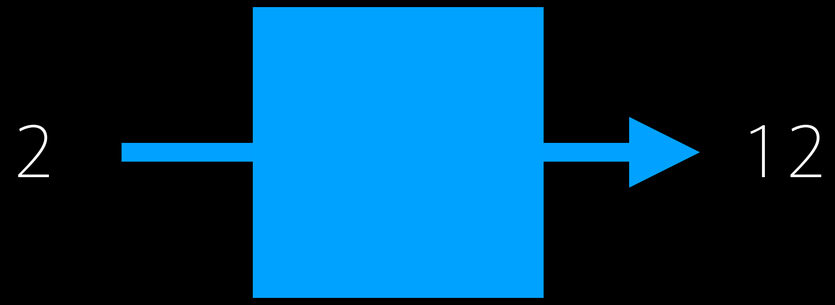
Design Programming

Spring 2018 - 7

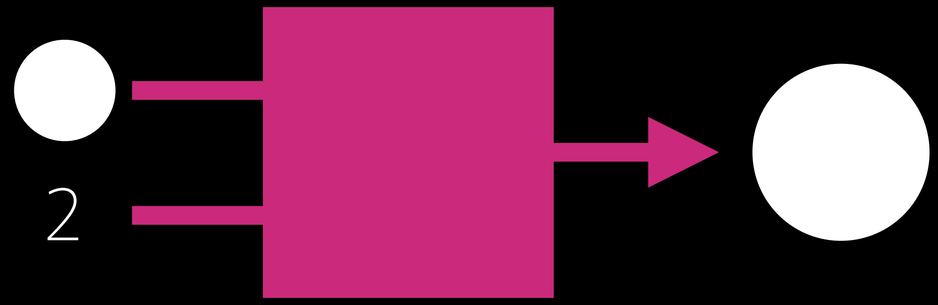
2018.5.30 Keio University, SFC

関数とは

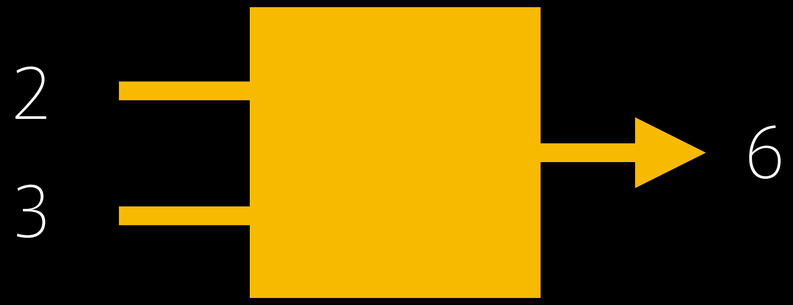
プログラムの最小単位の部品



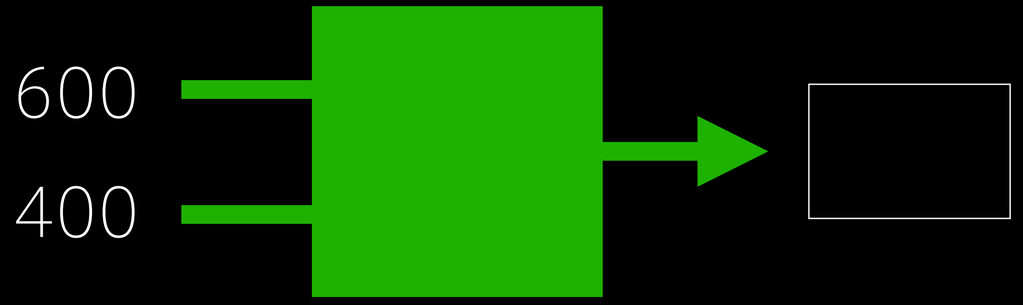
10を足す



拡大縮小



かけあわせる

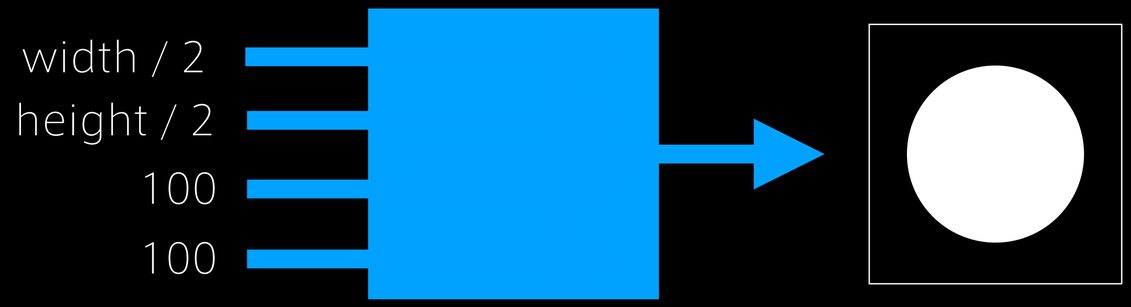


画面サイズを決める

抽象化

楕円を描く関数の中身は知らない

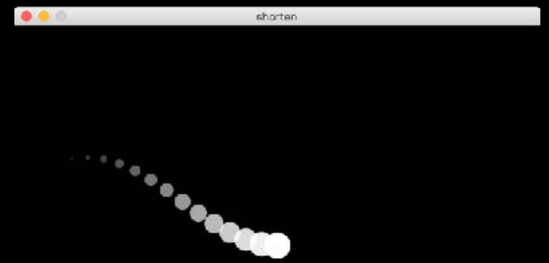
結果に注目するために細部は隠されている



楕円を描く

プログラムを短くする

```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    for (int i = 0; i < 15; i++) {
        fill(255, (15 - i) * 255 / 15);
        ellipse(
            width / 2 - 15 * i * width / 500,
            height / 2 + 50 * cos(radians(15 * i)),
            (15 - i) * 2,
            (15 - i) * 2
            );
    }
}
```



07/shorten/shorten.pde

プログラムを短くする

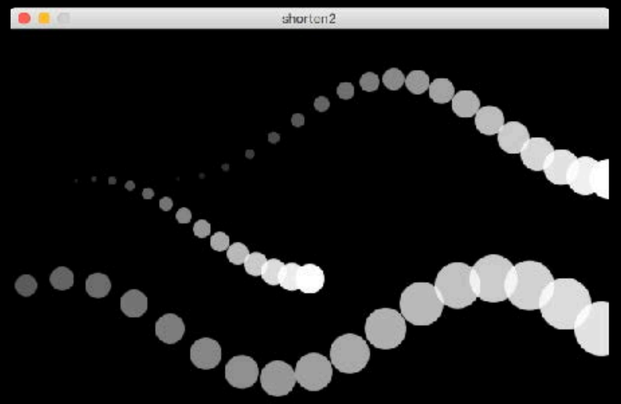
```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    for (int i = 0; i < 15; i++) {
        fill(255, (15 - i) * 255 / 15);
        ellipse(
            width / 2 - 15 * i * width / 500,
            height / 2 + 50 * cos(radians(15 * i)),
            (15 - i) * 2,
            (15 - i) * 2
            );
    }

        for (int i = 0; i < 20; i++) {
        fill(255, (20 - i) * 255 / 20);
        ellipse(
            width - 20 * i * width / 500,
            height / 4 + 50 * cos(radians(20 * i)),
            (20 - i) * 2,
            (20 - i) * 2
            );
    }

        for (int i = 0; i < 30; i++) {
        fill(255, (30 - i) * 255 / 30);
        ellipse(
            width + 100 - 30 * i * width / 500,
