クリエイティブ・ビジュアライゼーションWS

Processingの基本

基本図形を描く

色について

変数

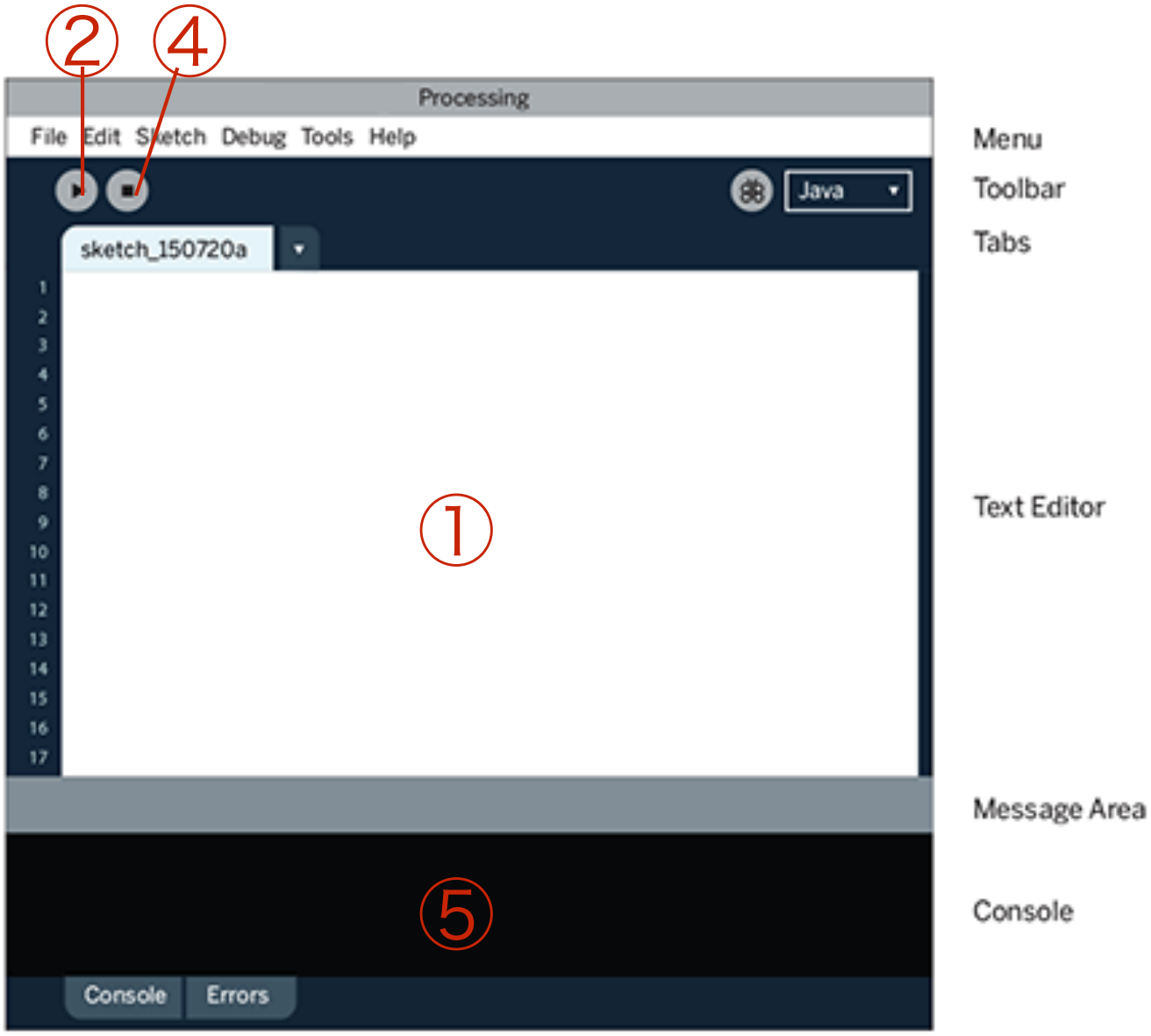
ワークショップ

山辺真幸（masaki@allianceport.jp）

Processingの基本

Processingの基本操作

Display Window



① editor

② run

③ window

④ stop

⑤ console

https://processing.org/tutorials/gettingstarted/

プログラミングの基本サイクル

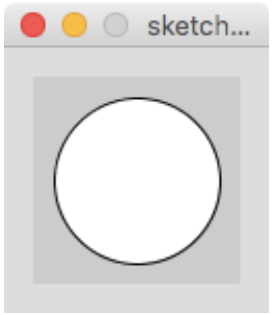
- Editor
  プログラムのソースコード（スケッチ）を書く
- Run
  スケッチを実行する
- Window
  描画結果が表示される
- Stop
  スケッチを停止する
- Console
  テキストメッセージやエラーコードが表示される

