■ PyPEN

2021 年 8 月 15 日

名古屋高等学校　中西渉 *

はじめる前に

この PDF ファイルは印刷していただいても構いませんが，リンクが埋め込まれていますので，実習に使うパソコン上の PDF ビューワやブラウザなどで開いていただくと便利です。

1　PyPEN について

先月末に，2025 年度からの共通テストに「情報」が導入されることが決定された [1] という報道がありました。それに先立って大学入試センターから試作問題 [2] やサンプル問題 [3] が公開されています。これらに含まれるプログラミングの問題では，DNCL という日本語をベースにした擬似言語が使われています。DNCL はこれまでのセンター試験や共通テストでも使われてきましたが，上述した試作問題等では若干文法が変化しました。特徴的なのは，これまではたとえば「もし〜ならば」が「を実行する」で終わるというようにブロック終端を示すフレーズがありましたが，新しい DNCL（以下では区別のために DNCL2 と呼びます）では Python のように文字下げでブロックを表現します（図 1 参照）。

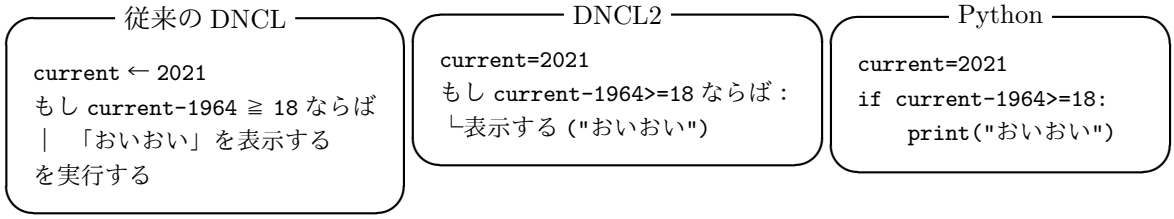

図 1: 言語の違い

これまでの DNCL[4] に対し，それで書かれたプログラムを実行するための環境として PEN[5] およびいくつかの派生プロジェクトがあります。DNCL2 についても，実行環境がいくつか既に作られています。たとえば

- どんくり [6]
- つちのこ [7]
- Picthon（ピクソン）[8]
- PyPEN[9]

* watayan@meigaku.ac.jp

があります（もっとも DNCL2 はまだ言語仕様がはっきりしないので，それぞれ微妙に違っているところがあります）。ここでは手前味噌ではありますが，筆者が開発している PyPEN を使って実習をしてみます。筆者のサイトにある https://watayan.net/prog/PyPEN/にアクセスしてください。ただし，Internet Explorer は使えないので，それ以外の Web ブラウザを使ってください。実行画面を図 2 に示します。左の欄にプログラムを書き込んでいくわけですが，その際に下にあるボタン（入力支援ボタン）を多用します。使い方は第 3 章で説明します。これと右クリックのメニューを使えば，日本語入力が Off のままでプログラムが入力できます。

図 2: PyPEN の実行画面

2 文法説明

2.1 変数，代入，四則演算

変数は値をいれるための名前をつけた箱のようなものです。PyPEN では英字で始まる英数字列で名前をつけます（日本語文字は使えません）。

変数に値を入れることを「代入」といいます。PyPEN では「**《変数》=《値》**」で代入を表します。ここで使われるイコールは数学のイコールとは意味が違っていて，むしろ「←」のようなものだと考えてください。たとえば「`n=n+1`」は数学ではありえない式ですが，PyPEN では「`n` に `n+1` の値を代入する」つまり「`n` ← `n+1`」のような意味になります。

なお，値は整数，実数，文字列，真偽という 4 つの型を持ちます。またこれらを [ ] でまとめて配列にすることもできます（後述）。

四則計算の記号のうち，加算，減算，乗算の記号はそれぞれ`+`，`-`，`*`で，表計算などでも同じなのですぐにわかると思いますが，除算については Python に合わせた記号を使っています。`/`は普

