

---

# プログラミング演習 (0)

## イントロダクション

---

中村、小林、三武、辻野

# まずは

---

- 教員とTAの紹介

# 次に

---

- 近くのひとと自己紹介して下さい

# Slackに入りました

---

- Slack にログインして「#講義-プログラミング演習2023」のチャンネルに入り、氏名が「B1-組-番号 名前」、表示名がアルファベット表記になっていることを確認する
  - ブラウザからは `fms-all.slack.com` へ
- 通知は適切にOFFにしておくが良いと思います（自分宛てとDMのみをONにする感じで）

# Gmailのアドレス収集

---

- Gmail のアドレスを提出する（課題提出に使うので超重要です！）
  - <https://forms.gle/2VUYAUQKwhaTrDXT7>
- これができないと課題提出できないので確実にやってください

# 何故 Java や C じゃないの？

---

- プログラミング嫌いを量産したくないから
  - 最初にJavaやCを学んだ学生さんの多くが挫折し、プログラミング自体を嫌いになってしまう
  - 苦手意識を払拭できず、コンピュータ自体も嫌いに
  - 一度嫌いになったら、そんじょそこらの教え方ではプログラミングできるようになりません...
- 例えば、プログラミングでウィンドウを出すこと自体が大変で、心が折れてしまうものです

# JavaやC言語でウインドウ

## Java

```
import java.awt.*;

public class Window01 {
    public static void main(String args[]){
        WindowTest windowText=new WindowTest();
    }
}

class WindowTest{
    WindowTest(){
        Frame frame;
        frame=new Frame("Window Test");
        frame.setSize(240, 240);
        frame.setVisible(true);

        Label label;
        label=new Label("Hellow Window");
        frame.add(label);
    }
}
```

## C言語

```
#include <windows.h>
#define WNDCLASSNAME TEXT("SampleClass")
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg,
{
    switch (msg) {
        case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
    }
    return DefWindowProc(hWnd, msg, wp, lp);
}

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, PSTR lpCmdLine, int
nCmdShow)
{
    HWND hWnd;
    WNDCLASS wc;
    MSG msg;
    wc.style          = CS_HREDRAW | CS_VREDRAW;
    wc.lpfWndProc     = WindowProc;
    wc.cbClsExtra     = 0;
    wc.cbWndExtra     = 0;
    wc.hInstance      = hInst;
    wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName   = NULL;
    wc.lpszClassName  = WNDCLASSNAME;

    if (!RegisterClass(&wc)) return 0;
    hWnd = CreateWindow(
        WNDCLASSNAME,
        TEXT("SampleWindow"),
        WS_OVERLAPPEDWINDOW | WS_VISIBLE,
        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
```

正直心が折れます

# もちろん

---

- Java や C言語の良いところは沢山
  - 高速に動作する
  - 色々な環境での開発に利用される
    - Windows, Apple, Linux, Web, SmartPhone, ...
  - ライブラリ（サポートしてくれるもの）が豊富
  - 多くの人が開発に利用している
  
- 2年次にJavaは扱う予定

# 何故 Processing か

---

- 視覚的なプログラム作成が容易
- プログラムの基礎を学ぶのに適している
- 最先端の研究でも実を言うと使われている
  - FMS学科の研究でもかなり使われています
- Processing は Java の上で動作しているものであり、Java にととても似ている
- Processing を勉強した後、Java を勉強するのはとても簡単！

# ちなみに

---

- Processingは描画だけではなく問題を解くことにも使える！

$$(3) \quad (x + y)(y + z)(z + x) = 0, \quad x + y + z = 20, \quad xyz \neq 0, \quad |x| \leq 20,$$

$|y| \leq 20, |z| \leq 20$  を満たす整数  $x, y, z$  の組は全部で **カキク** 個ある。

# ちなみに

- Processingは描画だけではなく問題を解くことにも使える！

(2)  $xyz = -240$  を満たす整数  $x, y, z$  の組は全部で **ウエオ** 個ある。

また  $xyz = -240$  かつ  $x + y + z = 0$  を満たす整数  $x, y, z$  の組は全部で

**カキ** 個ある。

[IV] 数列  $\{a_n\}$  は以下の条件 (i), (ii), (iii) を満たすとする。ただし、その第  $n$  項  $a_n$  を  $a(n)$  と表すことにする。

(i) すべての自然数  $n$  に対して、 $a(n)$  は自然数である。

(ii)  $n$  が 2017 以下の自然数のとき、 $a(n) = a(a(n + 201))$  である。

(iii)  $n$  が 2018 以上の自然数のとき、 $a(n) = n - 200$  である。

以下の問いに答えよ。

(1)  $a(2017)$  を求めよ。

(2)  $n$  が 1817 以上かつ 2017 以下の自然数のとき、 $a(n) = a(2017)$  が成り立つことを示せ。

(3)  $n$  が 1816 以下の自然数のとき、 $a(n) = a(2017)$  が成り立つことを示せ。

# 超重要

---

- ウェブ上のプログラムや他人のプログラムをコピーして使うことは厳禁！
  - プログラミング演習はプログラムが書けるようになることを目的としています！
  - 写すと時間とお金の無駄です！！
- 他人のプログラムを写しているような行為が発見された場合は、提供した人も、コピーした人も両者をカンニング扱いとします
  - コピーする人は本当に稚拙な書き換えしかしませんので要注意！！
- 「見せない」「写させない」が重要です

# 何故？

---

- X年前の約25%の学生がコピーに該当
  - もちろん、悪意のあったものはここまで多くはありませんが、そういうのを普通であるとしてらえている人が多いという事実がありました
- 他の講義でもレポートのコピーが問題に

# プログラミング

---

- ある目的を達成するプログラムの書き方は山のように存在している
  - ある積み木セットを利用して、4つの塔がある城を作れという課題が与えられたときに同じものができただろうか？
- また、プログラミングは文章を書くことと似ているため、そこには個性が現れる
  - 文章が一致することはとても少ない

# コピーとは？

---

- 他人から完成したプログラム自体を受け取り、それに何らかの手を加えること提出
- 他人から完成したプログラムの写真を受け取って、それを写して手を加えること提出
- 他人に完成したプログラムをディスプレイに表示し続けてもらい、それを写して手を加え提出

# 例

```
5 ←
6 float x=300; ←
7 float y=300; ←
8 int COLOR=0; ←
9 ←
10 void setup() { ←
11   size(800, 800); ←
12 } ←
13 ←
14 void draw() { ←
15   background(255); ←
16   noStroke(); ←
17   if ( COLOR%3==0 ) { ←
18     fill(255, 0, 0); ←
19   } else if ( COLOR%3==1 ) { ←
20     fill (0, 255, 0); ←
21   } else { ←
22     fill(0, 0, 255); ←
23   } ←
24   rect(x, y+100, 300, 100); ←
25   rec
26 } ←
27 ←
28 void
29 if
30 (
31   x=random(0, 501); ←
32   y=random(0, 501); ←
33 } ←
34 } [EOF]
```

```
5 float x=300; ←
6 float y=300; ←
7 int COLOR=0; ←
8 ←
9 void setup() { ←
10   size(800,800); ←
11 } ←
12 ←
13 void draw() { ←
14   background(255); ←
15   noStroke(); ←
16   if (COLOR%3==0) { ←
17     fill(255,0,0); ←
18   } else if (COLOR%3==1) { ←
19     fill(0,255,0); ←
20   } else { ←
21     fill(0,0,255); ←
22   } ←
23   rect(x,y+100,300,100); ←
24   rect(x+100,y,100,300); ←
25   && mous
31   y=random(0,501); ←
32 } ←
33 } ←
34 } [EOF]
```

