pi-1. プログラミング入門：Java 言語の基礎と実行環境の理解

（Java マスター講座：基礎から応用まで）

URL: https://www.kkaneko.jp/pro/pi/index.html

金子邦彦

CC BY NC SA

```
public class Main {
    public static void main(String[] args) throws Exception {
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
    }
}
```

Java プログラムの ソースコード

```
C:¥Users¥user>javac Main.java

C:¥Users¥user>java Main
big
15
```

Java プログラム実行のためのコマンド と実行結果

```
x = 100
if (x > 20):
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

Python

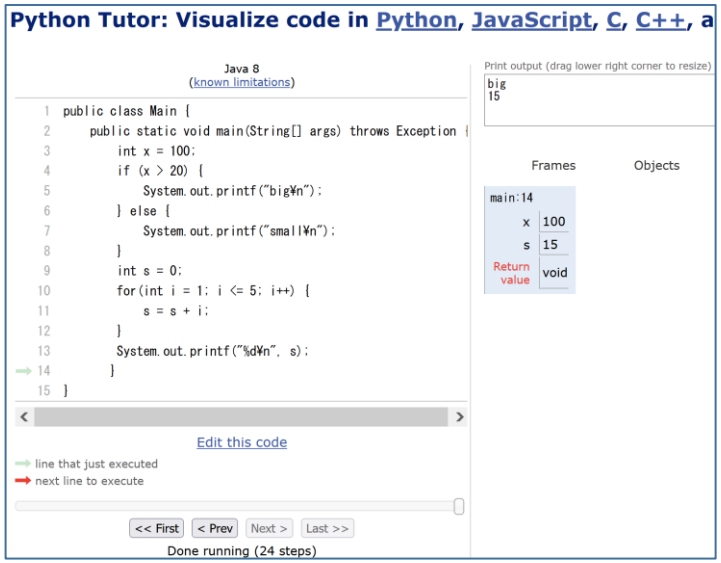
```
public class Main {
    public static void main(String[] args) throws Exception
{
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
    }
}
```

Java

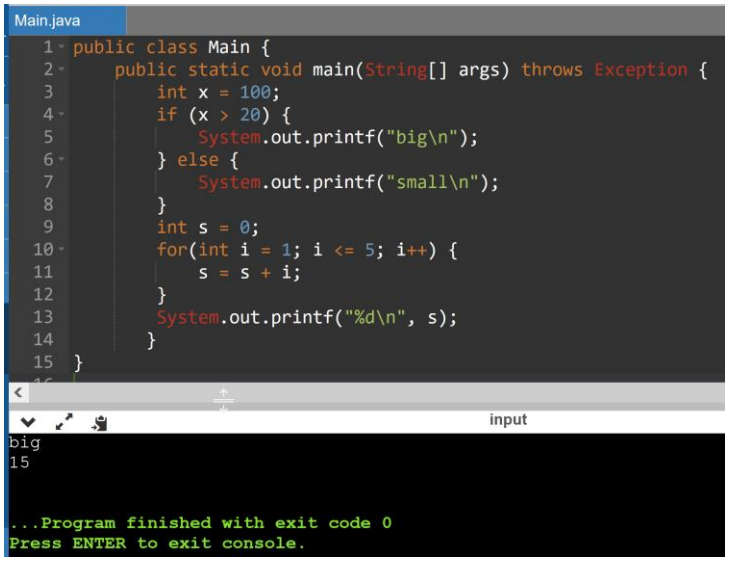
```
#include <stdio.h>
int main(void){
    int x, s, i;
    x = 100;
    if (x > 20) {
        printf("big¥n");
    } else {
        printf("small¥n");
    }
    s = 0;
    for(i = 1; i <= 5; i++) {
        s = s + i;
    }
    printf("%d¥n", s);
    return;
}
```

C

さまざまな プログラミング言語



オンラインでの Java プログラム実行（Java Tutor を使用）

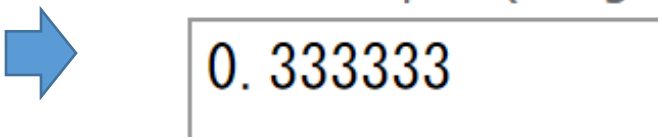


オンラインでの Java プログラム実行（GDB online を使用）

コンピュータは便利なものであるが，コンピュータを使うからといって，**計算が完璧に正確**というわけでは**ない**

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3         System.out.printf("%f\n", 1.0/3.0);
4     }
5 }
```

プログラム



実行結果

アウトライン

番号	項目
1-1	プログラミング
1-2	Java プログラムの実行方法
1-3	オンライン開発環境
1-4	Java Tutor での Java プログラム実行
1-5	GDB online での Java プログラム実行
1-6	計算誤差
1-7	さまざまなプログラミング言語
1-8	この授業の全体計画

