



---

# プログラミング演習 (3)

## 変数：計算とアニメーション

---

中村, 小松  
小林, 橋本

# 目標

---

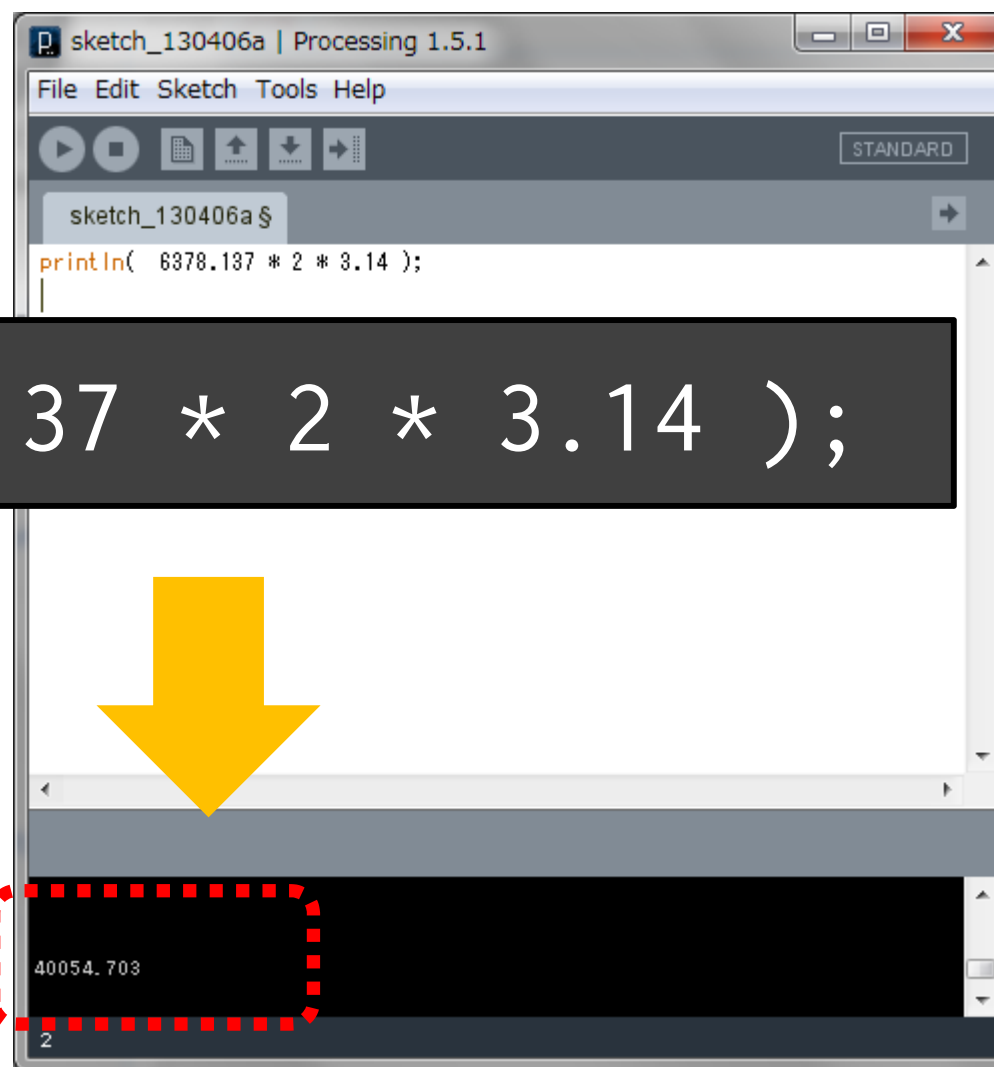


- Processing で計算してみよう
- Processing でアニメーションしよう
  - 計算の方法を理解する
  - 変数を理解する
  
- 課題： Processing でアニメーションしよう！

# 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?



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

l は 1 でも I でもなく  
L の小文字

\* は × 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(色々計算しよう)



- 半径は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？ ( $\frac{(jotei+katei)*takasa}{2}$ )

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！



**【整数】** ..., -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, ...

- ただし, 扱える数字には上限と下限があります

**【実数】** 3.14592653, 1.41421356, 2.23620679, ...

- 有理数のみ, 無理数はありません

**【真偽値】** true, false (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】** 中村聡史, Satoshi, Nakamura, ...

- 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
    - $50 + 100 = 150$
    - $50 * 100 = 5000$
    - $50 + 100.0 = 150.0$
    - $50.0 / 100.0 = 0.5$
    - $50 / 100.0 = 0.5$
    - $50 / 100 = 0$
    - $50 / 20 = 2$



