プログラミング学習環境PyPENは純粋な共通テスト演習環境であるべきか

中西渉[1,a)]

概要：大学入学共通テスト「情報」プログラミング問題で用いられるものに近い言語を使用した学習環境PyPENが，当該テストの演習のために用いられることは自然なことである．一方，これを高校情報科の授業のメイン環境とすることが望ましくないことは筆者を含め複数の情報教育関係者が主張している．本稿では，PyPENは受験対策環境に特化するべきなのか，という点について考察する．

Should the Programming Learning Environment PyPEN Be Dedicated to Common Test Practice?

Abstract: It is natural that the programming learning environment PyPEN, which uses a language similar to that used in the programming problems of the Common Test for University Admissions "Informatics", is used for exercises for the test. On the other hand, several information education stakeholders argue that it is BAD to use this as the main environment for high school informatics classes. This paper considers whether PyPEN should be specialized as an exam preparation environment.

1. はじめに

大学入学共通テスト（以下，共通テスト）では高等学校学習指導要領の改訂に対応して，令和7年度試験（2025年度）から「情報」が出題された．そのプログラミング問題については「授業で多様なプログラミング言語が利用される可能性があることから，受験者が初見でも理解できる大学入試センター独自のプログラム表記を用いる」とされている [1]．実際，事前に公開された試作問題 [2]・サンプル問題 [3] では，Pythonに似た文法をもつ独自言語が用いられた．特に試作問題の「概要『情報』」には「共通テスト用プログラム表記の例示」が掲載されており，これが共通テストで用いられる言語の見本のようなものと考えられてきたし，実際に行われた本試験・追試験でもこの表記に基づいて作成された問題が出題された．

高校「情報I」の教科書で実際に用いられている言語はPython，JavaScript，VBA，Scratchなどである．これらに替えて共通テスト用プログラム表記を授業で用いることは，多くの情報教育関係者が否定的な意見を述べており，筆者もそれに同意する．たとえば大学入試センター試験問題調査官である水野は「授業では教科書で使われている実用的なプログラム言語を使っていただきたい」と述べているし [4]，高野らはたとえ試験用のプログラム表記が厳密に定義された言語になったとしても，その構文規則を学習することは設計者自身も望んでいないだろうと考察している [5]．

一方，共通テストで出題される以上，高校現場でその演習が行われるのは当然である．実際の共通テストはPBTなのであるから，生徒は紙の上だけで答えを出すことを求められている．しかしプログラミングの問題である以上，実際にそのコードで動作を確認したいと考えることも自然である．筆者が開発しているプログラミング学習環境PyPENは共通テストのプログラミング問題で用いられるものに近い言語を使用しており，いくつかの高校などで演習に用いられているほか，参考書などでもこれを利用したサイトを用いているものがある．後述するようにPyPENは共通テストの演習のために開発したものではないのだが，結果的にそれに適したものになっているのは事実である．だとすれば，PyPENは共通テストの演習に特化した環境とすべきなのだろうか．本稿ではこの点について考察する．

[1] 名古屋高等学校
Nagoya Senior High School
[a)] watayan@meigaku.ac.jp

2. PyPEN について

2.1 PyPEN の概要

PyPEN は Web ブラウザ上で動作するプログラミング学習環境であり，Python に似た文法をもつ日本語ベースの独自言語を用いている．コードを入力して「実行」ボタンを押すと，そのコードが解釈実行され，結果が表示される．実行画面例を**図 1** に示す．

図 1 PyPEN の実行画面
Fig. 1 A screenshot of PyPEN

基本的な操作は左上のボタンで行う．実行ボタンはプログラム実行中は中断ボタンに変わり，一時停止してステップ実行や変数確認を行うことができる．また，コードを Python のコードに変換したり，コードを含めた状態で PyPEN の画面が開けるような URL を生成する機能もある．

画面下部にある入力支援ボタンで構文の雛形を入力することができる．また画面右部にあるフローチャートによりコードをフローチャート表示したり，逆にフローチャートを編集することでコードを生成することもできる（ユーザ定義関数を用いたコードには対応しない）．右下にあるのは簡単な File I/O を模した機能で用いるファイル（のようなもの）の管理画面である．