各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする

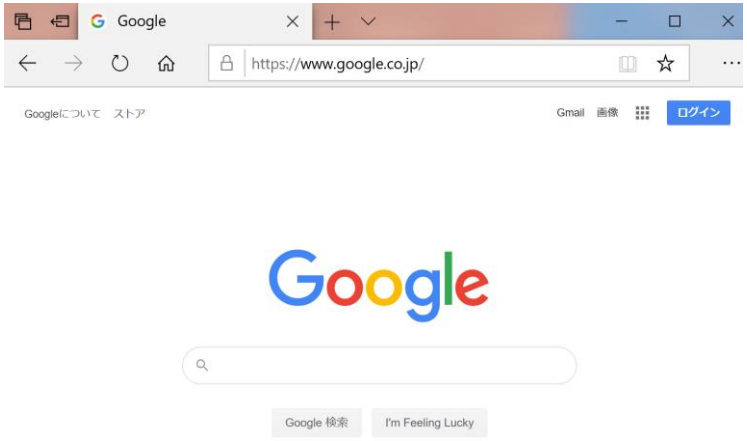
1-1. プログラミング

プログラム

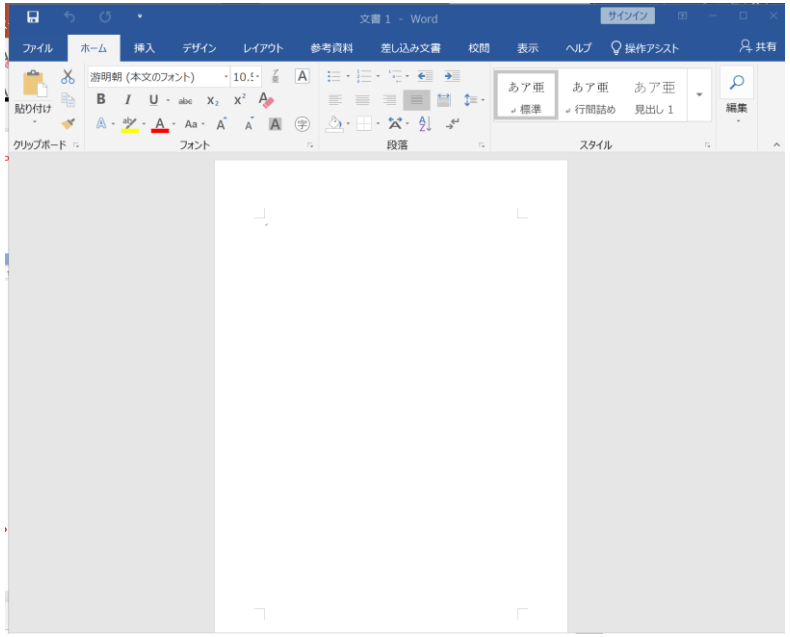
- **コンピュータ**は，**プログラム**で動く

- プログラムを設計，制作することはクリエイティブである

① さまざまなアプリ



Web ブラウザ



ワープロ
（マイクロソフト・ワード）

アプリでは，**プログラム**が動いている

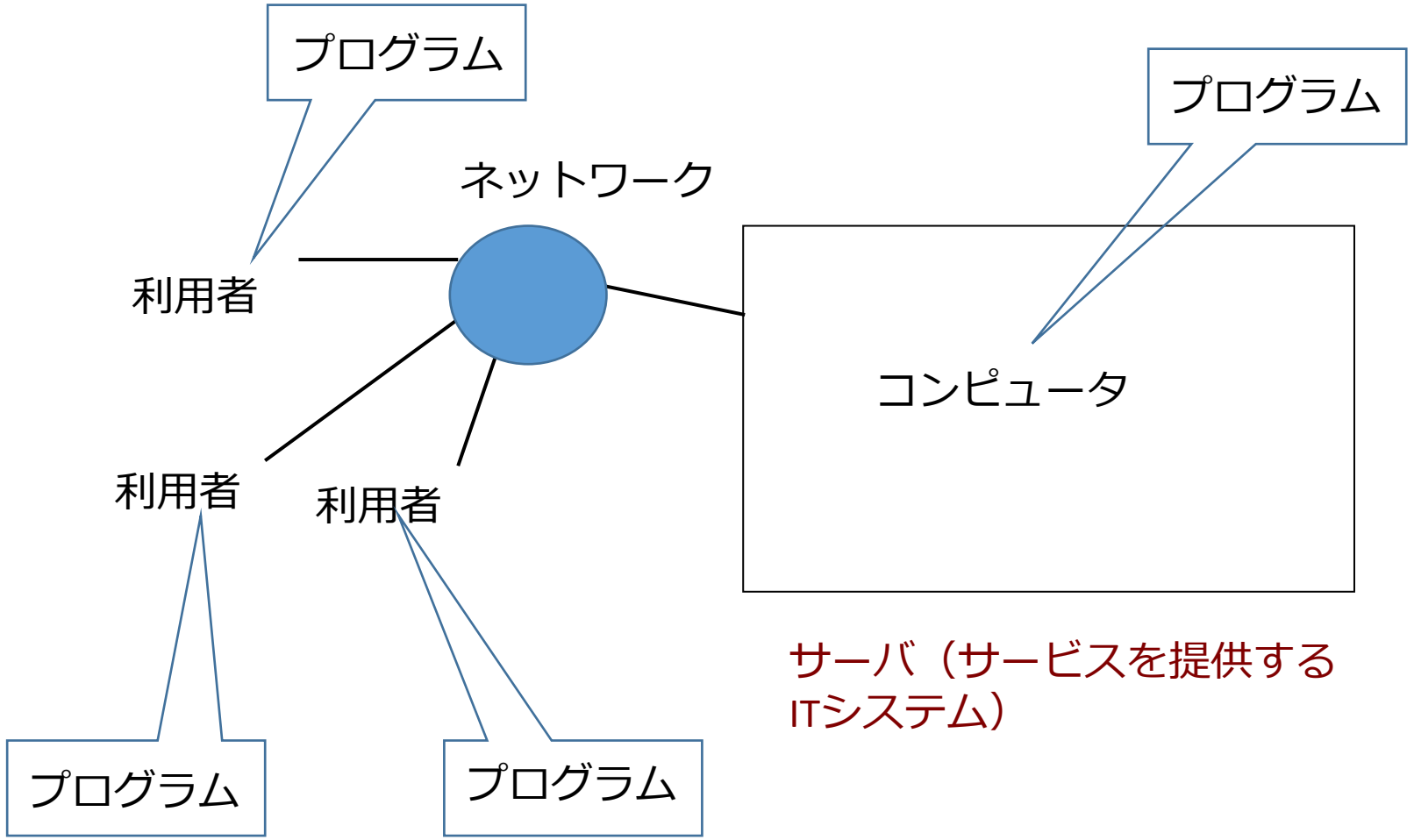
② コンピュータを細かくコントロール

```
In [7]: from keras.models import Sequential
   ...: model = Sequential()
   ...: from keras.layers import Dense, Activation
   ...:
   ...: model.add(Dense(units=64, input_dim=len(x_train[0])))
   ...: model.add(Activation('relu'))
   ...: model.add(Dense(units=max(set(y_train)) - min(set(y_train)) + 1))
   ...: model.add(Activation('softmax'))
   ...: model.compile(loss='sparse_categorical_crossentropy',
   ...:               optimizer='sgd',
   ...:               metrics=['accuracy'])
   ...: model.fit(x_train, y_train, epochs=200)
   ...: score=model.evaluate(x_test, y_test, batch_size=1)
   ...: print(score)
   ...: model.predict(x_test)
   ...: model.summary()
Epoch 1/200
3/3 [==============================] - 0s 5ms/step - loss: 1.0583 - accuracy:
0.3200
Epoch 2/200
3/3 [==============================] - 0s 0s/step - loss: 1.0530 - accuracy:
0.3200
Epoch 3/200
3/3 [==============================] - 0s 0s/step - loss: 1.0485 - accuracy:
0.3200
```

人工知能のプログラム（Python 言語）

ニューラルネットワークを作成している

③　コンピュータどうしがつながるときもプログラムが必要



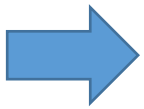
サーバ（サービスを提供するITシステム）

プログラミング (programming)

- **コンピュータ**は、**プログラム**で動く
- **プログラミング**は、**プログラムを設計、製作する**こと
- **何らかの作業**を、**コンピュータ**で実行させるために行う

```
public class YourClassNameHere {
    public static void main(String[] args) {
      int x = 100;
      int y = 200;
      System.out.printf("%d", x * y);
    }
}
```

プログラムの
ソースコード
（Java 言語）



```
20000
```

プログラムの
実行結果

ソースコード (source code)

- **プログラム**を，何らかの**プログラミング言語**で書いたもの
- 「**ソフトウエアの設計図**」ということも.

人間も読み書き，編集できる

```
public class YourClassNameHere {
    public static void main(String[] args) {
        int x = 100;
        int y = 200;
        System.out.println(x + y);
    }
}
```

100 × 200 を計算する Java 言語プログラム

プログラムが役に立つ理由

① プログラム次第で，様々な処理が可能.

② **プログラム**は，コンピュータでの様々な**処理**を**自動化**する

③ **プログラム**の**ソースコード**は，**作業記録**としても使うことができる．**いつでも再現できる**.

④ **プログラム**中の**値などを変えて再実行**も簡単

プログラミングで気を付けること

① コンピュータにも，**できないことがある**

② コンピュータを使うからといって，**計算が完璧に正確**というわけでは**ない**

③ **人間**がプログラムを作るとき，書き間違い，勘違い，思い込みなどによる**ミスがありえる**.