# 文字列の足し算は？

---

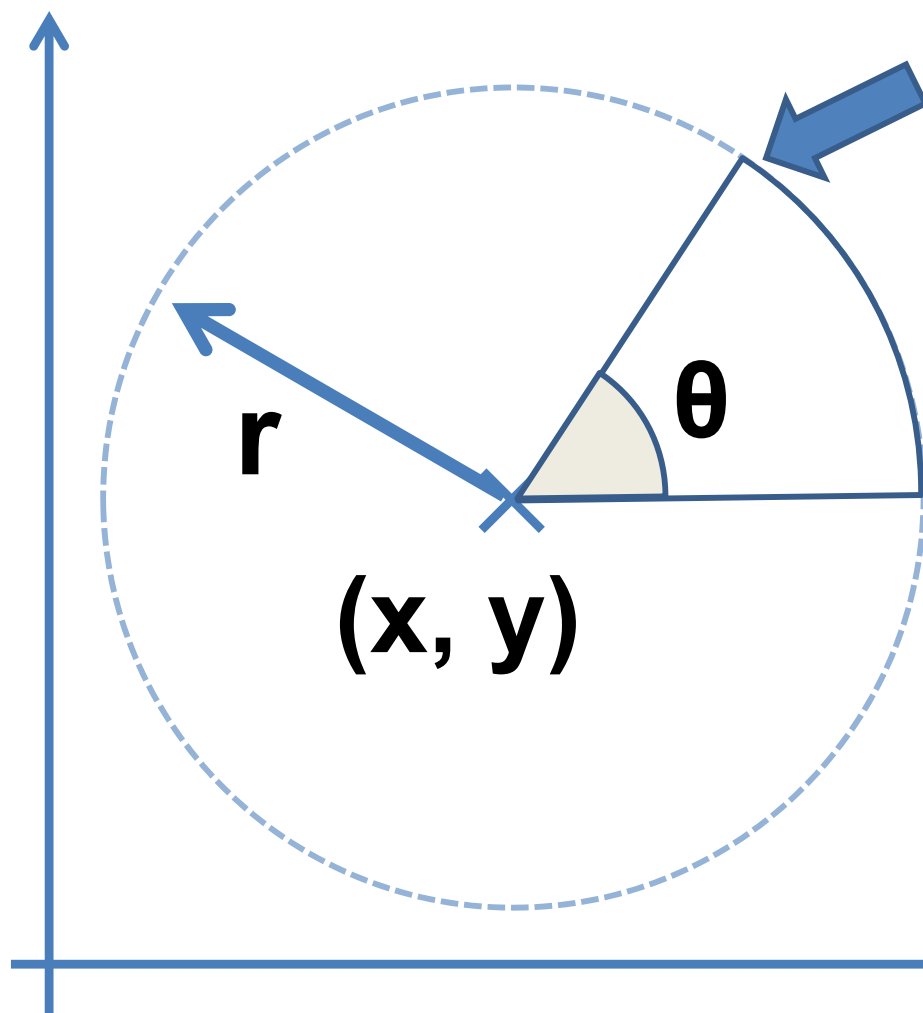


- "nakamura" + "satoshi"
  - "nakamurasatoshi" となります
- "60" + "20"
  - "6020" になります
- 60 + "20"
  - "6020" になります
- "60" + 20
  - "6020" になります
- 60 + 20
  - 80 になります

# 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？



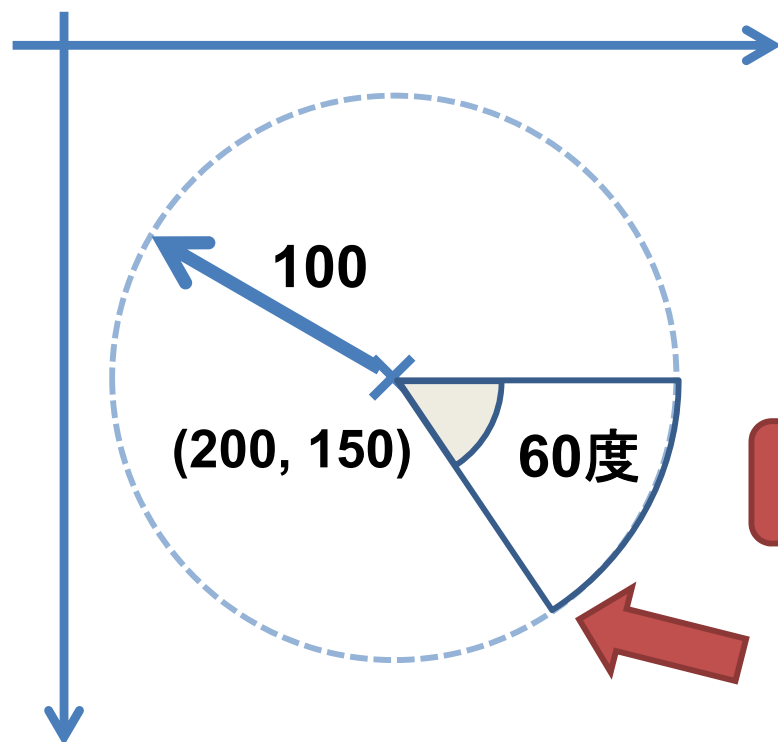
この点の座標は？

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

# 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

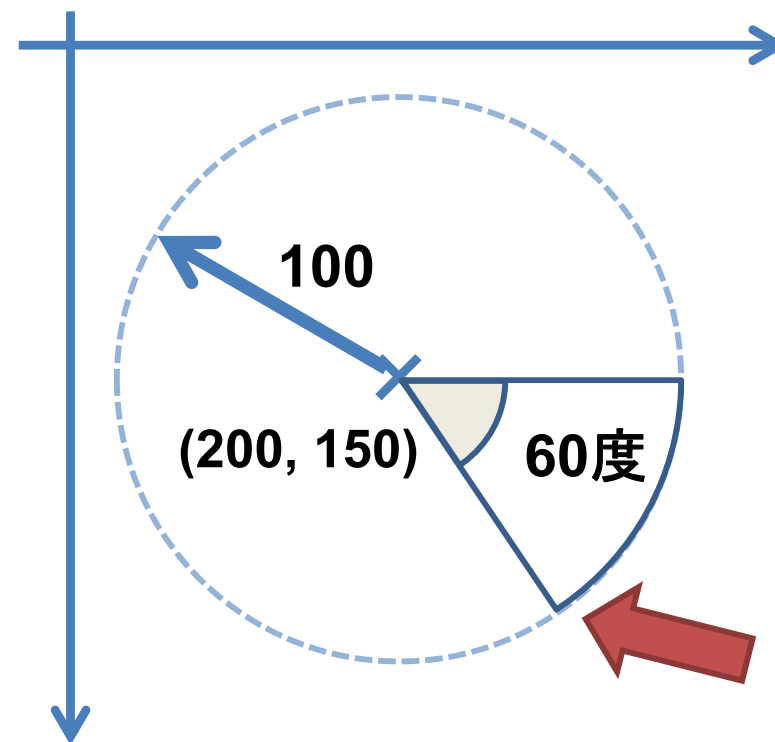


x 座標は

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

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??