COLORという変数を使ったのは2人だけ

```
float a;←  
float t;←  
float v;←  
float x;←  
float y;←
```

```
void mousePressed() {←  
  if (dist(mouseX, mouseY, 100, 100)  
    a=1;←  
}
```

```
void setup() {←  
  size(1000, 800);←  
  a=0;←  
  t=0;←  
  x=800;←  
  y=400;←  
  v=60;←  
}
```

```
void draw() {←  
  background(255);←  
  fill(255, 255, 255);←  
  ellipse(x, y, 150, 150);←  
  fill(255, 255, 255);←  
  ellipse(x, y-135, 120, 120);←  
  ellipse(x-30, y-150, 30, 30);←  
  ellipse(x+30, y-150, 30, 30);←  
  fill(0, 0, 0);←  
  ellipse(x-30, y-150, 10, 10);←  
  ellipse(x+30, y-150, 10, 10);←  
  fill(255, 255, 255);←  
  triangle(x, y-100, x-20, y-130,  
  line(x+75, y, x+135, y-75);←  
  line(x-75, y, x-135, y-75);←  
  ellipse(x-40, y+80, 30, 30);←  
  ellipse(x+40, y+80, 30, 30);←  
  fill(255);←  
  stroke(0);←
```

```
float a;←  
float t;←  
float v;←  
float x;←  
float y;←
```

```
void mousePressed() {←  
  if (dist(mouseX, mouseY, 100, 100)<75  
    a=1;←  
}
```

```
void setup() {←  
  size(600, 800);←  
  a=0;←  
  t=0;←  
  x=300;←  
  y=400;←  
  v=60;←  
}
```

```
void draw() {←  
  background(255);←  
  noStroke();//輪郭←  
  fill(210, 180, 140);←  
  ellipse(x, y, 200, 200);←  
  fill(210, 180, 140);//耳←  
  ellipse(x-100, y-100, 100, 100);←  
  fill(210, 180, 140);←  
  ellipse(x+100, y-100, 100, 100);←  
  fill(245, 222, 179);//中の顔←  
  ellipse(x, y+37, 100, 75);←  
  fill(245, 222, 179);//中の耳←  
  ellipse(x+50, y, 75, 75);←  
  fill(245, 222, 179);←  
  ellipse(x-50, y, 75, 75);←  
  fill(0);//目←  
  ellipse(x-50, y, 30, 30);←  
  fill(0);←  
  ellipse(x+50, y, 30, 30);←  
  fill(255);//白目←
```

```
float a;←  
float t;←  
float v;←  
float x;←  
float y;←
```

```
void mousePressed() {←  
  if (dist(mouseX, mouseY, 100, 100)  
    a=1;←  
}
```

```
void setup() {←  
  size(600, 800);←  
  a=0;←  
  t=0;←  
  x=300;←  
  y=400;←  
  v=60;←  
}
```

```
void draw() {←  
  background(255);←  
  ellipse(100, 100, 50, 50);←  
  stroke(255, 255, 100);//手←  
  strokeWeight(50);←  
  line(x-250, y-50, x-100, y-50);←  
  stroke(255, 255, 100);←  
  strokeWeight(50);←  
  line(x+100, y-50, x+250, y-50);←  
  stroke(255, 255, 255);←  
  stroke(255, 255, 100);//足←  
  strokeWeight(50);←  
  line(x-50, y+50, x-50, y+200);←  
  stroke(255, 255, 100);←  
  strokeWeight(50);←  
  line(x+50, y+50, x+50, y+200);←  
  strokeWeight(0);←
```

```
int card;←
int cardmai=0;←
int cardmai2=0;←
int cardmai3=0;←
int cardmai4=0;←
int cardmai5=0;←
float kyori;←
void setup(){←
  size( 900, 500 );←
}←
←
void draw(){←
  background(#999999);←
  if(cardmai>=1 && cardmai2>=1 && cardmai3>=1 && cardmai
    fill(255,0,0);←
    textSize(50);←
    text("Complete",350,400);←
  }else{fill(255);}←
  rect(40,50,140,250);←
  rect(200,50,140,250);←
  rect(360,50,140,250);←
  rect(520,50,140,250);←
  rect(680,50,140,250);←
  fill(0);←
  rect(40,400,170,70);←
  textSize(60);←
  text(cardmai,80,180);←
  text(cardmai2,250,180);←
  text(cardmai3,410,180);←
  text(cardmai4,570,180);←
  text(cardmai5,730,180);←
  fill(255);←
  textSize(50);←
  text("Gacha",50,440);←
}←
void mousePressed(){←
  if(mouseX>40 && mouseX<=210 && mouseY>400 && mouseY<4
    card=(int)random(1,6);←
    if(card==0){card=(int)random(1,6);}←
    if(card==1){←
      cardmai=cardmai+1;}else if(card==2){cardmai2=cardmai
```

```
int cardm=0;←
int cardm2=0;←
int cardm3=0;←
int cardm4=0;←
int cardm5=0;←
float kyori;←
void setup(){←
  size( 900, 500 );←
}←
←
void draw(){←
  background(#999999);←
  if(cardm>=1 && cardm2>=1 && cardm3>=1 && cardm4>=1 &&
    fill(255,0,0);←
    textSize(50);←
    text("Complete",350,400);←
  }else{fill(255);}←
  rect(40,50,140,250);←
  rect(200,50,140,250);←
  rect(360,50,140,250);←
  rect(520,50,140,250);←
  rect(680,50,140,250);←
  fill(0);←
  rect(40,400,170,70);←
  textSize(60);←
  text(cardm,80,180);←
  text(cardm2,250,180);←
  text(cardm3,410,180);←
  text(cardm4,570,180);←
  text(cardm5,730,180);←
  fill(255);←
  textSize(50);←
  text("Gacha",50,440);←
}←
void mousePressed(){←
  if(mouseX>40 && mouseX<=210 && mouseY>400 && mouseY<
    card=(int)random(1,6);←
    if(card==0){←
      card=(int)random(1,6);}←
    }←
```

# 特徴的なインデント

---

```
    text ("Gacha", 50, 440); ←  
    } ←  
void mousePressed() { ←  
←
```

```
    text ("Gacha", 50, 440); ←  
    } ←  
void mousePressed() { ←  
←
```

# 特徴的なスペース

---

```
17 | int flag; ←  
18 | int t ; ←  
19 | int end; ←
```

```
17 | int flag; ←  
18 | int t ; ←  
19 | int end; ←
```

```
5 int click=0;←
6 float t=0;←
7 float y=0;←
8 int x=500;←
9 ←
10 void setup() {←
11   size( 1000, 1000 );←
12 }←
13 ←
14 void draw() {←
15   y=650-1*(70*t-9.8*t*t/2);←
16   background( 255 );←
17   strokeWeight(5);←
18   noFill();←
19   ellipse(50, 50, 40, 40);←
20   line(0, height-50, 1000, height-50);←
21   drawCharacter1();←
22   if (click==1) {←
23     background( 255 );←
24     strokeWeight(5);←
25     noFill();←
26     ellipse(50, 50, 40, 40);←
27     line(0, height-50, 1000, height-50);←
28     t=t+0.1;←
29     drawCharacter1();←
30   }←
31   if (y>650) {←
32     background( 255 );←
33     strokeWeight(5);←
34     noFill();←
35     ellipse(50, 50, 40, 40);←
36     line(0, height-50, 1000, height-50);←
37     t=0;←
38     click=0;←
39     drawCharacter1();←
40   }←
41 }←
42 ←
43 void mousePressed() {←
44   if (dist(50, 50, mouseX, mouseY)<20) {←
45     click=1;←
46   }←
47 }←
48
```

```
5 int click=0;←
6 float t=0;←
7 float y=0;←
8 int x=500;←
9 ←
10 void setup() {←
11   size(800, 900);←
12 }←
13 ←
14 void draw() {←
15   y=585-1*(70*t-9.8*t*t/2);←
16   background(255);←
17   strokeWeight(5);←
18   noFill();←
19   ellipse(50, 50, 40, 40);←
20   line(0, height-50, 900, height-50);←
21   drawCharacter1();←
22   if (click==1) {←
23     background(255);←
24     strokeWeight(5);←
25     noFill();←
26     ellipse(50, 50, 40, 40);←
27     line(0, height-50, 900, height-50);←
28     t=t+0.1;←
29     drawCharacter1();←
30   }←
31   if (y>585) {←
32     background(255);←
33     strokeWeight(5);←
34     noFill();←
35     ellipse(50, 50, 40, 40);←
36     line(0, height-50, 800, height-50);←
37     t=0;←
38     click=0;←
39     drawCharacter1();←
40   }←
41 }←
42 ←
43 void mousePressed() {←
44   if (dist(50, 50, mouseX, mouseY)<20) {←
45     click=1;←
46   }←
47 }←
48
```