            3 * height / 4 + 50 * cos(radians(30 * i)),
            (30 - i) * 2,
            (30 - i) * 2
            );
    }
}
```

- 行数がかなり増えてしまう

  →可読性が低い

  →修正の際，書き換え箇所が増える

- 拡張する際また行数が増える



07/shorten2/shorten2.pde

プログラムを短くする

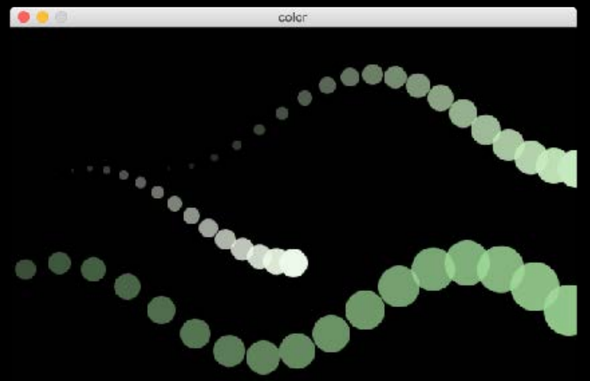
```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    for (int i = 0; i < 15; i++) {
        fill(255, (15 - i) * 255 / 15);
        ellipse(
            width / 2 - 15 * i * width / 500,
            height / 2 + 50 * cos(radians(15 * i)),
            (15 - i) * 2,
            (15 - i) * 2
            );
    }

    for (int i = 0; i < 20; i++) {
        fill(255, (20 - i) * 255 / 20);
        ellipse(
            width - 20 * i * width / 500,
            height / 4 + 50 * cos(radians(20 * i)),
            (20 - i) * 2,
            (20 - i) * 2
            );
    }

    for (int i = 0; i < 30; i++) {
        fill(255, (30 - i) * 255 / 30);
        ellipse(
            width + 100 - 30 * i * width / 500,
            3 * height / 4 + 50 * cos(radians(30 * i)),
            (30 - i) * 2,
            (30 - i) * 2
            );
    }
}
```

```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    nyoro(15, width / 2, height / 2);
    nyoro(20, width, height / 4);
    nyoro(30, width + 100, 3 * height / 4);
}

void nyoro(int n, int x, int y) {
    for (int i = 0; i < n; i++) {
        fill(255, (n - i) * 255 / n);
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i)),
            (n - i) * 2,
            (n - i) * 2
            );
    }
}
```