④ 「**プログラムが期待通りに動いているか**」の**テストが重要**

⑤ **ミスを減らす**ためにも，**「やりたいこと」を１回書いて済ませる**ことが大切．次のようなさまざまな手段がある

- **抽象化**
- **標準ライブラリ**
- **クラス階層**

⑥ **問題**をコンピュータで解くとき，解くべき**問題を深く理解**した上で，必要に応じて，**算法（アルゴリズム）**を活用する

1-2. Java プログラムの実行方法

プラットフォームとは

- もともとは、**大地**、**乗り降り場**等の意味

- IT では、ソフトウエア等を**動作させるのに必要**な**機器**や**ソフトウエア**のこと

  Windows 10 + パソコン

  Max OS X + パソコン

  Linux + サーバコンピュータ

  Android + スマホ

Java 言語の良さ

- さまざまな**プラットフォーム**で、**同じプログラムが動く（プラットフォーム非依存）**

私の見解

・**Java** の**登場前**は，「違うプラットフォームで動かすときは、プログラムの書き替えが**必要**」なのが常識

・Java の登場により，これが変化．人気の理由

・インターネットの普及により，Windows, Linux 等がミックスして動く IT システムが当たり前．Java は便利に利用できる．

Java 言語の特徴

1. さまざまな**プラットフォーム**で、**同じプログラムが動く**（**プラットフォーム非依存**）

2. **オブジェクト指向**の**プログラミング言語**である

3. 標準ライブラリ（標準機能として備わっているライブラリ）が充実している

4. C++言語と書き方が類似
   Java のことを「C++ の改良」という人も

Java のプログラムを動かすには

Java の**ソースコード**

```
public class Main {
    public static void main(String[] args) throws Exception {
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
       }
}
```

ファイル名: Main.java

コンパイル（ビルド）

バイトコード

自動結合

Java 仮想マシン

標準ライブラリ

全部がそろって，
１つの**アプリケーション**

コンパイル（ビルド）は、
ソースコードを**バイトコード**に
変換する操作

これで，プラットフォーム
非依存を達成

Java のプログラムのコンパイル（ビルド）と実行

Java のソースコード

```
public class Main {
    public static void main(String[] args) throws Exception {
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
      }
}
```

ファイル名: Main.java

Java のルール

Java のアプリケーションを起動すると，**main メソッド**が**実行される**

```
kaneko@www:~$ javac Main.java
kaneko@www:~$ java Main
big
15
kaneko@www:~$
```

javac は、**コンパイル（ビルド）**を行うコマンド

java は**アプリケーションの起動**を行うコマンド

1-3. オンライン開発環境

プログラム開発環境

プログラム開発環境は，**プログラミング**におけるさまざまなことを支援する機能をもった**プログラム**

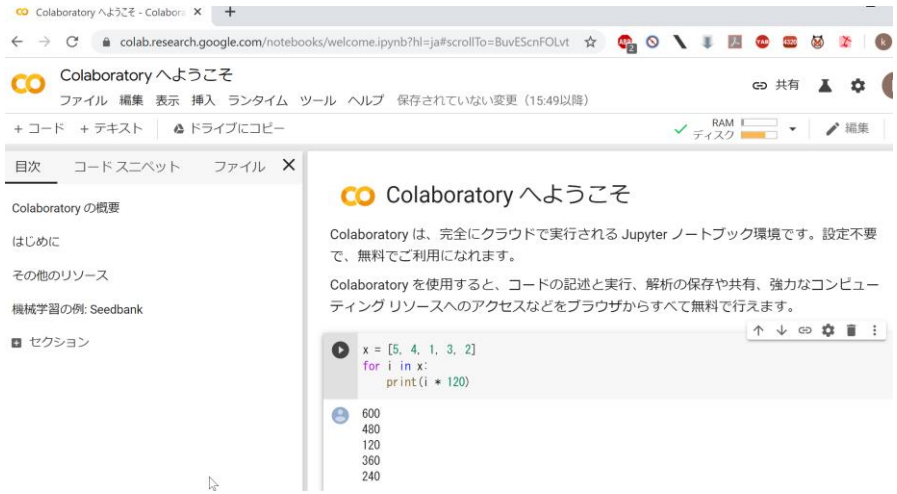
- プログラムの作成，編集（**エディタ**）
- プログラム中の誤り（**バグ**）の発見やテストの支援（**デバッガ**）
- プログラムの実行
- マニュアルの表示
- プログラムが扱うファイルのブラウズ
- プログラムの配布（**パッケージ機能**など），共有，共同編集
- バックアップ，バージョン管理

これらが簡単に行えるようになる

オンラインのプログラム開発環境

- **プログラム開発環境**の操作は，ウエブブラウザでできる
- 自分のパソコンに，特別なソフトをインストールする**必要がない**
- 機能制限がある場合が多い
- 利用登録の有無と内容，利用条件，料金については，利用者で確認のこと

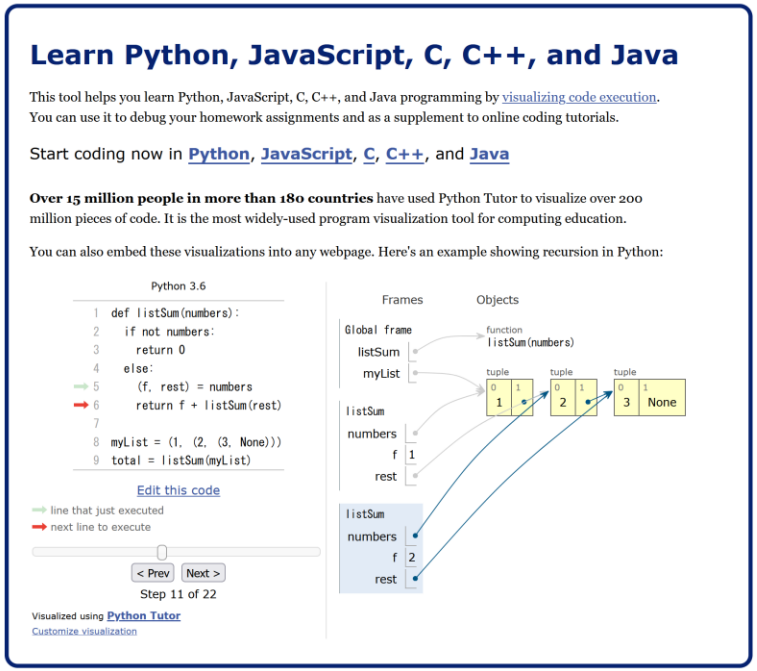
プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ①