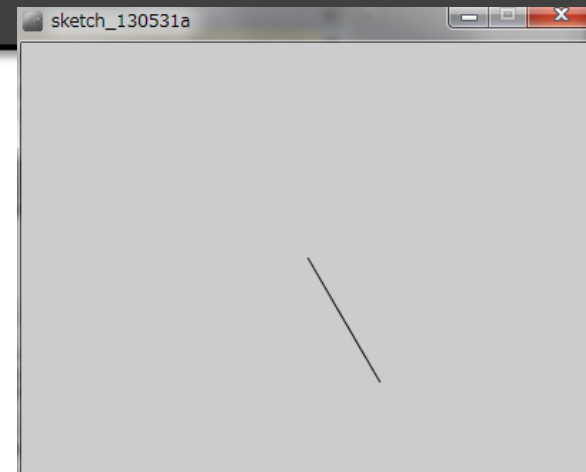
- $200+100*\cos((60/360)*2*PI)$ の計算は  
整数+整数\*cos((整数/整数)\*整数\*実数)
- 整数/整数 = 整数となるので, 60/360は本来  
0.1666...なのに小数点以下切り捨てで0に!  
 $200+100*\cos(0*2*PI) = 200+100*\cos(0) = 300$

# 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);
```

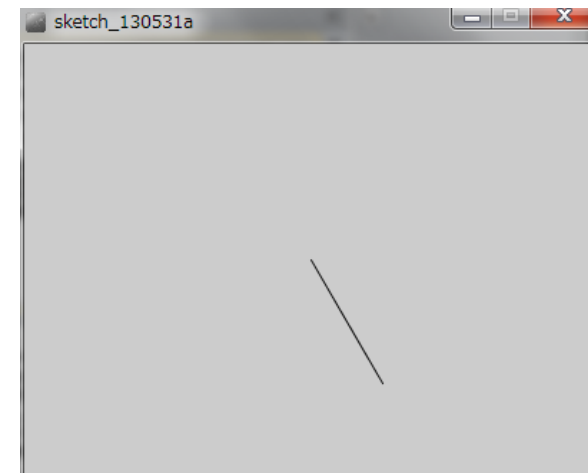


# 角度を指定した描画



- $x$ 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!

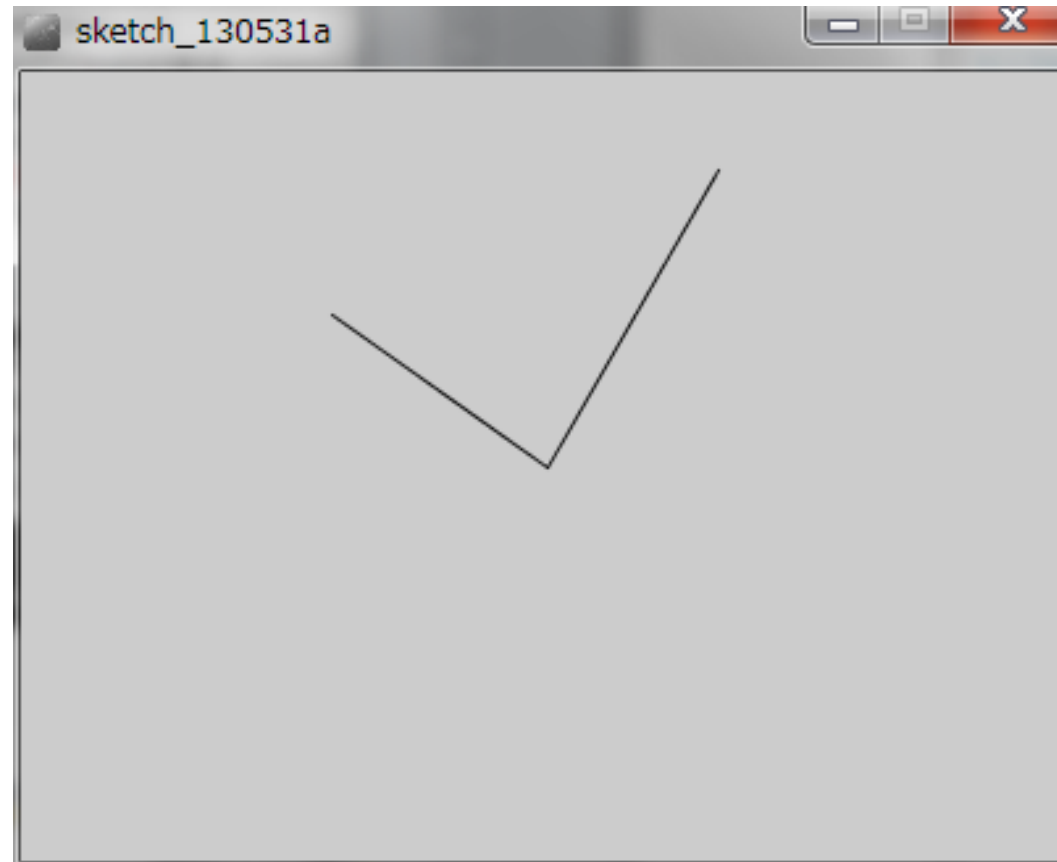
```
size( 400, 300 );  
float x = 200+100*cos( radians(60) );  
float y = 150+100*sin( radians(60) );  
line( 200, 150, x, y );
```



# 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 =

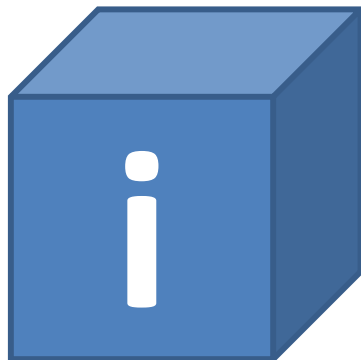
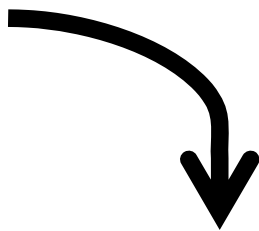
- 長針が指し示す10分の角度 =

- 長針の端の座標は( , )

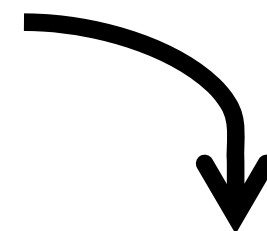


- 変数とはコンピュータ(プログラム)が何かしらの情報(数字やテキストなど)を記憶する箱
  - コンピュータは記憶の箱を沢山もっており, そこに値を代入(保存)したり, 取り出したりできる
  - 最後に代入された情報のみを記憶している

1



5.31





- 変数とは値を入れる箱（どのような箱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;
```



- 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
- 最初にどんな入れ物かを宣言する
  - 整数の入れ物か, 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



- mouseX, mouseY も変数
  - 現在のマウスカーソルのXY座標が記憶されている
  - コンピュータが勝手に記録してくれている
- 変数に値を格納する方法

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

```
int r;  
r = 10 * 5;
```

```
int r;  
r = 40;
```

```
int r = 10;
```

```
int r;  
r = 20;  
println( r*r*3.14 );
```

どれでもOK!

