

# pp-4. プログラミング基本用語など

(Python とプログラミングの基本)

URL: <https://www.kkaneko.jp/pro/colab/index.html>

金子邦彦



# ここで学ぶこと

- プログラミング, プログラム
- ソースコード
- プログラムの実行
- 開発環境
- ライブラリ類
- さまざまなプログラミング言語



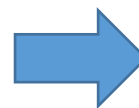
# プログラミング (programming)



- **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)
```

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
- 「ソフトウェアの設計図」ということも。  
人間も読み書き、編集できる
- 複数の**プログラミング言語**を使うことも

```
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 プログラムの実行手順例



- ソースコード

```
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 プログラムのソースコードを，foo.py のようなファイル名で保存しておく

- **プログラム**の起動は，シェル (Windows のコマンドプロンプトなど) から，コマンドで行える

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

# プログラミングで気を付けること



- **コンピュータ**は「万能のマシン」と言われることもある
- **プログラム**で行わせたい「作業」について、深い理解が必要
- **プログラム**中の誤り（**バグ**）を、コンピュータが自動で発見してくれるわけではない。
- 「**プログラム**が期待通りに動いているか」を検証する、**テスト**が必要

# 開発環境とは



**開発環境**とは、**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

# 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，ウェ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ェブサービス (オンラインの開発環境) の例



Google Colaboratory <https://colab.research.google.com/>

- **Python の開発環境**
- 人工知能, データサイエンス, その他多数のパッケージがインストール済み
- **コードセル, テキストセル**を複数ノートブックにまとめ, **保存や公開**できる
- 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**

# プログラム作成ができるウェブサービス (オンラインの開発環境) の例



× ⓘ 保護されていない通信 | pythontutor.com ☆ ABP

## VISUALIZE CODE AND GET LIVE HELP

Learn Python, Java, C, C++, JavaScript, and Ruby

[Python Tutor](#) (created by [Philip Guo](#)) helps people overcome a fundamental barrier to learning programming: understanding what happens as the computer runs each line of code.

Write code in your web browser, see it visualized step by step, and get live help from volunteers.

Related services: [Java Tutor](#), [C Tutor](#), [C++ Tutor](#), [JavaScript Tutor](#), [Ruby Tutor](#)

**Over five million people in more than 180 countries** have used Python Tutor to visualize over 100 million pieces of code, often as a supplement to textbooks, lectures, and online tutorials.

[Visualize your code and get live help now](#)

Python Tutor <http://www.pythontutor.com/>

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

# プログラム作成ができるウェブサービス (オンラインの開発環境) の例



The screenshot shows a web browser window with the URL `tutorialspoint.com/execute_python3_online.php`. The page title is "Execute Python-3 Online (Python v3.6.2)". The interface includes a code editor with the following Python code:

```
1 x = [5, 4, 1, 3, 2]
2 for i in x:
3     print(i * 120)
4
```

The output window shows the result of the execution:

```
$python3 main.py
600
480
120
360
240
```

Coding Ground

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

[m](#)

- Python, C, Java, JavaScript, R, Octave/MATLAB の他にも , SQL, bash, アセンブリ言語など多数の言語
- ファイル作成, ファイル読み書きの実習も簡単にできる
- **複数プログラムファイルの組み合わせも簡単にできる**

# プログラム作成ができるウェブサービス (オンラインの開発環境) の例



- paiza.IO <https://paiza.io/?locale=ja-jp>
  - Python, C, Java, JavaScript, R の他にも SQL など多数の言語
  - **操作が簡単. 表示は日本語.**
  - 一定の条件下でファイル操作も可能

# ライブラリ類



- **ライブラリ**とは

複数の**プログラム**が共有して使えるような機能を持った**プログラム**のこと。

多くの場合、**プログラム**の実行時に**リンク**（結合）される

- **パッケージ**（**モジュール**、**インクルードファイル**などともいう）

複数の**プログラム**が共有して使えるような機能を持った**ソースコード**

※ パッケージの種類、豊富は、プログラミング言語とに違う

# 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

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。



Python

初心者向  
け。その  
おかげで、  
多数の拡  
張機能も。



C / C++

コン  
ピュータ  
の性能を  
最大限引  
き出す。



R

「データ  
処理」に  
特化した  
コマンド  
言語



SQL

「データ  
ベース」  
に特化し  
たコマン  
ド言語



MATLAB /  
Octave

「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



*Database Lab.*

```
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 + 1
```

```
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



```
(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)))
```

関数型言語

# 全体まとめ



- プログラミングは、プログラムを設計，製作すること
- プログラム開発環境とは、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
- プログラミング言語は多数ある

# 関連資料



- **Python まとめページ**

<https://www.kkaneko.jp/cc/python/index.html>

- **Python とプログラミングの基本**

Google Colaboratory を使用.

<https://www.kkaneko.jp/cc/colab/index.html>

- **Python 入門 (全6回)**

Google Colaboratoryを使用.

<https://www.kkaneko.jp/cc/pf/index.html>

- **Python プログラミング演習 (全9回)**

Python Tutor, VisuAlgo を使用

<https://www.kkaneko.jp/cc/po/index.html>

- **さまざまな Windows アプリケーションのインストールと設定**

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