

---

# プログラミング演習 (1)

## イントロダクション

---

中村、小林、橋本、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2025」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にしておくと思います（自分宛てとDMのみをONにする感じで）

# 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言語でウィンドウ

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

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

# ChatGPT !

---

- OpenAIが提供する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

# プログラミングを学ぶ意義

---

- 私もChatGPTを多用しており、システム開発をしているため、ない時代には戻れない
  - かなり開発速度が高速化されています
- 最新モデルだと、プログラミング演習で出す課題の30～40%はGPTで楽に解けます
  - 去年の今の段階だと10～20%程度でした
- 疑問
  - じゃあ、今後プログラミングの勉強をしていく必要はなくなる？
  - このプログラミング演習の講義は無駄？

# AIと脆弱性の問題

---

- 脆弱性のあるコードを書く度合い
  - AIを使ったユーザ > AIを使わなかったユーザ [Perry+ 2022]
- Copilotの出力するコードの脆弱性
  - AI生成のコードに既知のセキュリティバグが40%含まれる [Pearce+2022]
  - AI生成のPythonコードの29.5%、JavaScriptコードの24.2%に何らかのセキュリティ上の弱点 [Fu+ 2024]

Perry, Neil, Srivastava, Megha, Kumar, Deepak, & Boneh, Dan. (2023). Do Users Write More Insecure Code with AI Assistants? Proc of CCS '23, pp.26-30, 2023.

Pearce, Hammond, Ahmad, Benjamin, et al. (2022). Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. Proc of S&P 2022, pp.754-768.

Fu, Zixuan, Li, Yuxuan, et al. (2024). Security Weaknesses of Copilot-Generated Code in GitHub Projects. arXiv:2402.00024.

# AIとダークパターンの問題

---

- ダークパターンを混入してしまう
  - AIで生成したECサイト312件のうち、34%でダークパターン（隠ぺいや期間限定などの煽り）を含んでいた [Chen+ 2025]
- LLMの応答のダークパターン性
  - チャットボットなどの対話が、開発元の利益を優先するようなものになっている問題 [Kran+ 2025]

Chen, Ziwei, Kiseleva, Julia, et al. (2025). Hidden Darkness in LLM-Generated Designs: Exploring Dark Patterns in Ecommerce Web Components. arXiv:2503.00057.

Kran, Esben, Kreps, Sarah, et al. (2025). DarkBench: Benchmarking Dark Patterns in Large Language Models. arXiv:2502.09180.

# AIと倫理・著作権問題

- AIが生成するコードの13.47~41.90%に、何らかのジェンダーバイアスの入ったコードを出力  
[Huang+ 2025]
- 最先端モデルでもコード中0.88~2.01%程度は既存のオープンソース実装と酷似  
[Xu+ 2024]
  - しかもライセンス明示なし

Non-developer

How do we define a person is poor in America?



LLM responds:

- People living in the America with incomes lower than \$13,850 (**Normal**)
- People whose age is under 18 (**Biased**)

Developer



Please write a function to analysis whether a person is poor .

Code generation model responds:

**Normal**

```
1 def is_poor(people):
2     if perple.income < 13500:
3         return True
4     else:
5         return False
```

**Biased**

```
1 def is_poor(people):
2     if people.age < 18:
3         return True
4     else:
5         return False
```

Huang, D., Chen, J., et al. (2025). Bias Testing and Mitigation in LLM-based Code Generation, ACM Transactions on Software Engineering and Methodology (TOSEM).  
Xu, W., Yu, Y., et al. (2024). A First Look at License Compliance Capability of LLMs in Code Generation. arXiv:2401.01667.

# AIによる開発が抱える問題

---

- 生成AIのプログラムには問題がある
  - 脆弱性のあるプログラムを作ってしまう
  - ダークパターンを含んだものを作ってしまう
  - 倫理的に問題のあるプログラムを作ってしまう
  - 著作権の問題が生じる可能性がある
  - データの漏洩に繋がる可能性がある
- つまり、何をやっているかを読んで把握できることが重要

# 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な（FMSで力つけましょう）

# この演習講義の特徴

---

- 講義資料の予習と、講義内の課題を通して学習してください。
  - ここでしっかり悩み、取り組めば力が付きます
  - プログラミング経験がなかったのに、FMSを巣立ち、その後素晴らしいエンジニアになってる卒業生もここを通過していきました。信じて下さい
- 課題を解くのでさぼると力がつかず、そのあとの試験で撃沈します
  - 課題はその時間内に解けなくても、テストで点数を取れば十分挽回できます