# 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

# 演算はどうするか？



$x + y$

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

$r * r * 3.14$

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

$10 \% a$

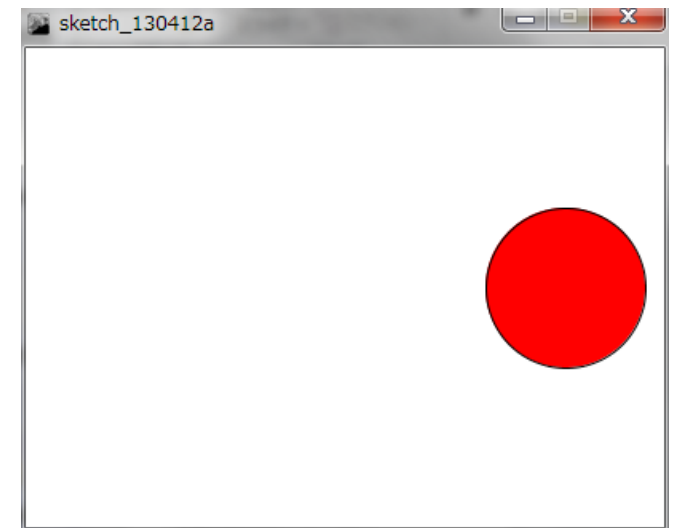
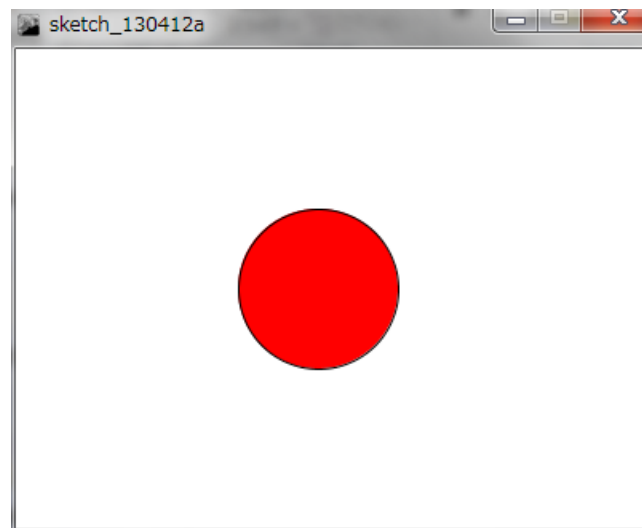
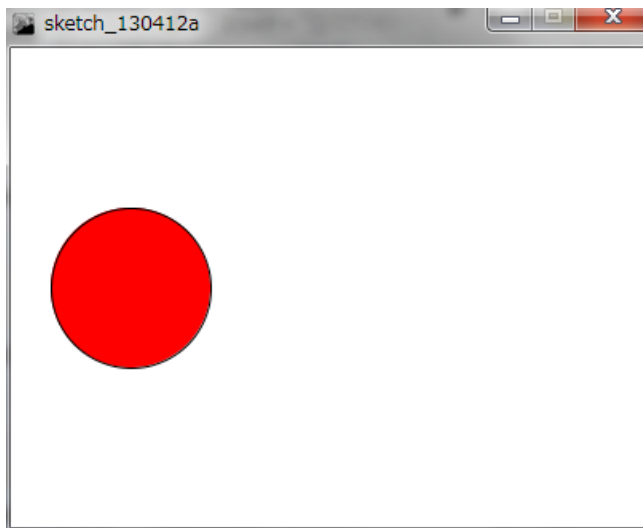
$x + y$	x に y を加える
$x - y$	x から y を引く
$x * y$	x に y を掛ける
$x / y$	x を y で割る
$x \% y$	x を y で割った余り
$+ x$	x そのもの
$- x$	x の + と - を反転

処理順序は算数と一緒に

# 円を動かしたい



(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たい. どうするか？





# 円を動かしたい



## (A1) 円を画面の左端から右端まで移動したい

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frameCountとする!

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

# 円を動かしたい



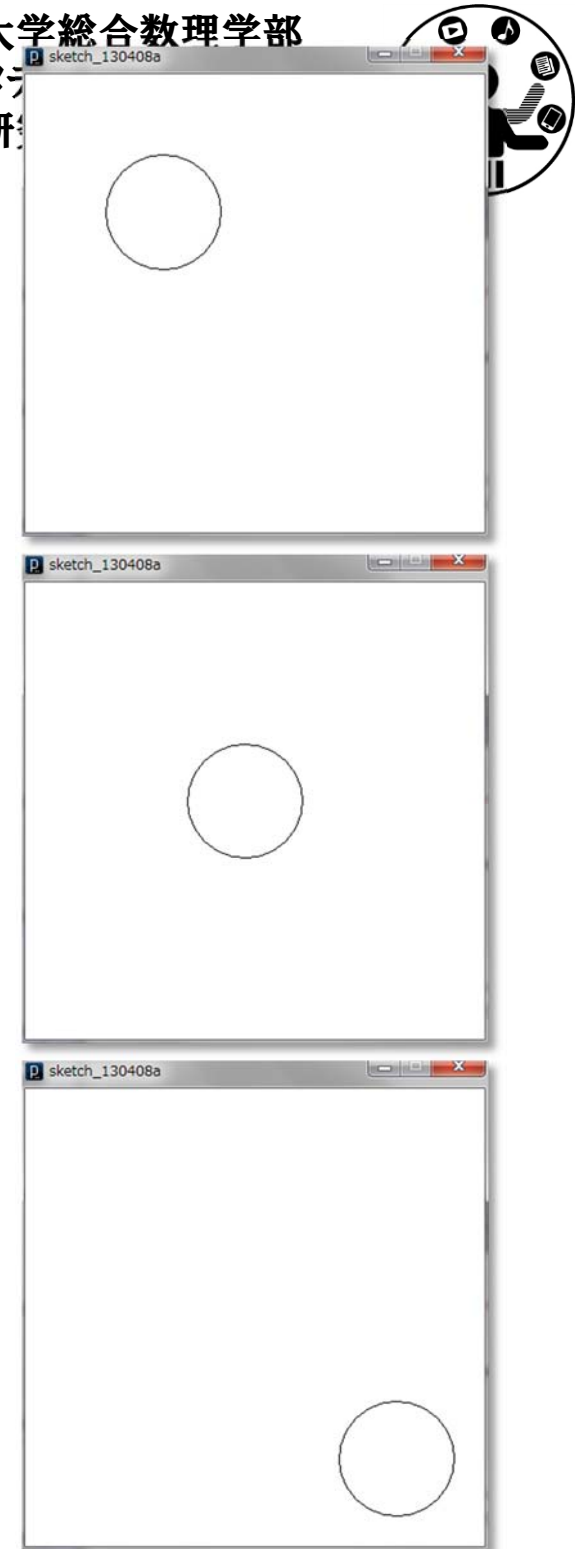
## (A2) 円を画面の左端から右端まで移動したい

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  
ellipse( x, 150, 100, 100 );

```
int x;  
void setup(){  
  size( 300, 200 );  
  x = 0;  
}  
void draw(){  
  background( 255, 255, 255 );  
  ellipse( x, 150, 100, 100 );  
  x = x + 1;  
}
```