Google Colaboratory

Python の開発環境
多数のパッケージがインストール済み
ノートブックにより、記録が簡単に残せる．ビジュアルな表示も簡単に可能
プログラムの共有も簡単

https://colab.research.google.com/

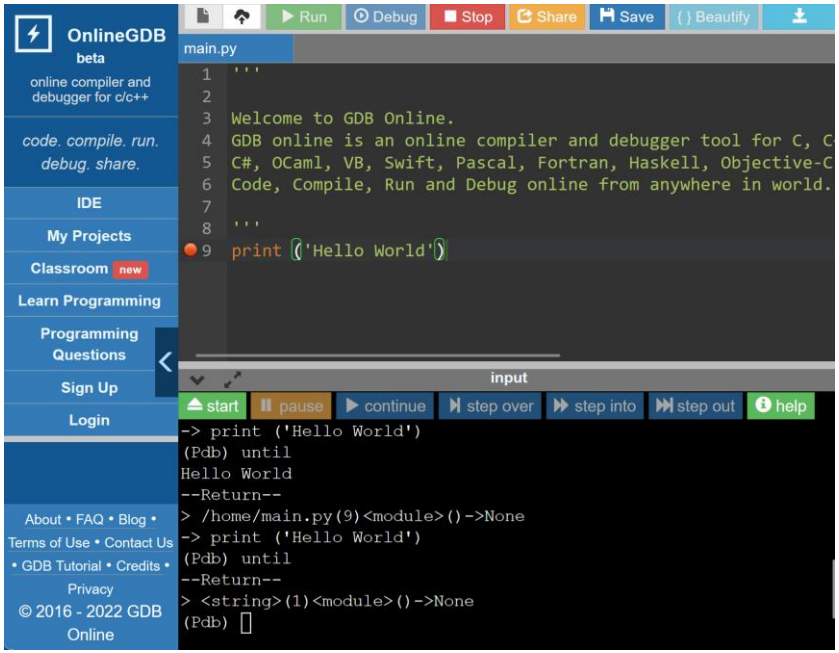


Java Tutor

Python, JavaScript, C, C++, Java
ステップ実行、オブジェクトの表示がビジュアルに

https://pythontutor.com/java.html

プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ②

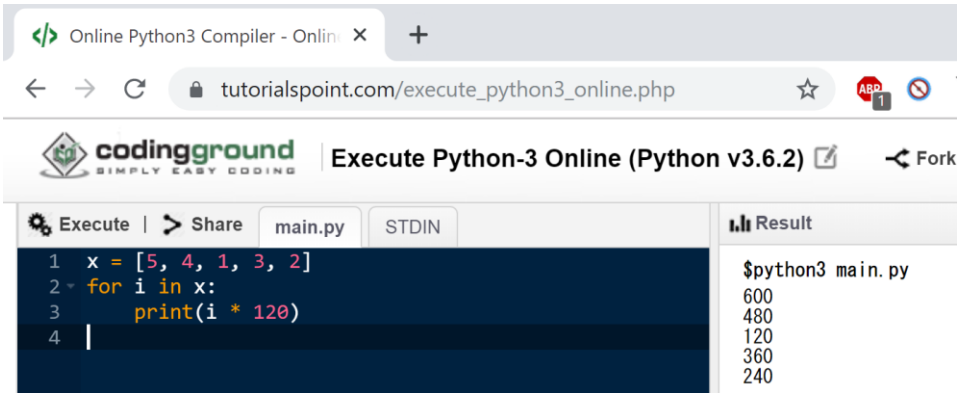


GDB online

C, C++, Java, Python, PHP, C#, OCam, VB, HTML, Ruby, Perl, Pascal, R, Fortran, Haskell, アセンブリ, Objective C, SQLite, Javascript, Prolog, Swift, Rust, Go, Bash

デバッガの機能あり

https://www.onlinegdb.com/



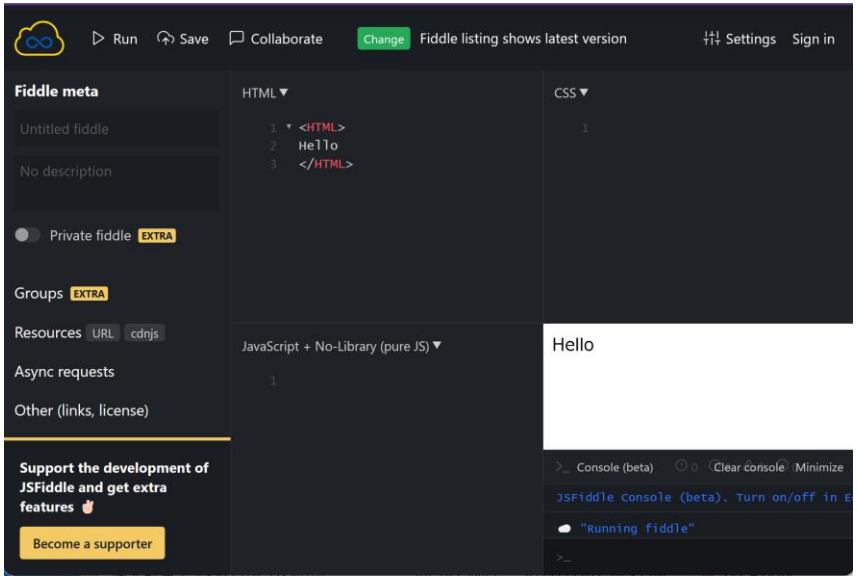
Coding Ground

Python, C, Java, JavaScript, R, Octave/MATLAB, SQL, bash, アセンブリ, MySQL, SQLite, その他多数

ファイル作成，ファイル読み書き，複数プログラムファイルの組み合わせ可能

https://www.tutorialspoint.com/codingground.htm

プログラム作成ができるウエブサービス（オンラインの開発環境）の例 ③



JSFiddle

HTML, CSS, JavaScript
見た目をオンラインで確認

https://jsfiddle.net/

Paiza.IO

Python, C, Java, JavaScript, R, MySQL
など多数
表示は日本語.
一定の条件下でファイル操作も可能

https://paiza.io/

無料のオンラインサービス

- 一定の条件下で無料で使える.
- 活用によって，ICTはより便利になる.