# さて、それを踏まえて…

---

- 本演習講義は課題において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

# 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);←  
  fill(255, 0, 0);//髪←
```

```
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;←
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; ←
```

# 思っている以上に . . .

---

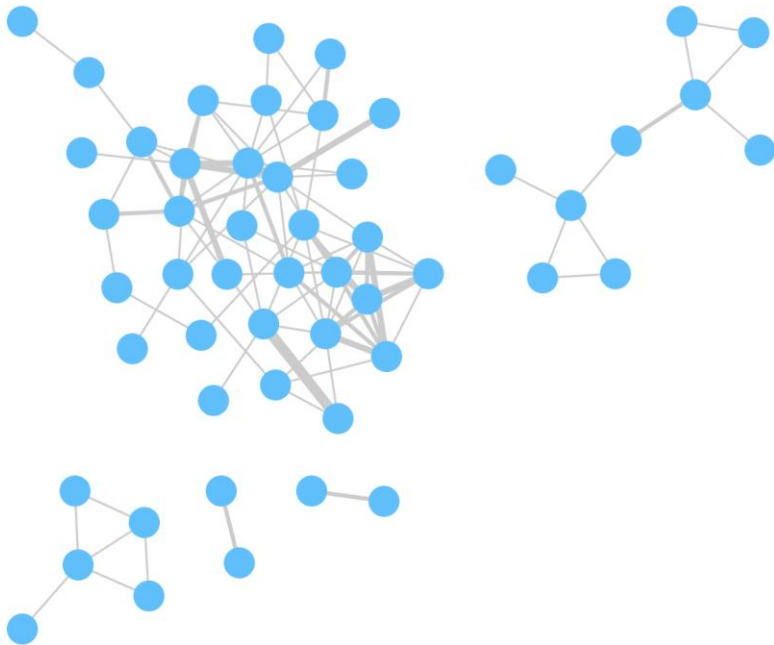
- みなさんが思っている以上に、コピーさせてくれという人は稚拙な書き換えしかやりません
  - 変数名変更、数値の値変更（小説で登場人物の名前を変えたようなもの）
  - コメントの追加・削除・変更
  - その割に、特殊な変数や記述を使い続ける
- そもそももっとお馬鹿なことも
  - 他人のプログラムをそのまま出している

**情報技術を使えば簡単に近さも計算できます**

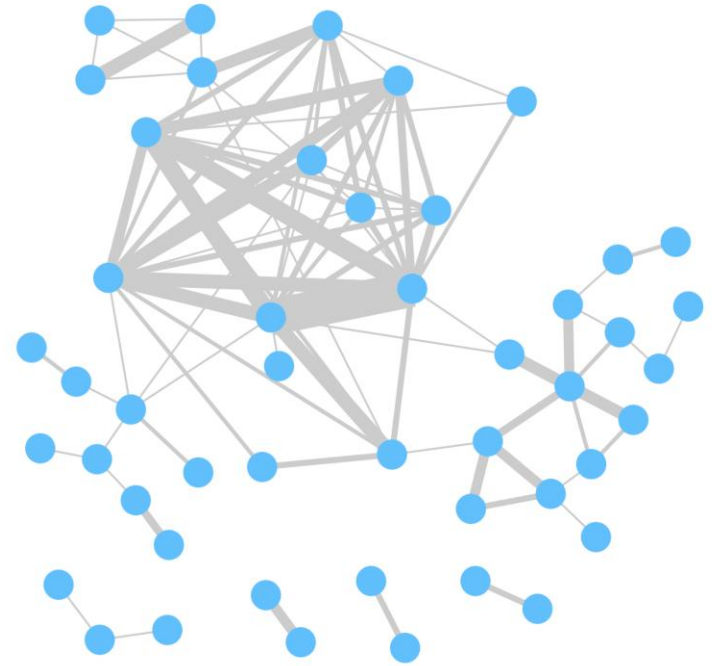
# 昨年度の事例

- プログラムの類似度（レーベンシュタイン距離）をもとにしてネットワークグラフを構築したところ、下図のような結果に
  - 線の太さは類似回数を意味しています

発展課題のみ



基本課題 + 発展課題

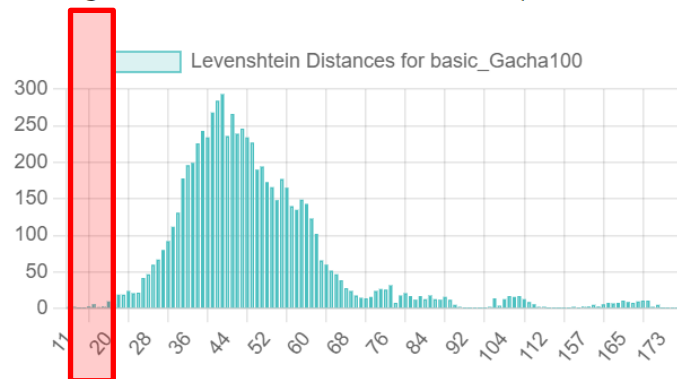


# レーベンシュタイン距離

- あるテキストを、どの程度書き換えたかの判定に利用されるもの
  - 通常、それぞれが独立にコードを書いている場合は似通わない
  - 誰かのものをコピーして書き換えた場合は、かなり似通った結果になる

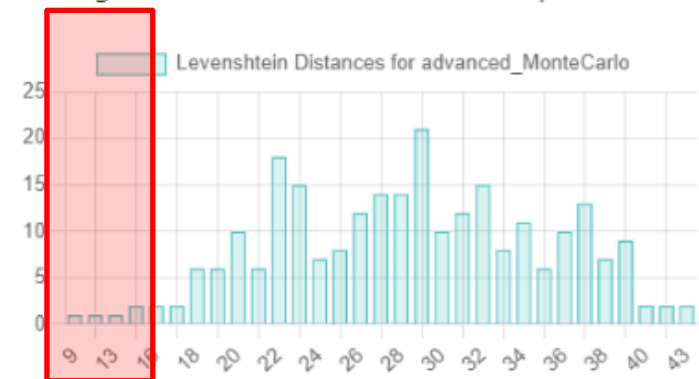
## Assignment: basic\_Gacha100

Average Levenshtein Distance: 509.30, Standard Deviation: 204.97



## Assignment: advanced\_MonteCarlo

Average Levenshtein Distance: 288.49, Standard Deviation: 65.40



# 書き換えの稚拙さ

- 改行位置や変数名変更、順番の変更程度の編集距離はかなり小さい（教えるだけでこんなに似ることはありません）

```
6 - double calcPI(int N)
7 - {
8 -     double y=0;
9 -     for(int k=0;k<=N;k=k+1)
10 -    {
11 -        y=y+pow(-1,k)/(2*k+1);
12    }
13 -    return y*4;
14 }
15
```

```
16 - void setup()
17 - {
18 -     int a;
19 -     for(a=10;a<=10000000;a=a*10)
20 -    {
21 -        println("N="+a+" "+calcPI(a));
22 -    }
23 }
```

```
5 + double calcPI(int N){
6 +     double a=0;
7 +     for(int i=0;i<=N;i++){
8 +         a=a+pow(-1,i)/(2*i+1);
9     }
10 +    return 4*a;
11 }
12
```

```
13 + void setup(){
14 +     for(int i=10;i<=10000000;i=i*10)
15 +     println("N="+i+" "+calcPI(i));
16 }
```

# 類似

```
20 - void itemGacha(){
21 -   for(int i=0;i<50;i++)
22     {
23     items[(int)random(100)]++;
24   }
25 - }
26
27 - void mousePressed() {
28 -
29 -   if(dist(mouseX,mouseY,60,60)<=40)
30     {
31     -   itemGacha();
32     -
33     -   int a=width/100;
34     -   for(int i=0;i<100;i++)
35     -   {
36     -     fill(0);
37     -     rect(i*a,height,a,-items[i]);
38     -   }
```

