# PyPENの大学入学共通テストへの対応状況

中西渉

watayan@meigaku.ac.jp
名古屋高等学校

情報処理学会第 87 回全国大会

# はじめに

2025 年度大学入学共通テスト

- 「情報」が追加
- プログラミング問題の擬似言語

→ プログラミング学習環境「PyPEN」を開発

PyPEN は模擬試験・共通テスト にどれだけ対応できるか

# 共通テスト用プログラム表記

## 教科書のプログラミング言語

- Python
- JavaScript
- VBA
- Scratch

## 共通テスト用プログラム表記

- 日本語ベース
- Python 風
- （以下では DNCL2）

# DNCL2のプログラム例

```
age を 1 から 60 まで 1 ずつ増やしながら繰り返す：
    もし age <= 17 ならば：
        表示する(age, ”歳です。”)
    そうでなければ：
        表示する(17, ”歳です。”)
        表示する(”おいおい”)
```

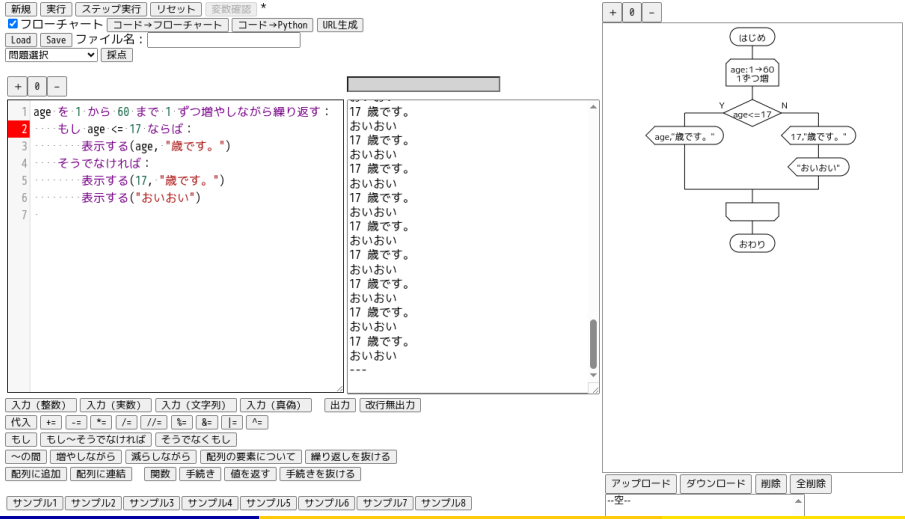
# Pythonだとこうなる？

```
for age in range(1, 61):
    if age <= 17:
        print(age, ”歳です。”)
    else:
        print(17, ”歳です。”)
        print(”おいおい”)
```

# PyPENについて

- ブラウザ上のプログラミング学習環境
- ローカルでも使用可能
- 言語はほぼDNCL2
- MITライセンスで公開
- https://github.com/watayan/PyPEN.git

# PyPENの実行画面


新規
実行
ステップ実行
リセット
変数確認
フローチャート
コード→フローチャート
コード→Python
URL生成
Load
Save
ファイル名：
問題選択
採点
+
0
-
1 age を 1 から 60 まで 1 ずつ増やしながら繰り返す：
2 ····もし age <= 17 ならば：
3 ····表示する(age, "歳です。")
4 ····そうでなければ：
5 ····表示する(17, "歳です。")
6 ····表示する("おいおい")
7
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
17 歳です。
おいおい
---
入力（整数）
入力（実数）
入力（文字列）
入力（真偽）
出力
改行無出力
代入
+=
-=
*=
/=
//=
%=
&=
|=
^=
もし
もし～そうでなければ
そうでなくもし
～の間
増やしながら
減らしながら
配列の要素について
繰り返しを抜ける
配列に追加
配列に連結
関数
手続き
値を返す
手続きを抜ける
サンプル1
サンプル2
サンプル3
サンプル4
サンプル5
サンプル6
サンプル7
サンプル8
+
0
-
はじめ
age:1→60
1ずつ増
age<=17
Y
N
age,"歳です。"
17,"歳です。"
"おいおい"
おわり
アップロード
ダウンロード
削除
全削除
--空--