# アニメーションしてみよう

```
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



```
int i;
```

変数  $i$  を定義

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

```
  i=0;
```

$i$  に 0 を代入

```
}
```

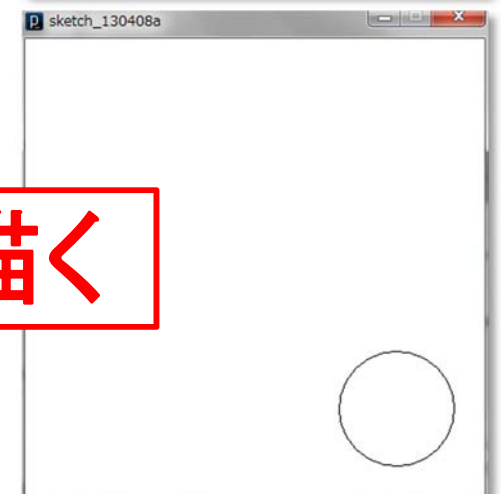
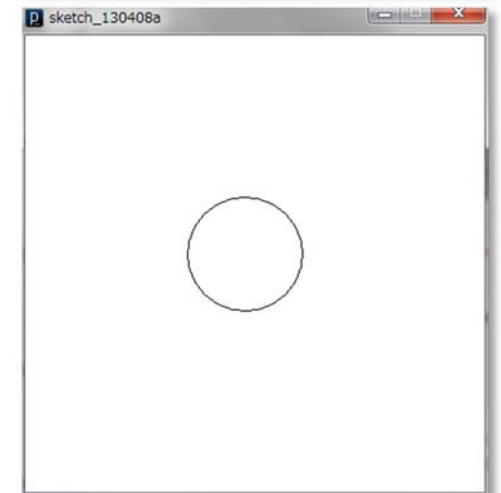
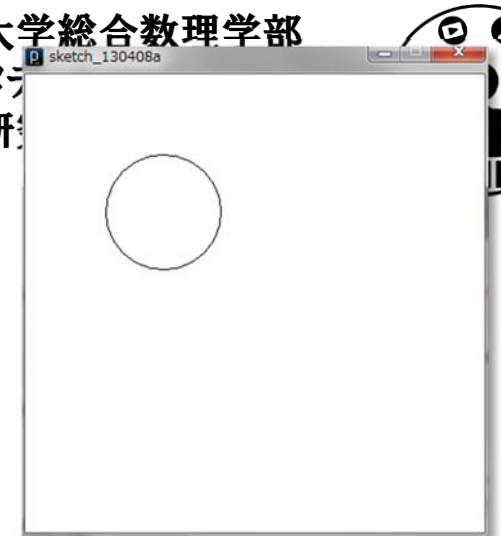
```
void draw(){  
  background( 255, 255, 255 );  
  ellipse( i, i, 100, 100 );
```

```
  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



```
int x;  
int y;  
void setup(){  
  size( 400, 300 );  
  x = 0;  
  y = 0;  
}  
void draw(){  
  background( 255, 255, 255 );  
  ellipse( x, y, 100, 100 );  
  x = x + 2;  
  y = y + 1;  
}
```

変数  $x$  を定義

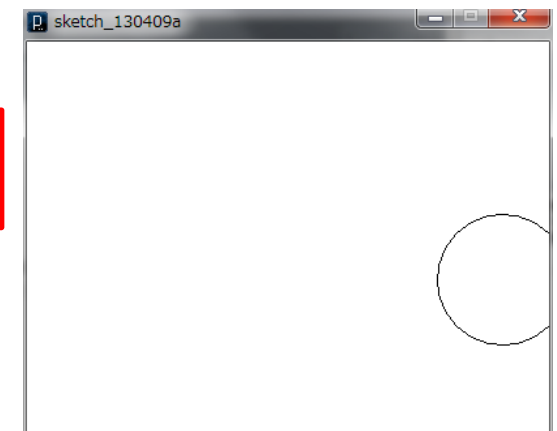
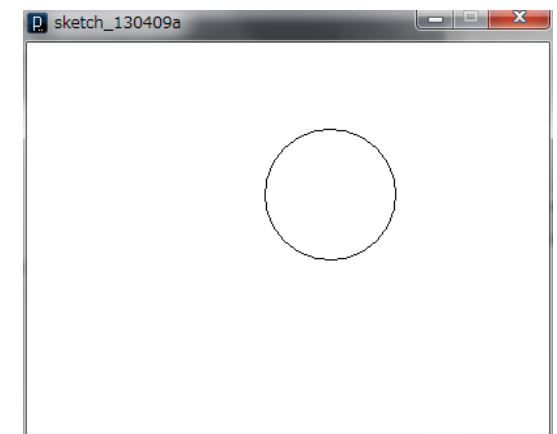
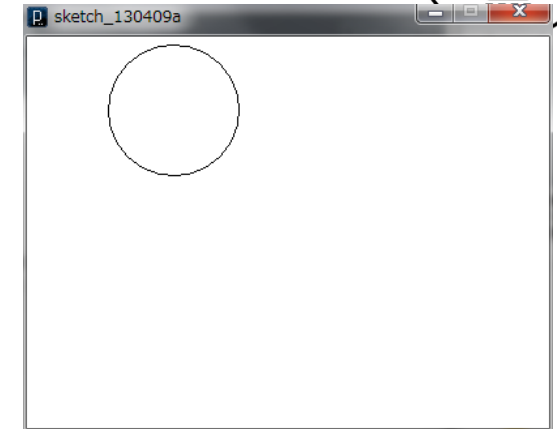
変数  $y$  を定義