プログラミングの基本サイクル

エディタに書いて実行

```
ellipse(50, 50, 80, 80);
```

結果



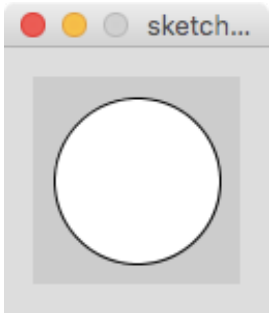
プログラミングの基本サイクル

プログラムの意味

```
ellipse(50, 50, 80, 80);
```

「楕円(ellipse)を描きなさい。
　中心は左から50px、上から50px、幅は80px、高さは80pxで。」

結果



プログラミングの基本サイクル

プログラムの意味

```
ellipse(50, 50, 80, 80);
```

「楕円(ellipse)を描きなさい。
中心は左から50px、上から50px、幅は80px、高さは80pxで。」

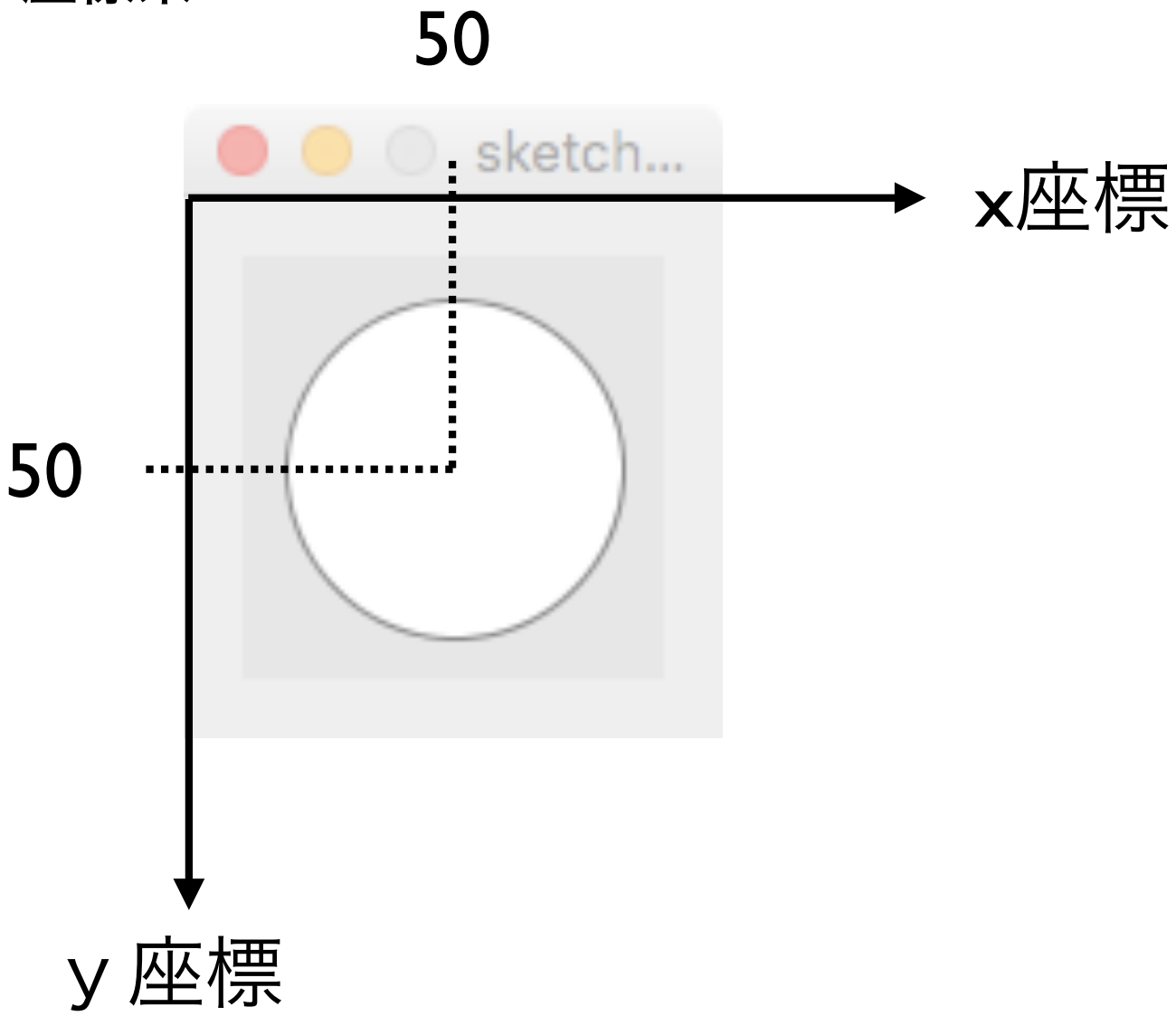
「メソッド（関数）」

「パラメータ（引数）」

ellipseには4つのパラメータ。パラメータは「,」で区切る

プログラミングの基本サイクル

ウィンドウの座標系

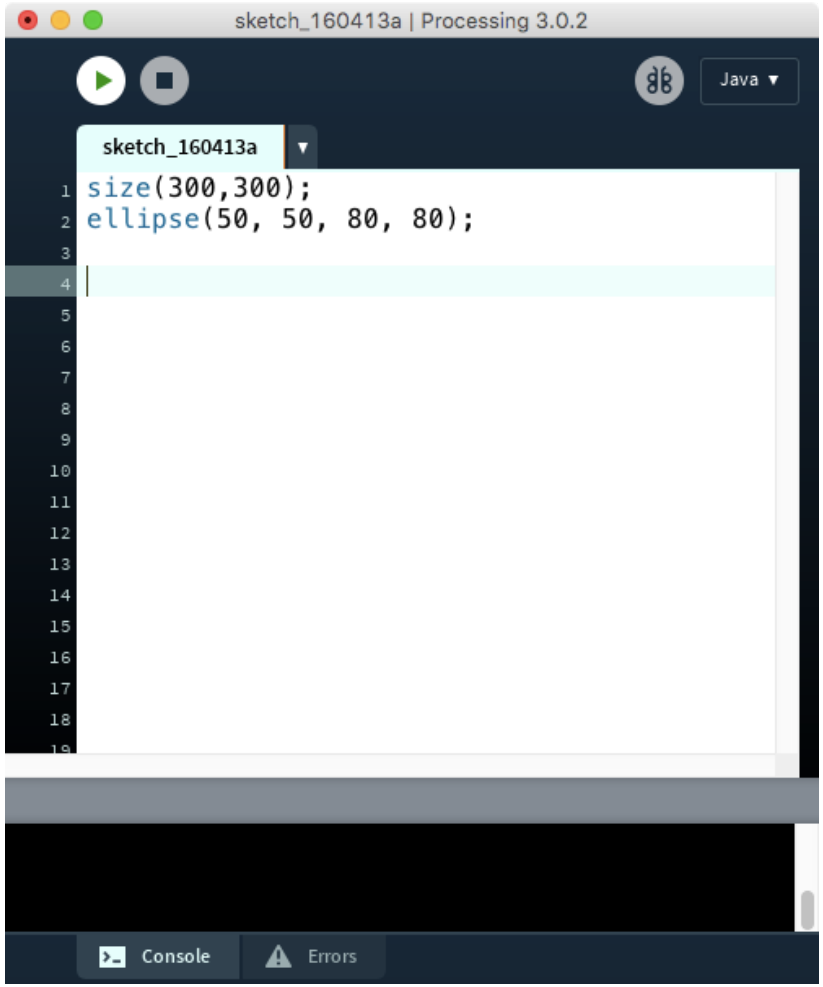


プログラミングの基本サイクル

ウィンドウのサイズを指定する

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