【マナー】

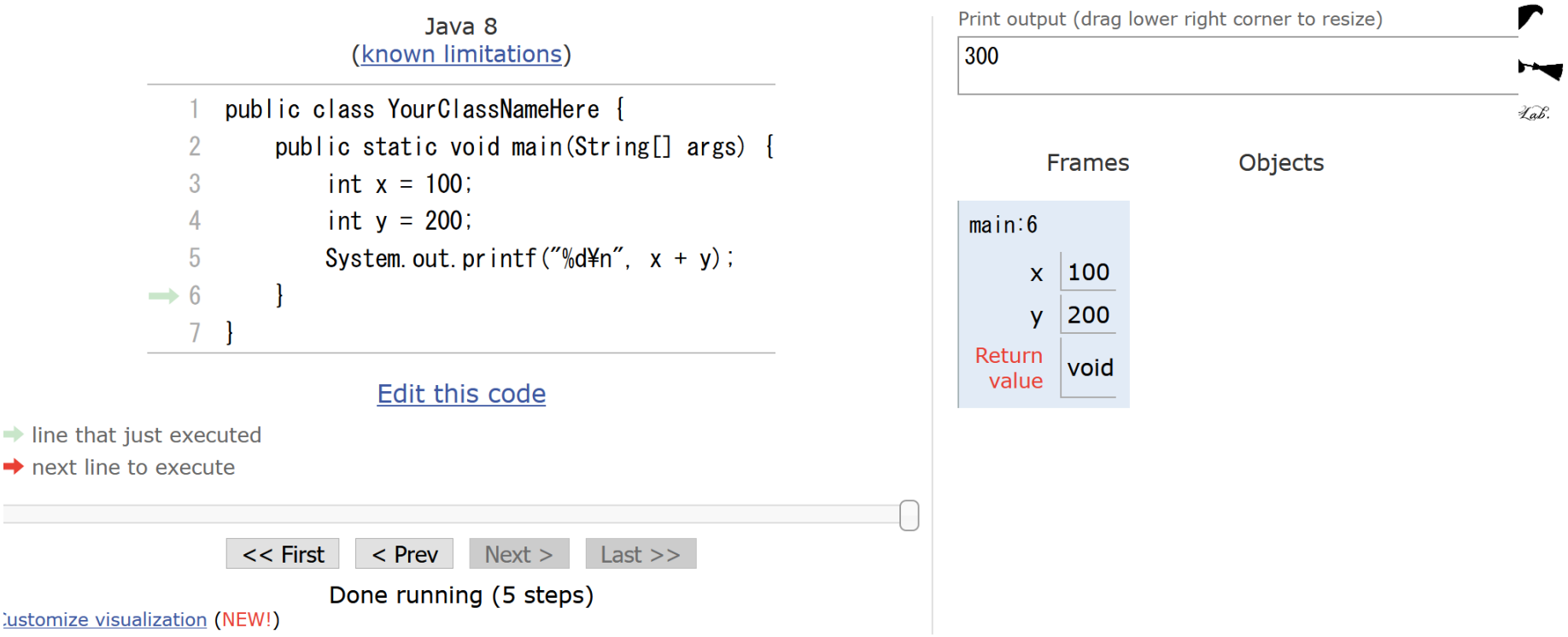
・作者が定める**利用条件**を確認

・**著作権**を尊重

・安全意識：秘密にしたいデータをアップロードしない

・広告等が表示される場合がある

1-4. Java Tutor での Java プログラム実行



unsupported features

Java などのプログラミング言語の体験，演習ができる**オンラインサービス**

Java Tutor

http://www.pythontutor.com/

オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

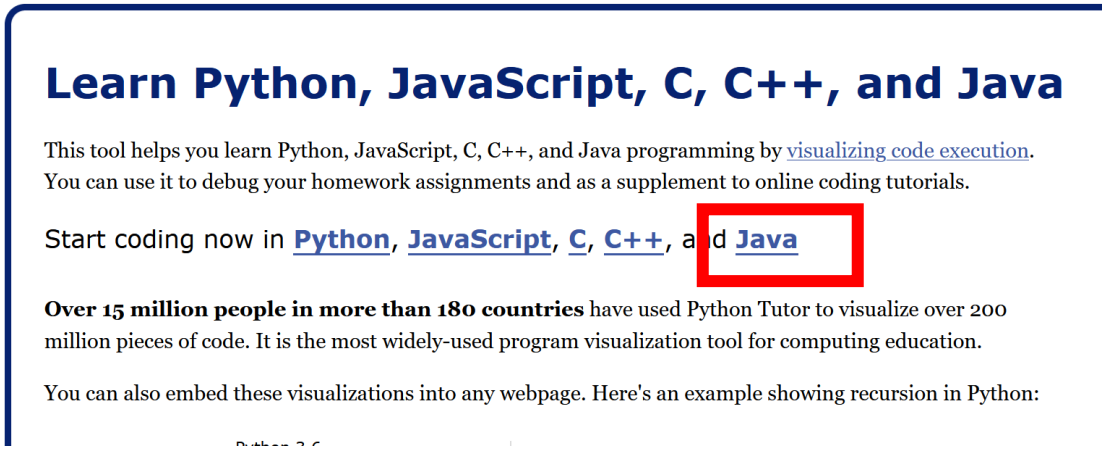
Java Tutor の起動

① **ウェブブラウザ**を起動する

② **Java Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

③ 「**Java**」をクリック　⇒ **編集画面**が開く



Java Tutor の編集画面

Java debugger and visualizer - Java Tutor - Learn Java programming by visualizing code (also debug Python, JavaScript, C, and C++ code)

Write code in Java 8

「Java 8」になっている

```
1 public class YourClassNameHere {
2     public static void main(String[] args) {
3
4     }
5 }
```

最初から main メソッドの
ひな形が入っている

エディタ
（プログラムを書き換えることができる）

Visualize Execution

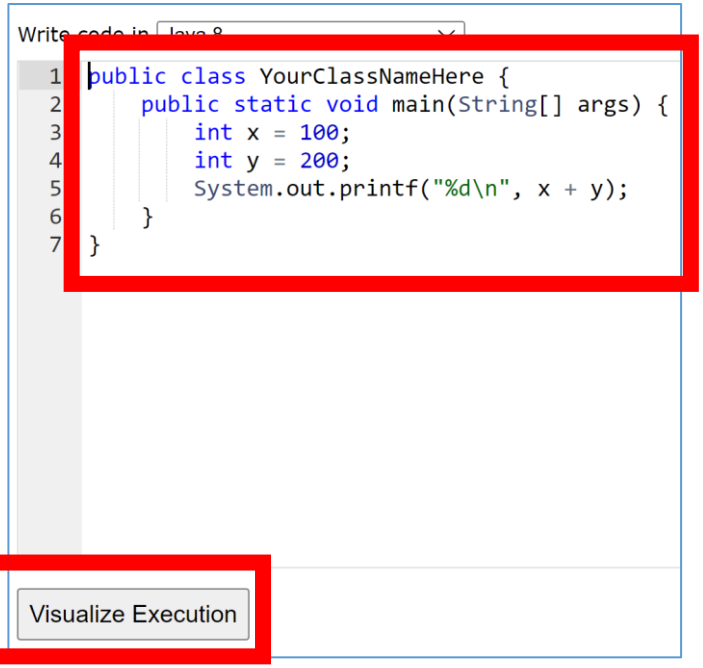
実行のためのボタン

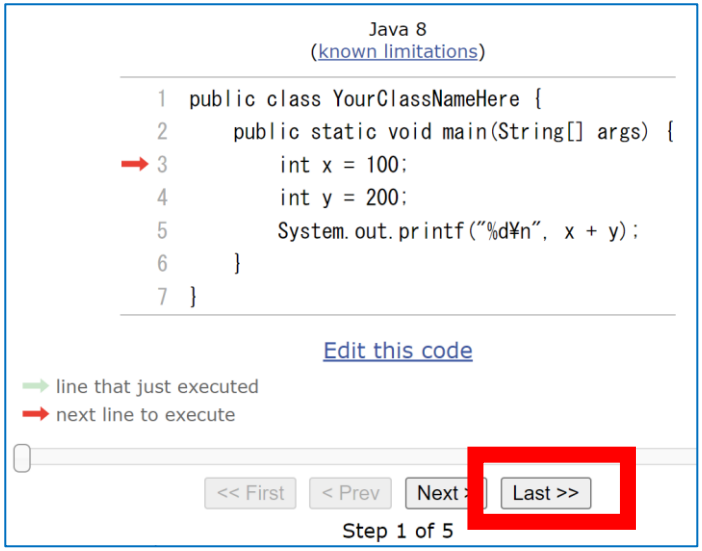
inline primitives, don't nest objects [default] | draw pointers as arrows [default]

Show code examples

Generate permanent link

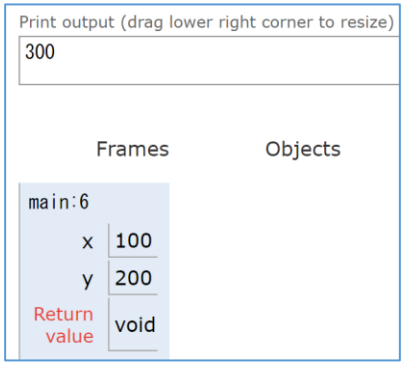
Java Tutor でのプログラム実行手順

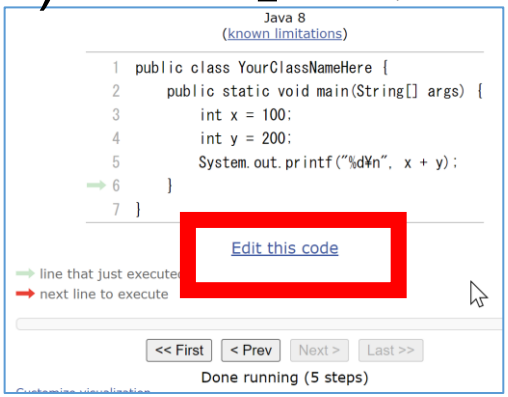




(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える

(2)「**Last**」をクリック.





(3) **<u>実行結果を確認</u>**する.

