

我々が用いる プログラミング学習環境について

中西渉

watayan@meigaku.ac.jp
名古屋高等学校

2022 年 6 月 24 日

自己紹介

氏名 中西渉

所属 名古屋高等学校 情報科教諭

免許取得 現職教員認定講習会

所属学会 情報処理学会, 日本情報科教育学会

Web サイト <https://watayan.net>

Twitter @watayan

情報教育関係の主なプログラム

PenFlowchart, WaPEN, PyPEN

<https://www.nagoya-gakuin.ed.jp>にあるいろいろ



1 はじめに

新学習指導要領開始

→「情報 I」が始まった学校も

プログラミングが全員に課される

本稿の主旨：

プログラミング実習でどのような環境が必要か

注意：

- 筆者および勤務校の PC 環境は多数派と異なる
 - 私立なので情報教室の管理は自由
- 他校の事情とは違う面が多々ある

2 プログラミング学習環境

出版社	書名	使用言語			
		P	J	V	S
東書	新編情報 I	○	○		
東書	情報 I Step Forward! (理論編) (実習編)	○			
		○	○	○	○
実教	高校情報 I Python	○			
実教	高校情報 I JavaScript		○		
実教	最新情報 I			○	
実教	図説情報 I				○
開隆堂	実践 情報 I			○	
数研	高等学校 情報 I 巻末実習	○	○	○	
		○		○	
数研	情報 I Next	○	○	○	
日文	情報 I 巻末資料	○			
		○	○		
日文	情報 I 図解と実習				○
第一	高等学校 情報 I 巻末資料			○	
		○			

※ P=Python, J=JavaScript, V=VBA, S=Scratch

2.1 教科書で想定されている環境

教科書で扱われている主な言語

- Python
- JavaScript
- VBA
- Scratch

※ VBAとScratchは実習環境が決まっている
→本稿では言及しない

Python の場合

- 教科書では明示されていない
- スクリーンショットは IDLE や Google Colaboratory を見かける

JavaScript の場合

- OS 付属のテキストエディタ + Web ブラウザ
- HTML に埋め込み

雛形となる HTML ファイル

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>タイトル</title>
  </head>
  <body>
    <script>
      <!--ここにコードを書く-->
    </script>
  </body>
</html>
```


2.2 プログラミング学習環境

情報教室 PC のさまざまな問題…

- ソフトウェアのインストールができない
- スペックが低い

→ 生徒に持たせた端末の方が自由？

Excelは低スペック PC でもわりと動く

→ VBAの実習が「現実的」？

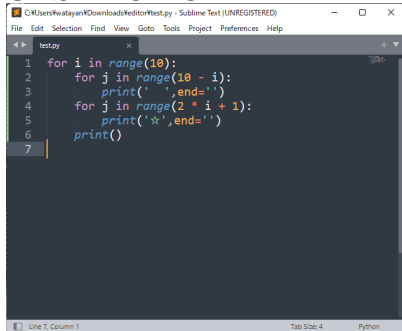
2.2.1 テキストエディタについて

「メモ帳」なんて使いたくない
→だったら何を？

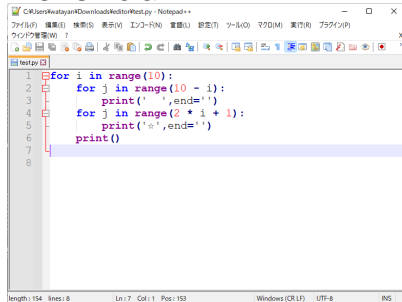
無料のテキストエディタ（ポータブル版）がいろいろある

- Sublime Text
- NotePad++
- サクラエディタ
- Visual Studio Code

Sublime Text



NotePad++



- 簡単な設定で日本語化

- 評価用であれば無料
- 日本語化が面倒

サクラエディタ