# 行数は違えど…

```
5 float t=0;↵
6 void setup(){↵
7   size(1000,600);↵
8   background(255);↵
9   fill(255,0,0);↵
10  ellipse(800,height-100,20,20);↵
11  ellipse(900,height-100,20,20);↵
12  fill(255);↵
13 }↵
14 void draw(){↵
15   float x=100*t*cos(radians(58));↵
16   float y=600-100*t*sin(radians(58))+4.9*t*t;↵
17   ellipse(x,y,20,20);↵
18   t=t+0.1;↵
19   ↵
20 } [EOF]
```

```
5 float t=0;↵
6 ↵
7 void setup() {↵
8 ↵
9   size(1000, 600);↵
10  background(255);↵
11  fill(255, 0, 0);↵
12  ellipse(800, height-100, 20, 20);↵
13  ellipse(900, height-100, 20, 20);↵
14  fill(255);↵
15 }↵
16 void draw() {↵
17   fill(255);↵
18 ↵
19   float x=100*t*cos(radians(58));↵
20 ↵
21   float y=600-100*t*sin(radians(58))+4.9*t*t;↵
22 ↵
23   ellipse(x, y, 10, 10);↵
24 ↵
25   t=t+0.1;↵
26 } [EOF]
```

```
5 int diceA=1;←
6 int diceB=5;←
7 ←
8 ←
9 void setup() {←
10 size(600, 400);←
11 }←
12 ←
13 void draw() {←
14 background(255);←
15 //画面を豪華にしよう!←
16 if (diceA==diceB) {←
17 background(255, 255, 0);←
18 textAlign(CENTER);←
19 fill(255, 0, 0);←
20 textSize(30);←
21 text("NICE!!!!", 300, 300);←
22 }←
23 stroke(0);←
24 strokeWeight(3);←
25 fill(255);←
26 //diceA←
27 rect(150, 150, 100, 100);←
28 //diceB←
29 rect(350, 150, 100, 100);←
30 ←
31 noStroke();←
32 //////////////////////////////////////////////////←
33 //dice check←
34 switch( diceA ) {←
35 case 1:←
36 fill(255, 0, 0);←
37 ellipse(200, 200, 20, 20);←
38 break;←
39 ←
40 case 2:←
41 fill(0);←
42 ellipse(175, 175, 20, 20);←
43 ellipse(225, 225, 20, 20);←
44 break;←
45 ←
46 case 3:←
47 fill(0);←
48 ellipse(175, 225, 20, 20);←
49 ellipse(200, 200, 20, 20);←
50 ellipse(225, 175, 20, 20);←
51 break;←
```

```
5 int diceA=1;←
6 int diceB=5;←
7 ←
8 void setup() {←
9 size(600,600);←
10 background(255);←
11 }←
12 void draw(){←
13 if (diceA==diceB) {←
14 background(255, 255, 0);←
15 textAlign(CENTER);←
16 fill(255, 0, 0);←
17 textSize(30);←
18 text("OK!!", 300, 300);←
19 }else{←
20 background(255);←
21 }←
22 stroke(0);←
23 strokeWeight(3);←
24 fill(255);←
25 //diceA←
26 rect(150, 150, 100, 100);←
27 //diceB←
28 rect(350, 150, 100, 100);←
29 ←
30 noStroke();←
31 switch(diceA) {←
32 case 1:←
33 fill(255, 0, 0);←
34 ellipse(200, 200, 20, 20);←
35 break;←
36 ←
37 case 2:←
38 fill(0);←
39 ellipse(175, 175, 20, 20);←
40 ellipse(225, 225, 20, 20);←
41 break;←
42 ←
43 case 3:←
44 fill(0);←
45 ellipse(175, 225, 20, 20);←
46 ellipse(200, 200, 20, 20);←
47 ellipse(225, 175, 20, 20);←
48 break;←
49 ←
50 case 4:←
51 fill(0);←
```

# 思っている以上に . . .

---

- みなさんが思っている以上に、コピーさせてくれという人は稚拙な書き換えしかやりません
  - 変数名変更、数値の値変更（小説で登場人物の名前を変えたようなもの）
  - コメントの追加・削除・変更
  - その割に、特殊な変数や記述を使い続ける
- そもそももっとお馬鹿なことも
  - 他人のプログラムをそのまま出している

**情報技術を使えば簡単に近さも計算できます**

# 提供してもらわないように！

---

- みなさんがコピーしてしまうことにより、コピー提供者も迷惑を被ります
- この講義を前提に、
  - エンタテインメントプログラミング演習（1年夏）
  - プログラミング演習2（1年秋）
  - メディアプログラミング実習（2年春）
  - アルゴリズム基礎（2年春）
  - Webプログラミング実習（2年秋）
  - があるため、逃げたらずっと苦しむだけ。勉強にもなりませんのでしんどいなりに立ち向かいましょう！
- これまで英語や数学とかを勉強してきたときどうしています？ どれくらいの時間をかけてきました？

# 提供しないように！

---

- 良かれと思ってやりたくなる気持ちはわかります
- 教えるのが面倒でこれみとけって気分になることもあるでしょう
  - でも、同一レポート（課題）の提出があった場合、提供された側が正直に申告しない場合は提供元・提供先の両方が処罰の対象となります
  - そんなしょうもないことで人生を狂わせてはもったいないので、くれぐれも注意してください
- 他の講義でも問題になりつつあります。絶対にレポートはコピーさせないこと！

# これまで来た質問

---

- 教えたら類似するのでは？
  - 多少は類似するでしょう。ただ、積み木でお城を作るという課題で同じ形、同じ色の積み木を同じように積むことはあり得るでしょうか？
  - 見せなければ一致しない
- 教えることを禁止するのか？
  - ぜひとも教えて下さい、ただコピーが発生するような行為は慎んでください
- 他人のプログラムを見てはだめなのか？
  - 教えるときは見ざるを得ませんので、その時は見て下さい。ただ、見せて教えないで下さい。

# ChatGPT !

---

- OpenAIが提供するGPT-3/GPT-4をベースとしたBotとのチャットサービス
- 様々な対話が可能だが、プログラミングについても教えてもらうことが可能

# 例えばこの問題

(3)  $(x + y)(y + z)(z + x) = 0$ ,  $x + y + z = 20$ ,  $xyz \neq 0$ ,  $|x| \leq 20$ ,

$|y| \leq 20$ ,  $|z| \leq 20$  を満たす整数  $x, y, z$  の組は全部で  個ある。

## 質問の例

$(x+y)(y+z)(z+x)=0$ ,  $x+y+z=20$ ,  $xyz \neq 0$ ,  
 $|x| \leq 20$ ,  $|y| \leq 20$ ,  $|z| \leq 20$  を満たす  $x, y, z$  の  
組の数を求める Processing のプログラムを書  
いて下さい

# 完璧なプログラムを出力

S

$(x+y)(y+z)(z+x)=0$ ,  $x+y+z=20$ ,  $xyz \neq 0$ ,  $|x| \leq 20$ ,  $|y| \leq 20$ ,  $|z| \leq 20$ を満たす $x, y, z$ の組の数を求める  
Processingのプログラムを書いて下さい



以下はProcessingで与式を解くプログラムの例です。このプログラムは、与式を満たす $x, y, z$ の全ての組み合わせを試して、条件を満たす組み合わせの数をカウントします。

```
processing Copy code

void setup() {
  size(400, 400);
  int count = 0; // 条件を満たす組み合わせの数をカウントする変数

  for (int x = -20; x <= 20; x++) {
    for (int y = -20; y <= 20; y++) {
      for (int z = -20; z <= 20; z++) {
        if ((x + y) * (y + z) * (z + x) == 0 && x + y + z == 20 && x * y * z != 0) {
          count++;
        }
      }
    }
  }


  println(count); // 条件を満たす組み合わせの数を表示する
}

void draw() {
  // 何もしない
}
```

このプログラムは、3重のforループを使用して、 $x, y, z$ の全ての組み合わせを試します。次に、与式が0に等しく、 $x + y + z$ が20に等しく、 $xyz$ が0に等しくない場合、条件が満たされていると判断され、count変数が1つ増加します。最後に、条件を満たす組み合わせの数が出力されます。

# 解説もお願いできる

S このプログラムを解説してほしい

 このプログラムは、3重のforループを使用して、x、y、zの全ての組み合わせを試して、与式を満たすx、y、zの組の数を求めるものです。以下に、このプログラムの詳細な解説を示します。

```
processing Copy code  
  
void setup() {  
  size(400, 400);  
  int count = 0; // 条件を満たす組み合わせの数をカウントする変数
```

setup関数は、Processingの標準的な関数で、プログラムが実行される最初に一度だけ呼び出されます。ここでは、ウィンドウのサイズを指定し、count変数を初期化しています。

```
processing Copy code  
  
for (int x = -20; x <= 20; x++) {  
  for (int y = -20; y <= 20; y++) {  
    for (int z = -20; z <= 20; z++) {
```

3重のforループを使用して、x、y、zの全ての組み合わせを試しています。各ループでは、-20から20までの整数値を変数x、y、zに順番に代入していきます。

```
processing Copy code  
  
if ((x + y) * (y + z) * (z + x) == 0 && x + y + z == 20 && x * y * z != 0) {  
  count++;  
}
```


if文を使用して、与式が0に等しく、x + y + zが20に等しく、xyzが0に等しくない場合、条件が満たされていると判断され、count変数が1つ増加します。

```
processing Copy code  
  
}  
}  
}  
  
println(count); // 条件を満たす組み合わせの数を表示する
```