(4)「**Edit this code**」をクリックして**編集画面**に戻る

Java Tutor 使用上の注意点①

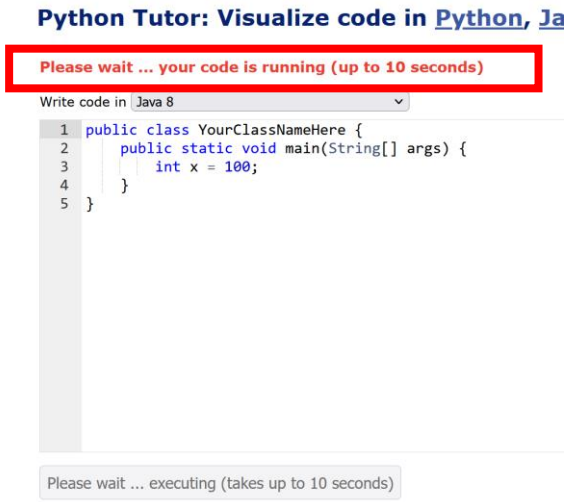
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  過去の文法ミスに関する確認表示

  邪魔なときは「**Close**」

Java Tutor 使用上の注意点②

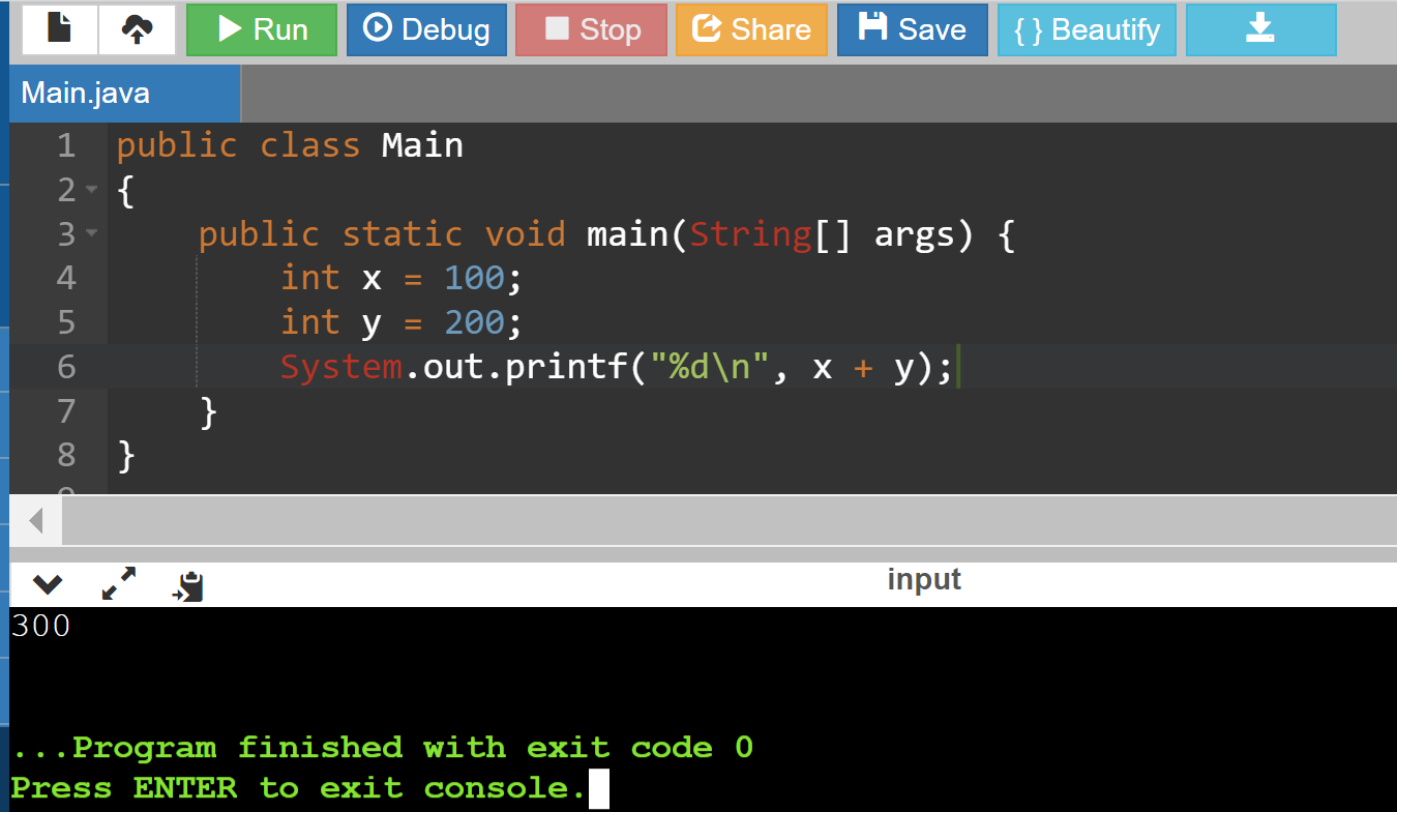
「**please wait ... executing**」のとき，**１０秒ほど待つ**.



→ **混雑しているとき**は，**「Server Busy・・・」というメッセージが出る**ことがある.

混雑している．**少し（数秒から数十秒）待つ**と自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

1-5. GDB online での Java プログラム実行



Java などのプログラミング言語の体験，演習ができる**オンラインサービス**

GDB online

https://www.onlinegdb.com

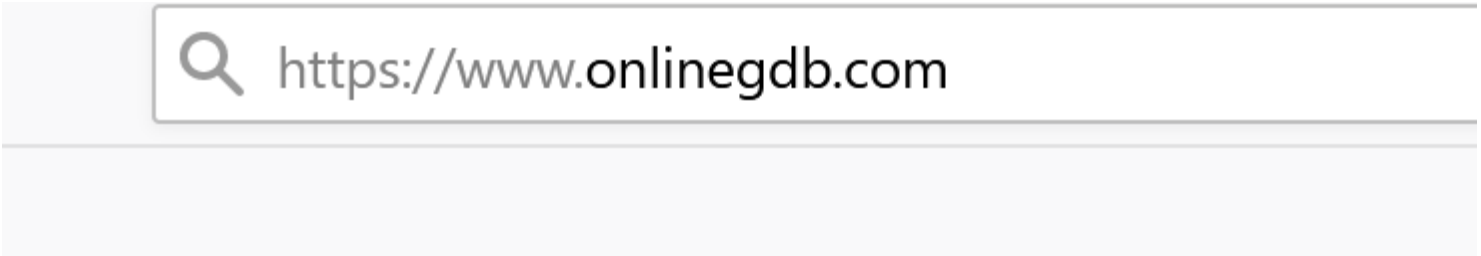
オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