```

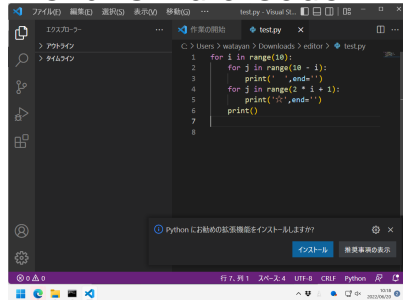
1 for i in range(10):
2     for j in range(10 - i):
3         print(" ",end='')
4     for j in range(2 * i + 1):
5         print('☆',end='')
6     print()
7

```

7行 1列 CRLF CRLF UTF-8 REC 挿入

- 最初から日本語
- Tab キーでスペースでなく Tab 文字が入る

Visual Studio Code



- 日本語化は拡張機能
- 拡張機能の管理ができるか (言われるままにインストール?)
- ポータブル版は data フォルダを作る

テキストエディタの初期状態で確認したいポイント

- 設定変更のしやすさ
- 日本語化のしやすさ
- Tab はスペースか Tab 文字か
- Ctrl+O でどのフォルダが開くか

2.2.2 Pythonの実行環境について

実行環境

- オンライン
 - Google Colaboratory
 - Bit Arrow
 - paiza.IO
 - Wandbox
- ローカルマシン
 - Python のインストールが必要

Google Colaboratory

Colab Notebooks - Google 機械学習.ipynb - Colab

https://colab.research.google.com/drive/1-zITtUuLy0myXsZFu8_7Ky9tCvz_gP9\$scrollTo=88Rr_1-j6y

機械学習.ipynb

ファイル 編集 挿入 ランタイム ツール ヘルプ すべての変更を保存しました

+ コメント 共有 設定

RAM ディスク 編集

+ コード + テキスト

事前準備

手書き数字を読みとる機械学習のプログラムを実行してあります。最初の5行は必要なライブラリを読み込んでいます。

ここでは教師ありの機械学習をするので、手書き数字のデータを読み込まなくてはなりません。これを自前で用意するのは大変なので、scikit-learnの手書き文字データをdigitsを読み込んでいます。

続いて機械学習用のサポートベクターマシンを用意し、それにdigitsの手書き文字データと正解を読み込ませて学習させています。

```
[1] from sklearn.datasets import load_digits
from sklearn.svm import LinearSVC
from PIL import Image
import numpy as np
from IPython.display import display

digits = load_digits()
machine = LinearSVC()
machine.fit(digits.data, digits.target)

/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1280: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  ConvergenceWarning,
  LinearSVC()

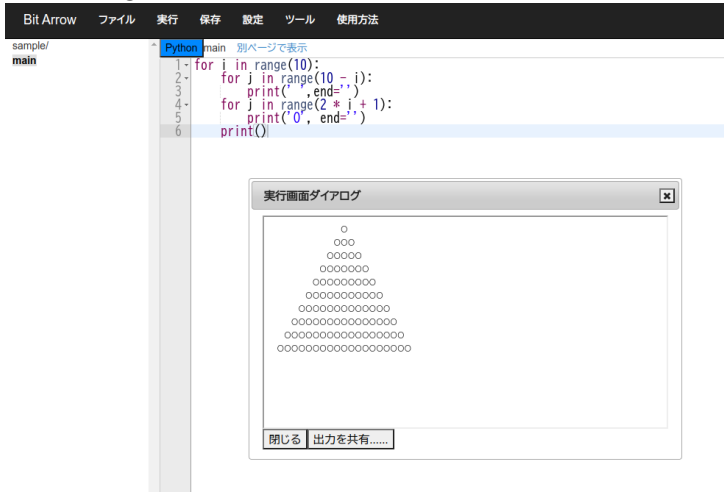
どのようなデータが入っているかを見てみましょう。このプログラムはややこしいので、実行だけしてみてください。手書き文字データと正解の精度100点が表示されます。
```

```
[ ] bg = Image.new('L', (512, 512))
for i in range(10):
    data = np.array(np.split(digits.data[i], 8]) * 16.8
    image = Image.fromarray(data).resize((32, 32), Image.BOX)
    bg.paste(image, ((14*10)*15, (11/10)*15, (16/10)*15+32, (11/10)*15+32))
    print(digits.target[i], end='\n' if i%10==9 else '\t')
```

✓ 1秒 完了時間: 0.50

- テキストとコードが書ける
- マークダウン記法
- セルをまたぐ変数の共有に注意

Bit Arrow



- いくつかのライブラリも使える（サーバ上で実行）
- 他の言語も使える

Wandbox

言語
Python

コンパイラ
CPython 3.10.2

オプション
実行時オプション

企業スポンサー
セオライドテクノロジーズ
株式会社フィックスターズ

個人スポンサー
[Siv3D](#)

