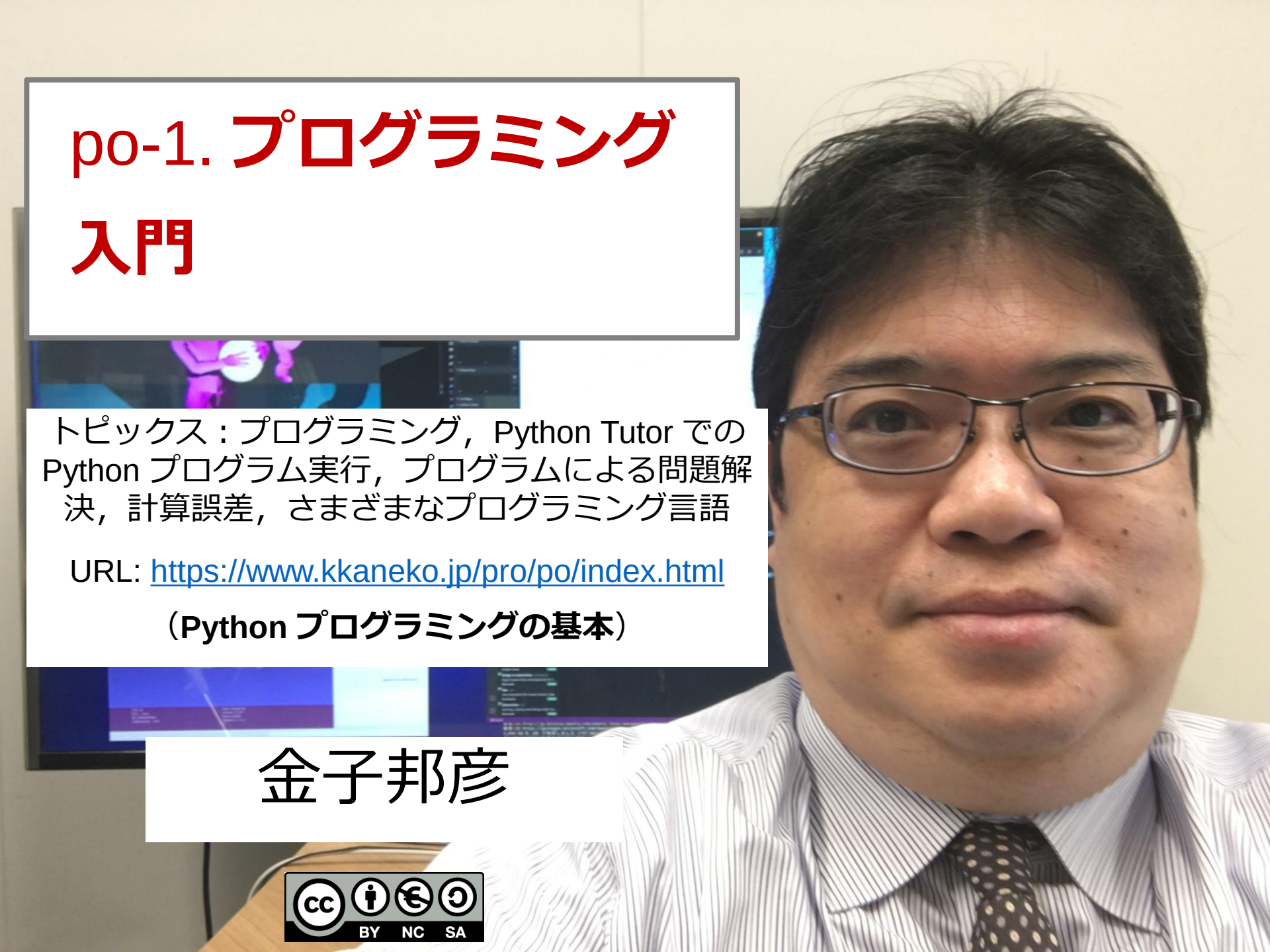
po-1. プログラミング入門



トピックス：プログラミング，Python Tutor での Python プログラム実行，プログラムによる問題解決，計算誤差，さまざまなプログラミング言語

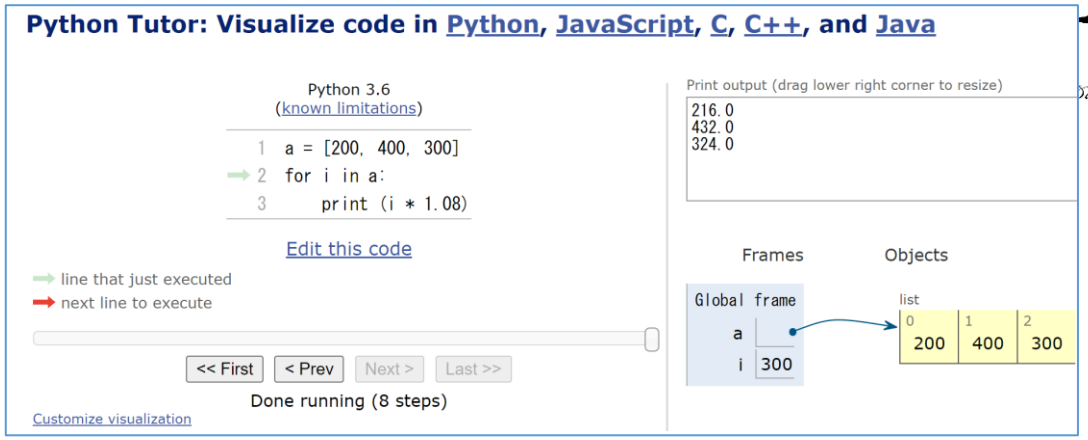
URL: https://www.kkaneko.jp/pro/po/index.html

（**Python プログラミングの基本**）

金子邦彦

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```

Python プログラムの
ソースコード



オンラインでの Python プログラム
実行（Python Tutor を使用）

```python
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

```java
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

```c
#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

24 と **18** の**最大公約数**を求めたい

```
1  import math
2  print( math.gcd(24, 18) )
```

プログラム

Print output (drag lower right c

```
6
```

実行結果

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  print(100 * 1.1)
2  print(150 * 1.1)
3  print(200 * 1.1)
```

プログラム

Print output (drag lower right corner to resize)

```
110.00000000000001
165.0
220.00000000000003
```

Frames Objects

実行結果

アウトライン

	項目
1-1	プログラミング
1-2	Python Tutor での Python プログラム実行
1-3	プログラムによる問題解決
1-4	計算誤差
1-5	さまざまなプログラミング言語
1-6	９回の全体計画