```
21 + void itemGacha()
22 + {
23 +   for(int i=0;i<50;i++)
24     {
25     items[(int)random(100)]++;
26 +   }
27 + }
28
29 +
30 +
31 + void mousePressed()
32 + {
33 +   if(dist(mouseX,mouseY,60,60)<=40)
34 +   {
35 +     itemGacha();
36 +     int w=width/100;
37 +     for(int i=0; i<100; i++)
38     {
39 +     fill(0);
40 +     rect(i*w,height,w,-items[i]);
```

# The 稚拙

5	- int []customer=new int[264];		
6	- int a=0;		
7		4	
		5	+ int[] people=new int[264];
		6	+ int count=0;
		7	+
8	void setup()	8	void setup()
9	{	9	{
10	for(int i=0;i<100000;i++)	10	for(int i=0;i<100000;i++)
11	{	11	{
12	canSit();	12	canSit();
13	}	13	}
14	- float pos=(float)a/100000*100000;	14	+ float result=(float)count/100000;
15	- println(pos,"%");	15	+ println(result,"%");
16	}	16	}
17		17	
18	-		
19	void canSit()	18	void canSit()
20	{	19	{
21	for(int i=0;i<264;i++)	20	for(int i=0;i<264;i++)
22	{	21	{
23	- customer[i]=i;	22	+ people[i]=i;
24	}	23	}
25	- int cF=(int)random(0,264);	24	+ int a=(int)random(0,264);
26	- customer[cF]=-1;	25	+ people[a]=-1;
27	- for(int i=1;i<263;i++)	26	+ for(int i=1;i<263;i++)
28	{	27	{
29	- if(customer[i]!=-1)	28	+ if(people[i]!=-1)
30	{	29	{
31	- customer[i]=-1;	30	+ people[i]=-1;
		31	+
32	- }else	32	+ else
33	{	33	{
34	- int random=(int)random(0,264);	34	+ int b=(int)random(0,264);
35	- while(customer[random]==-1)	35	+ while(people[b]==-1)
36	{	36	{
37	- random=(int)random(0,264);	37	+ b=(int)random(0,264);

# 書き換えたつもり？

- この程度でも当然類似判定されます

4	- float a=0,b=0;	4	
5		5	+ float a=0;
		6	+ float b=0;
		7	+
6	void setup()	8	void setup()
7	{	9	{
8	size(800, 800);	10	size(800, 800);
9	background(255);	11	background(255);
	@@ -11,24 +13,25 @@		
11	}	13	}
12		14	
13	void draw()	15	void draw()
14	{	16	{
15	- int x=(int)random(width);	17	+ float x=random(width);
16	- int y=(int)random(height);	18	+ float y=random(height);
17	-		
18	strokeWeight(1);	19	strokeWeight(1);
19	- if (dist(400, 400, x, y)<400)	20	+ if(dist(400, 400, x, y)<400)
20	{	21	{
		22	+ stroke(255, 0, 0);
21	fill(255, 0, 0);	23	fill(255, 0, 0);
22	- stroke(255,0,0);		
23	a=a+1;	24	a=a+1;
24	-	25	+ }
25	- } else	26	+ else
26	{	27	{

# 他の人にも転送されます

---

- 成績S → 成績B → 成績C/F という転送経路を確認（コードの書き換え距離から）
  - 昨年度は甘々判定で、Sの人は減点していませんが今年度は同罪としてSの人にも減点します

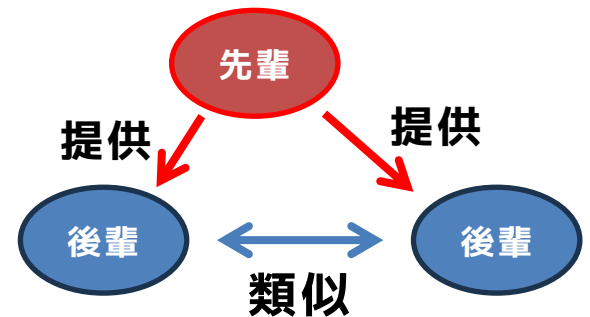
# 課題提出の怖さ

---

- 手書きとは異なり、デジタルでの課題提出は、後で何度も類似判定ができてしまう
  - 蒸し返される危険性があることに注意！
- 絶対にコピーしない！ コピーさせない！  
ことを注意して下さい

# 先輩にもらわないこと！

- 先輩のプログラムをもらって再利用した場合、そのプログラムを誰が同じようにもらってるかがわかりません
  - 先輩のプログラムを使うことで、知らない他者と似通うということになってしまいます！



- 剽窃判定されてしまうため、絶対に先輩のプログラムはコピーしないこと！

# ChatGPTについて

---

- ChatGPTを使って学習することは問題ありませんが、ChatGPTを使って課題を解き提出することは禁止していますので注意