```

1 for i in range(10):
2     for j in range(10 - i):
3         print(' ', end='')
4     for j in range(2 * i + 1):
5         print("O", end='')
6     print()

```

\$ python3 prog.py

実行

```

      O
     OO
    OOO
   OOOO
  OOOOO
 OOOOOO
OOOOOOO
OOOOOOOO
OOOOOOOOO
OOOOOOOOOO
OOOOOOOOOOO
OOOOOOOOOOOO
OOOOOOOOOOOOO
OOOOOOOOOOOOOO
OOOOOOOOOOOOOOO
OOOOOOOOOOOOOOO

```

終了コード: 0

- 入力は事前に入れておく
(インタラクティブにはできない)
- 他の言語も使える (さまざまなバージョン)

ローカルマシンでの実行

- ① コマンドプロンプト（または PowerShell）で手書き
- ② IDLE で手書き
- ③ テキストエディタで書いてコマンドプロンプトで実行
- ④ テキストエディタで書いて IDLE で実行
- ⑤ IDLE のエディタで書いて IDLE で実行
- ⑥ テキストエディタ上で実行
- ⑦ IDE 上で実行
- ⑧ Jupyter Notebook などの上で実行

(1) コマンドプロンプトで手書き

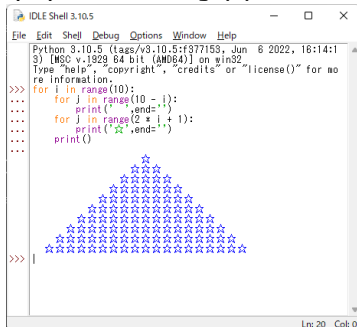


```

コマンドプロンプト - python
(c) Microsoft Corporation. All rights reserved.

C:\Users\watayan>python
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>> for i in range(10):
...     for j in range(10 - i):
...         print(' ',end='')
...     for j in range(2 * i + 1):
...         print('☆',end='')
...     print()
...
  
```

(2) IDLE で手書き

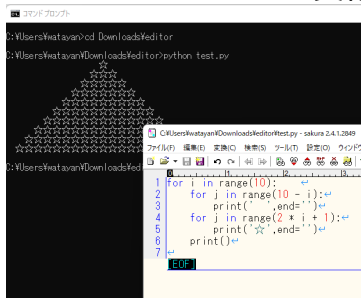