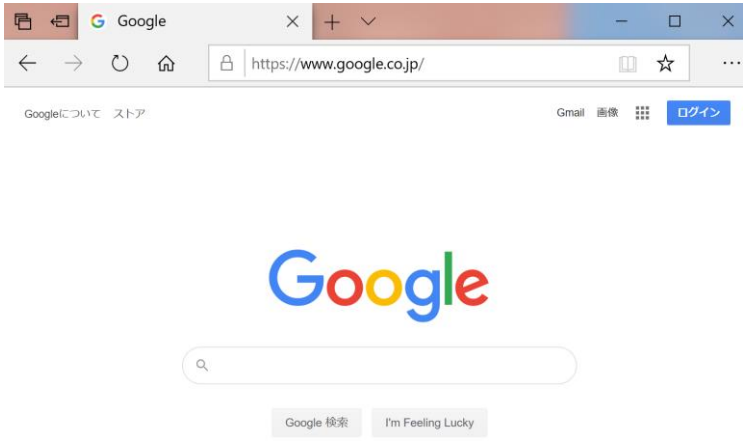
1-1. プログラミング

プログラム

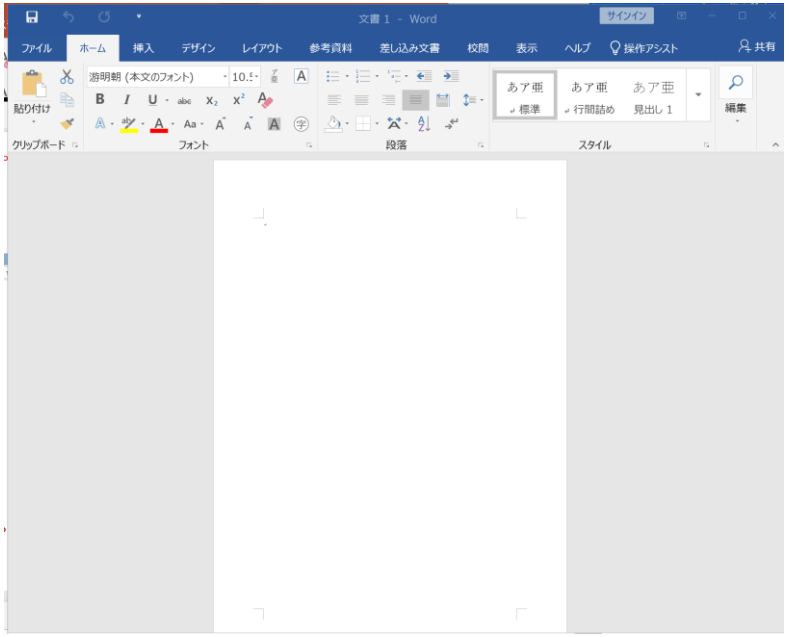
- コンピュータは，**プログラム**で動く

- **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

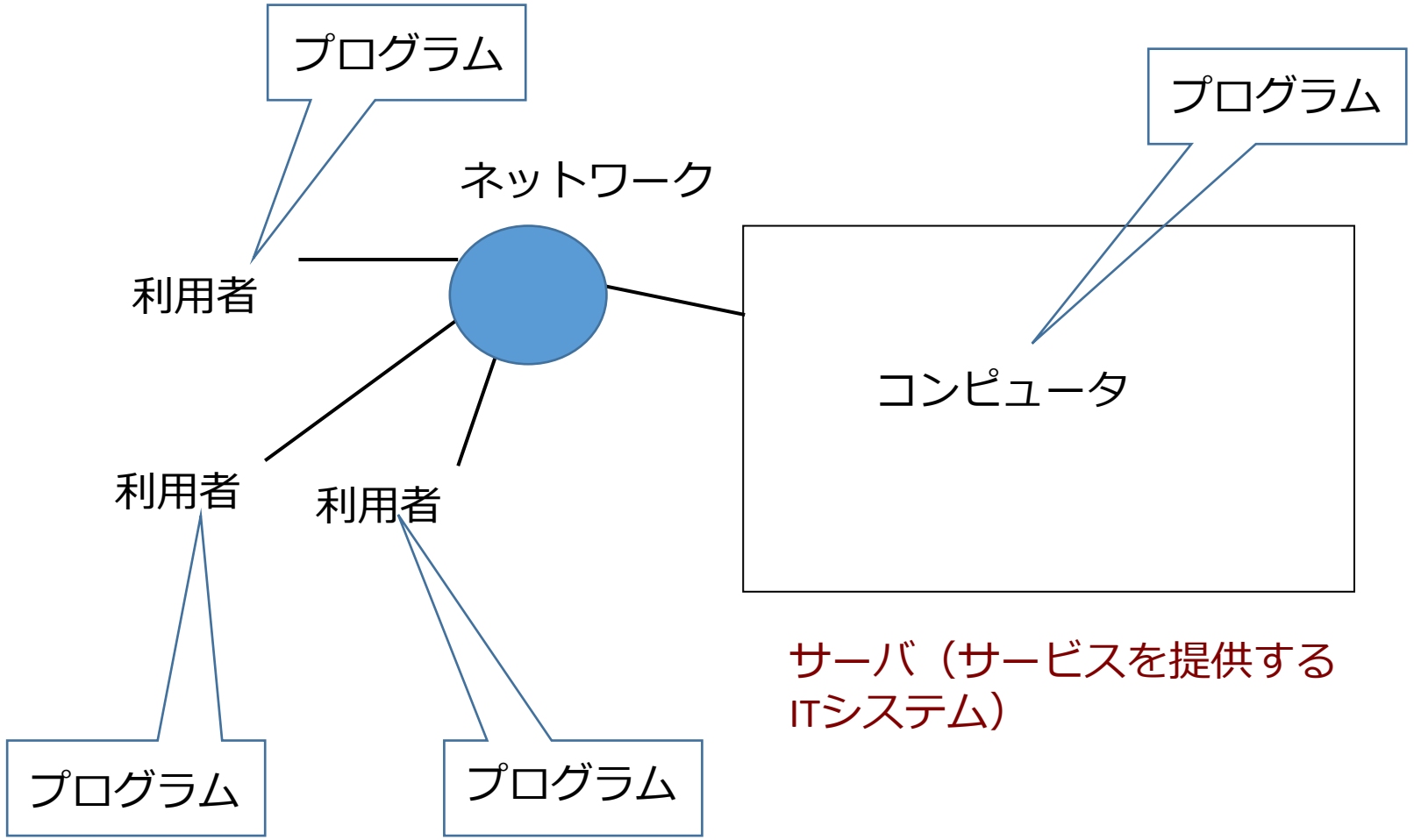
② コンピュータを細かくコントロール


```
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



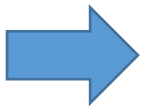
サーバ（サービスを提供するITシステム）

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

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```

Python プログラムの
ソースコード



```
216.0
432.0
324.0
```

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**

```
import picamera
camera = picamera.PiCamera()
camera.capture("1.jpg")
exit()
```

Raspberry Pi で，カメラを使って撮影し，画像を保存するプログラムのソースコード（Python 言語）

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**
- **モジュール，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. Python Tutor での Python プログラム実行

Python

- プログラミング言語
- 「**入門者に学習しやすい**」とされる
- **多数の拡張機能**（外部プログラムのインポートによる）

Python の主なキーワード

- print　　表示
- type　　型名（クラス名）の取得
- if, else　　条件分岐
- for, while　　繰り返し
- def　　関数定義
- return　　関数の評価値
- class　　クラス定義
- __init__　　オブジェクトの生成（コンストラクタ）
- self　　クラス定義内で自オブジェクトへアクセス
- vars　　オブジェクトの属性名と値
- super　　親クラス（スーパークラス）

Python プログラムの実行

① ソースコードを**ファイルに保存**し，python コマンドで実行