GDB online で Java を動かす手順

① ウェブブラウザを起動する

② 次の URL を開く

https://www.onlinegdb.com

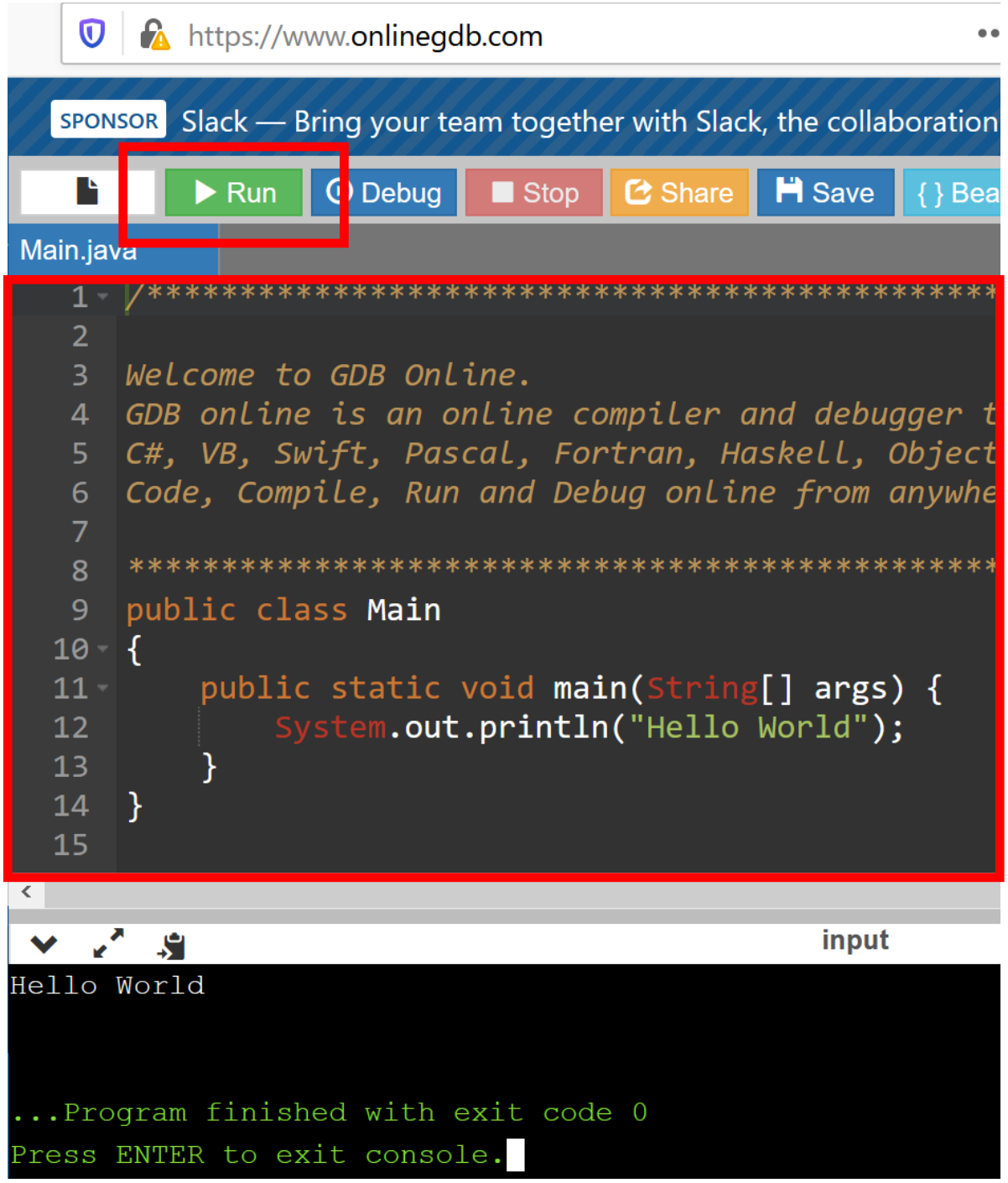


③「Language」のところで，「Java」を選ぶ

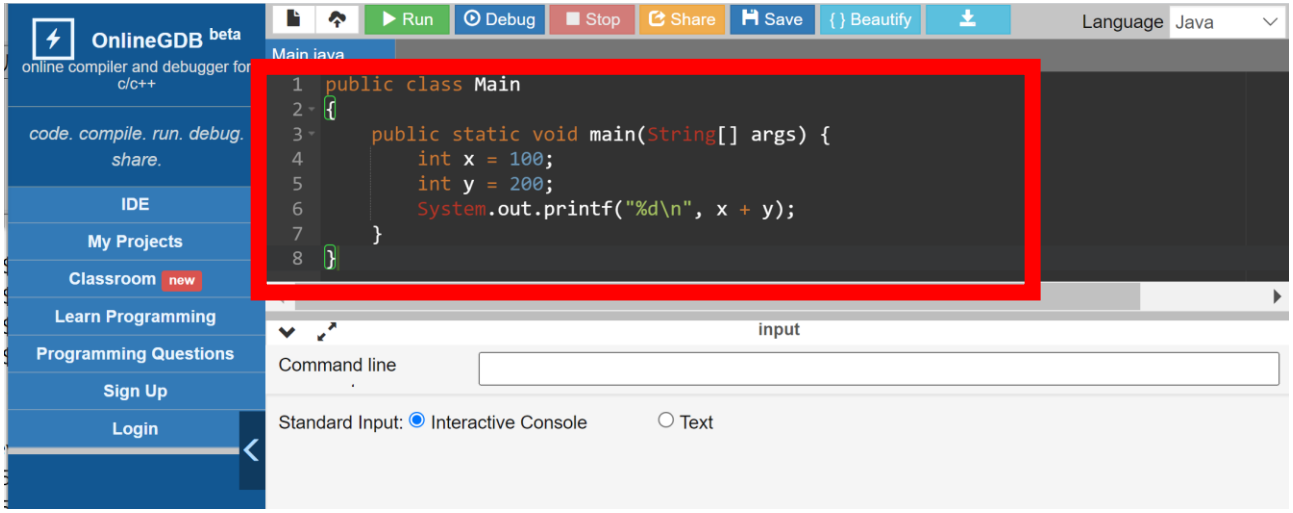
実行ボタン

エディタ画面

プログラムを
書き換えることができる

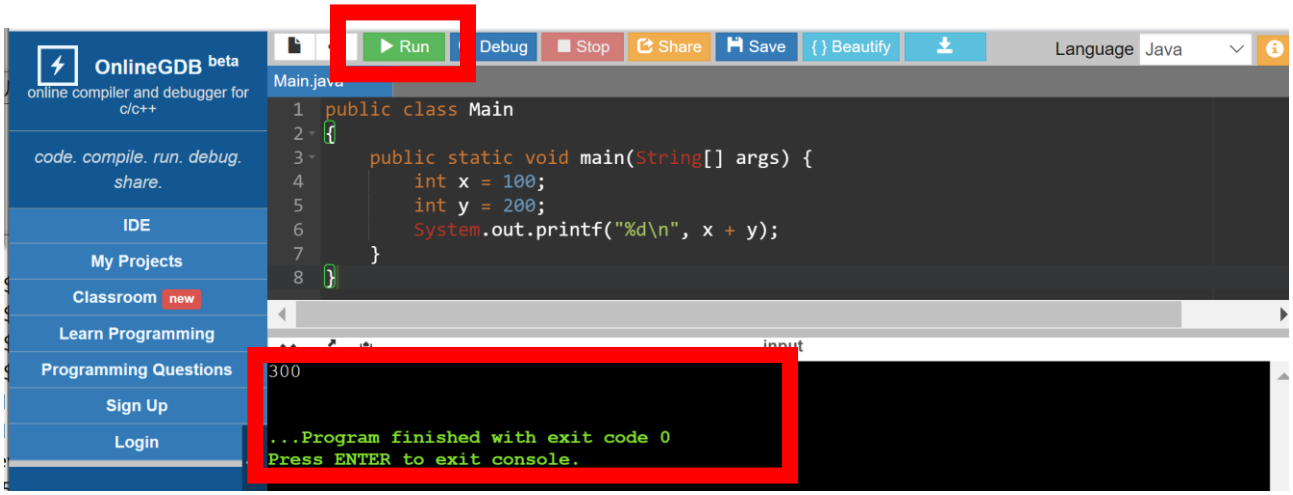


④ ソースコードを入れる



⑤ 実行．実行結果を確認

「**Run**」をクリック．



1-5. 計算誤差

コンピュータで「１÷３」を求めるとどうなると思いますか

1. 0.33333333333333333333333333333333 と無限に表示される
2. 計算できない
3. 正確な値が表示されない（誤差を含む）

演習

資料：**43 ～ 45**

【トピックス】

- **計算誤差**

① Java Tutor のエディタで次のプログラムを入れる

```
public class YourClassNameHere {
    public static void main(String[] args) {
        System.out.println(1.0/3.0);
    }
}
```

② 実行するために，「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認

③ 「**Edit this code**」をクリックして，エディタの画面に戻る

④ Java Tutor のエディタで次のプログラムを入れる

```
public class YourClassNameHere {
    public static void main(String[] args) {
        System.out.println(6 * 1.1);
    }
}
```