```

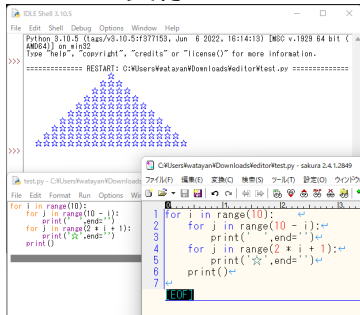
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>> for i in range(10):
...     for j in range(10 - i):
...         print(' ',end='')
...     for j in range(2 * i + 1):
...         print('☆',end='')
...     print()
...
  
```

- やりたくない…
- コピペしようとしてハマることが予想される

(3) テキストエディタで書いて コマンドプロンプトで実行



(4) テキストエディタで書いて IDLE で実行

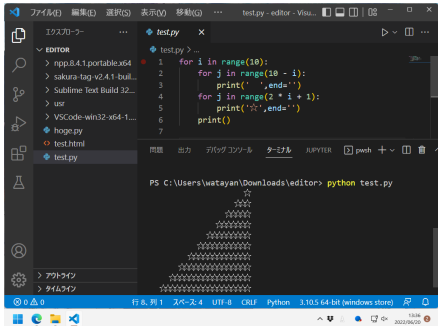


- ファイルを置いたフォルダで迷いそう

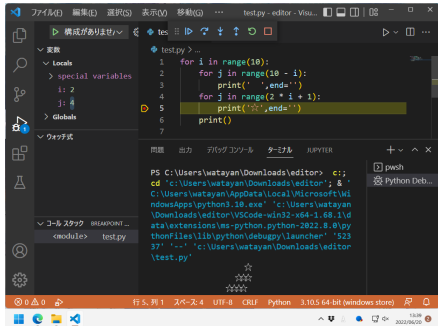
[illegible]

- 2画面の使い分けが正しくできるか

(6) テキストエディタ上で実行 (Visual Studio Code)
(ターミナルで実行) (デバッグ実行)

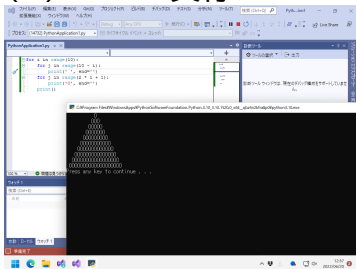


- 該当のフォルダを開かないとファイル操作がややこしい



- 多機能な分，操作が複雑

(7) IDE で実行

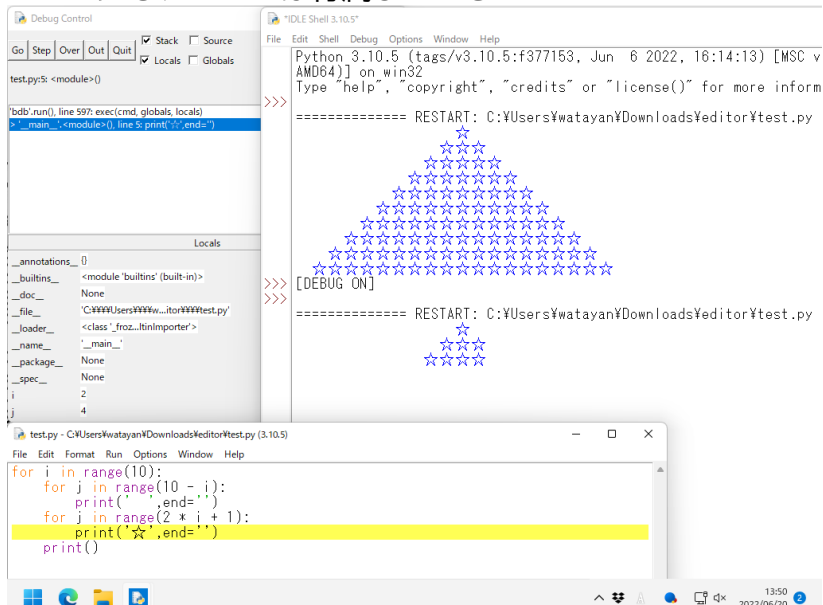


(8) Jupyter Notebook などの上で実行



- こんな環境が用意できるなら誰も困らないのでは？

IDLE にもデバッガは付属しているが…



The screenshot shows the IDLE 3.10.5 environment. The left pane displays the `test.py` file with the following code:

```
for i in range(10):
    for j in range(10 - i):
        print(' ',end='')
    for j in range(2 * i + 1):
        print('☆',end='')
    print()
```

The right pane shows the IDLE Shell with the following output:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v
AMD64]] on win32
Type "help", "copyright", "credits" or "license()" for more inform
>>>
===== RESTART: C:\Users\watayan\Downloads\editor\test.py
          ☆
        ☆☆☆
      ☆☆☆☆☆
    ☆☆☆☆☆☆☆
  ☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
[DEBUG ON]
===== RESTART: C:\Users\watayan\Downloads\editor\test.py
          ☆
        ☆☆☆
      ☆☆☆☆☆
    ☆☆☆☆☆☆☆
  ☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆
```

The bottom pane shows the `test.py` file in the C:\Users\watayan\Downloads\editor\test.py (3.10.5) window, displaying the same code as the left pane.