JavaScript で実装されており，サーバの機能を利用しないため，ユーザ自身の Web サーバやローカルのファイルシステムに置いても実行することができる．プログラムは GitHub で MIT ライセンスに基づき公開している [6]．

2.2 PEN から PyPEN に至る経緯

西田らが開発した初学者用プログラミング学習環境 PEN（Programming Environment for Novices）は，大学入試センター試験の科目「情報関係基礎」で用いられている手順記述言語 DNCL をベースにした xDNCL という言語を用いている [7]．筆者は 2006 年に行われた「教育用プログラミング言語に関するワークショップ 2006」[8] に参加して PEN を知り，勤務校の授業で用いてきた．

その後，筆者はフローチャートでプログラムを生成できる機能を PEN に追加した PenFlowchart を開発した [9]．しかし PEN や PenFlowchart は Java アプリケーションであるため，Java 実行環境やインストールが必要であることから使用が敬遠されることがあった．そのため Web アプリケーションでほぼ同等の機能を持つ WaPEN を開発した [10]．

2019 年に実施された第 12 回全国高等学校情報教育研究会全国大会（和歌山大会）の休み時間に「WaPEN の文法を Python っぽい感じにしてみてはどうか」と中野由章氏から提案があり，これを受けて開発したのが PyPEN である [11]．当初は共通テストを意識したものではなかったが，結果的に共通テストのプログラミング問題で用いられるものに近い言語となった．その後も共通テスト用プログラム表記についての情報が得られるたびに，PyPEN の言語仕様をそれに合わせるようにはしてきた．

2.3 PyPEN で用いる「言語」の細部

共通テスト用プログラム表記はあくまでも「表記」であって，仕様が定義された「言語」ではないのだから，細部が問題になるような出題はありえないと考えて良い．たとえば共通テスト用プログラム表記が例示されている [2] には，この表記とフローチャート・Python3・JavaScript・VBA との対比が掲載されているので，これらの言語間で異なる部分については出題されないだろう．

具体例を述べる．共通テスト用プログラム表記の「`i` を `0` から `9` まで `1` ずつ増やしながら繰り返す」に対応するコードは Python では「`for i in range(0, 10, 1):`」であり，JavaScript では「`for (i = 0; i < 10; i++)`」，VBA では「`For i = 0 To 9`」であるが，繰り返しを抜けたあとの `i` の値は Python では 9，JavaScript や VBA では 10 というように異なる．このことから，繰り返しを抜けたあとではループ変数の値を参照してはいけないというルールが徹底されなければならないことがわかる．たとえば線形探索を**図 2** のように書いてしまうと，JavaScript や VBA では正しく動作するが，Python では必ず「見つかった」と表示されてしまう．

```
a = 【外部からの入力】
list = [3, 5, 2, 8, 6]
kosu = 要素数 (list)
i を 0 から kosu - 1 まで 1 ずつ増やしながら繰り返す：
    もし list[i] == a ならば：
        繰り返しを抜ける
もし i < kosu ならば：
    表示する (' 見つかった')
そうでなければ：
    表示する (' 見つからなかった')
```

図 2 ループ変数を繰り返し外で参照する例
Fig. 2 An example of referencing the loop variable outside the loop

PyPEN はプログラムの実行環境でもあるのだから，共通テスト用プログラム表記を模した「言語」を定義して用いる必要がある．共通テストの演習に使われることを考慮すれば，PyPEN には出題されるプログラムができるだけそのまま実行できることが要求されていると考えるべきであろう．しかし逆に言えば，それ以外の部分は独自に定義してもいいし，しなくてはいけないものである．以下では，PyPEN の言語の細部について行った最近の改良点を述べるが，これらは共通テストの演習を行う上では気にする必要がないものと考えている．

2.3.1 参照の値渡し

以前は関数の引数や変数への代入はすべて値渡しで行っていたが，電気通信大学 2025 年度入試問題 [12] で pop，push に相当する操作が使われていたことを受けて，リストや辞書については参照の値渡しをするよう変更した．pop はリストの末尾の要素を削除して，その値を返すものであるから，**図 3** が本来の形である．しかし値渡しのままでは引数のリストを変更できないので，**図 4** のように一時変数を用いるなどして実現する必要がある．これは明らかにコードの可読性を悪くするので，図 3 のような書き方ができるようにした．なお，従来通りの値渡しをするためには copy 関数を用いればいい．

```
list =[1, 2, 3]
a = pop(list) # a = 3, list = [1, 2]
```