⑤ 実行するために，「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認

結果を確認

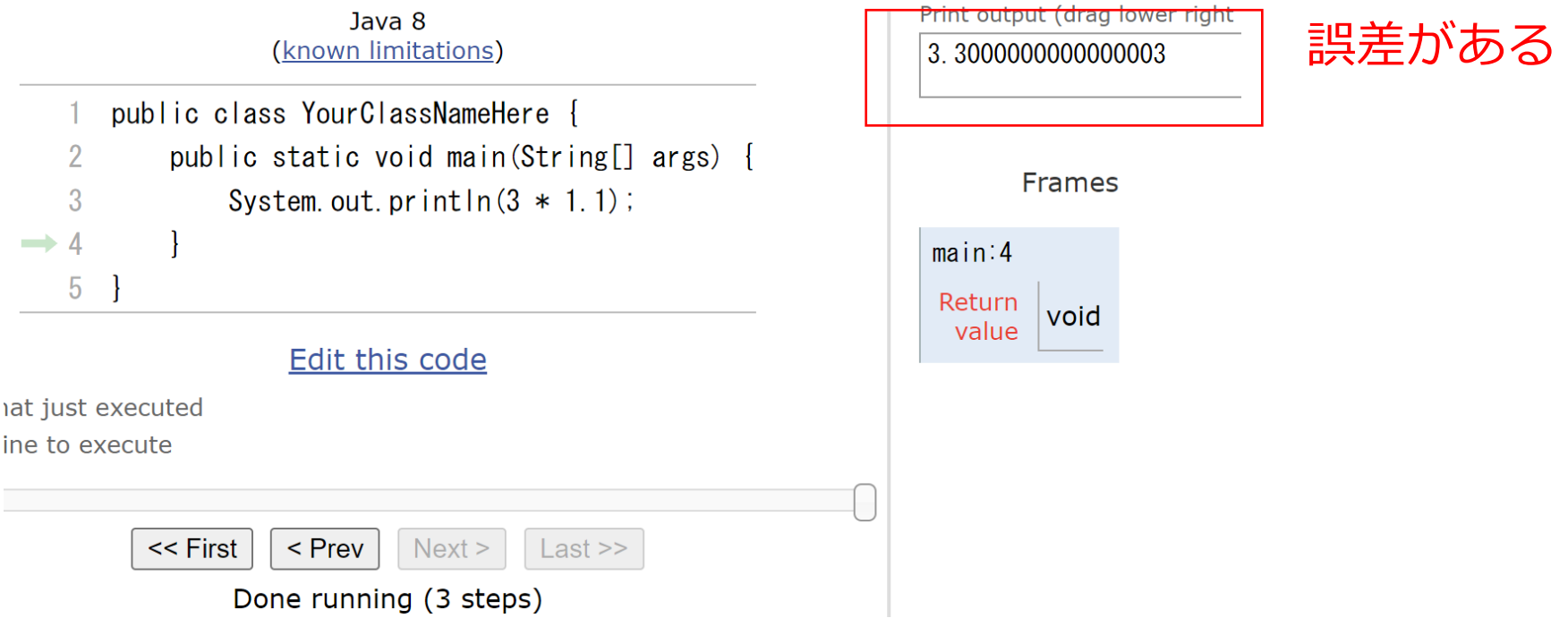
誤差がある

⑥ 「**Edit this code**」をクリックして，エディタの画面に戻る

⑦ Java Tutor のエディタで次のプログラムを入れる

```
public class YourClassNameHere {
    public static void main(String[] args) {
        System.out.println(3 * 1.1);
    }
}
```

⑧ 実行するために，「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認



- コンピュータだから「計算が完璧に正確」という思い込みはしないこと

  1 ÷ 3 を計算して表示させると、

  正確な値が表示されない（誤差を含む）

- 誤差があっても、十分に役に立つ

- 誤差を許しているから、計算が効率的に済むという考え方もある

1-6. さまざまなプログラミング言語

プログラミングを学ぶときに気を付けること

- **プログラミング言語**には，種類が**数多く**ある

- **基礎**となる知識が大事.

　一度，あるプログラミング言語で**基礎をマスター**しておけば，**他のプログラミング言語**でも**応用が利く**，という考え方も

プログラミング言語は複数ある

- **複数のプログラミング言語を学ぶことは大事.**
  賛成できますか？

- 「１つを知っていれば，**どの言語も大体似ている**ので，**応用が利く**」という考え方もある.

- 「やりたいこと，学びたいことに向いた言語を，**そのときどきで選ぶ**のが，一番良い」とも.

- 人によって「**好きな言語**が違う」ということも

さまざまなプログラミング言語

- Python
- C
- Java
- JavaScript
- R
- Octave
- Scheme

など

ここで行う作業

1．20 より大きければ「big」、
　　さもなければ「small」と表示

2．0 + 1 + 2 + 3 + 4 + 5 を求める

国家資格取得にも関係する
（Java, Python, C/C++）

なぜプログラミング言語はたくさんあるのでしょうか？

それぞれ特徴がある	**Java**	**Python**	**C / C++**	**R**	**SQL**	**MATLAB / Octave**
	どのコンピュータでも同じプログラムが動く． 普及度はトップレベル．	初心者向け．そのおかげで，多数の拡張機能も．	コンピュータの性能を最大限引き出す．	「データ処理」に特化したコマンド言語	「データベース」に特化したコマンド言語	「数値計算」，「信号処理」などに特化したコマンド言語

Python プログラム見本

```
x = 100
if (x > 20):
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

- すぐに実行できる
- さまざまな「パッケージ」で機能を拡張できる
- Windows でも Linux でも，ほぼ同じプログラムで動く

Java プログラム見本

```
public class Main {
    public static void main(String[] args) throws Exception {
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
       }
}
```

- Windows でも Linux でも Android アプリでも，同じプログラムで動く

C プログラム見本

```
#include <stdio.h>
int main(void){
    int x, s, i;
    x = 100;
    if (x > 20) {
        printf("big¥n");
    } else {
        printf("small¥n");
    }
    s = 0;
    for(i = 1; i <= 5; i++) {
        s = s + i;
    }
    printf("%d¥n", s);
    return;
}
```

・コンピュータの決め細かなコントロール

・高速実行できるチューニング

JavaScript プログラム見本

Webアプリに向く

```
process.stdin.resume();
process.stdin.setEncoding('utf8');
var util = require('util');
var x = 100;
if (x > 20) {
    process.stdout.write('big¥n');
} else {
    process.stdout.write('small¥n')
}
var s = 0;
for(var i = 1; i <= 5; i++) {
    s = s + i;
}
process.stdout.write(util.format('%d¥n', s));
```

R プログラム見本

```
x <- 100
if (x > 20) {
    print("big")
} else {
    print("small")
}
s <- 0
for (i in c(1,2,3,4,5)) {
    s <- s + i
}
print(s)
```

データ専門家向け

Octave プログラム見本

```
x = 100
if (x > 20)
  printf("big¥n")
else
  printf("small¥n")
endif
s = 0
for i = [1 2 3 4 5]
  s = s + i
endfor
printf("%d", s)
```

行列計算，信号処理など
に向く

Scheme プログラム見本

関数型言語

```
(define (decide x)
    (cond
        ((> x 20) "big")
        (else "small")))
(define (sum n)
    (cond
        ((= n 0) 0)
        (else (+ (sum (- n 1)) n))))
(begin
    (print (decide 100))
    (print (sum 5)))
```

1-7 この授業の全体計画

この授業

- **プログラム**に上達するとよいことがたくさんある
- この授業では，**プログラミング**に関する基礎，大切なことを学ぶ．
- **プログラミング**の初心者を対象

この授業の
主な内容

入門

- オブジェクト
- メソッド
- データの種類
- クラス
- 配列
- 条件分岐，繰り返し

発展

- **バグのないプログラム**を作成するのに役立つ実践
- **プログラムの設計法**
- プログラムでできる**種々の機能**（タイマー，グラフィックスなど）

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/pro/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/pro/pi/index.html

- **Java プログラム例**

https://www.kkaneko.jp/pro/java/index.html