2.2.3 JavaScriptの実行環境について

教科書はWebブラウザを想定

Node.jsは？

```
ファイル(F) 編集(E) 表示(V) ブックマーク(B) 設定(S) ヘルプ(H)
watayan@wata-x280:~$ node
Welcome to Node.js v12.22.5.
Type ".help" for more information.
> for(var i = 0; i < 10; i++)
... {
...   console.log(i);
... }
0
1
2
3
4
5
6
7
8
9
undefined
> 
```

- prompt, alert, document.write が使えない
- 標準入力の処理が難しい
出力は console.log

JavaScript を…

HTML 内に書くか

```
<script>  
    document.write("Hello, World.");  
</script>
```

- 1 ファイルで完結する
- いちいち前後の HTML を書くのは面倒

別ファイルに書くか

```
<script src="hello.js" type="text/javascript">  
</script>
```

- ファイル名だけ書き換えれば済む
- 複数ファイルを管理する手間

複数ウィンドウ

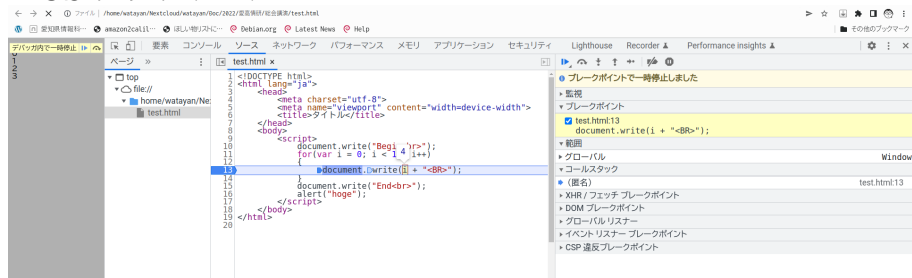
複数ファイル

→種々のトラブルが予想される

- ファイル間違い・フォルダ間違い
- ファイルの保存忘れ
- リロード忘れ

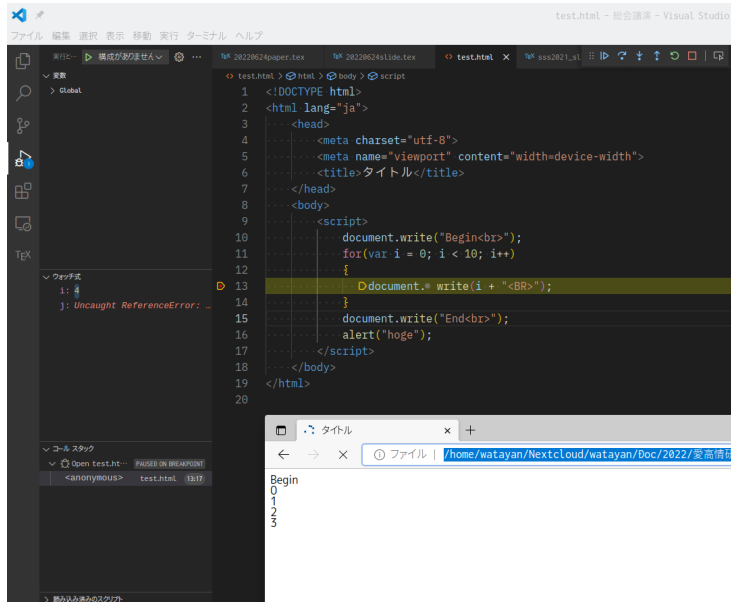
急募：これらの防止策

Web ブラウザのデベロッパーツール



- ブレークポイントと変数のウォッチができれば…
- キー操作は覚えないとつらい

テキストエディタでデバッグ (Visual Studio Code)



3 おわりに

大学入学共通テスト への導入

- 試作問題（検討イメージ）
- サンプル問題

プログラミング問題では DNCL（日本語ベースの擬似言語）

旧 DNCL

```
n を 1 から 10 まで 1 ずつ増やしながら、
| もし  $n \% 3 \neq 0$  ならば
| | n を表示する
| を実行し、そうでなければ
| | 「( ´ · ω · ` )」を表示する
| を実行する
を繰り返す
```

新 DNCL

```
n を 1 から 10 まで 1 ずつ増やしながら：
| もし  $n \% 3 \neq 0$  ならば：
|   ↳ 表示する (n)
|   そうでなければ：
|   ↳ ↳ 表示する ("( ´ · ω · ` )")
```

（試作問題・サンプル問題）

（情報関係基礎）

DNCLの実行環境は多々ある（旧・新とも）私自身も作っている
→ DNCLで実習をすれば一石二鳥？

それはすべきでない！！

- 問題解決の道具には不足
- 教科書掲載のプログラム言語で入試は対応可能
- この程度の「読み替え」ができないようでは意味がない