図 3 本来求められる pop の例
Fig. 3 An example of pop as intended

```
list = [1, 2, 3]
temp = pop(list) # temp = [3, [1, 2]]
a = temp[0]      # a = 3
list = temp[1]   # list =[1, 2]
```

図 4 値渡しによる pop の例
Fig. 4 An example of pop using pass-by-value

この変更については辰己丈夫氏から「過去に PyPEN で書かれたコードが動作しなくなるのではないか」という指摘があった．確かにリストや辞書を引数に取る関数をユーザ定義している場合には，関数の中でそれらをローカル変数扱いできるものだとして書いたコードの動作が違ってしまうため，PyPEN を用いて独自教材を作っている人にとってはこれが問題になることは十分にありえる．

だからといって過去の資産に縛られて従来の仕様を維持することが望ましいとは考えない．また，今後もこのような仕様変更が必要になることは十分にありえることであるが，それを Git のログで追跡してもらうというのは現実的でない．今回については「必要に応じて copy 関数を用いる」で対応してもらうことになるが，そういったことを利用者にアナウンスするため，Facebook に PyPEN グループを作成した．今後も仕様変更があった場合には，そこで告知していく予定である．

2.3.2 BigInt への対応

PyPEN では数値は整数型と実数型が区別して用いられ，整数型は JavaScript の Safe Integer の範囲（$2^{53}-1$ 程度）に制限していた．これはオーバーフローという概念を知ることができて有用である反面，作れるプログラムを制限することにもなっていた．そこで整数型については内部的に JavaScript の BigInt 型を用いるように変更した．これにより，桁数の大きい整数を扱うことができるようになった．

たとえば従来の PyPEN では階乗の計算は**図 5** のように 18! までしか実行できなかったが，BigInt に対応することによって**図 6** のようにほぼ制限なく実行できるようになった．

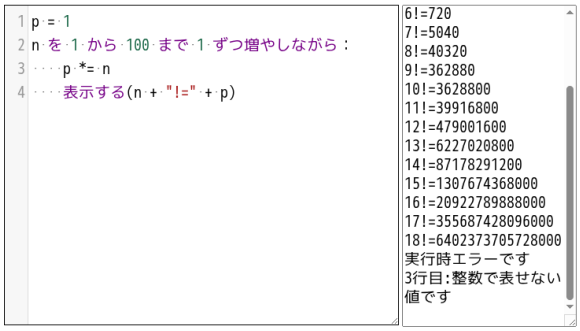

図 5 従来の階乗計算
Fig. 5 Factorial calculation in the previous version

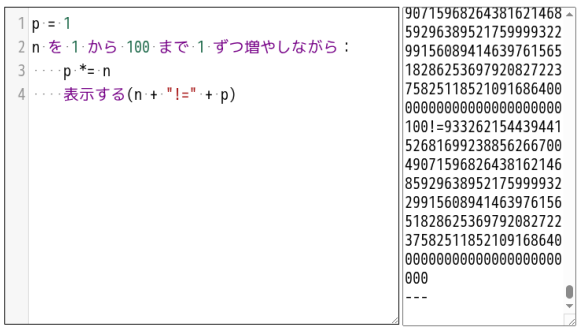

図 6 BigInt 対応後の階乗計算
Fig. 6 Factorial calculation with BigInt support

2.3.3 関数・手続きの一本化

従来の PyPEN では戻り値を返すものを「関数」，返さないものを「手続き」と区別していた．これは構文解析において命令は文になれるが，式が文になれないようにしていたことによる制限であった．たとえば「x に 3 を加える」という処理は「x = x + 3」あるいは「x += 3」と書くものであって，「x + 3」とだけ書いても意味がないことを理解させたいと考えたためである．

しかしこれは別の混乱を招くとも考えられる．たとえば前述した push，pop の例でいえば，push は値を返さないから「手続き」，pop は値を返すから「関数」である．しかし