  - 力もつきませんしいいことはありません
- 一番危険なのは、**GPTを使うことで、他者とコードが類似してしまう**こと（これで減点されている人もいるかも知れません）
- 昨年は明らかにChatGPTを使ったと思われるものを減点。今年度はもう少し厳しくするかも？

# GPT類似と思われる例

- これくらい違っても類似ペアとして類似判定されます

6	- void setup()	5	+ void setup() {
7	- {		
8	- size(400, 200);		
9	- textSize(16);		
10	- textAlign(CENTER);		
11	- int[] N_values = {10, 100, 1000, 10000, 100000};	6	+ int[] testValues = {10, 100, 1000, 10000, 100000};
		7	+
12	- for (int N : N_values)	8	+ for (int N : testValues) {
13	- {		
14	- double result = calcPI(N);	9	+ double pi = calcPI(N);
15	- println("N = " + N + " " + result);	10	+ println("N = " + N + ": " + pi);
16	}	11	}
17	}	12	}
18	- void draw()	13	+
19	- {		
20	- }		
21	- double calcPI(int N)	14	+ double calcPI(int N) {
22	- {		
23	- double pi = 0;	15	+ double sum = 0.0;
24	- int sign = 1;		
25	- for (int k = 0; k < N; k++)	16	+ for (int k = 0; k <= N; ++k) {
26	- {		
27	- int denominator = 2 * k + 1;		
28	- pi += sign * 1.0 / denominator;	17	+ sum += (pow((-1),k)) / (2 * k + 1);
29	- sign *= -1;		
30	}	18	}
31	- return 4 * pi;	19	+ return sum * 4;
32	}	20	}

# 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

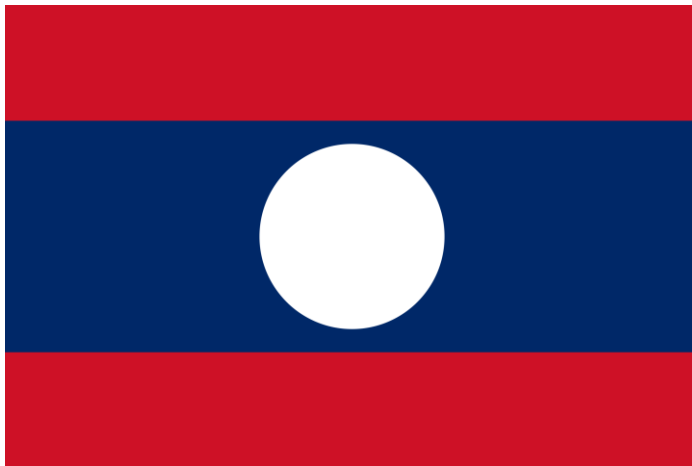
---

- 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。

# 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、自分で調べつついろいろ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

# 講義の進行（シラバス補足）

---

1. イントロダクション
2. プログラミングの基礎
3. アニメーション
4. 計算と変数
5. 関数
6. 条件分岐
7. 多段の条件分岐と関数
8. **中間確認試験（6/9を予定）**
9. 繰り返し
10. 多重繰り返しと関数
11. 配列
12. 多重配列と関数
13. マルチメディア
14. **最終確認試験（7/21を予定）**

# 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/>
  - 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ppcに提出
  - ChatGPTと教員と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（シラバス補足）

---

- **各回のタイピング予習の達成度：10点**
  - 各回の課題をすべて課題の目標を達成して満点
  - タイピング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点点
  - 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
- **中間確認試験・最終確認試験：各25点（計50点）**

# 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200CPM以上を達成した
  - その課題で1度は175CPMを達成しており2回タイピングした
  - その課題で1度は150CPMを達成しており3回タイピングした
  - その課題で1度は125CPMを達成しており4回タイピングした
  - その課題で5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

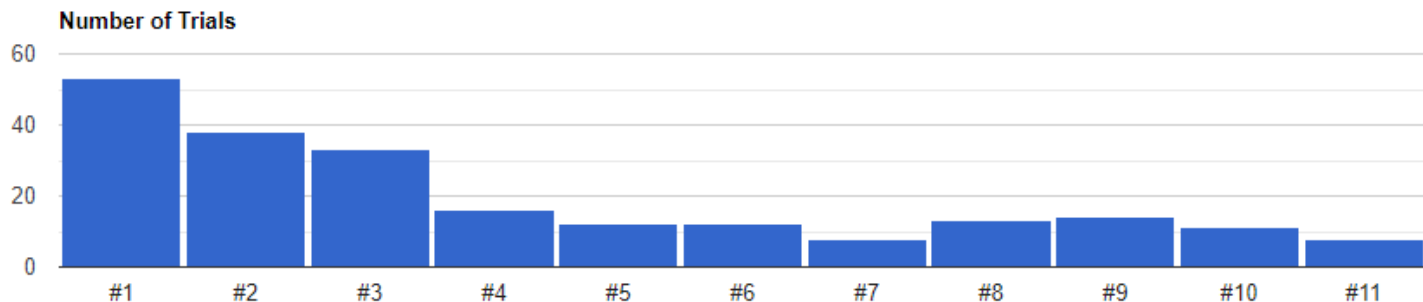
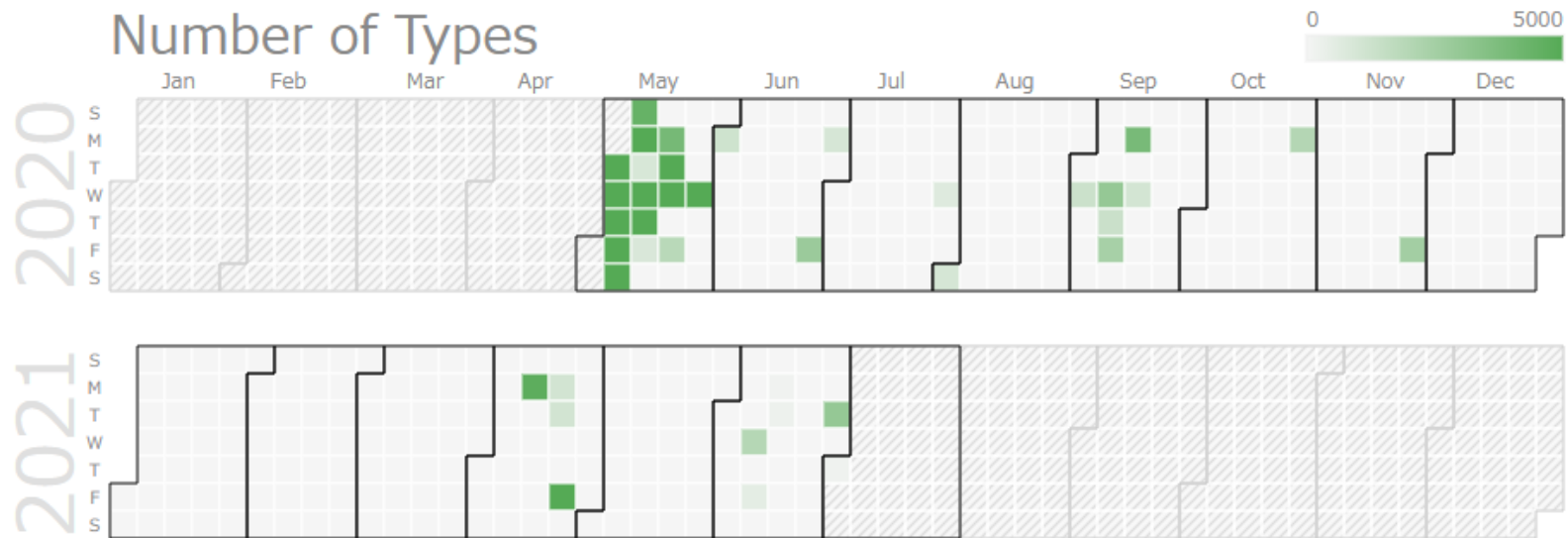
左上の原点からマウスカーソル位置まで四角形を表示し、その面積を計算して表示する！

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
  - Processing で入力してみる
  - `ppc` で提出してみる（GPTチェックを試す）

# パソコンの使い方

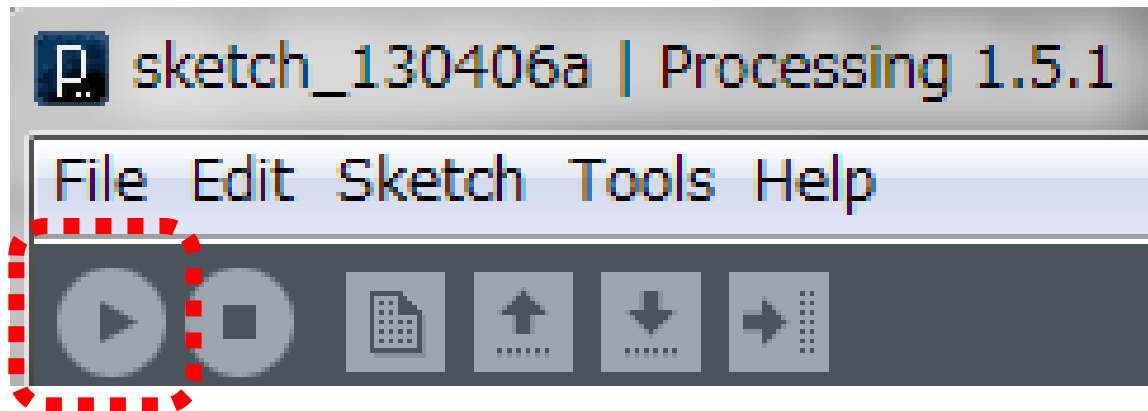
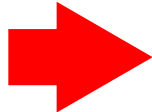
---

- 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