高さ300px、幅300px



プログラミングの基本サイクル

演習

任意のサイズのウィンドウを準備し、
任意の位置に任意の大きさで楕円を描く。

基本図形を描く

基本図形のメソッド

- 点（描画点の座標）

  point(300, 300);

- 線（開始点、終了点の座標）

  line(0, 0, 100, 100);

- 四角形（開始点の座標、幅、高さ）

  rect(10, 10, 100, 50);

- 楕円（中心の座標、幅、高さ）

  ellipse(30, 30, 100, 100);

- 三角形（頂点の座標を３つ）

  triangle(30, 75, 58, 20, 86, 75);

基本図形のメソッド

- 四角形のモード変更

  rectMode(CENTER);
  rect(10, 10, 100, 50);

  →中心座標と幅と高さ

基本図形のメソッド

- 四角形のモード変更

  rectMode(CENTER);
  rect(10, 10, 100, 50);

  →中心座標と幅と高さ

基本図形のメソッド

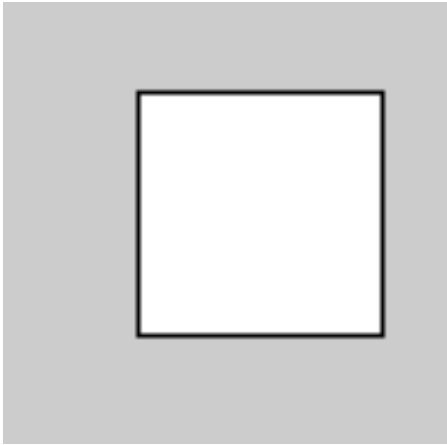
演習

```
size(200,200);
rectMode(CENTER);