変数  $x$  に 0 を代入

変数  $y$  に 0 を代入

$x, y$  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 );
}
```

A screenshot of the Processing 2.0.3 IDE. The window title is "sketch\_140513a | Processing 2.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for play, stop, refresh, and zoom. The code editor shows the same code as the previous block. The output console at the bottom displays the following text:

```
y = 304
x = 610
y = 305
x = 612
y = 306
x = 614
y = 307
x = 616
y = 308
x = 618
y = 309
```

The page number "14" is visible in the bottom left corner of the IDE window.

# 値の動きを見てみよう！



```
Basho basho;  
int x=0;  
int y=0;  
void setup(){  
    size( 400, 300 );  
    basho = new Basho( this );  
}  
void draw(){  
    background( 255, 255, 255 );  
    ellipse( x, y, 100, 100 );  
    x = x + 2;  
    y = y + 1;  
}
```

1期生の川連先輩作！

<https://github.com/kvvzr/Basho>

Basho.pdeをダウンロードしてプログラムの  
右側のタブに追加し2行書くだけ！



# 計算結果を表示してみよう



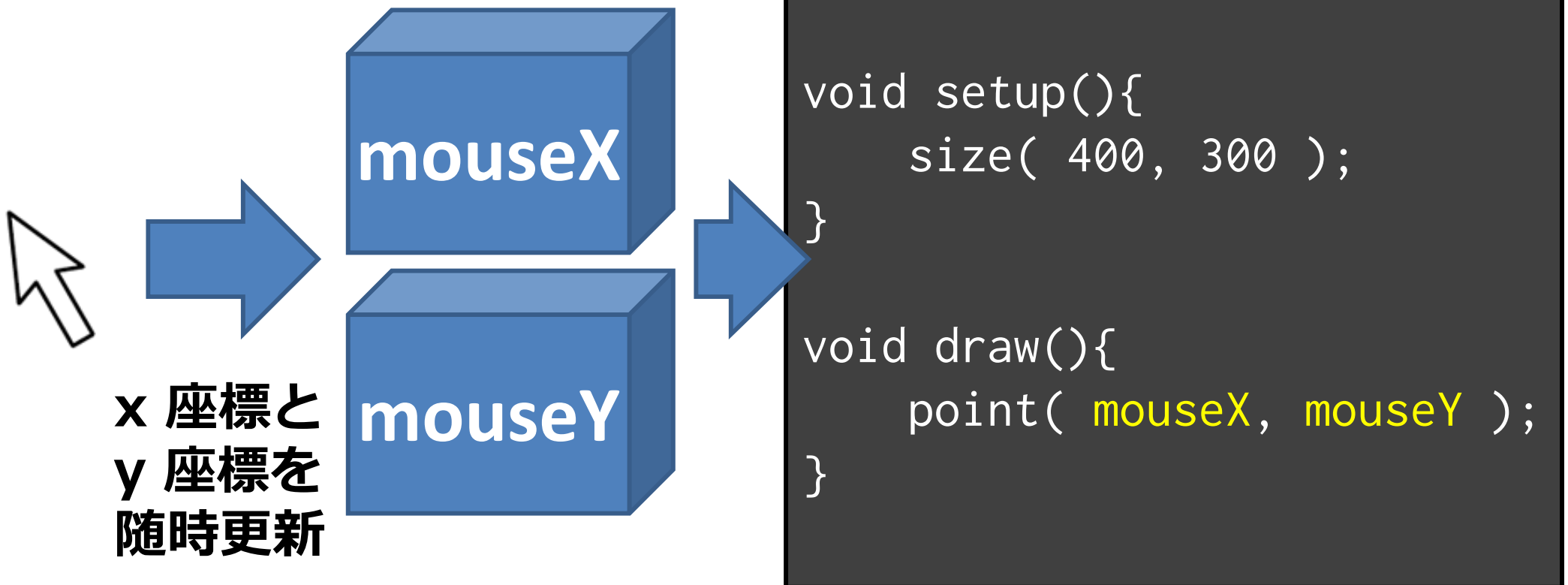
- 入力して実行してみましよう  
– 半径から面積を求める

```
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
```



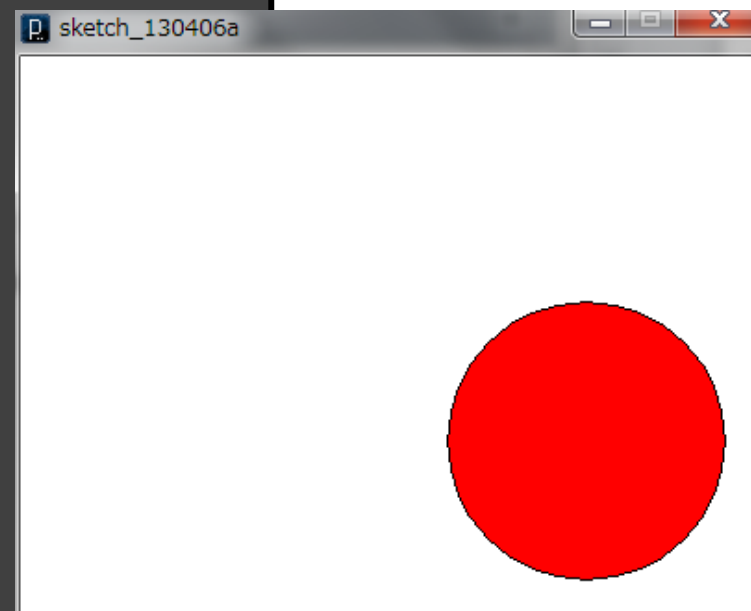
- mouseX や mouseY も変数
  - コンピュータが勝手に値を更新してくれている



# 赤丸をマウスの場所に



```
void setup(){  
  size( 400, 300 );  
}  
  
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX, mouseY, 150, 150 );  
}
```



## マウスのXY座標

- mouseX と mouseY はカーソルの位置
- draw の中で mouseX や mouseY を利用するとその点に絡めた描画が可能



- 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 );
}
```