# PyPEN の機能

## Python コードを出力

新規 実行 ステップ実行 リセット 変数確認 *

フローチャート コード→フローチャート コード→Python URL生成

Load Save ファイル名：

問題選択 採点

\+ 0 −

```
1 age を 1 から 60 まで 1 ずつ増やしながら繰り返す：
2 ····もし age <= 17 ならば：
3 ········表示する(age, "歳です。")
4 ····そうでなければ：
5 ········表示する(17, "歳です。")
6 ········表示する("おいおい")
7
```

```
for age in range(1, 60+1, 1):
    if age <= 17:
        print(age, '歳です。')
    else:
        print(17, '歳です。')
        print('おいおい')
```

## コード入りの URL を生成

新規 実行 ステップ実行 リセット 変数確認 *

フローチャート コード→フローチャート コード→Python URL生成

Load Save ファイル名：

問題選択 採点

\+ 0 −

```
1 age を 1 から 60 まで 1 ずつ増やしながら繰り返す：
2 ····もし age <= 17 ならば：
3 ········表示する(age, "歳です。")
4 ····そうでなければ：
5 ········表示する(17, "歳です。")
6 ········表示する("おいおい")
7
```

```
https://watayan.net/prog/PyPEN/?code=eJx9jzEKAwkAQRXtPNWzlgoXbmEYPYyFowxlEMEmjCMGAhRJQAkLASbE9zEeTzitkNmK1OMNwfN68zw7HI4IsyRE4gsyp19XXDbxvohScPXYJZApOwDk4Vqg8F5Cwuq_A69c1bZEORBQh7-sPyAV6nCsKLj6En2p7KLOLnkGitrIdMs_jSdt8NBFj3y7egGdNqo0LSPxP44LfIi_QgEOw_sRvY2vFln_o
```

# 情報教育関係者（作者を含む）の意見

- PyPEN で授業をするな！
  「情報」のテーマは問題解決
  問題解決のためにはもっと自由な言語・環境
- 演習に使うのはアリ…かな？
  参考書・解説サイト 等で使われている

# 模擬試験のプログラム問題

問題からプログラムだけを抜粋

- 2023 年度進研高 2 模試
- 2023 年度全統共通テスト 高 2 模試
- 2024 年度大学入学共通テスト 進研模試
- 2024 年度駿台 atama+ 共通テスト 模試
- 2024 年度ベネッセ・駿台マーク模試

# 2023年度進研高2 模試

```
Kurasumei = [”1A”, ”1B”, ”1C”, ”1D”, ”2A”, ”2B”, ”2C”, ”2D”, ”3A”, ”3B”, ”
Kekka = [””, ””, ””, ””, ””, ””, ””, ””, ””, ””, ””, ””]
Tensu = [42, 24, 13, 27, 11, 49, 65, 67, 54, 60, 65, 3]
kurasu_num = 12
ichiban = 1
i = 0
n = -1
i を 0 から kurasu_num まで 1 ずつ増やしながら繰り返す：
    もし ichiban < Tensu[i] ならば：
        ichiban = Tensu[i]
        n = i
Kekka[n] = ”最優秀賞”
表示する(Kekka[n], ”は”, Kurasumei[n], ”です。”)
```

# 2023年度進研高2 模試（続き）

```
Kurasumei = ["1A", "1B", "1C", "1D", "2A", "2B", "2C", "2D", "3A", "3B", "
Kekka = ["敢闘賞", "敢闘賞", "敢闘賞", "敢闘賞", "敢闘賞", …, "敢闘賞"]
(略)
i を 0 から kurasu_num まで 1 ずつ増やしながら繰り返す：
    もし ichiban < Tensu[i] ならば：
        ichiban = Tensu[i]
        n = i
        Kekka[n] = "最優秀賞"
x を 0 から kurasu_num まで 1 ずつ増やしながら繰り返す：
        表示する(Kurasumei[x], "は", Kekka[x], "です。")
```

- 改良しようとしてエンバグしてる！

# 2023年度全統共通テスト高２模試

```
関数 ランダム整数():
        整数(7 * 乱数()) + 1 を返す
Seito = ["アカリ", "ダイキ", "エミ", "ヒロキ", "ナナ", "ミサキ", "シュン"]
Seki = ["空席", "空席", "空席", "空席", "空席", "空席", "空席"]
i を 1 から 7 まで 1 ずつ増やしながら繰り返す:
    number = ランダム整数()
    Seki[i] = Seito[number]
```

- 配列は１オリジン
- 自作関数は中身だけを問う（定義方法は不明）
- どういう不具合かを考えさせる

# 2023年度全統共通テスト高２模試（続き）

```
（略）
i を 1 から 7 まで 1 ずつ増やしながら繰り返す：
    number = ランダム整数()
    Seito[number] == ”席決定” の間繰り返す：
        number = ランダム整数()
    Seki[i] = Seito[number]
    Seito[number] = ”席決定”
```