両者は対になる操作であるから別の扱いをするのは不合理である．そこで式を文とすることを許容し，関数と手続きをすべて関数として扱うように変更した（「手続き」はキーワードとしては残してあるが「関数」と同じ扱いである）．

このことにより副次的に「a = b = 3」のような連鎖代入も可能になった．一方で代入などに使われない式が文として存在できるようになってしまったが（前述の「x + 3」がエラーにならない），これは指導者側が注意すればよいと考えている．

2.3.4 組み込み関数の充実

学習の初期ではリストの要素の合計を求めたり，ソートや探索を自分で実装すること自体が学習の要素となる．そのため従来の PyPEN では組み込み関数を必要最小限のものにとどめていた．しかしその時期をすぎれば，いちいちそのような基本的な関数を手書きすることは非効率であるから，sum や sort のような基本的な関数が用意されるべきだという考え方もある．

そこで追加の関数を別ファイルに実装し，それを組み込むかどうかは設定ファイルで指定できるようにした．追加したのは sum, max のような基本的なものから，gcd, pnorm など数学的なもの，shuffle, sort, next_permutation などリスト操作を行うものなどである．実装が難しいものは [13] などを参考にした．

2.3.5 正規表現

共通テストで正規表現を扱うことはないと考えているが，実用的な用途では正規表現は有用である．そこで JavaScript の正規表現を利用した match 関数を実装した．これは正規表現にマッチした文字列や，その中のグループにマッチした部分文字列をリストで返すものである（マッチしなければ空リストを返す）．

正規表現を用いるときの処理の流れは，まずマッチしたかどうかで分岐し，マッチした場合には取り出した部分文字列について処理を行う，というものになることが多い．そのためにはマッチしたかどうかを真偽値として処理する必要があるので，条件判断を真偽値以外でも行えるようにした．具体的には，次のものを False と扱う．

- 数値の 0
- '0' の連続，'false'（大文字・小文字は問わない），'偽' といった文字列
- 空文字列，空リスト，空辞書

これによって，**図 7** のように match 関数の返り値をそのまま条件判断に用いることができる．

```
もし match('土|日', '月月火水木金金') ならば：
    表示する ('週末まで頑張ろう')
そうでなければ：
    表示する ('終末まで頑張ろう')
```

図 7 match の返り値による分岐
Fig. 7 Conditional execution based on match results

2.3.6 range もどき

Python では回数の決まったループには range が使われることが多いが，これは少し癖のあるクラスである．range(start, stop, step) で数列を生成するものであるが，start から始まった値が stop までではなく，stop の手前までで終わる．一方，共通テスト用プログラム表記では range に相当するものはなく，BASIC の FOR〜NEXT に似た「〜から〜まで〜ずつ増やしながら」という構文が用いられる．

Python で range(1, 10, 1) は 1 から 9 までの数列を生成するが，共通テスト用プログラム表記で「1 から 10 まで 1 ずつ増やしながら」は 1 から 10 までの値を扱う．井手は情報 I の授業でどの言語を用いても共通テスト形式の問題の得点に大きい影響はないとしながらも，Python の range が stop の値を含まないことに起因する（他の言語に見られない）誤答傾向があることを指摘している [14]．

そこで PyPEN にも range 相当の機能を追加した．ただし Python の range オブジェクトのようにシーケンス型としての実装は難しいので，range が生成する数列をリストとして返すものである．共通テスト用プログラム表記では Python の「for i in ...」にあたるような要素をなめる構文がないので，PyPEN では独自に「〜の要素〜について」という構文を用意している．これにより，1 から 10 までの整数を順に表示するコードは**図 8** のように書ける．

```
range(1, 11) の要素 i について：
    表示する (i)
```

図 8 PyPEN での range の使用例
Fig. 8 An example of using range in PyPEN

2.3.7 Python のキーワード導入

図 8 は Python でいう「for i in range(1, 11):」に相当するものであるが，語順が逆なこともあって同じ構文とはとらえにくいと考えられる．そこで PyPEN では Python のキーワードや構文をある程度そのまま用いることができるようにした．たとえば上記のコードは PyPEN 上でも「for i in range(1, 11):」とそのまま書くことができる（**図 9**）．構文解析の際にはどちらのキーワード・構文も受け付けるので，混ぜ書きさえできてしまう．