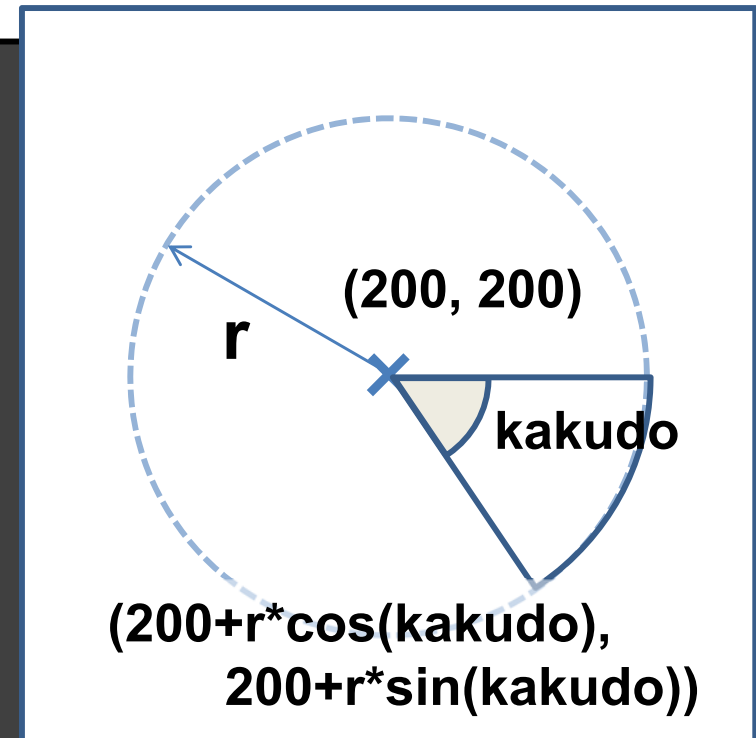


- 円をぐるぐる回転させてみる

```
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！

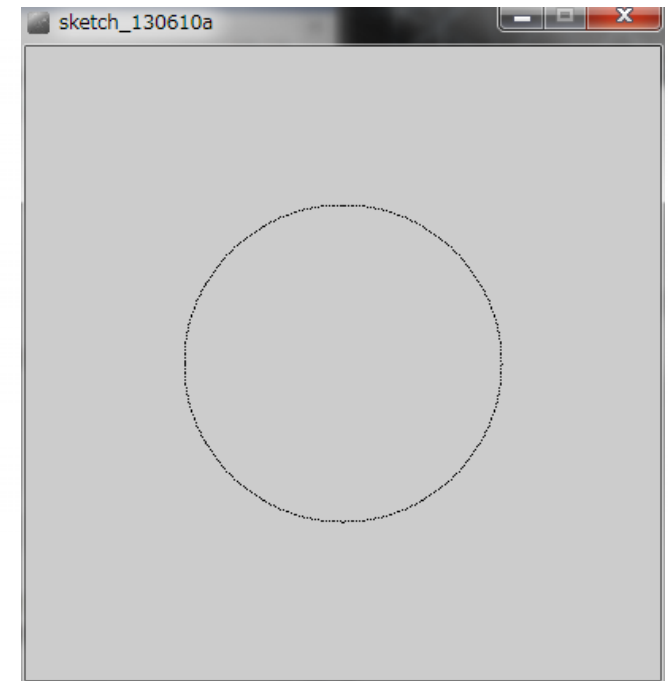
# 円を点で描画してみる



(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 `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;
}
```



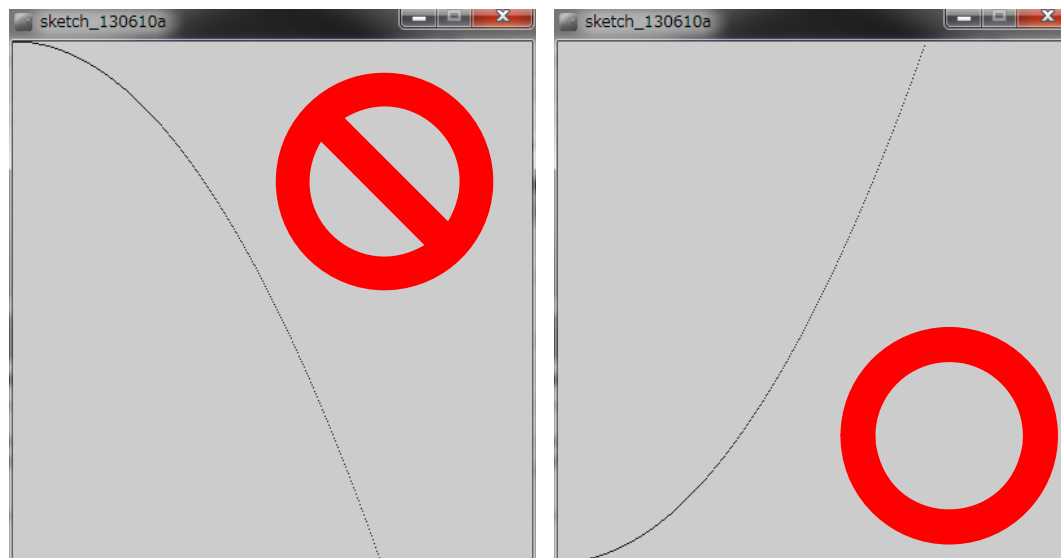
# 放物線を点で描画してみる



(Q)  $y = \frac{x^2}{200}$  のグラフを描くにはどうするか？

- ヒント(考え方)

- 整数の変数  $x$  を用意して  $x$  を draw の度に増やす
- $y$  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？



# 放物線を点で描画してみる




- y 座標は下方向に進む

– ウィンドウの一番下を 0 とするため,  $400-x*x/200$

```
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, 400-y );
  x=x+1;
}
```