```
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)
```

ソースコードを
ファイルに保存

```
kaneko@www:/tmp$ python foo.py
big
15
```

python コマンドで実行.
実行時にファイル名を指定.

- Python のインストール必要
  https://www.python.org
- Windows では，**python** コマンドで実行
- 終了は exit()

Python プログラムの実行

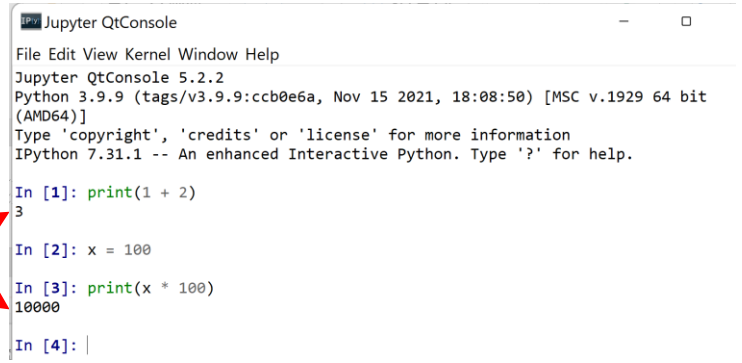
② **Python コンソール**を使用．Python プログラムを入れるたびに結果が得られる（**対話的実行**と言ったりする）．

実行結果