3重のforループを抜けた後、count変数には条件を満たす組み合わせの数が格納されています。println関数を使用して、この数をコンソールに表示します。

以上が、与式を解くProcessingのプログラムの解説です。

S もう少し噛み砕いて教えてほしいです

 このプログラムは、3つの変数x、y、zがある条件式 $(x + y)(y + z)(z + x) = 0$ を満たす場合のx、y、zの組み合わせを探し、その組み合わせの数をカウントします。また、x、y、zの絶対値が20以下で、xyzが0でない場合に限定されます。

まず、3重のforループを使用して、x、y、zの全ての組み合わせを試します。各forループでは、x、y、zの値が-20から20までの整数で順番に代入されます。これにより、x、y、zがすべて20までの整数である可能性があるため、総当たりで全ての組み合わせを試すことができます。

次に、if文を使用して、条件が満たされているかどうかを判断します。最初の条件は、 $(x + y)(y + z)(z + x) = 0$ であることです。これは、3つの括弧のうち、少なくとも1つが0に等しくなければならないことを意味します。次の条件は、 $x + y + z = 20$ であることです。最後に、xyzが0でないことを確認する条件があります。

条件をすべて満たす場合、count変数を1つ増やします。最後に、条件を満たす組み合わせの数を表示します。

以上が、このプログラムの簡単な解説です。

# 以下のプログラムを解説して

S 以下のプログラムを解説して

```
int[] card = new int[10];
// カードの枚数を初期化
for(int i=0; i<10; i++)
{
    card[i]=0;
}
// 100回カードをランダムに引く
for(int i=0; i<100; i++)
{
    // 0から9までの数字をランダムに得る
    // つまり0から9の数字を等確率で得る
    int id=(int)random(0, 10);
    // そのidのカードの枚数を増やす
    card[id]++;
}
// 結果を1枚ずつ出力する
for(int i=0; i<10; i++)
{
    println("No." + i + " is " + card[i]);
}
```

 このプログラムは、配列を用いてランダムにカードを引いた結果を集計し、それぞれのカードの枚数を表示するものです。以下に、各行の機能を説明します。

```
arduino Copy code
int[] card = new int[10];
```

‘int’ 型の要素を持つ ‘card’ という名前の配列を宣言し、その要素数を ‘10’ に設定します。


```
css Copy code
for(int i=0; i<10; i++)
{
    card[i]=0;
}
```

 Regenerate response

S 以下のプログラムを解説して

```
// 初期化する
void setup()
{
    // 画面サイズを400x300
    size(400, 300);
    // 背景を青色(0,0,255)に設定
    background(0, 0, 255);
}

// 描画する (何度も呼び出される)
void draw()
{
    // 塗りつぶし色を赤色(255,0,0)に設定
    fill(255, 0, 0);
    // 左上(0,0)から横幅frameCount, 縦幅300の四角形を描画
    rect(0, 0, frameCount, 300);
}
```

 このプログラムは、Processingというプログラミング言語で、時間とともに四角形を描画し、その背景を青色に設定するものです。以下に、各行の機能を説明します。

```
scss Copy code
void setup()
{
    size(400, 300);
    background(0, 0, 255);
}
```

‘setup()’ 関数は、Processing が実行される最初のタイミングで実行されます。‘size()’ 関数でウィンドウのサイズを ‘400x300’ に設定し、‘background()’ 関数でウィンドウの背景色を青色に設定します。

```
scss Copy code
void draw()
{
    fill(255, 0, 0);
    rect(0, 0, frameCount, 300);
}
```

 Regenerate response

# とはいえ

- やっぱりお馬鹿さん
  - うまく学びに繋げる方法を考えていきましょう

以下は、Processingで日本の国旗を描画するプログラムです。

```
processing
Copy code

void setup() {
  size(500, 300); // ウィンドウサイズの設定
}

void draw() {
  background(255); // 背景色の設定

  // 赤い円を描画
  fill(255, 0, 0);
  noStroke();
  ellipse(width/2, height/2, height*2/3, height*2/3);

  // 白い十字を描画
  stroke(255);
  strokeWeight(10);
  line(width/2, height/2 - height/6, width/2, height/2 + height/6); // 縦線
  line(width/2 - height/6, height/2, width/2 + height/6, height/2); // 横線
}
```

Processingで表示するプログラムを書いて

Processingでドラえもんを表示するプログラム例です。



```
sketch_230331a
sketc...

{
  400); // ウィンドウのサイズを設定

{
  (255, 255, 204); // 背景色を設定

// 青い体を描画
fill(85, 184, 255);
noStroke();
rect(100, 100, 200, 200, 20, 20, 0, 0);

// 白い手を描画
fill(255);
rect(70, 150, 40, 100, 10);
rect(290, 150, 40, 100, 10);
```



# さて

---

- プログラマ、エンジニアの仕事はなくなっていくのだろうか？
- 今のところ私の答えは「NO」
  - 新しいものを作り出すことは出来ない
  - ひとのためのシステムの開発は難しい
  - プログラムを書かせるための記述をするには課題を分割する能力が必要で、また生成されたプログラムに手を入れるには、その素養が必要
  - ただ、駆け出しエンジニア・プログラマ的な職はなくなるかも？（FMSで力つけましょう）
  - NoCodeはかゆいところに手が届かない

# さて、プログラミングとは

---

- カップラーメンを作る方法をどう伝える？
  - 「お湯入れて3分待つだけ」
  - で、ちゃんと伝わる？
- 以下の図形を言葉だけで指示して描いてもらおうときどう指示する？



# さて、プログラミング習得

---

- 習得する方法には主に、2つの道があります

## (1) 作りたい欲にまかせて、自分で調べつつ色々なプログラムを作っていく

- 本当はこちらが上達速度も早くおすすめですよ
- ただ誰もがができる道ではないとはわかっています

## (2) ひたすら写経していく

- 作りたい欲がない場合、一番手っ取り早いのが勘所を掴むため写経することになります。ひたすら書いて覚える！
- そのためにシステムも用意しました `typing.run`

– 他で、競技プログラミングをやる手もあるよ

# ということで

---

- 「作りたい欲」がでないと思う人は、まずは `typing.run` を利用してプログラムを何度も写経してみてください
  - そのうち、なにか作りたいものが見えてくる人も多いものです
  - また、予習資料にあるプログラムや課題、過去の課題などにも挑戦してみましよう！
- タイピング練習と思って、まずはプログラムをどんどん書いてみましょう！

# ちなみに

---

- なぜ `typing.run` をやるの？
  - プログラミングでつまづきやすいポイント
    - 英語アレルギー
    - 命令名のケアレスミス
    - `{ } ( ) , ;`などの記法
    - タイプ速度が遅い
  - こうした問題をタイピングにより解決
    - 英語になれる
    - 命令名の入力ミスはしない
    - 記法になれる
    - タイプ速度が早くなる
- コード補完があるしタイピング意味ないのでは？
  - コード補完 + 高速タイピングこそが強い
  - コンピュータを使ううえでタイピングは必須

# 進め方

---

- 講義資料は事前にPDFで配布
  - PDFは必ず予習しておくこと
    - 資料は <https://lecture.nkmr.io/>
  - 本来「小テスト」を講義の最初に実施していたのですが typing.run を予習のかわりに
- 課題を配布
  - 基本課題と発展課題の両方を電子的に配布
  - 説明の後、各自課題に取り組む
- 課題を共有フォルダに提出
  - 教員とTAで課題のチェックを実施
  - 基本課題が終わるまでは帰ってはだめ！
  - 発展課題は講義中にはチェックしませんので確実に動作しているかどうかを周りのひとなどと確認しましょう！

# 講義の流れ

---

- 13:30-13:50 解説
- 13:50-14:00 課題提示
- 14:00-16:30 課題を解く + 提出する
- 16:30-17:00 課題解説

# 評価

---

- **各回のタイピング予習の達成度：20点**

- 各回の課題をすべて課題の目標を達成して満点
- 予習の点数については次ページで

- **基本課題：30点**

- 基本課題は必ず終わらせて提出してください
- 時間内に採点します

- **発展課題：10点**

- 発展課題は講義の指定時間内に完成させると加算
- 次回の講義開始前までは50%を満点として採点
- 時間内には採点しません

- **最終試験または課題：40点**

- 最終試験が実施できない場合は、課題提出で代替します

# タイピング予習

---

- 各回の予習点数
  - 予習点数 = 各回の点 \* (達成課題ポイント合計 / 課題数)
- 1ポイントとなる条件
  - その課題で1度は250CPM以上を達成した
  - その課題で1度は225CPMを達成しており2回タイピングした
  - その課題で1度は200CPMを達成しており4回タイピングした
  - その課題で1度は175CPMを達成しており6回タイピングした
  - その課題で1度は150CPMを達成しており8回タイピングした
  - その課題で10回タイピングした
- 他人にやってもらうことは不正に相当します
  - タイピング速度が普段と明らかに違う、普段と特殊キーのタイプ速度が違うなどあった場合、不正とみなし点数剥奪となります。