rect(100,100,20,100);
ellipse(100,70,60,60);
ellipse(81,70,16,32);
ellipse(119,70,16,32);
line(90,150,80,160);
line(110,150,120,160);
```

多角形のメソッド

- 複数の頂点座標をvertexメソッドで指定する。
  先頭でbeginShape、最後にendShapeを実行する。

```
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



https://processing.org/reference/beginShape_.html

色について

色についての基本１

- 背景の色を決める（パラメータは明るさ、255=白、0=黒）

  background(255);

- 線の色を決める

  stroke(0);

- 塗りつぶしの色を決める

  fill(150);

- 線の色を指定しない

  noStroke();

- 塗りつぶしの色を指定しない

  noFill();

一度実行すると、他の色を指定するまで描画が影響されるので注意

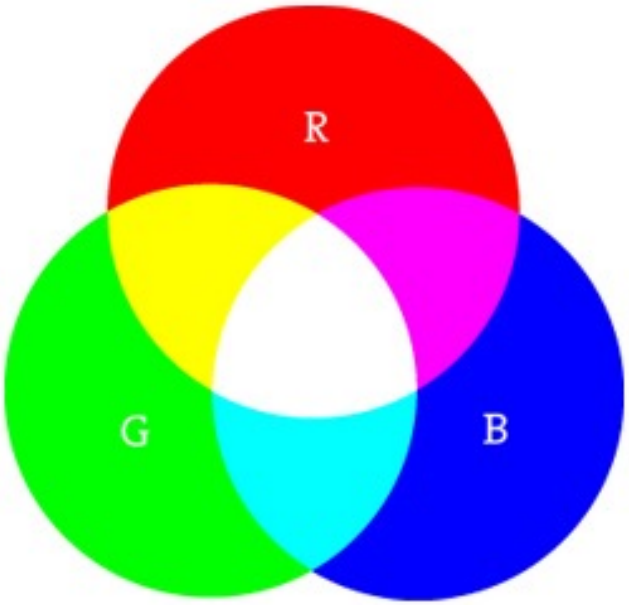
色についての基本１

演習