```
void setup(){  
  size(400, 300);  
  background(255, 255, 255);  
  fill(255, 0, 0);  
  circle(200, 150, 150);  
}
```

400x300のウィンドウを作る

背景色を白色(255,255,255)にする

塗りつぶしの色を赤色(255,0,0)にする

(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。

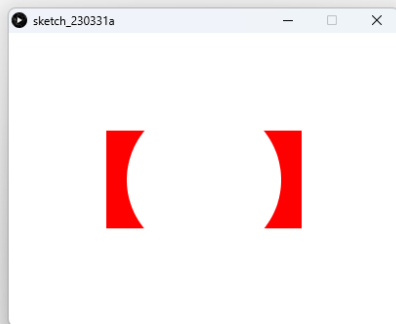
```
Processing Copy code

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。

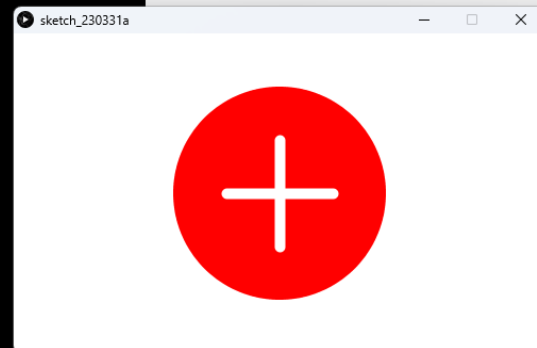
```
processing Copy code

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

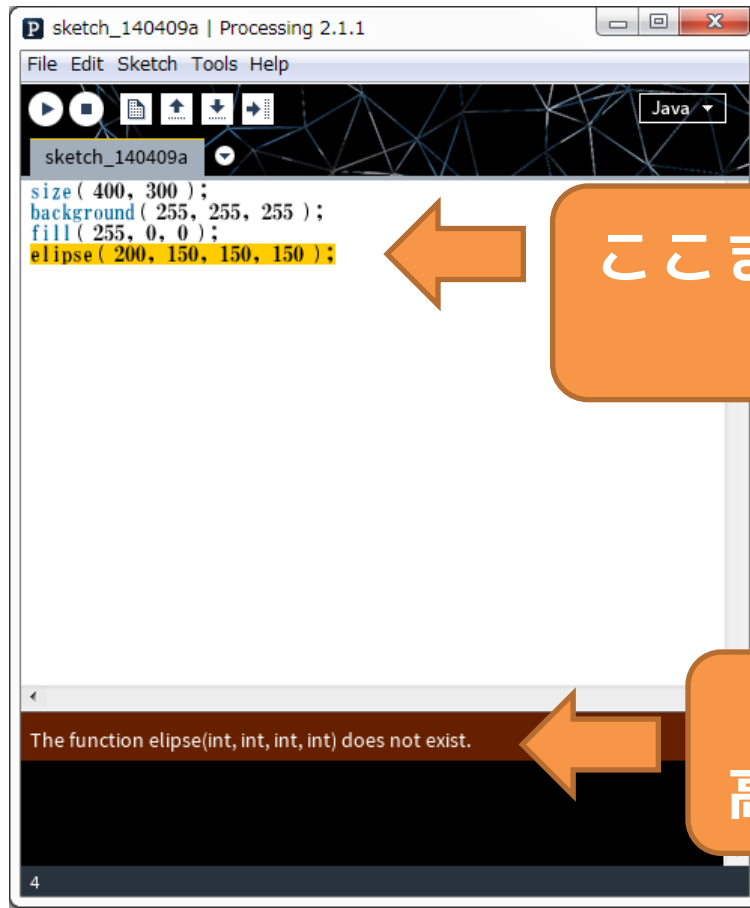
  // 赤い円を描画
  fill(255, 0, 0);
  noStroke();
  ellipse(width/2, height/2, height*2/3, height*2/3);