```
C:¥Users¥user>python
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022
Type "help", "copyright", "credits" or "license"
>>> x = 100
>>> if (x > 20):
...     print("big")
... else:
...     print("small")
...
big
>>> s = 0
>>> for i in [1, 2, 3, 4, 5]:
...     s = s + i
...
>>> print(s)
15
>>>
```

python コマンド

実行結果



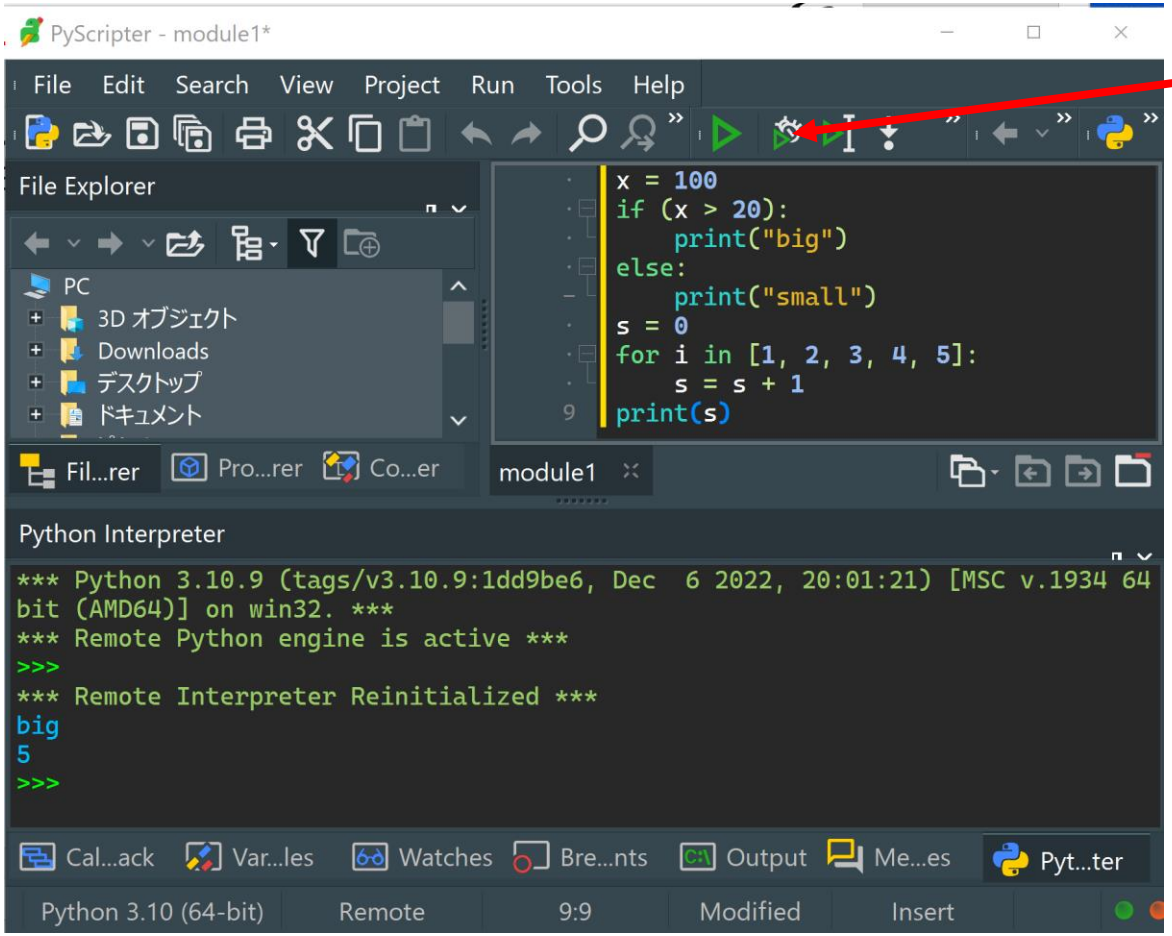
Jupyter QtConsole

- Python のインストール必要
  https://www.python.org
- Windows では，**python** コマンドで実行
- 終了は exit()

- インストール必要
https://www.kkaneko.jp/tools/win/tools.html#python
- 「**jupyter qtconsole**」で起動

Python プログラムの実行

③ Python ソースコードの編集，実行機能を持った**アプリを利用**



実行ボタン

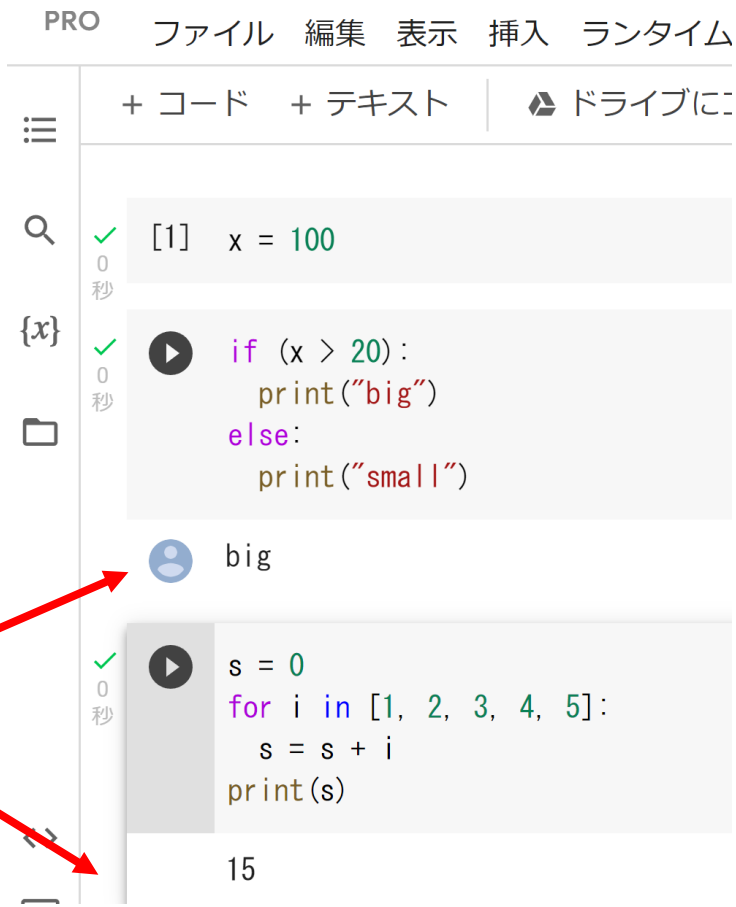
編集画面

実行結果

PyScripter の画面

Python プログラムの実行

④ **Python のノートブック**を使用．Python プログラムを，**コードセル**の中に入れておく．**コードセル**内のプログラムは，編集，実行可能．



Python プログラム

Python プログラム

Python プログラム

実行結果

まとめ

Python プログラムの実行にはさまざまな方法がある

① ソースコードを**ファイルに保存**し，python コマンドで実行

② **Python コンソール**を使用．Python プログラムを入れるたびに結果が得られる（対話的実行と言ったりする）．

③ Python ソースコードの編集，実行機能を持った**アプリを利用**

④ **Python のノートブック**を使用．Python プログラムを，**コードセル**の中に入れておく．**コードセル**内のプログラムは，編集，実行可能．

Python の使い方（Windows パソコン）

- **Python 処理系の起動**（Windows の場合）
  - **python** または **py -3.10** （3.10 はバージョン番号）
- **pip の起動**（Windows の場合）