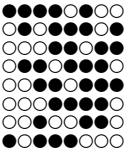
- 不具合を直してみた
- 実行時間にバラツキが！

# 2023年度全統共通テスト高２模試（続き）

```
関数 ランダム整数改(a):
        整数(a * 乱数()) + 1 を返す
mitei = 7
i を 1 から 7 まで 1 ずつ増やしながら繰り返す:
    number = ランダム整数改(mitei)
    count = 0, j = 0
    count < number の間繰り返す:
        j = j + 1
        もし  Seito[j] != ”席決定” ならば:
            count = count + 1
    Seki[i] = Seito[j]
    Seito[j] = ”席決定”
    mitei = mitei - 1
```

# 2024年度大学入学共通テスト 進研模試

```
Kekka = [[0, 0, 0, 0, 1, 0, 1, 1], […]]
shiro = 0
kuro = 0
i = 0
tate を 0 から 7 まで 1 ずつ増やしながら繰り返す：
    yoko を 0 から 7 まで 1 ずつ増やしながら繰り返す：
        shiro = shiro + Kekka[tate][yoko]
        i = i + 1
kuro = i - shiro
```



# 2024年度大学入学共通テスト進研模試（続き）

```
もし kuro > shiro ならば：
    表示する（”黒の勝ち”）
そうでなくもし kuro == shiro ならば：
    表示する（”引き分け”）
そうでなければ：
    表示する（”白の勝ち”）
```

- 公開１週間前にバグ発見！
- （どう直したかは秘密）読者への課題とする？

# 2024年度駿台atama+共通テスト模試

```
（komugito, sato, syurui を外部から入力）
もし syurui == 0 ならば：
    kosu = komugiko ÷ 20
    もし kosu > sato ÷ 15 ならば：
        kosu = sato ÷ 15
そうでなければ：
    kosu = komugiko ÷ 30
    もし kosu > sato ÷ 10 ならば：
        kosu = sato ÷ 10
表示する(kosu)
```

- 与えられた小麦粉，砂糖で作れるマフィンとクッキーの個数
- syurui == 0 ならマフィンだけ，そうでなければクッキーだけ

# 2024年度駿台atama+共通テスト模試（続き）

```
関数 最大数(syurui, komugiko, sato):
    (前ページの内容)
m_saidai = 0, k_saidai = 0, u_saidai = 0
m_kosu を0から最大数(0, komugiko, sato) まで 1 ずつ増やしながら繰り返す:
        k_kosu = 最大数(1, komugiko - m_kosu * 20, sato - m_kosu * 15)
        uriage = m_kosu * 150 + k_kosu * 100
        もし uriage > u_saidai ならば:
            m_saidai = m_kosu
            k_saidai = k_kosu
            u_saidai = uriage
表示する(m_saidai, k_saidai, u_saidai)
```

- マフィンの個数を0から最大数まで変えて，売上の最大値を探す

# 余談

前のページのスライドを VS Code で作っていると…

```
\begin{frame}[t,containsverbatim]\frametitle{2024年度駿台atama$+$共通テスト模試（続き）}
    \begin{verbatim}
関数 最大数(syurui, komugiko, sato)：
    （前ページの内容）
m_saidai = 0, k_saidai = 0, u_saidai = 0
m_kosu を0から最大数(0, komugiko, sato) まで 1 ずつ増やしながら繰り返す：
    k_kosu = 最大数(1, komugiko - m_kosu * 20, sato - m_kosu * 15)
    \end{verbatim}
\end{frame}
```

Copilot おそるべし！

# 2024年度ベネッセ・駿台マーク模試

```
Buin = ["伊藤", "加藤", "小林", "佐藤", "鈴木", "高橋", "田中", …, "渡辺"]
sentaku = 3
i を 0 から sentaku - 1 まで 1 ずつ増やしながら繰り返す：
    bango = 整数(乱数() * 10 + 1)
    表示する(i + 1, "人目：", Buin[bango - 1])
```