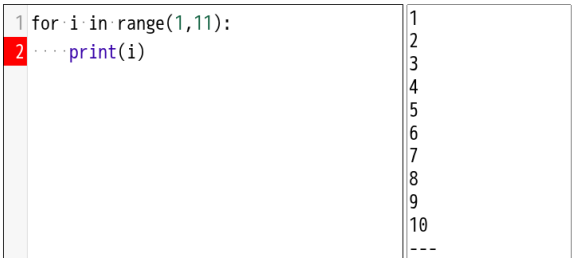

図 9 PyPEN 上での Python コード実行例
Fig. 9 An example of executing Python code on PyPEN

このことにより，簡単な Python のコードなら PyPEN 上で動作させることが可能になった．PyPEN から Python に移行することのハードルが下がることを期待する．

3. 考察

3.1 試験対策教材としての PyPEN

共通テストで Python などの実用的な言語を用いないのは，生徒が学習してきた環境による差をなくすためと考えられる．このように試験のために作られた言語として有名なものには，情報処理技術者試験のアセンブリ言語 CASL とそのための仮想計算機 COMET があり，これに対応するシミュレータが多数開発されてきた．おそらくそれらの中には試験対策のために開発されたものもあるとは思われるが，低級言語や CPU，アーキテクチャなどの学習を目的とする事例も多く見受けられる．

PyPEN を作るきっかけの大元である PEN は前述したように，大学入試センター試験（当時）の「情報関係基礎」のプログラミング問題のために作られた手順記述言語 DNCL そのものでなく，それを元にした xDNCL と言う言語を定義して用いている．[7] によれば，PEN を開発するにあたって DNCL に注目したのは日本語ベースの記述言語であることや，説明なしで入試に用いられるわかりやすさなどを評価したためであったのだし，DNCL をそのまま使うのでなく変数の型定義や命令・構文・組み込み関数を拡充したのも学習者にとってよりよい学習環境とするためであったと述べられている．そのことからも，PEN は試験対策だけを重視して作られたものではないと考えられる．実際，筆者が勤務校で PEN を用いたのも，初学者にとってハードルの低い学習環境を求めたのであって，決して試験対策のためではなかった．

翻って PyPEN はどうであるか．共通テストの対策環境として用いられることは自然なことであるが，PyPEN 自体は共通テストの演習のために開発されたものではない．むしろ，Python に似た文法をもつ日本語ベースの独自言語を用いることで初学者にとってハードルの低い学習環境を提供することを目的として開発したものである．その結果として共通テストのプログラミング問題で用いられるものに近い言語となったにすぎない．とはいえ，PyPEN が共通テストの演習環境として「ちょうどいい」ものとして使われる以上，教材として使えることを最低限の要件として満たしておく必要はある．

ところで，最近の IT パスポート試験や基本・応用情報技術者試験のプログラミング問題ではそのための擬似言語を用意している．2025 年に行われた技術書同人誌即売会[*1]の会場でこれらの試験のための指導をしているという人たちと話す機会があり，PyPEN のキーワードや構文解析部分を情報技術者試験の言語用に書き直せば，これらの試験のための学習環境として使えるのではないかと語っていた．そのことからも，プログラミング言語（に類するもの）があればその実行環境の需要はあるものなのだと考えることができる．

[*1] 技術書同人誌博覧会 12(10/26)，技術書典 19(11/16)．

3.2 日本語ベースのプログラミング環境としての PyPEN

多くのプログラミング言語はキーワードとして英語由来のものを用いている．単純な単語が多いにも関わらず，これが初学者にとってのハードルになるという指摘は以前からなされてきた．前述した CPU シミュレータの中にはニーモニックを漢字にしたものもあるが [15]，日本語表記によって初学者にとってのハードルを下げるという発想は共通のものである．PyPEN を大学の授業で用いている例もあるが，それは導入が用意であることが理由の一つである．

DNCL や共通テスト用プログラム表記は本来プログラム言語というよりも，アルゴリズム記述のための疑似コードに近いものである．逆にいえば，アルゴリズムそのものの伝達にはコードよりも適しているといえる．筆者は PyPEN からフローチャートを取り去りたいと考えているが，それはコード自体がフローチャートと同じだけの表現を持っていると考えるからである．だとすると日本語であることはむしろ優位性となるだろう．