# タイピング予習

- 達成課題ポイントは0 or 1ではなく、実数値とします
- あるCPMを達成している場合に、n回がノルマとしてあげられている場合で、m回実施していた場合、 $m/n$ が達成課題ポイントとして付与されます
  - 現在、第1~3回分については、過去のノルマが生きてるので少し表示がおかしいです（後ほど修正される予定です）。

## (1) 円を画面の中央に表示したい！

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！）

430CPM (2回)  | 1位/50人中  
typo min:3 max:5 avg:4

[edit](#)

## (2) マウス位置に合わせて円を2つ描画するには？

計算を利用してマウスの位置に合わせて円を2つ描画しよう

411CPM (2回)  | 2位/44人中  
typo min:0 max:4 avg:2

[edit](#)

## (3) 円を繰り返し動かしたい

300x200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

390CPM (1回)  | 2位/40人中  
typo min:4 max:4 avg:4

[edit](#)

## (4) 左上から右下に円を動かそう

左上から右下に円を動かし、その値の変化を見てみよう。

366CPM (3回)  | 1位/38人中  
typo min:5 max:7 avg:5.7

[edit](#)

## (5) ローカル変数とグローバル変数

それぞれの変数で円を描画して、グローバル変数とローカル変数の振舞のの違いを知る

402CPM (1回)  | 2位/25人中  
typo min:3 max:3 avg:3

[edit](#)

## (6) 四角形と面積を計算

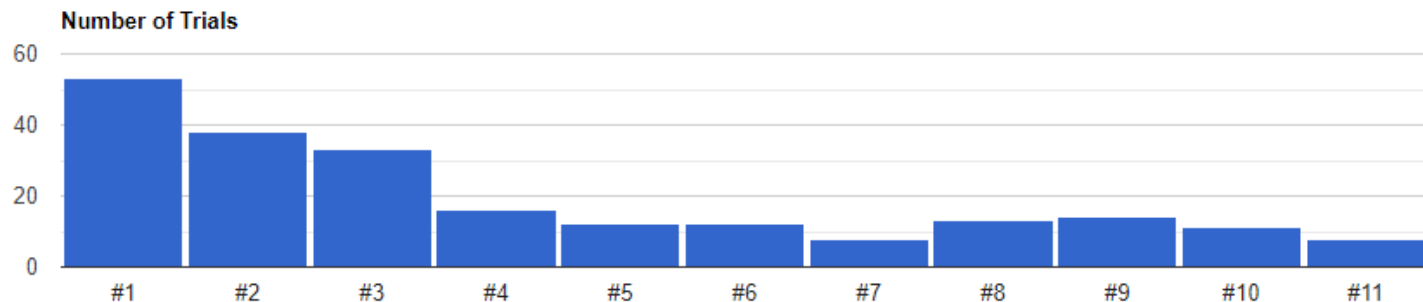
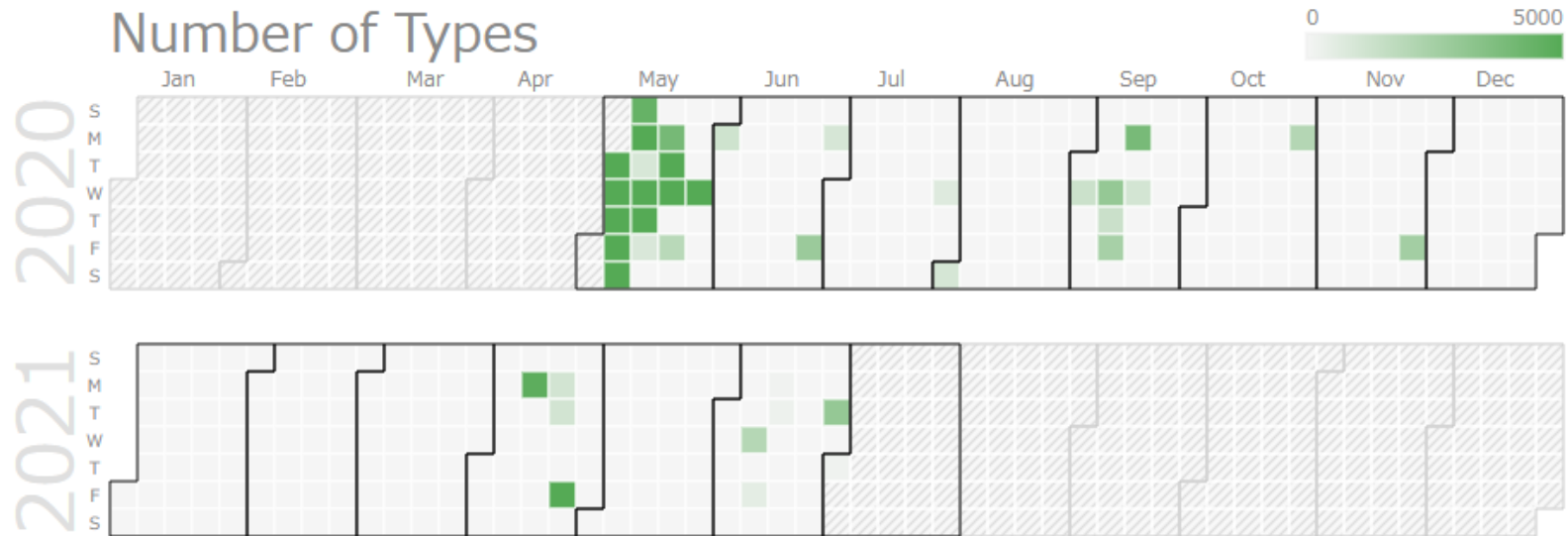
左上の原点からマウスカーソル位置まで四角形を表示し、その面積を計算して表示する！

383CPM (2回)  | 1位/32人中  
typo min:5 max:10 avg:7.5

[edit](#)

# タイピング予習

- typing.run のダッシュボード機能にアクセスすることによって取組状況を確認できます



# typing.run の使い方

---

- Meiji mailでログイン
  - タイピングできることを試して下さい
  - (1) 最初のプログラムを入力しましょう！

# 大学外からの利用

---

- 自宅から接続する場合は、多要素認証をしておくかVPN接続をする必要があります！
  - 多要素認証をしておくのと、2ヶ月くらいは特に入力する必要なくなるのでおすすめ
    - <https://www.meiji.ac.jp/isc/mm-mfa/>
  - VPN接続の方法はこちら
    - <https://www.meiji.ac.jp/mind/ssl-vpn/>
- typing.run で第1回について、達成しているのに1ptいってないのは去年の設定が残っているせいですので気にしないで下さい

# 今回の目標

---

- 終わった部分
  - `typing.run` の使い方を学ぶ
- これから
  - 20～30分程度でパソコンの使い方（高橋）
    - この20～30分は、わかっていないひと向けですのでわかっていないひとは `typing.run` または次回の予習などをしておいてください。
  - Processing で入力してみる
  - Google Formで提出する練習をする

# パソコンの使い方

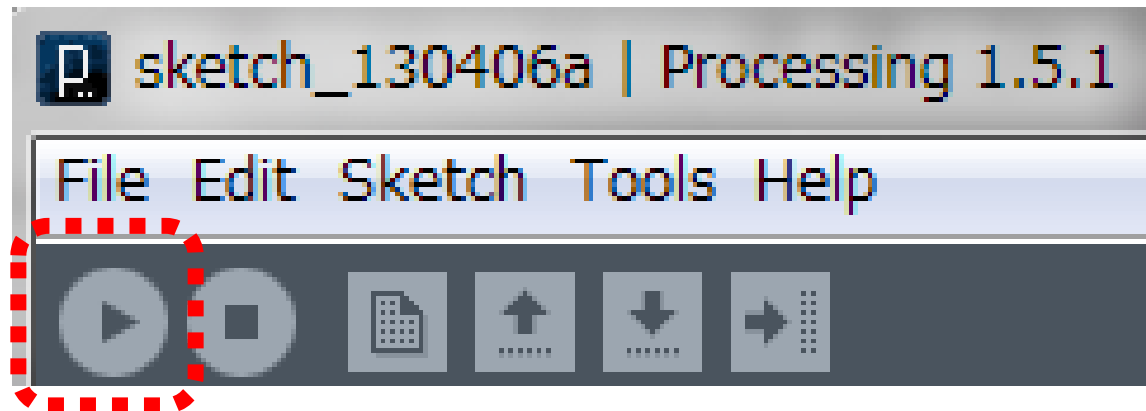
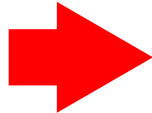
---