```
size(200, 200);
background(255);
stroke(0);
fill(150);
rect(50,50,75,100);
```

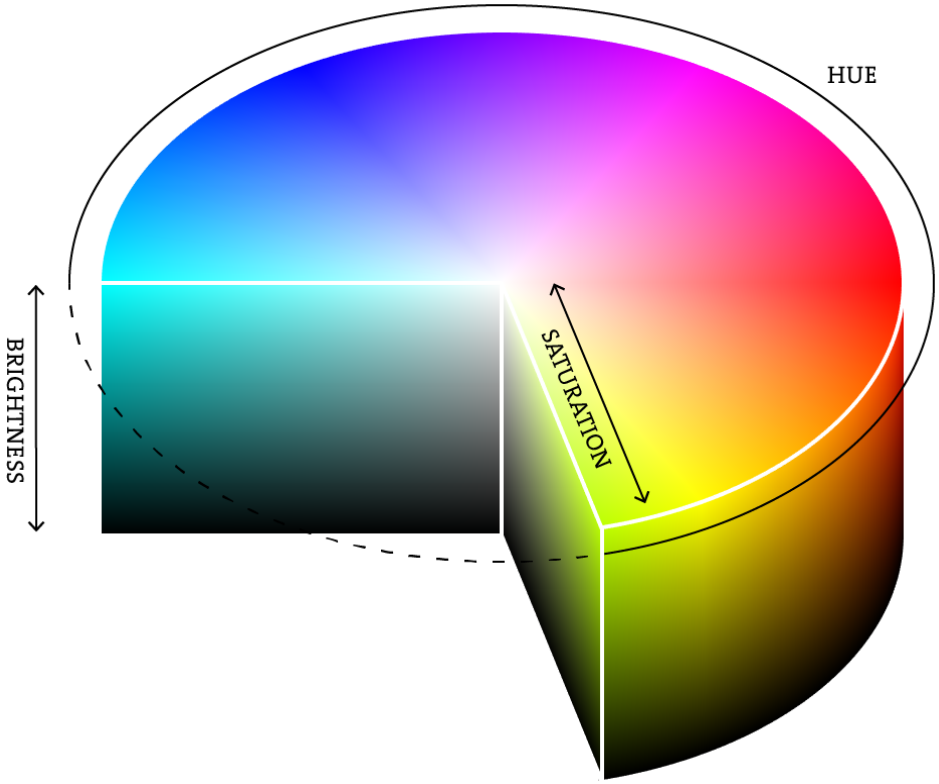
色についての基本２ カラーモード

RGBモード



Red,Green,Blue明るさを
指定して色を決定

HSBモード



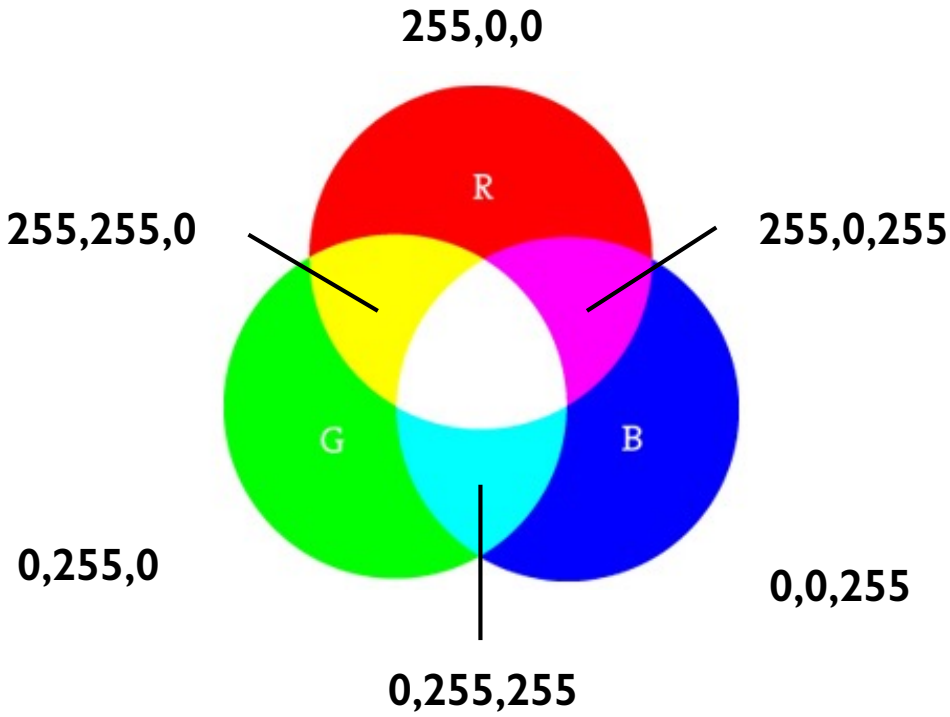
色相（hue）明度（brightness）
彩度（saturation）を指定して
色を決定

色についての基本２ カラーモード

RGBモードをセットする