  - コマンドプロンプトを管理者として実行し，**python -m pip** または **py -3.10 –m pip**（3.10 はバージョン番号）
- **Python 開発環境**
  - Jupyter Qtconsole　　jupyter qtconsole で起動
  - Nteract　　jupyter nteract で起動
  - Jupyter Lab　　jupyter lab で起動
  - spyder　　spyder で起動

Python 処理系と開発環境のインストール手順は、次のページ等で説明

https://www.kkaneko.jp/cc/tools/index.html

開発環境とは

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
- 公開，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

Python Tutor

- プログラミング学習を行える**オンラインサービス**

  http://www.pythontutor.com/

- Web ブラウザを使う

- たくさんの言語を扱うことができる

  Python, Java, C, C++, JavaScript, Ruby など

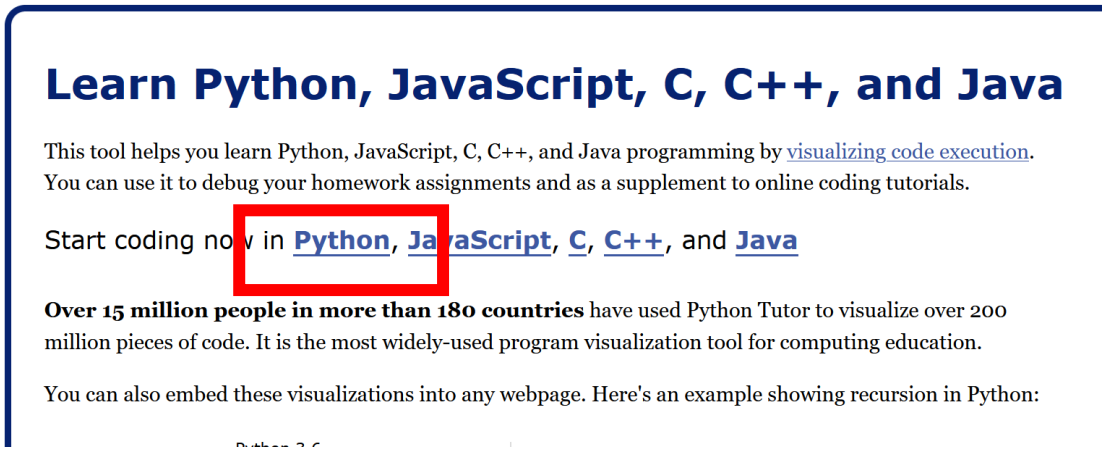
Python Tutor の起動

① **ウェブブラウザ**を起動する

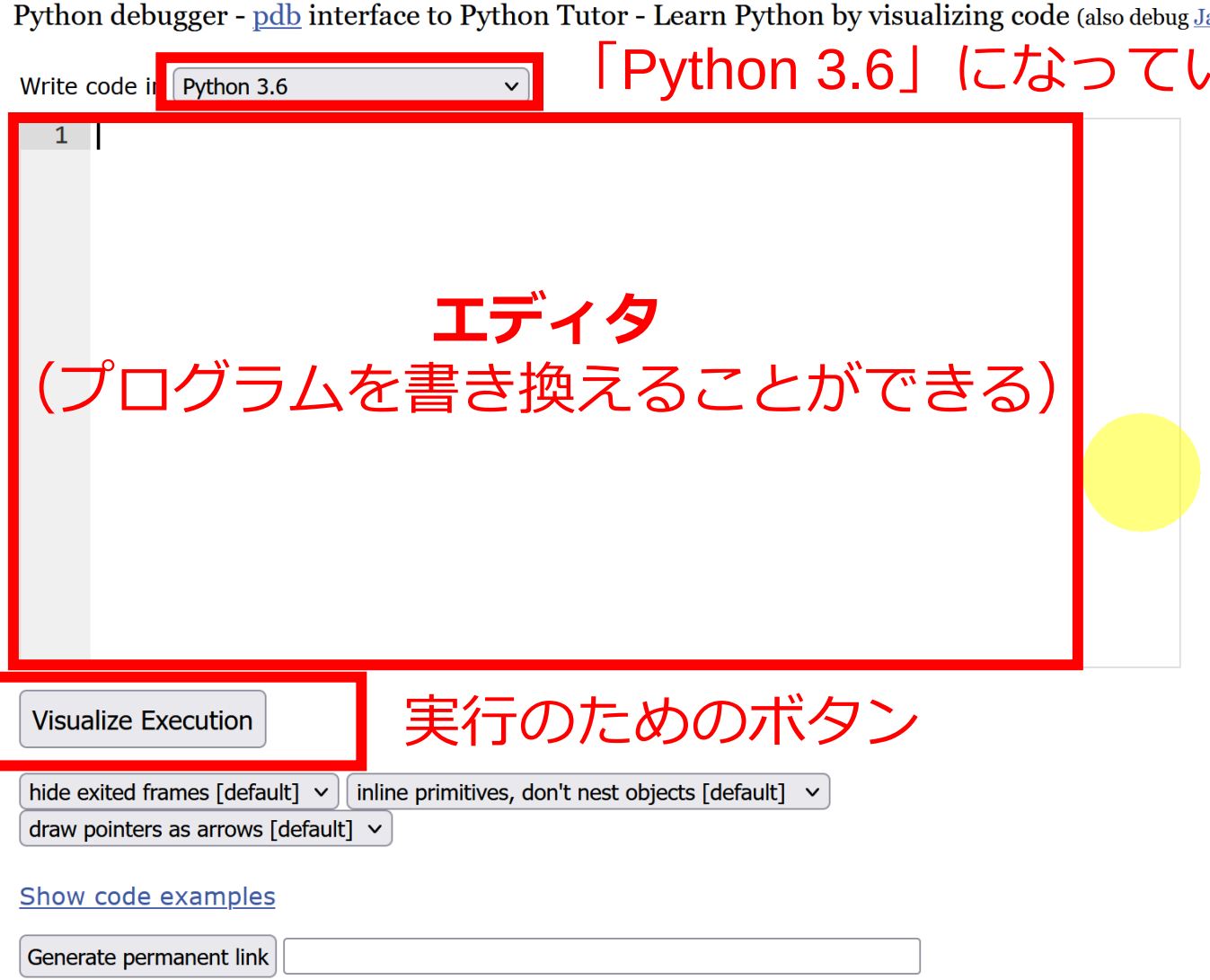
② **Python Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

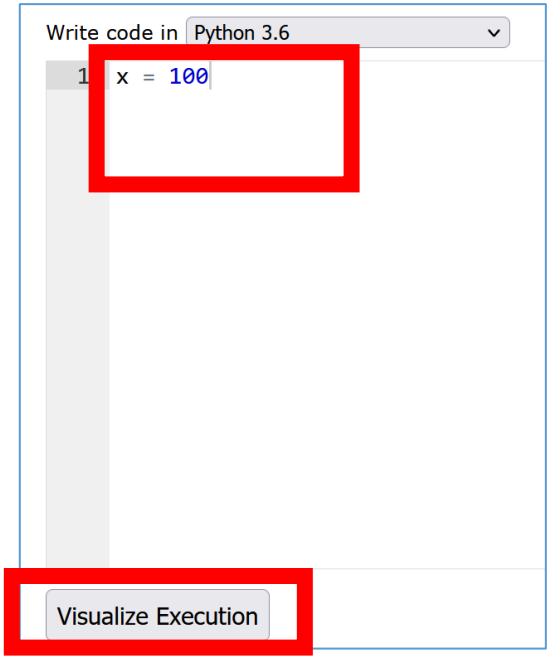
③ 「**Python**」をクリック　⇒ **編集画面**が開く



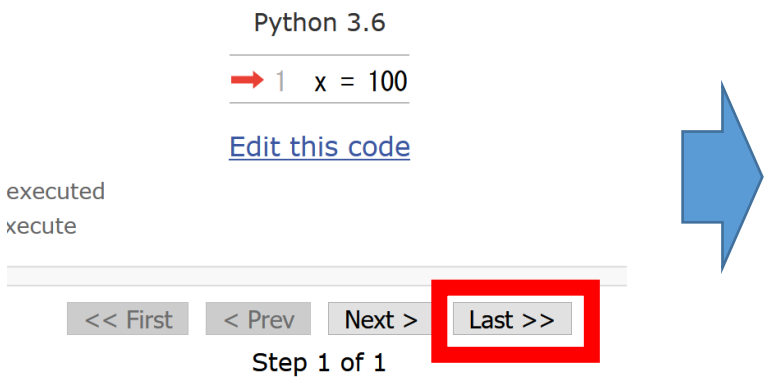
Python Tutor の編集画面



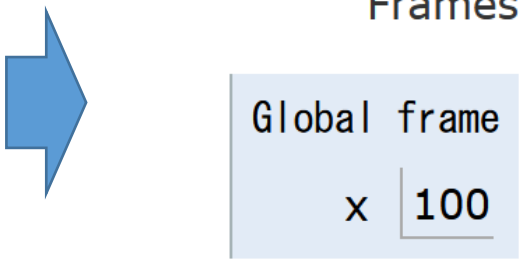
Python Tutor でのプログラム実行手順



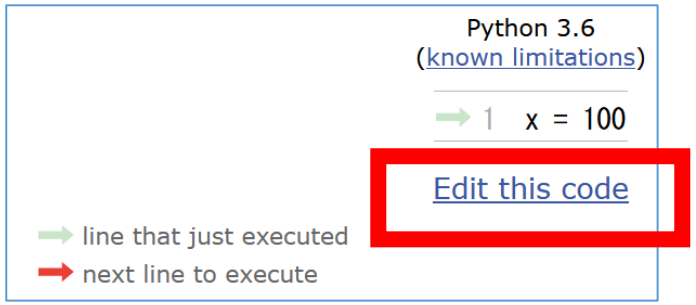
(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える



(2)「**Last**」をクリック.



(3) **実行結果を確認**する.



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

Python Tutor 使用上の注意点①