しかし日本語でのコード記述は漢字変換などのブレを許容しなくてはいけないなど，扱いが難しい面もある．PyPEN でもある程度は漢字の開き方や送り仮名の違いを許容しているが，あらゆるケースをカバーすることはできない．そのため PEN で採用されていた「入力支援ボタン」を用意している．現代風の UI では入力補完が一般的だとは思われるが，入力支援ボタンを並べることで，使える構文やキーワードを一覧できることも利点だと考えられる．

3.3 Web ブラウザ上の Python 実行環境との比較

共通テスト用プログラム表記が Python の表記を真似ていることは疑いがないことであり，PyPEN も Python のキーワードを取り入れることで簡単な Python のコードを実行できるようになったことは既に述べた．PyPEN は共通テスト用プログラム表記に近い言語から始めてこのように改良してきたのだが，Web ブラウザ上で Python のコードを実行できる環境は既に多く存在しているのだから，それらをベースにして共通テスト用プログラム表記が使えるように改造するという，逆向きのアプローチもありえたのではないか．

たとえば Python を JavaScript で実装する Skulpt というプロジェクトがある [16]．ほんの数十行の HTML ファイルで**図 10** のような Python 実行環境を作ることができる．おそらく構文解析部分を書き換えれば，共通テスト用プログラム表記のコードも実行できるのではないかと考えてい

るが，まだ試みてはいない．PyPEN は無限ループが止められるようにするなどの処理のため実行速度が遅いので，このような高速な環境をベースにできれば，より快適な学習環境が提供できるかもしれない．

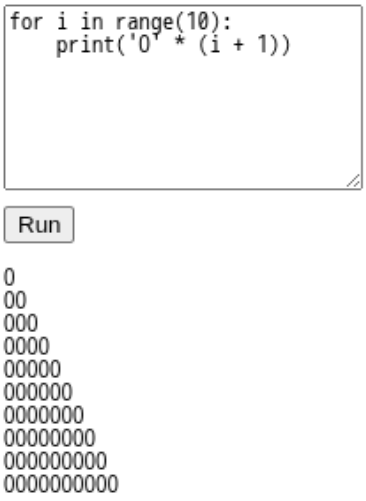

図 10 Skulpt を用いた Python 実行環境の例
Fig. 10 An example of a Python execution environment using Skulpt

4. おわりに

共通テスト用プログラム表記は試験のためのものであるから，独自の「仕様」を持つことが許される．しかしそれが他の言語における「常識」と衝突する場合には，慣れた者ほど違和感や混乱を覚えることがある．たとえば DNCL には「論理演算は左から結合する」という規則が明記されていた．これにしたがえば「`false` かつ `false` でない」は「(`false` かつ `false`) でない」なので `true` と解釈されなくてはいけない．しかし多くの言語では `not` の優先順位が高いので，そのような言語に慣れている者は「`false` かつ (`false` でない)」と誤解してしまいがちである[*2]．では DNCL でも「でない」の優先順位を高くすればよかったのかと言えば，そんな単純な話ではない．受験生の多くはプログラミング言語の「常識」は知らないのだから，規則はできるだけ単純にすべきである．そう考えると「左から」だけの簡単な規則に軍配が上がっても仕方がない．

また，その DNCL の規則が生徒が将来本格的なプログラミング言語を使うときに混乱の元になるかというと，必ずしもそうはいえないだろう．なぜなら彼らは DNCL に「熟練」しないからである．プログラマであれば使う言語の演算子の優先順位に熟練する必要はあり，微妙な判断もできる必要がある．しかし受験生は DNCL を使うのは共通テストのときだけであり，その後に DNCL を使うことはない．だからこそ規則の単純さが重んじられるべきでもあるといえる．

では PyPEN はどちら側であるべきなのか．願わくば，PyPEN が共通テスト用プログラム表記の学習環境としてだけでなく，初学者にとってハードルの低い日本語ベースのプログラミング学習環境としても広く用いられることを期待したい．そのためには共通テスト用プログラム表記の制約に縛られすぎないことも重要であると考えているし，縛るような制約を設けないでいただきたいとも考えている．

[*2] 佐久間拓也氏の指摘による．

付記

本予稿は情報処理学会が研究報告原稿作成用に提供している LaTeX スタイルファイル [17] を LuaLaTeX 対応に改変して組版を行った．エラーなしで処理ができる最低限の改変にとどめたものであるが，GitHub に公開するので修正等いただければ幸いである [18].