- 基本のお話

# Processing とは

- Processing プロジェクトは2001年春に開始
- アート、デザイン界向けにJava拡張として
- プログラムすることをスケッチするという
- 多くのライブラリを導入することで各種の処理が可能に
- プログラムの実行は再生ボタンを押すだけ

再生ボタン



# 想像してみましよう

---

- 画面にウィンドウを作るのってどうする？
- 国旗のようなものはどうやって作る？
- どうやったら沢山の平行な線を描ける？
- キャラクタはどうやって描画する？
- キャラクタを動かすにはどうしたら良い？

# 最初に

- 下記のコード（プログラム）を入力して、再生ボタン（実行ボタン）を押してみましょう

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

400x300のウィンドウを作る

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

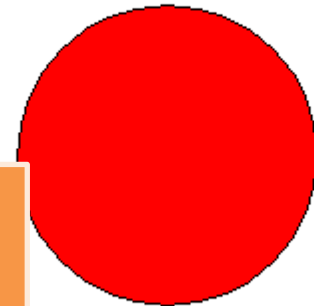
背景色を白色(255,255,255)にする

```
fill(255, 0, 0);
```

塗りつぶしの色を赤色(255,0,0)にする

```
circle(200, 150, 150);
```

(200,150)を中心として直径150の円を描く  
塗りつぶしの色は最後に指定した赤色



# ちなみに

## • ChatGPTさんは不思議なコードを出力

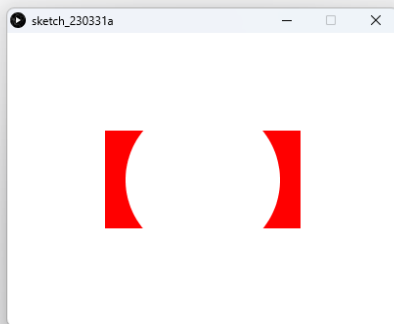
S Processingで日本の国旗のプログラムを書いて

以下はProcessingで日本の国旗を描画するプログラムです。円を2つ描いて、中心が重なるように配置しています。

```
Processing Copy code

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

void draw() {
  background(255); // 背景色を白に設定
  fill(255, 0, 0); // 赤い色を設定
  noStroke(); // 枠線をなしに設定
  rectMode(CENTER); // rect関数の座標を中心に合わせる
  rect(width/2, height/2, height*2/3, height/3); // 日本の国旗の赤い部分を描画
  ellipseMode(CENTER); // ellipse関数の座標を中心に合わせる
  fill(255); // 白い色を設定
  stroke(255); // 枠線の色を白に設定
  strokeWeight(height/12); // 枠線の太さを設定
  ellipse(width/2, height/2, height*4/9, height*4/9); // 日本の国旗の白い部分を描画
}
```



プログラムを実行すると、サイズが400x300ピクセルのウィンドウが開き、日本の国旗が描画されます。円の大きさは`height\*4/9`で、高さの2/3の幅があります。円の中心は、ウィンドウの中心に合わせてあります。

注意：Processingでは、`setup()`関数はプログラムの最初に1回だけ実行されます。そして、`draw()`関数は、プログラムが実行された後、一定の間隔で繰り返し実行されます。詳しくは、Processingの公式ドキュメントを参照してください。

以下は、Processingで日本の国旗を描画するプログラムです。

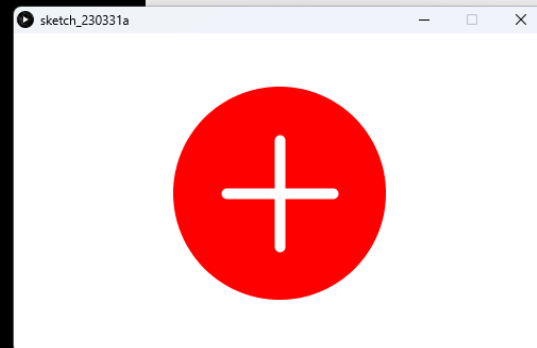
```
processing Copy code

void setup() {
  size(500, 300); // ウィンドウサイズの設定
}

void draw() {
  background(255); // 背景色の設定

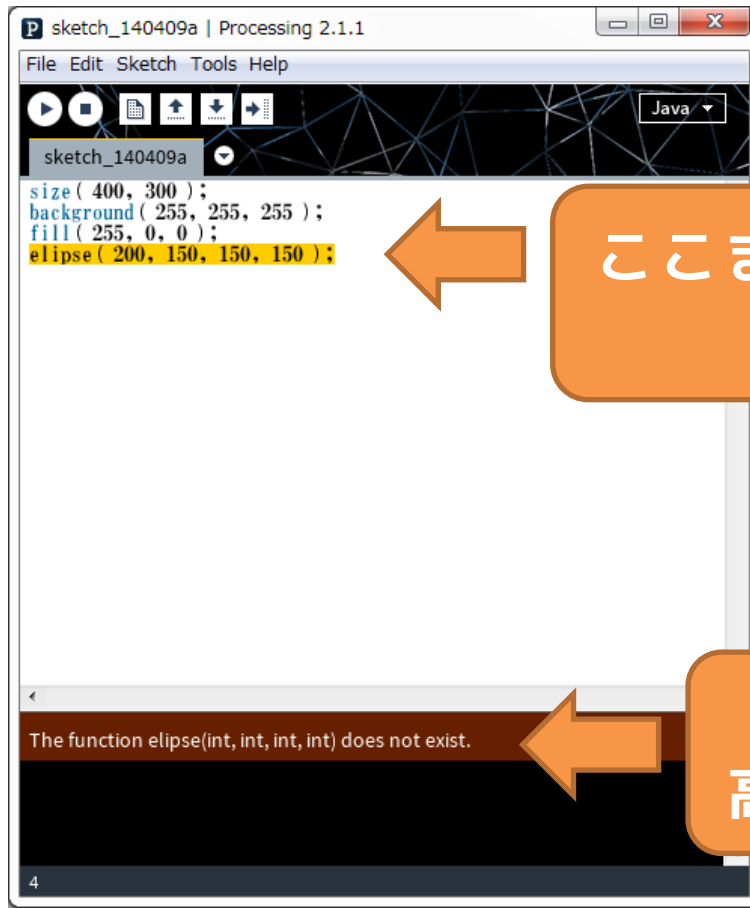
  // 赤い円を描画
  fill(255, 0, 0);
  noStroke();
  ellipse(width/2, height/2, height*2/3, height*2/3);

  // 白い十字を描画
  stroke(255);
  strokeWeight(10);
  line(width/2, height/2 - height/6, width/2, height/2 + height/6); // 縦線
  line(width/2 - height/6, height/2, width/2 + height/6, height/2); // 横線
}
```



# 動かない??

- 何かプログラムが読み込めないエラーが発生した時、Processingはハイライトしてそこら辺がおかしいということを教えてくれます



ここまたは、ココらへんがおかしい  
という意味

エラーメッセージ  
高校レベルの英語なので読もう

# 動かない??

The function `ellipse(int, int, int, int)` does not exist.  
(訳) `ellipse(int,int,int,int)`という関数はないよ!



**ellipse は ellipse の間違い!**

**注意してみると ellipse は青色に  
ellipse は黒色になっており違う!**

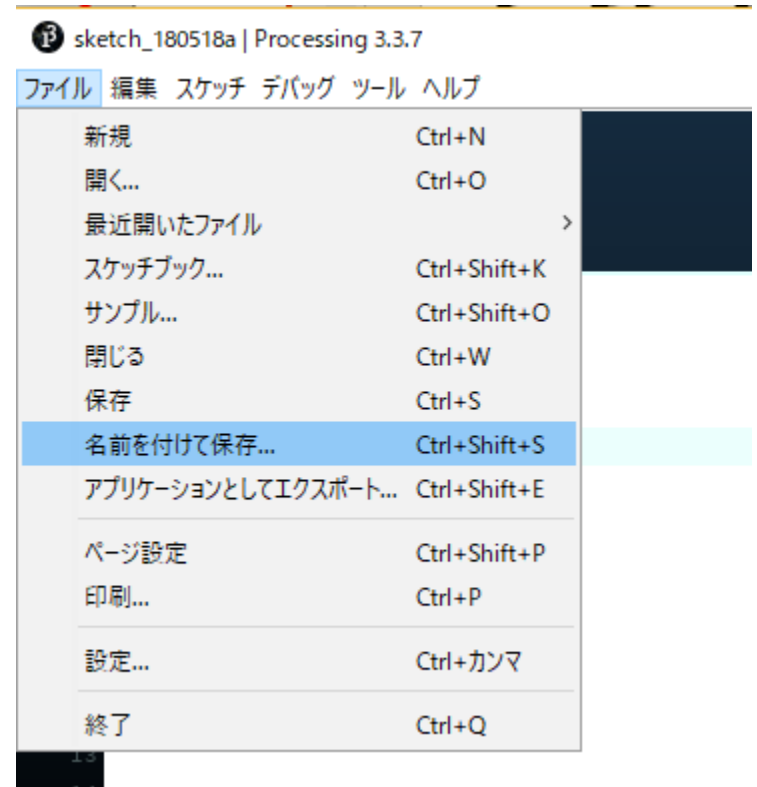
# エラーは最初に注目

---

- エラーメッセージは沢山表示されます
  - 一部おかしなところがあると、そこから他の部分もおかしいと判断されることがよくあります
  - 表示されるエラーメッセージは、最初にどんなメッセージが表示されているか注目しましょう
- ここにエラーがある！と完璧に推定することはコンピュータには難しい
  - その行または、その前後の行におかしなところがないかをチェックしよう！
  - 例えば、セミコロンが抜けている場合に、次の行でエラーが出ます