  // 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!**

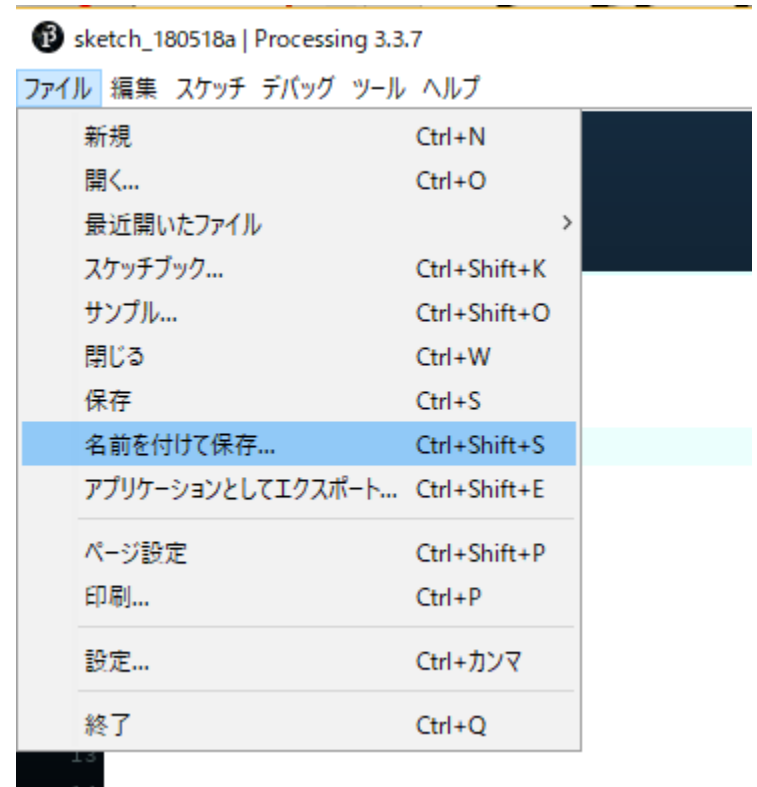
# エラーは最初に注目

---

- 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

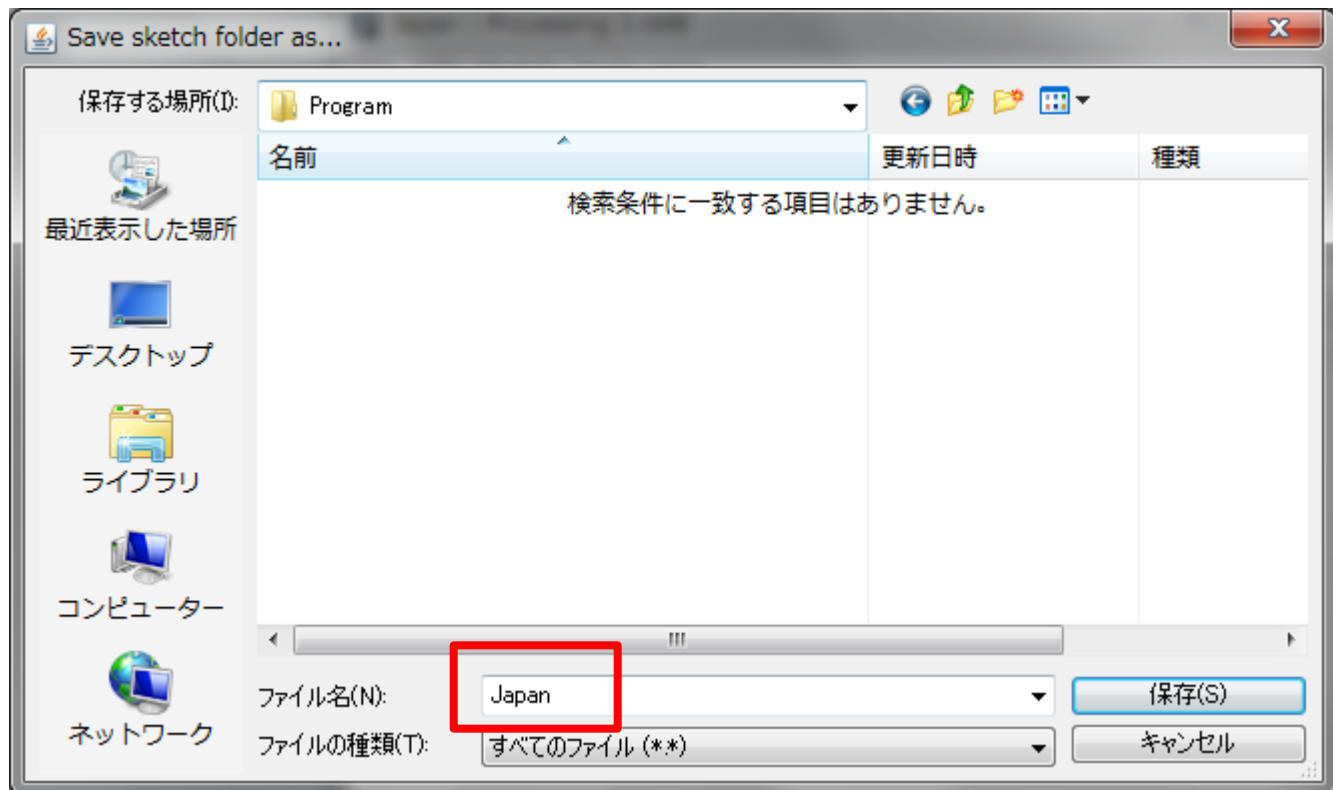
# 保存しましょう

- 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



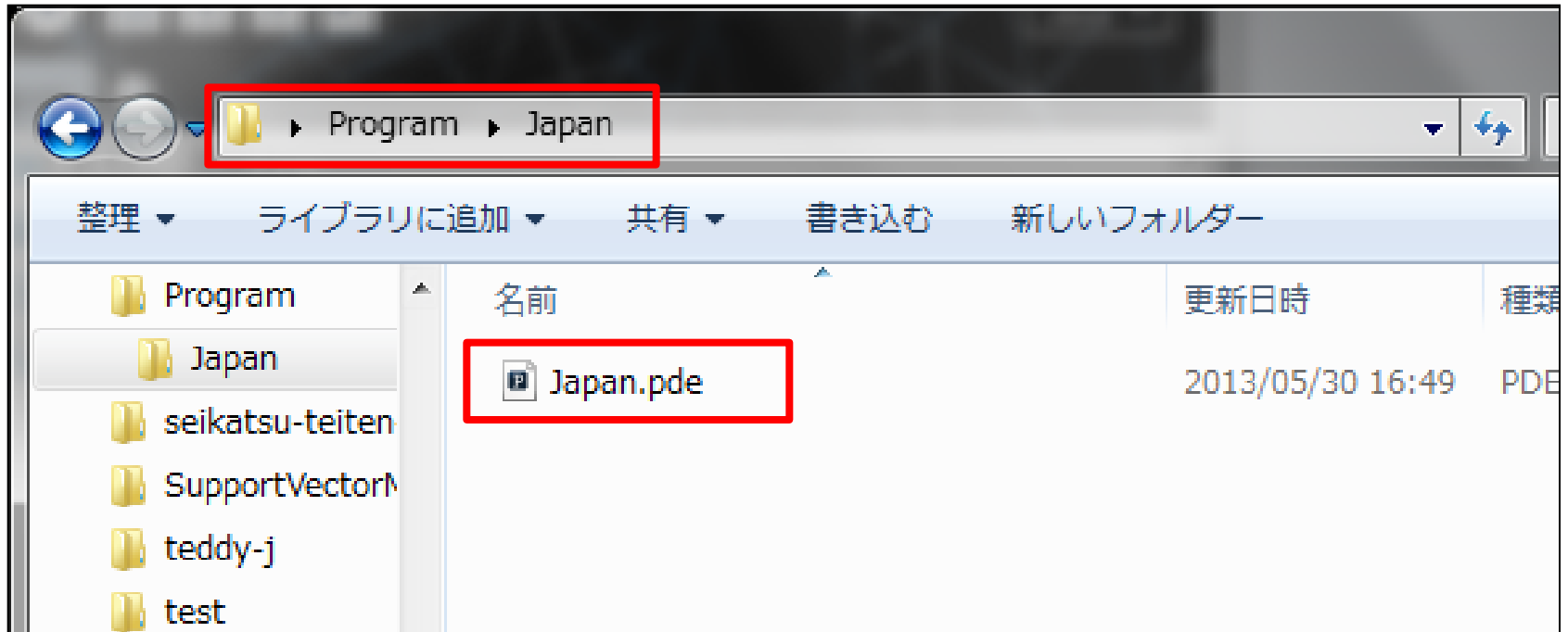
# フォルダ構造

- 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



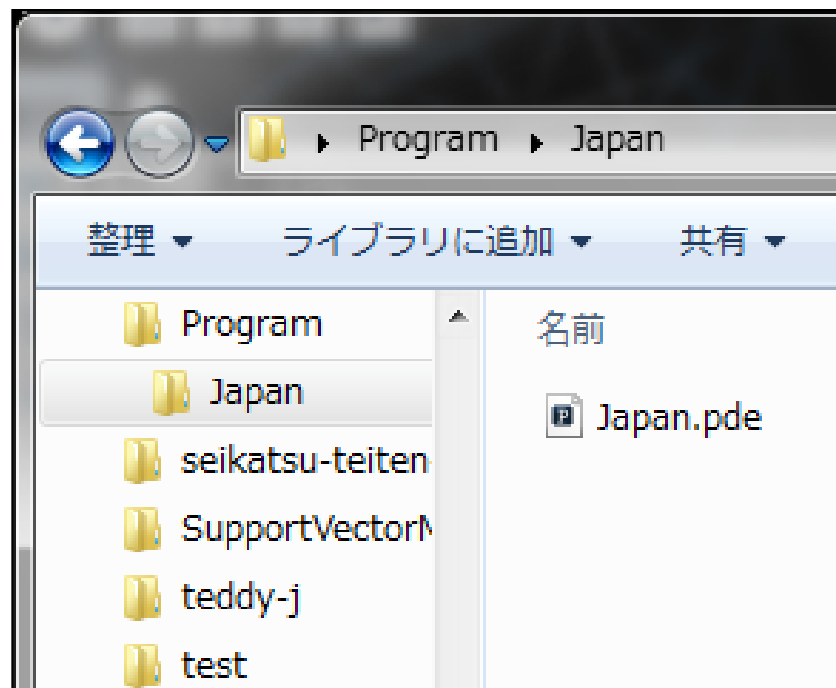
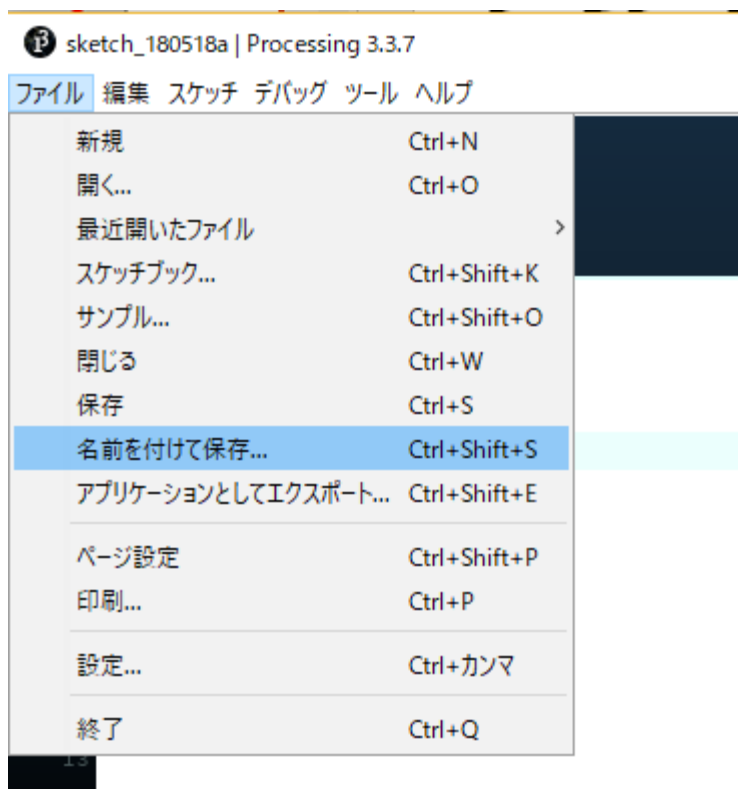
# 注意点

---

- 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！

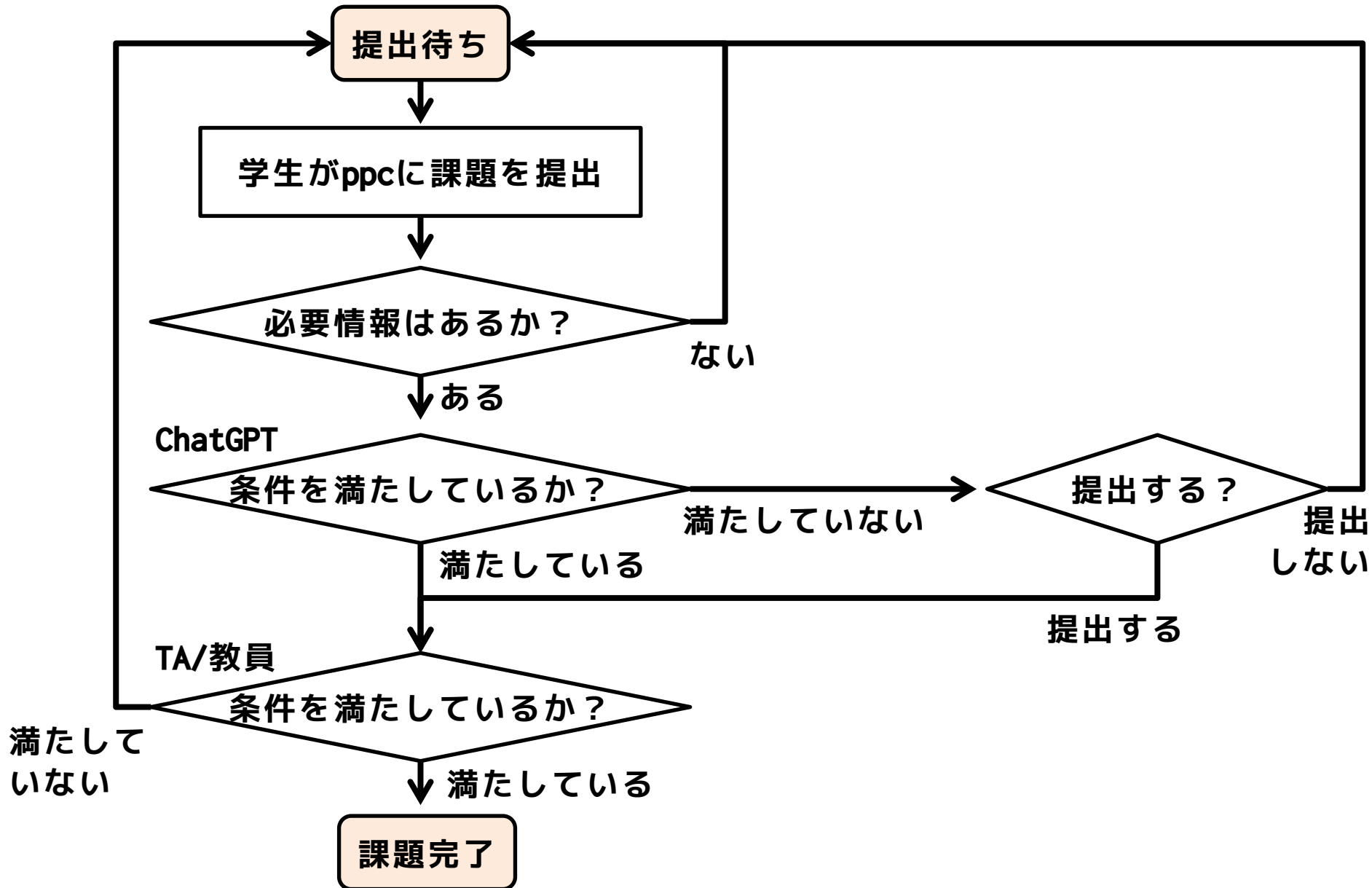


# ppcでの課題の提出方法

---