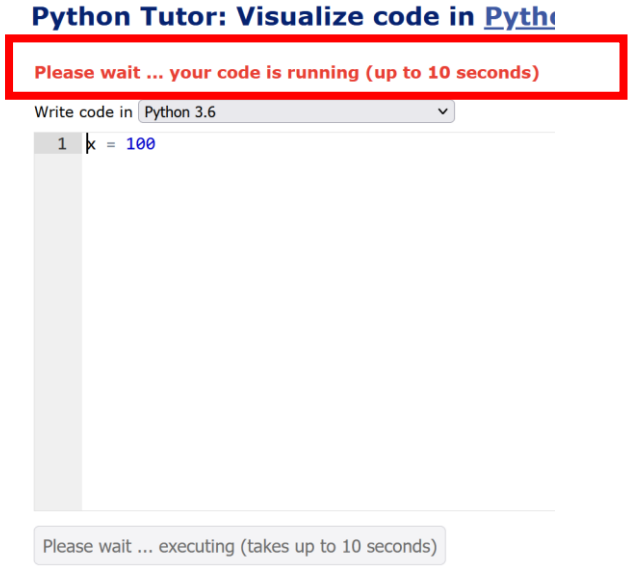
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  過去の文法ミスに関する確認表示

  邪魔なときは「**Close**」

Python 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）

演習

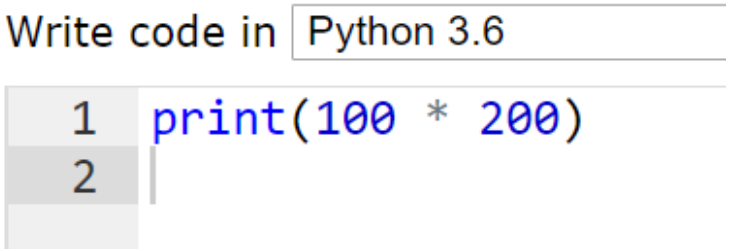
資料：**31 ～ 34**

【トピックス】

- **Python Tutor の使い方**

① Python Tutor のエディタで次のプログラムを入れる

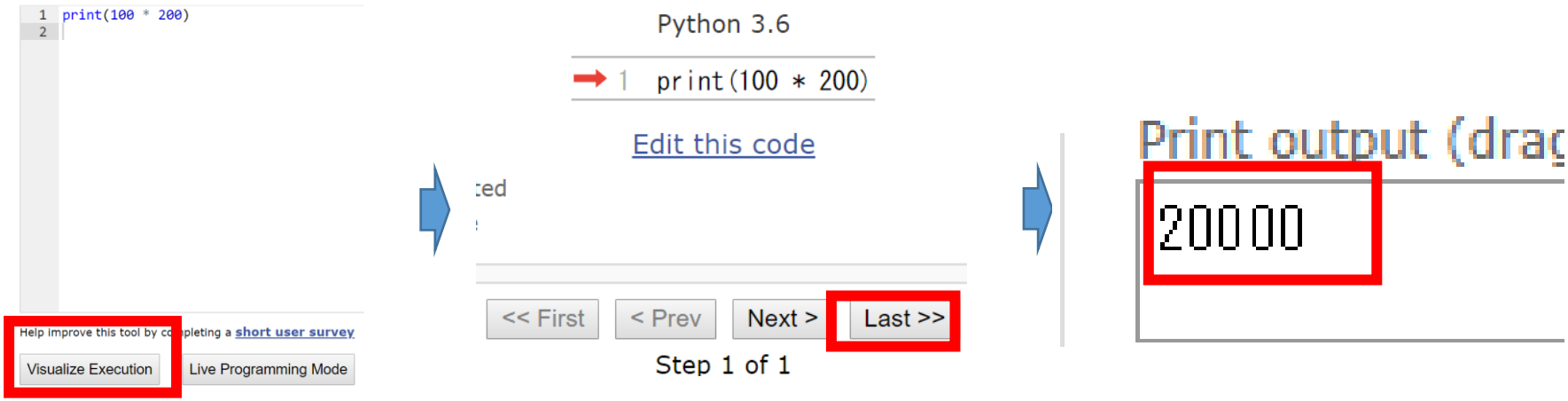
print(100 * 200)



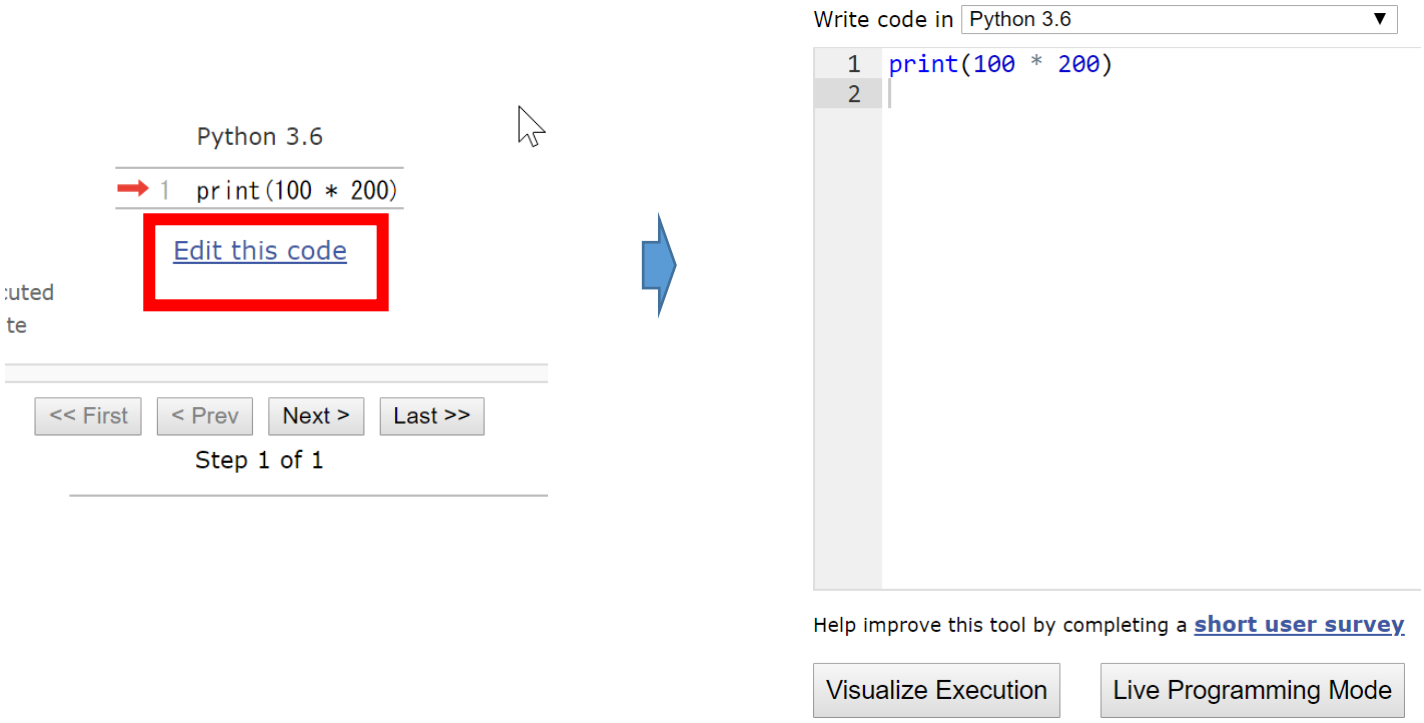
すべて**半角文字**

「*」は掛け算の記号

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

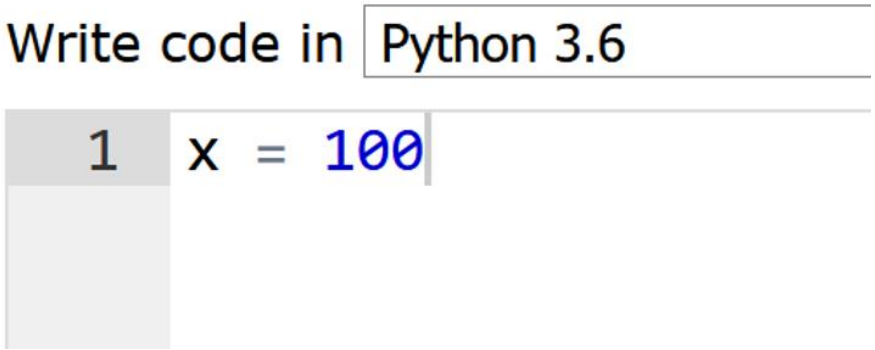


③ 「**Edit this code**」をクリックして，エディタの画面に戻る



④ Python Tutor のエディタで次のプログラムを入れる

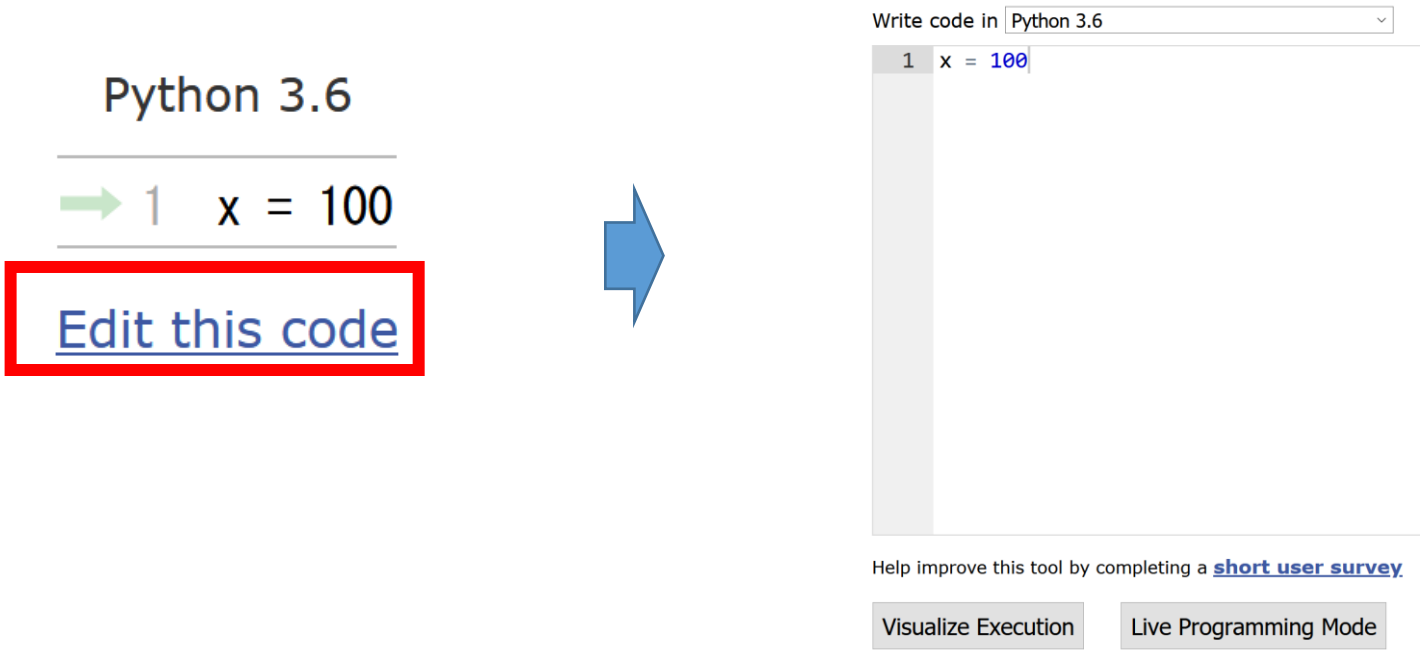
x = 100



すべて**半角文字**

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

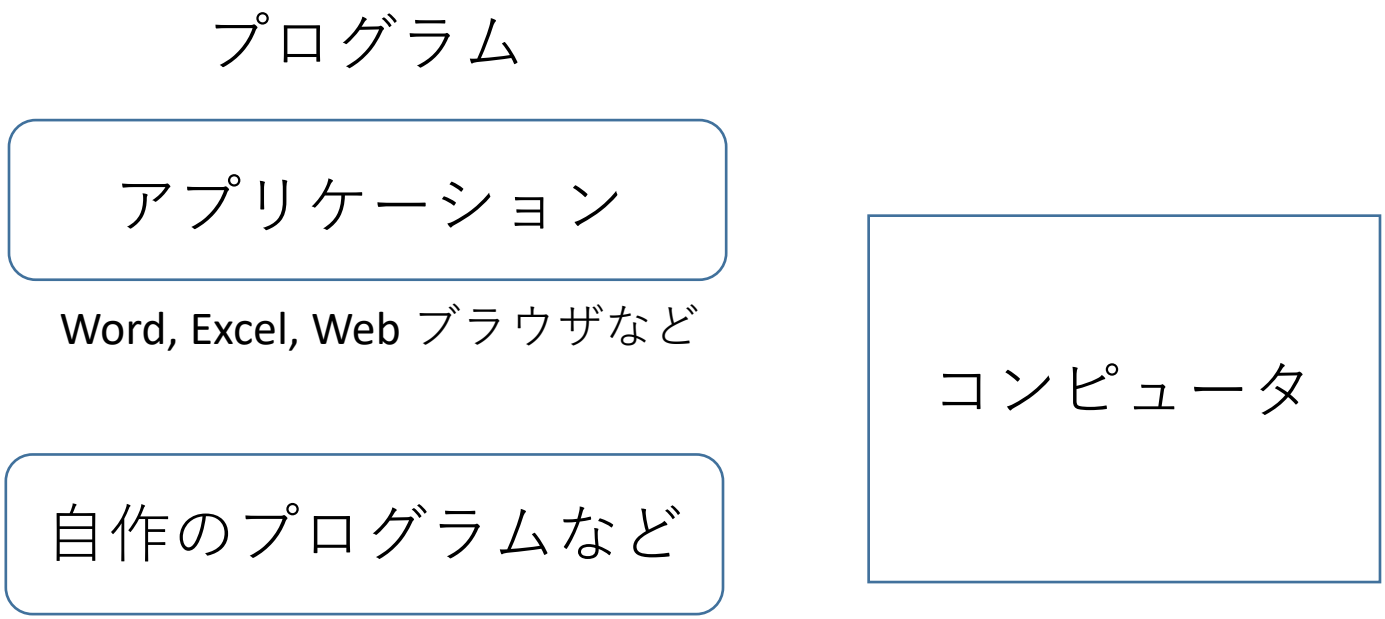
⑥「**Edit this code**」をクリックして，エディタの画面に戻る



1-3. プログラムによる問題解決

プログラムは何の役に立つか

- コンピュータを使い，さまざまな問題を解くことができる
- 人間は，コンピュータを使いこなす（コンピュータが人間の能力を増幅する）．
- コンピュータへの指令を行うのがプログラム.



演習

資料：**38 ～ 43**

【トピックス】

・プログラムでできること

プログラムで出来ることの例

計算問題

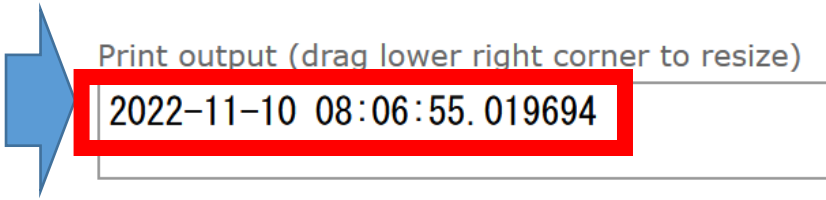
- 現在の日時
- 最大公約数
- 平方根
- 円周率
- 三角関数

その他，データ処理，データ送受信，AI，グラフィックスなど，コンピュータによる情報処理や情報通信

① Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

オペレーティングシステム（コンピュータ）のタイマーを利用．**いまの日時が表示される**