```
colorMode(RGB,255,255,255,255);
```

R,G,B,αの最大値

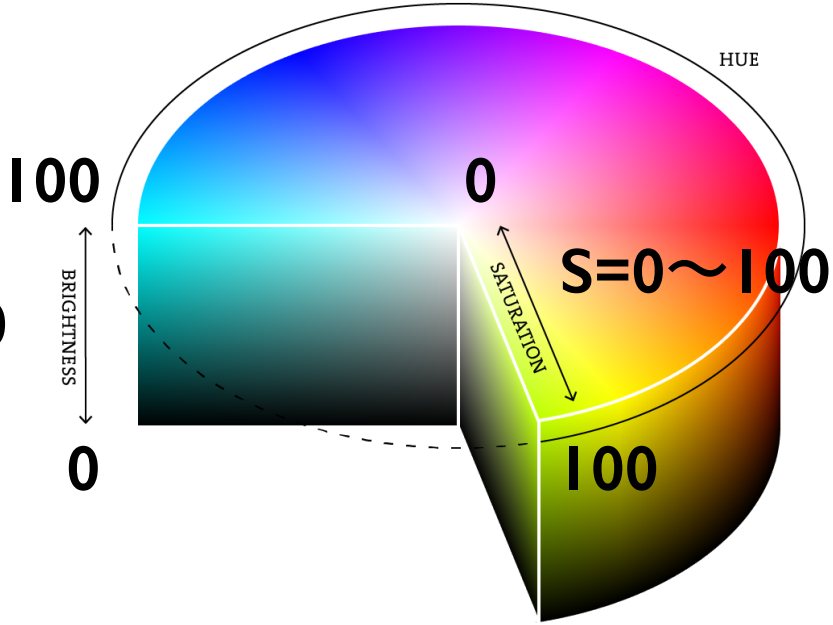


色についての基本２ カラーモード

HSBモードをセットする

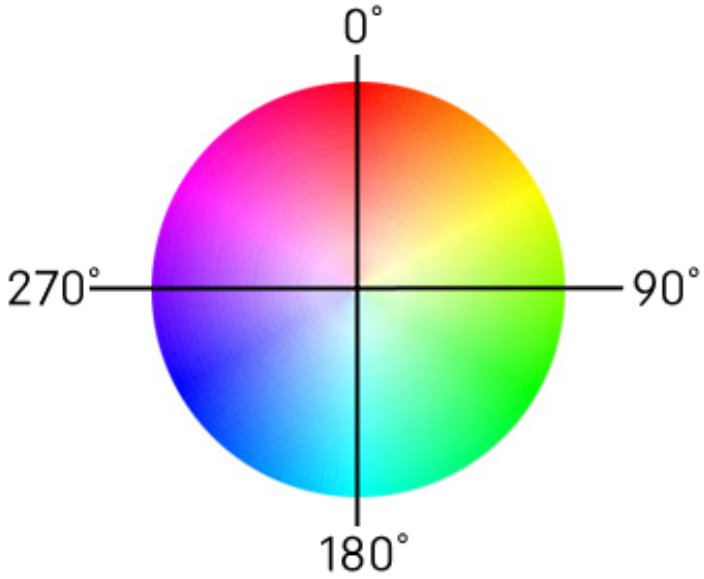
```
colorMode(HSB,360,100,100,100);
```

H,S,B,αの最大値



B=0～100

S=0～100



H=0～360

色についての基本２ カラーモード