07/shorten3/shorten3.pde

色のパラメータを与える

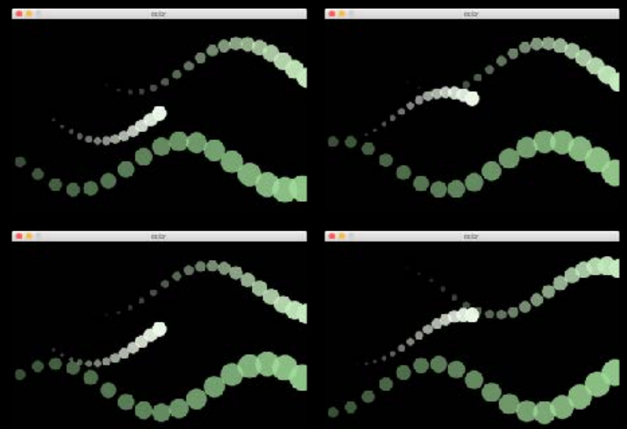
```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    nyoro(15, width / 2, height / 2, color(237, 248, 233));
    nyoro(20, width, height / 4, color(199, 233, 192));
    nyoro(30, width + 100, 3 * height / 4, color(161, 217, 155));
}

void nyoro(int n, int x, int y, color c) {
    for (int i = 0; i < n; i++) {
        fill(c, (n - i) * 255 / n);
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i)),
            (n - i) * 2,
            (n - i) * 2
            );
    }
}
```



07/color/color.pde

アニメーションさせる

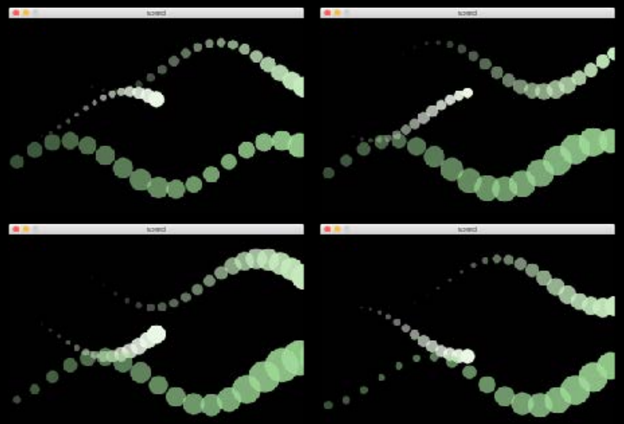
```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    nyoro(15, width / 2, height / 2, color(237, 248, 233), 1.0);
    nyoro(20, width, height / 4, color(199, 233, 192), 3.0);
    nyoro(30, width + 100, 3 * height / 4, color(161, 217, 155), 0.4);
}

void nyoro(int n, int x, int y, color c, float sp) {
    for (int i = 0; i < n; i++) {
        fill(c, (n - i) * 255 / n);
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i + frameCount * sp)),
            (n - i) * 2,
            (n - i) * 2
            );
    }
}
```