```
import datetime
now = datetime.datetime.now()
print(now)
```



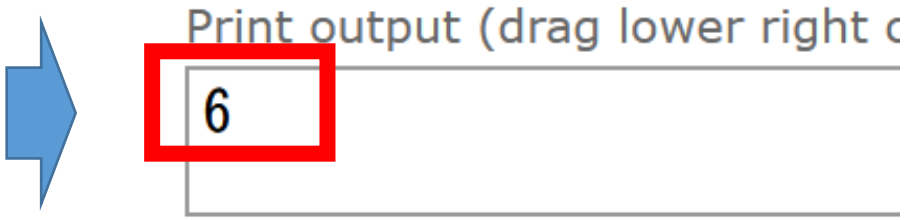
結果を確認

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

② Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

24 と **18** の**最大公約数**を求めたい

```
import math
print( math.gcd(24, 18) )
```



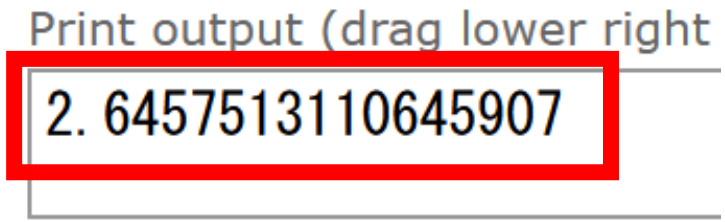
結果の
「**6**」を確認

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

③ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

面積が **7** の正方形の一辺の長さは？