```
size(400,400);

colorMode(RGB,255,255,255,255);
fill(255,0,0,255);  //fill(R,G,B,alpha);
ellipse(100,200,300,300);

colorMode(HSB,360,100,100,100);  //fill(H,S,B,alpha);
fill(115,100,100,100);
ellipse(300,200,300,300);
```

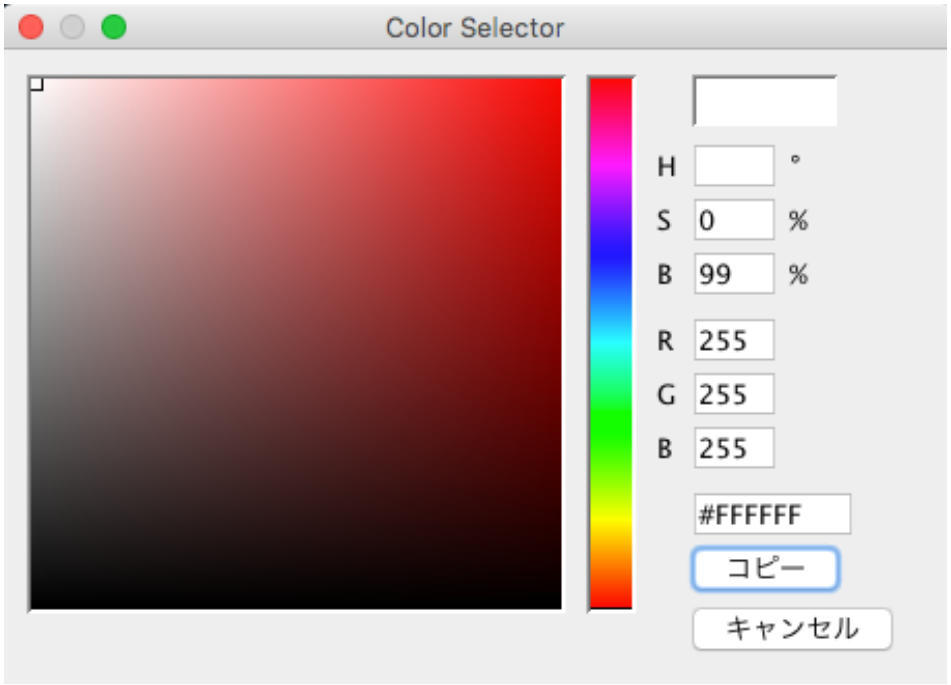
fillやstrokeの第4パラメータはalpha（不透明度）を指定できる

255＝不透明

0＝完全な透明

色についての基本２ カラーモード

Color Selector



Processingの「ツール」＞「色選択...」で呼び出せる。

マウスでピックアップした色のRGB値を調べることができる。

色についての基本２ カラーモード

パラメータをRGBで指定し、任意の色で描画する