07/speed/speed.pde

粒の大きさにノイズを与える

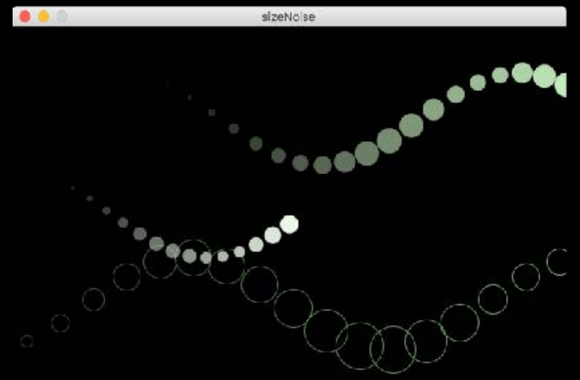
```
void setup() {
    size(600, 400);
    noStroke();
}

void draw() {
    background(0);
    nyoro(15, width / 2, height / 2, color(237, 248, 233), 1.0);
    nyoro(20, width, height / 4, color(199, 233, 192), 3.0);
    nyoro(30, width + 100, 3 * height / 4, color(161, 217, 155), 0.4);
}

void nyoro(int n, int x, int y, color c, float sp) {
    for (int i = 0; i < n; i++) {
        fill(c, (n - i) * 255 / n);
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i + frameCount * sp)),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100)
            );
    }
}
```



07/sizeNoise/sizeNoise.pde

塗りのパラメータを与える

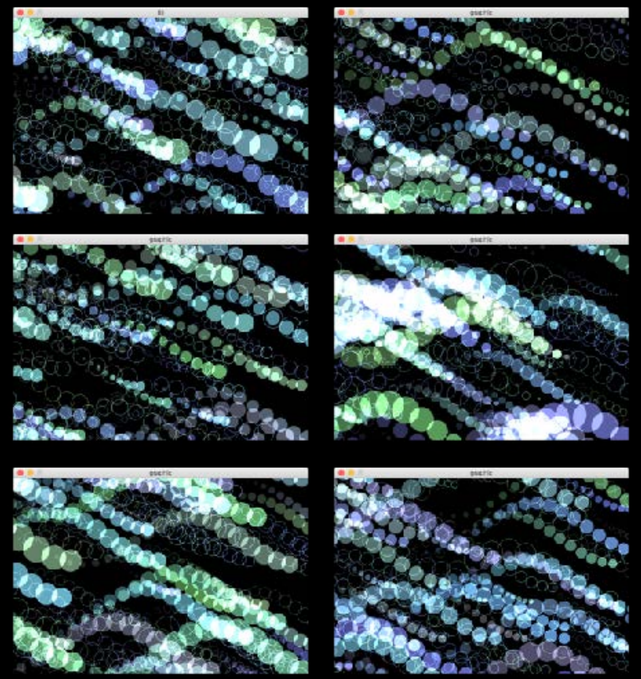
```
void setup() {
    size(600, 400);
}

void draw() {
    background(0);
    nyoro(15, width / 2, height / 2, color(237, 248, 233), 1.0, true);
    nyoro(20, width, height / 4, color(199, 233, 192), 3.0, true);
    nyoro(30, width + 100, 3 * height / 4, color(161, 217, 155), 0.4, false);
}

void nyoro(int n, int x, int y, color c, float sp, Boolean fill) {
    for (int i = 0; i < n; i++) {
        if (fill) {
            noStroke();
            fill(c, (n - i) * 255 / n);
        } else {
            noFill();
            stroke(c, (n - i) * 255 / n);
        }
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i + frameCount * sp)),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100)
            );
    }
}
```



07/fill/fill.pde

グラフィックパターン

```
void setup() {
    size(600, 400);
    noLoop();
    blendMode(ADD);
}

void draw() {
    background(0);
    for (int i = 0; i < 100; i++) {
        nyoro(
            int(random(10, 30)),
            int(random(- 100, width + 100)),
            int(random(-100, height + 100)),
            color(random(100, 110), random(100, 180), random(100, 220)),
            int(random(10)),
            boolean(int(random(2)))
            );
    }
}

void nyoro(int n, int x, int y, color c, float sp, Boolean fill) {
    for (int i = 0; i < n; i++) {
        if (fill) {
            noStroke();
            fill(c, (n - i) * 255 / n);
        } else {
            noFill();
            stroke(c, (n - i) * 255 / n);
        }
        ellipse(
            x - n * i * width / 500,
            y + 50 * cos(radians(n * i + frameCount * sp)),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100),
            (n - i) * 4 * noise(float(i * 10 + frameCount) / 100)
            );
    }
}
```



07/graphic/graphic.pde

返り値

なにか描画するためではなく，

関数の実行結果として「数値」や「その他のデータ型」を受け取る

そのデータのことが「返り値」

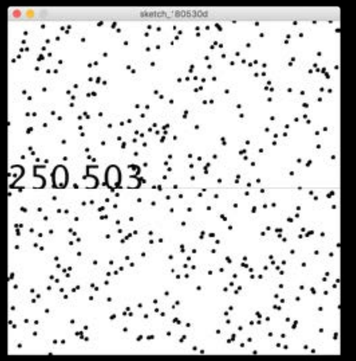
返り値なし	返り値あり
`size();`	`random();`
`ellipse();`	`noise();`
`println();`	`second();`