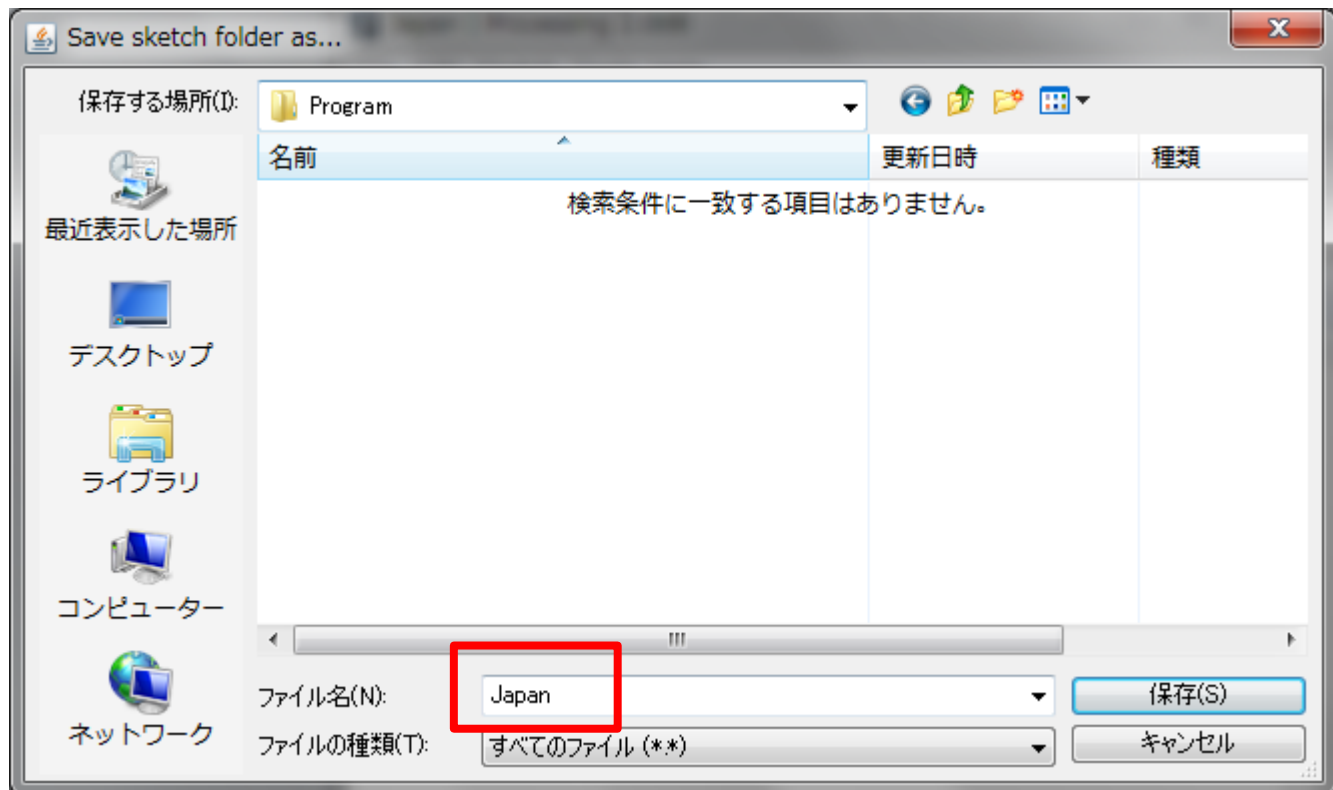
# 保存しましょう

- ファイル → 名前を付けて保存 で保存
  - 名前をつけて保存
  - どんどん保存しましょう
- ファイル → 新規
  - プログラムを新規作成



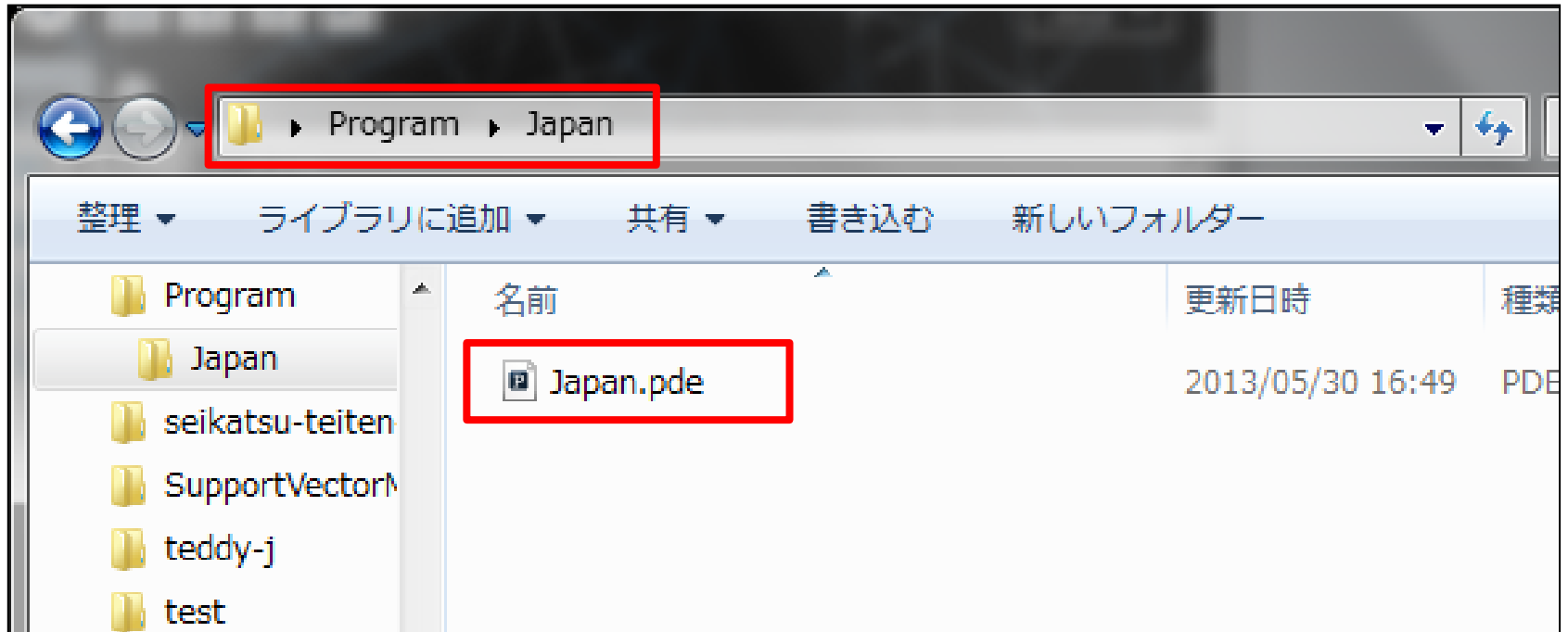
# フォルダ構造

- Processingのプログラムは、フォルダとセット
  - (例) スケッチ (プログラム) を、デスクトップに Program というフォルダを作成し、「Japan」という名前で保存する場合 (WindowsもMacも同じ)



# フォルダ構造

- 「Japan」という名前で保存すると。。。
  - 下図のように、名前が「Japan」というフォルダの下に、「Japan.pde」というプログラムが保存される
  - Japanというフォルダは自動生成される