参考文献

[1] 大学入試センター：令和 7 年度試験　大学入学共通テスト問題作成方針，(オンライン)，入手先 ⟨https://www.dnc.ac.jp/kyotsu/kako_shiken_jouhou/r7/#monsaku⟩ (参照 2026.01.11).
[2] 大学入試センター：令和 7 年度試験の問題作成の方向性，試作問題等，(オンライン)，入手先 ⟨https://www.dnc.ac.jp/kyotsu/kako_shiken_jouhou/r7/r7_kentoujoukyou/r7mondai.html⟩ (参照 2026.01.11).
[3] 大学入試センター：サンプル問題 (『地理総合』，『歴史総合』，『公共』，『情報』)，(オンライン)，入手先 ⟨https://www.dnc.ac.jp/kyotsu/kako_shiken_jouhou/r7/r7_kentoujoukyou/#anchor02⟩ (参照 2026.01.11).
[4] 水野修治：大学入学共通テスト新科目『情報 I』〜サンプル問題等とそのねらい〜，河合塾 (オンライン)，入手先 ⟨https://www.wakuwaku-catch.net/kouen210801/04/⟩ (参照 2025.12.20).
[5] 高野志歩，田村みゆ，富岡真由，秋信有花，倉光君郎：擬似コードから考える自然言語を活かしたプログラミング言語，情報教育シンポジウム論文集，Vol. 2021, pp. 147–151 (2021).
[6] watayan: PyPEN, (online), available from ⟨https://github.com/watayan/PyPEN⟩ (accessed 2025.12.22).
[7] 西田知博，原田章，中村亮太，宮本友介，松浦敏雄：初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol. 48, No. 8, pp. 2736–2747 (2007).
[8] 西田知博：PEN: 大学入試センター用言語を用いたプログラミング教育，情報処理学会 教育用プログラミング言語に関するワークショップ 2006 報告集，pp. 62–67 (2006).
[9] 中西渉：PenFlowchart の開発，情報処理学会研究報告コンピュータと教育 (CE)，Vol. 2012–CE–113, No. 13, pp. 1–6 (2012).
[10] 中西渉：WaPEN...DNCL の Web ブラウザ上の実行環境におけるフローチャートなどの実装，情報教育シンポジウム論文集，Vol. 2018, No. 31, pp. 210–214 (2018).
[11] 中西渉：プログラミング学習環境 PyPEN の開発，日本情報科教育学会第 13 回研究会研究報告書, pp. 19–22 (2019).
[12] 電気通信大学：過去の入試問題 (2025 年度前期日程入試問題)，(オンライン)，入手先 ⟨https://www.uec.ac.jp/education/undergraduate/admission/exam.html⟩ (参照 2025.12.22).

[13] Milton Abramowitz and Irene A. Stegun: *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Applied Mathematics Series, No. 55, U.S. Government Printing Office, Washington, D.C. (1972). 10th printing with corrections, originally published 1964.
[14] 井手広康：情報 I におけるプログラミング言語の選択が大学入学共通テストの解答に及ぼす影響，情報処理学会論文誌教育とコンピュータ (TCE)，Vol. 9, No. 1, pp. 1–10 (2023).
[15] 金尹悦，安井浩之，吉野邦生：初中等教育向け教育用 CPU シミュレータの開発，情報処理学会第 74 回全国大会講演論文集，No. 1, pp. 807–808 (2012).
[16] Scott Graham and contributors: Skulpt, (online), available from ⟨https://skulpt.org/⟩ (accessed 2026.01.14).
[17] 情報処理学会：研究報告原稿作成について，（オンライン），入手先 ⟨https://www.ipsj.or.jp/kenkyukai/genko.html⟩（参照 2026.01.15）.
[18] watayan: 情報処理学会研究報告用クラスファイル（LuaLaTeX 対応版），（オンライン），入手先 ⟨https://github.com/watayan/ipsj_cls_for_techrep⟩（参照 2026.01.15）.