```
import math
print( math.sqrt(7) )
```



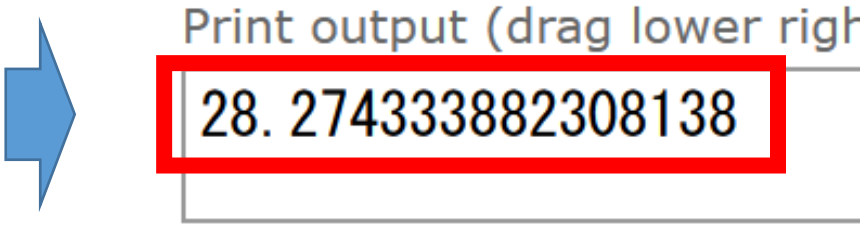
結果の
「**2.6457513110645907**」
を確認（結果は近似値）

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

④ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

半径 **3** の円の面積は？

```
import math
print( 3 * 3 * math.pi )
```



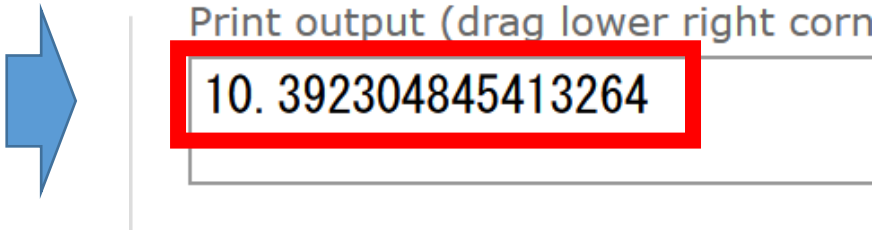
結果を確認

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

⑤ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

三角形の２辺の長さが，**4** と **6** で，その間の角度が **60** 度のとき，面積は (1/2) × **4** × **6** × sin(**60**)

```
import math
print( (1/2) * 4 * 6 * math.sin(60 * math.pi / 180) )
```



結果を確認

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

1-4. 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：**47 ～ 51**

【トピックス】

- **計算誤差**

① Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  print(1/3)
```

Print output (drag lower

0. 3333333333333333

結果を確認

0.0000000000000003333333333333333333333333333333333...
が誤差！

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

② Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  print(1/6)
```

Print output (drag lower righ

```
0.16666666666666666
```

結果を確認

計算誤差がある

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

③ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  print(1/9)
```

Print output (drag lower ri

```
0.1111111111111111
```

結果を確認

計算誤差がある

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

④ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

print(0.333333333333333 * 3)
※ 「3」は15個. （すべて半角文字）

```
1  print(0.333333333333333 * 3)
```

Print output (drag lower r

0. 9999999999999989

結果を確認

計算誤差がある

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認.
「**Edit this code**」をクリックすると，エディタの画面に戻る

⑤ Python Tutor のエディタで次のプログラムを入れ，実行し，結果を確認する

```
1  print(100 * 1.1)
2  print(150 * 1.1)
3  print(400 * 1.1)
```

Print output (drag lower righ

```
110.00000000000001
165.0
440.00000000000006
```

結果を確認

計算誤差がある場合と無い場合がある

「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

- 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

1-5. 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

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

データ専門家向け

```
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)
```

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)))
```

まとめ

- **プログラミング言語**には**さまざまな種類**がある

- 「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」という考え方もある

1-6. ９回の全体計画

9回の全体計画

1	プログラミング入門	プログラミング，Python Tutor での Python プログラム実行，プログラムによる問題解決，計算誤差，さまざまなプログラミング言語，９回の全体計画
2	Python プログラミングの基本	オブジェクトとメソッド，引数，代入，データの種類，制御，コードコンバット（Code Combat）の紹介
3	式の抽象化と関数	式，変数，式の抽象化と関数，関数定義，def，関数呼び出し
4	条件分岐，ステップ実行	条件分岐，if，else，ステップ実行
5	リスト，辞書	リスト，辞書
6	繰り返し（ループ），ステップ実行	繰り返し（ループ），for，in，ステップ実行
7	モジュール，標準ライブラリ，算法（アルゴリズム）	モジュール，インポート，import，サブモジュール，パッケージ，標準ライブラリ，算法（アルゴリズム）
8	クラス，メソッド，オブジェクト生成	クラス定義，オブジェクト生成，class，def，__init__，メソッドアクセス，属性アクセス，self
9	クラス階層，継承	クラス階層，継承，super，dir，プログラム開発環境，オンライン開発環境

９回で行うこと

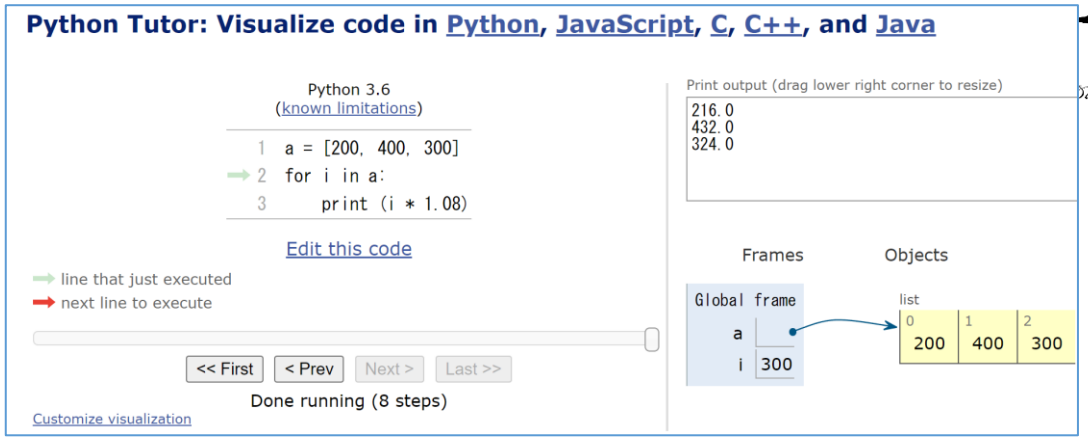
- **９回の資料**で，**Python とプログラミングを学ぶ.**
  - Python プログラムの書き方
  - 抽象化
  - モジュール
  - クラス階層

Python と，
プログラミングの基礎の
両方を学ぶ

- **Python プログラム実行による演習付き**（Python Tutor などを利用）
- 「**プログラミング言語 Python** で学んだ知識，スキルは，**他のプログラミング言語**を使う時にも役に立つ」という考え方も

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```

Python プログラムの
ソースコード



オンラインでの Python プログラム
実行（Python Tutor を使用）

```python
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

```java
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

```c
#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

プログラミング

- コンピュータによりさまざまな問題を解くとき，**プログラミングが役立つ**

  （例）現在の日時，最大公約数，平方根，三角関数など

- 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  print(1/3)
```

```
Print output (drag lower
0.3333333333333333
```

- **さまざまなプログラミング言語がある**．用途や状況に応じて使い分ける
- 「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

Python 関連ページ

- Python まとめページ

https://www.kkaneko.jp/tools/man/python.html

- Python 入門（スライド資料とプログラム例）

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

- Python プログラミングの基本（スライド資料とプログラム例）

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

- Python プログラム例

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

- 人工知能の実行（Google Colaboratory を使用）

https://www.kkaneko.jp/ai/ni/index.html

- 人工知能の実行（Python を使用）（Windows 上）

https://www.kkaneko.jp/ai/deepim/index.html