400);
}

void draw()
{
  background(255, 255, 255);
  fill(255, 0, 0);
  float x = 200 + 100 * cos(kakudo);
  float y = 200 + 100 * sin(kakudo);
  ellipse(x, y, 100, 100);
  kakudo = kakudo + (5.0 / 180.0) * PI; // 5度ずつ増やす
}
```



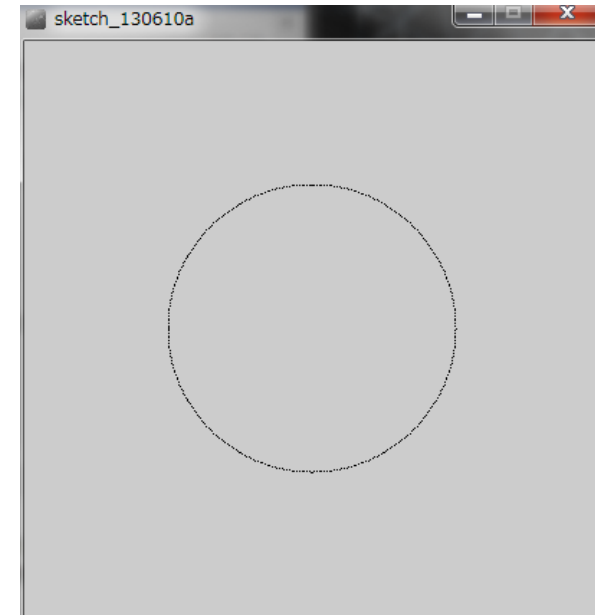
**cos や sin の単位はラジアン！**

# 円を点で描画してみる



(Q) 400x400のウィンドウ上に半径100の円を点で描画するにはどうするか？

- ヒント（考え方）
  - 円の中心座標は (200, 200)
  - point で点を描く
  - 角度を  $i$  とし draw の度に増やす
  - 点を描く場所は
    - $x = 200 + 100 * \cos(\text{radians}(i))$
    - $y = 200 + 100 * \sin(\text{radians}(i))$となる



# 円を点で描画してみる



- 角度を指定する変数（整数）を  $i$  とし, どんどん増やしていく
- ラジアンに変換するのは  $\text{radians}(i)$

```
int i = 0;

void setup()
{
  size(400,400);
}
void draw()
{
  float x = 200 + 100 * cos(radians(i));
  float y = 200 + 100 * sin(radians(i));
  point(x, y);
  i = i + 1;
}
```

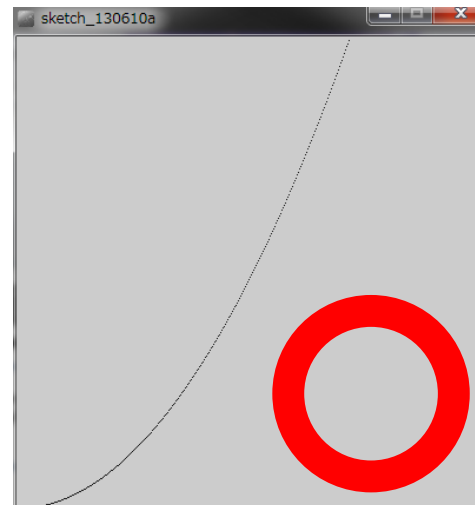
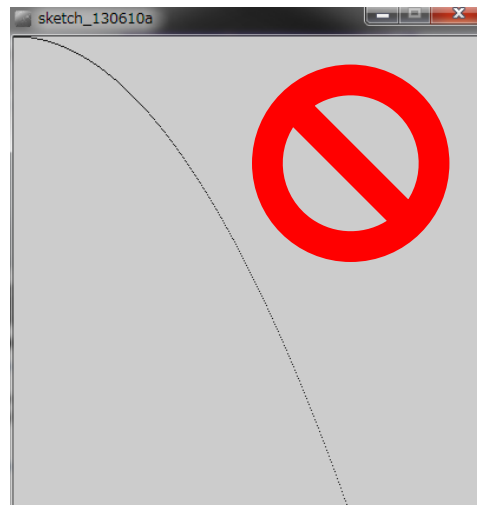
# 放物線を点で描画



(Q)  $y = \frac{x^2}{200}$  グラフを描け

– ヒント (考え方)

- 整数の変数  $x$  を用意して  $x$  を `draw()` のたびに増やす
- $y$  座標の値を  $x * x / 200$  で計算する
  - 上下を反転させるにはどうするか？



# 放物線を点で描画




- $y$  座標は下方方向に進む
  - ウィンドウの一番下を0とするため,  $height - x*x/200$