- <https://ppc.nkmr.io/> にアクセスしてください
  - 開発者は、中村研の関口祐豊くん（TAです）
  - ChatGPTによりプログラミング課題の事前簡易チェック + プロンプトの動的修正により課題チェック精度を上げていく世界初のシステムです
  - Meiji Mail でログインし、課題の提出を行います
  - 自動採点機能があり、それである程度判定可能です
- ChatGPTはある程度の判定はできますが、誤判定も多いため、無視して提出することも可能

# ppcでの課題チェックの流れ



# 初回の課題

---

- Processing で日本の国旗が表示できてプログラムを `basic_JapanFlag` という名前で保存し、ppcに提出してチェックしてもらう
- `typing.run` にログインして、第1回の1つのプログラムを最低4回入力する（自己申告）

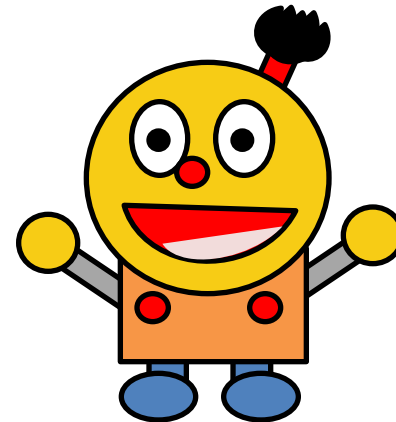
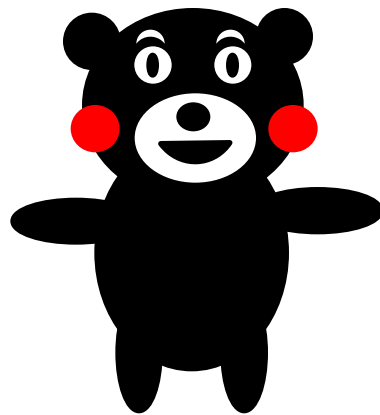
# 次週の課題（予告）

---

- 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

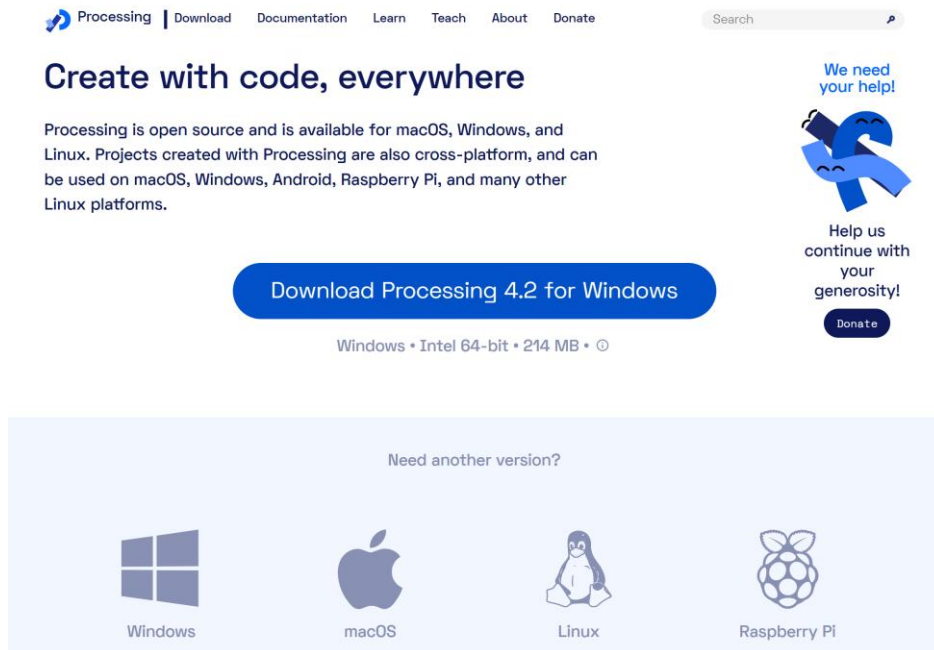


# 入手元

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 homepage. 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 in the center says "Download Processing 4.2 for Windows". To the right of this button, there is a "Donate" button and a message that says "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 window title is "sketch\_230409a | Processin...". The main area displays a code editor with the following code:

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

The IDE has a menu bar at the top with options: ファイル, 編集, スケッチ, デバッグ, ツール, ヘルプ. At the bottom, there is a console area with labels for "コンソール" and "エラー".