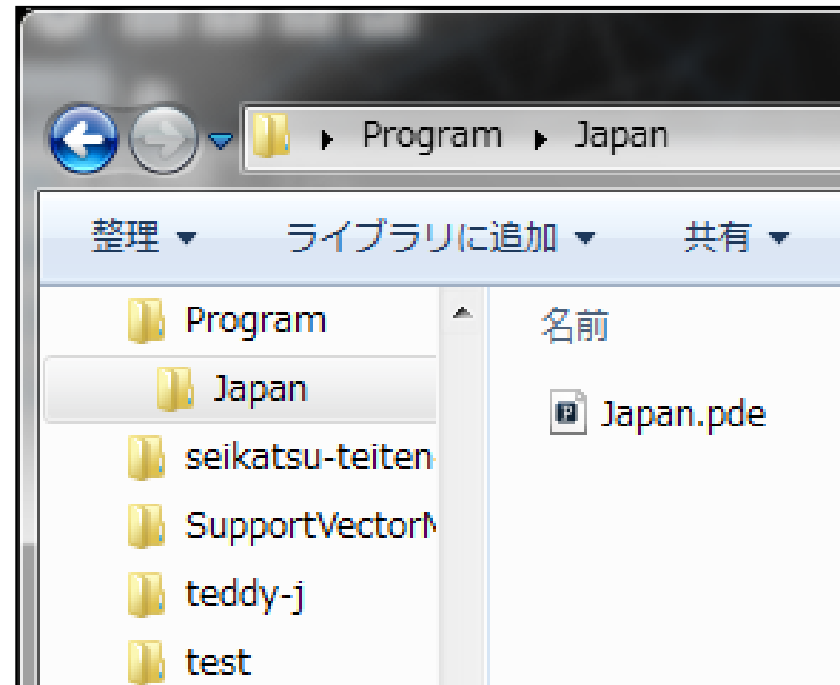
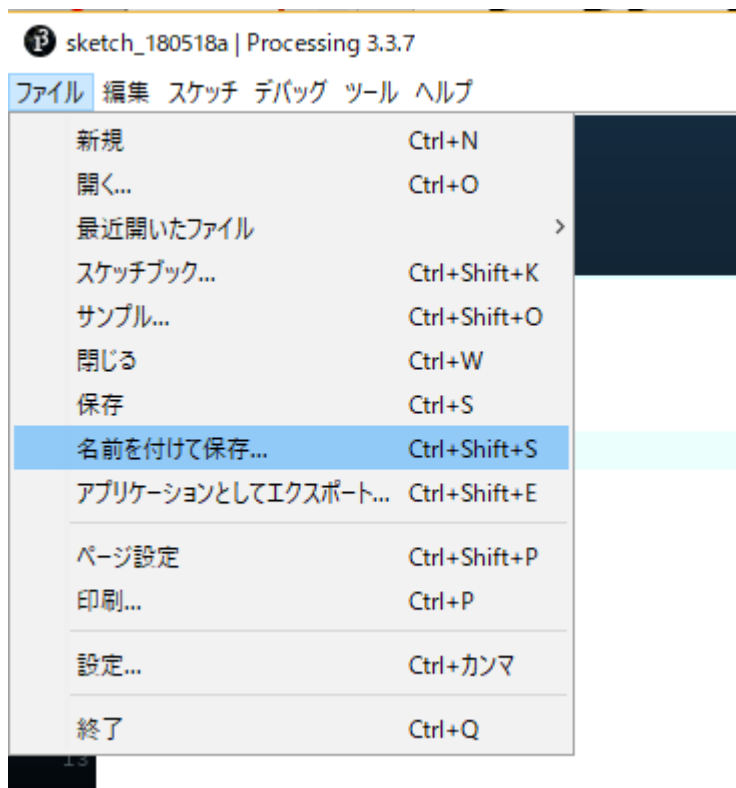
# 注意点

---

- Processingの名前は英数字のみ（ただし、1文字目は英文字のみ）
  - Processingでは、日本語の名前でプログラムを保存することはできません
  - Processingでは、プログラム名に空白や記号は使えません

# 開き方

- メニューの ファイル→ 開く から選んで開く！
- 関連付けを行うとダブルクリックで開けるようになる！



# 課題の提出方法

---

- ここからは小林先生に

# 初回の課題

---

- Google Form からCheckを出す：組・番号・名前
- Processing で日本の国旗が表示できてプログラムを保存およびGoogle Driveに提出する
- typing.run にログインして、第1回の4つのプログラムを最低1回入力する（自己申告）

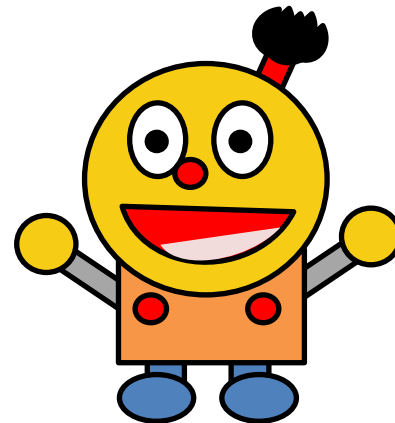
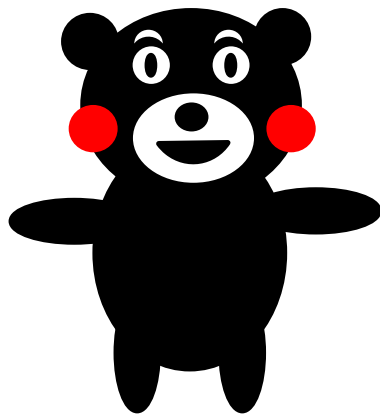
# 次週の課題（予告）

---

- 基本課題 スケッチ名: **basic\_Character**

- 500x500のウィンドウを作成し、その中に好きなキャラクターの絵をProcessingで描画しよう！

- なお、キャラクターには手、足、頭、目があるようにせよ
- 今後の課題で使う可能性があるため、キャラクターは残しておけ

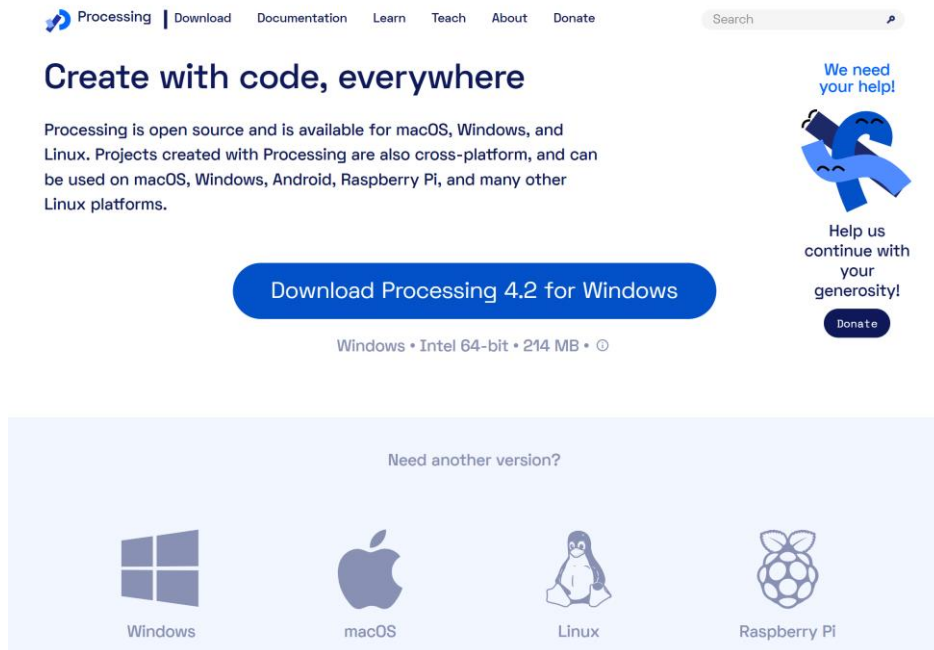


# 入手元

Processingを下記のURLよりダウンロードして  
ファイルを解凍して利用しよう！

お金に余裕があれば寄付も！

<https://processing.org/download/>



The screenshot shows the Processing website's download page. At the top, there is a navigation bar with links for Download, Documentation, Learn, Teach, About, and Donate. Below the navigation bar, the main heading reads "Create with code, everywhere". A paragraph of text explains that Processing is open source and available for macOS, Windows, and Linux. A prominent blue button labeled "Download Processing 4.2 for Windows" is centered on the page. To the right of this button, there is a "Donate" button and a graphic of a blue ribbon with the text "We need your help! Help us continue with your generosity!". At the bottom of the page, there is a section titled "Need another version?" with icons for Windows, macOS, Linux, and Raspberry Pi.



The screenshot shows the Processing IDE interface. The top menu bar includes options like ファイル (File), 編集 (Edit), スケッチ (Sketch), デバッグ (Debug), ツール (Tools), and ヘルプ (Help). The main workspace displays a sketch named "sketch\_230409a" with the following code:

```
1 void setup(){  
2   size(300, 300);  
3 }  
4  
5 void draw(){  
6   circle(mouseX, mouseY, 30);  
7 }  
8  
9
```