```
size(400,400);
colorMode(RGB,255,255,255,255);

fill(255,0,0,255);
ellipse(100,200,300,300);

fill(0,255,0,255);
ellipse(300,200,300,300);
```

変数

変数とは

さまざまなデータを格納

データを使って演算や描画に利用

```
int a;
int b;
int c;
a = 10;
b = 5;
c = a + b;
println(c);
```

変数とは

```
int a;
int b;
int c;
a = 10;
b = 5;
c = a + b;
println(c);
```

変数の宣言

```
int a;
```

整数型の変数aが使用できるようになる

```
int a, b, c;
```

一度に複数の変数を宣言することも可能

代入

```
a = 10;
```

aに整数値10を代入する

```
int a = 10;
```

宣言と代入を同時に行うことも可能

変数とは

```
int a;
int b;
int c;
a = 10;
b = 5;
c = a + b;
println(c);
```

演算

```
c = a + b;
```

変数aの値と変数bの値を加算してcに代入

変数の値を表示

```
println(c);
```

```
println(a+b);
```

コンソールに変数の値を表示するprintlnメソッド

変数のデータ型

基本のデータ型

型の種類	名前	例
整数型	int	5, 100
少数型	float	3.14, 10.5
文字型	char	'A'
真偽値型	boolean	true, false
カラー型	color	RGBの値

変数のデータ型

基本のデータ型を拡張した「オブジェクト」

オブジェクトの種類	名前	例
テキスト	String	"hello world"
画像	PImage	画像ファイル
フォント	PFont	フォントファイル

ビルトイン変数

予め用意されている変数

宣言や代入は必要なく、特定の値を常に保持している

よく使うビルトイン変数

`width`	ウィンドウの幅
`height`	ウィンドウの高さ
`mouseX`	マウスポインタの現在位置の`x`座標
`mouseY`	マウスポインタの現在位置の`y`座標

代入や計算

tutorial000「変数の代入や計算」

https://github.com/CreativeVisualization/texts-tutorials/blob/master/tutorials_processing/tutorial_000_variables/tutorial_000_variables.pde

データ型の異なる代入や計算

intに少数を代入するとエラーになる。

floatに整数を代入すると勝手に少数に変換される。10→10.0

```
int a = 20;
float b = 10;
float c;
c = a + b;
println(c);
```

キャストについて

強制的に変数のデータ型を変換する。

float型aを整数型の値を一時的に整数型にする (int)a

```
float a = 3.14;
int b;
b = (int)a;
println(b);
println(a);
```

演算の省略記法

変数aに定数を加算する

```
a = a +5;
a += 5
```

変数aに1を加算する

```
a = a +1;
a++;
```

変数aから定数を減算する

```
a = a - 5;
a -= 5;
```

変数aから1を減算する

```
a = a -1;
a--;
```

演習I

縁の色は青、塗りつぶしの色は赤で円（ellipse）を描く。
ただし、縁の色、塗りつぶしの色、中心座標、幅、高さには変数を用いること。
座標や幅高さに用いる変数は、整数型か少数型か、両方で宣言して確かめる。

ヒント

```
fill(c1);
stroke(c2);
ellipse(x,y,w,h);
```

演習2

演習1で使用した変数を流用してもうひとつの円を描く

ヒント

```
fill(c1);
stroke(c2);
ellipse(x,y,w,h);

fill(c2);
stroke(c1);
ellipse(y,x,h,w);
```

乱数について

自分で数値を代入すれば意図した状態しか作り出せない。
乱数を発生させ数値の代入をコンピュータに任せてみる。
（ただし、一定の制御は必要。）
rondomメソッドを使う。

```
float result;
result = random(0,10);
println(result);
```

乱数について

0〜10の間のランダムな数値が少数型で発生することがわかる。

整数値を得たい場合は、intメソッドで変換する。

```
int result;
result  = int(random(0,10));
println(result);
```

演習3

実行するごとにランダムな位置に描かれるように演習2のプログラムを改良する。