# 予習問題

---

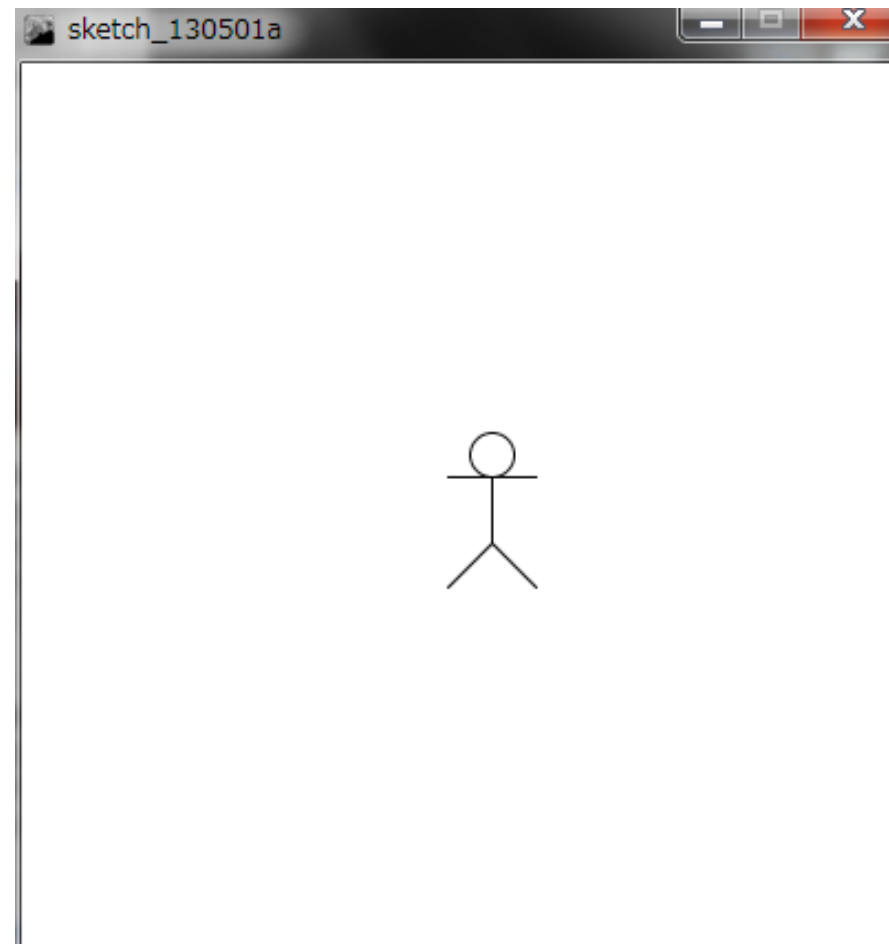


- 四角形を左から右に移動してみましよう
- 円を描きつつ, その面積も表示してみましよう
- 色を黒色から赤色に変更してみましよう
- 放物線を描いてみましよう

# 棒人間を描く



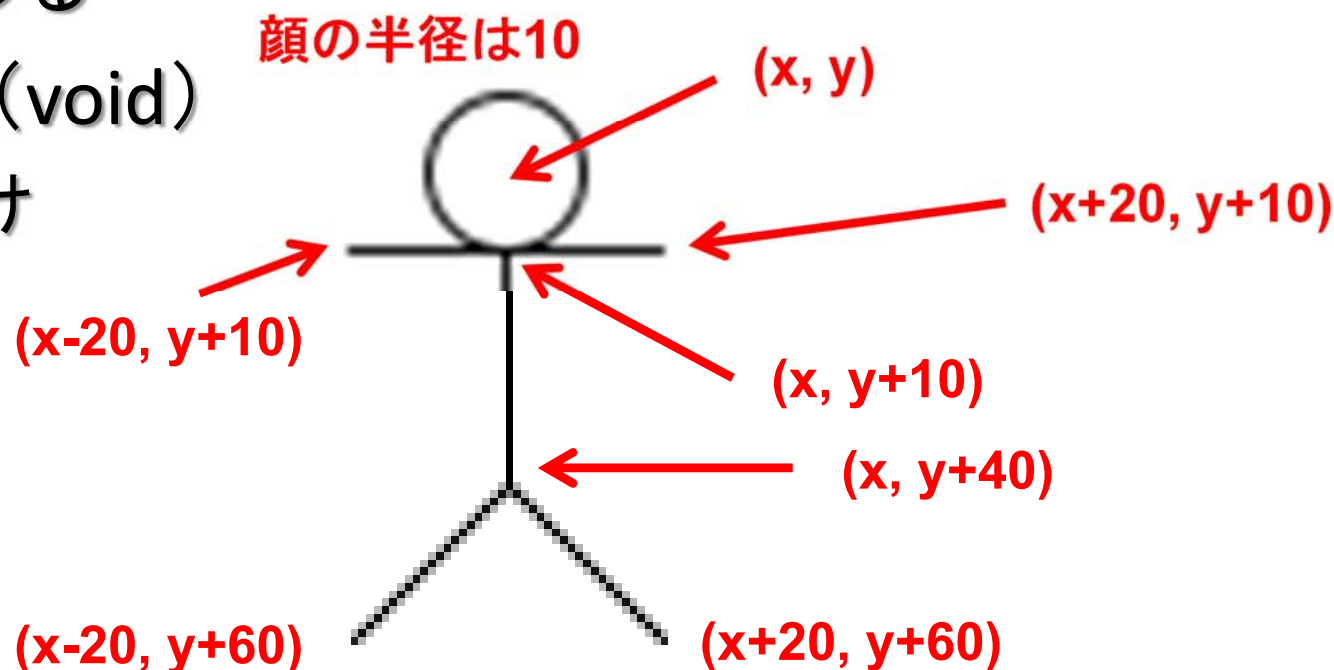
(Q) 棒人間の位置を変数で指定 & 変更する





- 考え方

- 棒人間は, 顔の中心の座標  $(x, y)$  を与えると, 勝手に体と手と足を描くものにする
- 棒人間の中心の座標を  $(x, y)$  としたときのそれぞれの座標を決める
- 返り値はなし (void)
  - 描画するだけ



# 棒人間を描く



- 変数  $x, y$  の場所に棒人間を描く
- 変数の値を変更するだけで, 手軽に場所を変更できる!

```
int  x = 50;  
int  y = 100;  
size( 400, 400 );  
ellipse( x, y, 20, 20 );  
line( x, y+10, x, y+40 );  
line( x-20, y+10, x+20, y+10 );  
line( x, y+40, x-20, y+60 );  
line( x, y+40, x+20, y+60 );
```



- 棒人間を画面の左から右までアニメーションしてみよう
  - 変数  $x$  と  $y$  をどんどん変化させる
- 棒人間をカーソルの位置に表示しましょう
  - $mouseX$ ,  $mouseY$  を利用する

# 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;
```

ローカル変数

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

```
}
```

```
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×

# 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