- 10人の中から3人を選ぶ
- 重複を許すバグ！
- 配列が0オリジンなのにbangoで1を足すのはどうなの？

# 2024年度ベネッセ・駿台マーク模試（続き）

```
Buin = [”伊藤”, ”加藤”, ”小林”, ”佐藤”, ”鈴木”, ”高橋”, ”田中”, …, ”渡辺”]
Tosen = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ninzu = 0
sentaku = 3
ninzu < sentaku の間繰り返す：
    bango = 整数(乱数() * 10 + 1)
    もし Tosen[bango - 1] == 0 ならば：
        Tosen[bango - 1] = 1
        ninzu = ninzu + 1
        表示する(ninzu, ”人目：”, Buin[bango - 1])
```

- 重複を許すバグを修正

# 2024年度ベネッセ・駿台マーク模試（続き）

```
buinsu = 要素数(Buin)
ninzu < sentaku の間繰り返す：
    bango = 整数(乱数() * buinsu + 1)
    もし Tosen[bango - 1] == 0 ならば：
        Tosen[bango - 1] = 1
        ninzu = ninzu + 1
ninzu = 1
i を 0 から buinsu - 1 まで 1 ずつ増やしながら繰り返す：
    もし Tosen[i] == 1 ならば：
        表示する(ninzu, "人目：", Buin[i])
        ninzu = ninzu + 1
```

- いまさら buinsu が変数に？

# 共通テストのプログラム問題

- 本試験
- 追試験

# 共通テスト本試験

```
Akibi = [5, 3, 4]
buinsu = 3
tantou = 1
buin を 2 から buinsu まで 1 ずつ増やしながら繰り返す：
    もし Akibi[buin] < Akibi[tantou] ならば：
        tantou = buin
表示する（”次の工芸品の担当は部員”, tantou, ”です。”）
```

- 一番早く「空き」になる人が次の工芸品の担当
- 配列が１オリジン！

# 共通テスト本試験（続き）

```
Nissu = [4, 1, 3, 1, 3, 4, 2, 4, 3]
kougeihinsu = 9
Akibi = [1, 1, 1]
buinsu = 3
kougeihin を 1 から kougeihinsu まで 1 ずつ増やしながら繰り返す：
    tantou = 1
    buin を 2 から buinsu まで 1 ずつ増やしながら繰り返す：
    もし Akibi[buin] < Akibi[tantou] ならば：
        tantou = buin
    表示する（”工芸品”, kougeihin, ”部員”, tantou, …）
    Akibi[tantou] = Akibi[tantou] + Nissu[kougeihin]
```

- あれ，いつもの「改良」は？

# 共通テスト追試験

- ごみ収集が題材
- 配列が 1 オリジン！
- あとでごみが「可燃・不燃」から 7 種類に

# PyPENでの対応

模試・共通テストの問題を見て感じた課題

1. 独自表記
2. 複数の代入を１行で書くこと
3. 配列の初期化方法
4. 二次元配列の添字
5. 組み込み関数
6. ユーザ定義関数
7. 配列は０オリジンか１オリジンか

# (1) 独自表記

- 「÷」は手書きや印刷ならいいが…
- 「//」で良かったのでは？
- ｜とか └とか必要か？

# (2) 複数の代入を 1 行で書くこと

a = 0, b = 1 のような表記

- DNCL の頃からある
- Python なら a, b = 0, 1
- …必要か？

# (3) 配列の初期化方法

DNCL 配列は長さ不定で初期化できる
「Hairetsu を 0 で初期化する」みたいな

DNCL2 配列は長さ不定で初期化できる（はず）

模試・共通テスト の問題 具体的な要素で初期化

PyPEN Python と同様
「10 個の””」「[””] * 10」などの表記も可能

PyPEN は Python への移植性を残したい
配列用のクラスを作るのはいまいち…

# （4）二次元配列の添字

Data[4, 2]
Data[4][2]

実は DNCL2 は前者
模試で後者が使われている Python は後者

PyPEN は両対応

# (5) 組み込み関数

問題に応じて用意される

- 要素数 (《配列》)…配列の要素数
- 乱数 ()…0 以上 1 未満の乱数

PyPEN は同機能の関数を持つが…関数名が漢字にできない

# （6）ユーザ定義関数

定義の表記方法は不明

# (7) 配列は 0 オリジンか 1 オリジンか

問題によって異なる
共通テスト本番で 1 オリジンが出題

- 1 オリジンにするか？
  ありえない！
- 切り替え可能にするか？
  実装の苦労が報われる気がしない
- Data[0] を空けておけばいいだけ

# まとめ

- 共通テスト の表記は「聖典」になるか？
- 完全な服従追随が求められるのか？
- DNCL2 に「厳密な仕様」は似合わない
  「読めばわかる」がすべて