```
int x = 0;

void setup()
{
  size(400, 400);
}


void draw()
{
  int y = x * x / 200;
  point(x, y);
  x = x + 1;
}
```



```
int x = 0;

void setup()
{
  size(400, 400);
}

void draw()
{
  int y = x * x / 200;
  point(x, height - y);
  x = x + 1;
}
```



# 予習問題



- 600x400のウィンドウ内の上下左右の位置に一辺が100ピクセルの正方形を左から右に移動してみましょう
- 画面内に円を描き, その円の塗りつぶし色を黒色から赤色に変更していきましょう
  - (255, 0, 0)から(0, 0, 0)に徐々に変更
- $y = \frac{(x-100)^2}{200}$ の放物線を描け

# 変数の値の入れ替え



- どうやって変数の値を入れ替えるの？

変数の型	変数名	値
int	x	5
int	y	3
:	:	



```
x = y;  
y = x;
```



変数の型	変数名	値
int	x	3
int	y	3
:	:	

# 変数の値の入れ替え



- 変数を追加する

```
int temp = x;  
x = y;  
y = temp;
```



変数の型	変数名	値
int	x	3
int	y	5
int	temp	5

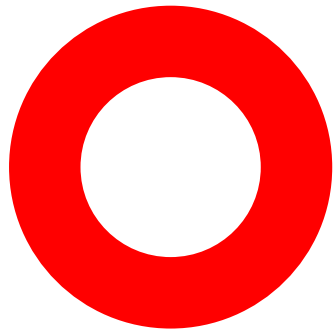


# 変数に対する注意点



- 同じ変数は定義することが出来ない
  - コンピュータは変数の名前で判断しているので同じ名前の変数は、1つしか存在できない
    - 同姓同名だと人間でも困りますよね

```
int x;  
int y;
```



```
int x;  
int x;
```



```
int x;  
float x;
```



# 変数に対する注意点



- グローバル変数とローカル変数
  - {} 内で定義されている変数は他から利用出来ない
  - {} 外で定義されている変数はどこからでも利用可能

```
int x = 10;
```

グローバル変数

```
void setup()
```

```
{
```

```
  int y = 20;
```

```
  size(400, 300);
```

```
}
```

ローカル変数

```
void draw()
```

```
{
```

```
  ellipse(x, y, 20, 20);
```

```
}
```

x はグローバルなので使えるけど  
y はsetup内でローカルなのでx

# ローカルとグローバル



- どうなる？

```
int globalX; // グローバル変数を定義

void setup()
{
  size(300, 200);
  // グローバル変数に値を入れる
  globalX = 0;
}

void draw()
{
  int localX; // ローカル変数を定義
  localX = 0; // ローカル変数に値を入れる
  background(255);
  globalX++;
  localX++;
  // それぞれの値で描画してみる
  ellipse(globalX, 50, 100, 100);
  ellipse(localX, 150, 100, 100);
}
```

# 変数に対する注意点



- グローバル変数とローカル変数
  - {} 内で定義されている変数は他から利用出来ない
    - 友達など内輪でのニックネームみたいなもの
  - {} 外で定義されている変数はどこからでも利用可能
    - 世の中全般で通じる名前
  - {} 内で定義されている変数と、同じ名前の変数を別の {} 内で定義することは可能（違う内輪なので）
  - グローバル変数として定義した変数を、ローカル変数にすると名前がかぶるので問題有り