返り値

```
void setup() {
    size(500, 500);
    fill(0);
    textSize(50);
}

void draw() {
    background(255);
    int sum = 0;
    for(int i = 0; i < width; i++) {
        int rand = int(random(height));
        ellipse(i, rand, 5, 5);
        sum += rand;
    }
    line(0, average(width, sum), width, average(width, sum));
    text(str(average(width, sum)), 0, average(width, sum));
}

float average(int n, int s) {
    float avg;
    avg = (float)s / (float)n;
    return avg;
}
```

型を明記する.

returnを使って返したい値を指定する.



07/returnFunc/returnFunc.pde

演習1

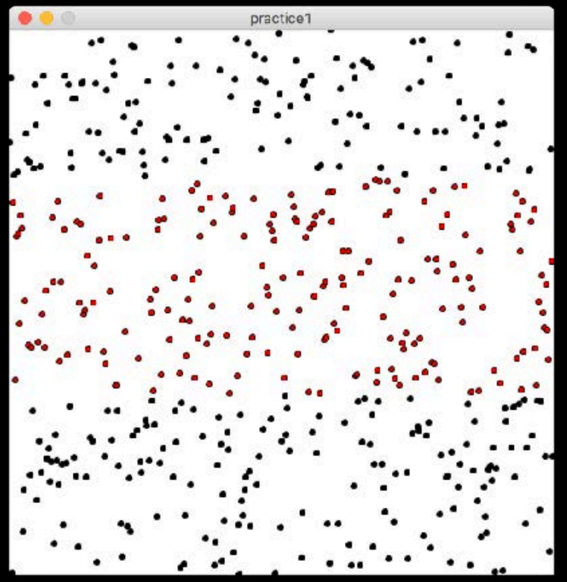
07/returnFunc/returnFunc.pdeを利用して,

縦が前フレームの平均値から100以内の場合塗り色が赤を返す関数を書き,

描画に適用しなさい.

演習1

```
int prevSum = 0;

void setup() {
    size(500, 500);
    fill(0);
}

void draw() {
    background(255);
    int sum = 0;
    for (int i = 0; i < width; i++) {
        int rand = int(random(height));
        fill(separate(width, prevSum, rand));
        ellipse(i, rand, 5, 5);
        sum += rand;
    }
    prevSum = sum;
}

color separate(int n, int s, int y) {
    float avg;
    avg = (float)s / (float)n;
    color c;
    if (abs(y - avg) < 100) {
        c = color(255, 0, 0);
    } else {
        c = color(0);
    }
    return c;
}
```

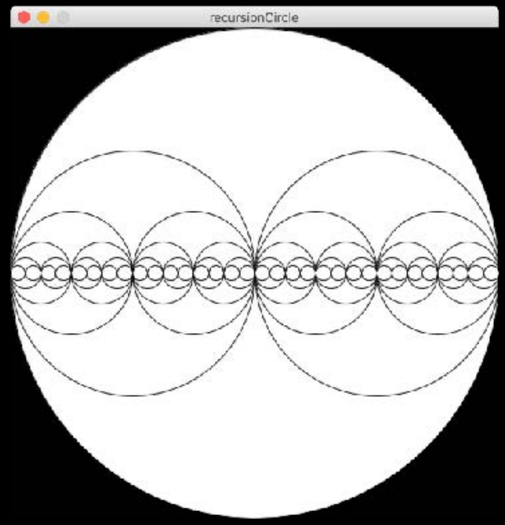


07/practice1/practice1.pde

再帰

```
void setup() {
    size(500, 500);
}

void draw() {
    background(0);
    circles(width / 2, width / 2, width / 2, 5);
}

void circles(float x, float y, float r, int n) {
    ellipse(x, y, r * 2, r * 2);
    if (n > 0) {
        float nextR = r / 2;
        circles(x + nextR, y, nextR, n - 1);
        circles(x - nextR, y, nextR, n - 1);
    }
}
```

関数の中で関数を再帰的に呼び出す



07/recursionCircle/recursionCircle.pde

課題

- 今日学習した「関数の利用，返り値，再帰」を使用してスケッチを１つアップロードしなさい.
  自作の関数を必ず１つは使用すること.
  アップロード先は「07 function」とする.
  ※今日学習していない分野でもすでに自分が知っている技術は使用可能とする.

  〆切：6月5日24時まで

次回

- 配列とアニメーション