通の割り算で，商は実数になります。整数の割り算で商と余りをだすものについては，商を//，余りを%で表します。

加減より乗除が優先されるのは数学と同じです。優先順位を変えるためにカッコを使うことができますが，使うのは ( ) だけです。数学のように () と {} を使い分けるようなことはしないので気をつけてください。

2.2 入力，出力

入力は「**《変数》に《型》を入力する**」という構文です。型は上述した整数，実数，文字列，真偽のどれかです。

出力は「**表示する (《値》)**」または「**改行無しで表示する (《値》)**」です。値はコンマ区切りで複数並べることができます。

2.3 制御構造

制御構造は図 3 にあげたものがあります。いずれも見れば意味がわかると思います。なお「《条件》」は 2 つの値を==，!=，>，<，>=，<=で比較するというものです（==は等しい，!=は等しくないという意味)。また複数の条件を and や or でつないだり，not を前置して否定することもできます。また特殊な条件として「**《配列》の中に《値》**」というのがあります。

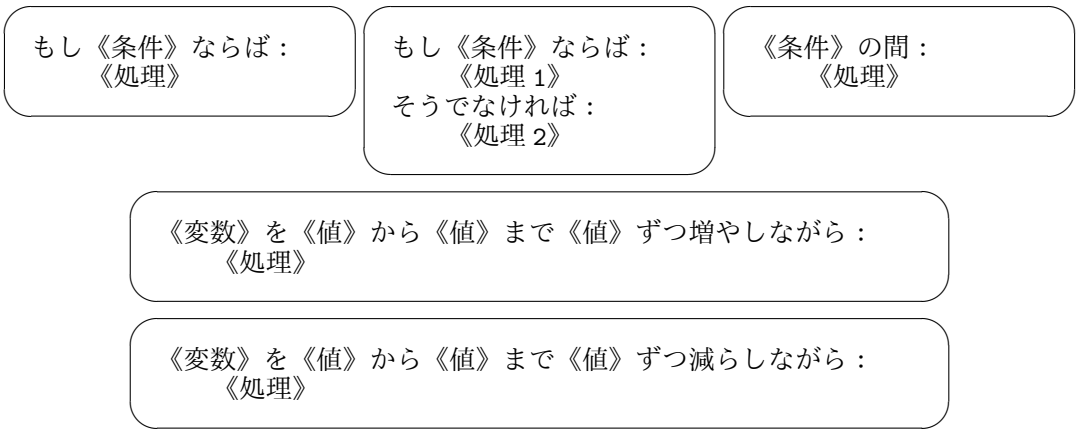

図 3: 制御構造

2.4 関数，手続き

複数の処理をまとめて関数や手続きにすることができます。表記方法は図 4 の通りです。関数は必ず値を返す必要があります。手続きは値を返すことができません。

関数 《関数名》(《仮引数》):
    《処理》
    …
    《値》を返す

手続き 《手続き名》(《仮引数》):
    《処理》
    …
    手続きを抜ける
    …

図 4: 関数・手続き

3 ガチャのプログラム

では実際に一つプログラムを作ってみましょう。最初はガチャのプログラムです。ガチャというのは俗称ですが，複数の景品のどれかがもらえるようなクジを引くものを指しているようです。商店に置いてあるガチャガチャ（カプセルトイ）や，オンラインゲームのアイテム取得などが代表的です。ここでは次のようにルールを決めて，何回クジを引くかをシミュレーションしてみましょう。

- 景品の種類は 10 種類
- すべての景品の当たる確率は等しい
- すべての景品を集めたら終了

3.1 1 回ずつやってみる

ではさっそく作っていきます。プログラムの全体は例 1 の通りです。まず必要なのはクジを引いた回数を記録する変数と，どの景品が取得済みかを記録する変数です。これをそれぞれ n，a とします。

例 1

```
n=0
a=[0,0,0,0,0,0,0,0,0,0]
a!=[1,1,1,1,1,1,1,1,1,1] の間:
    k=random(9)
    a[k]=1
    n=n+1
    表示する (n,a)
```

3.1.1 代入（入力支援ボタンの使い方）

まず n に 0 を代入するわけですが，ここで「入力支援ボタン」の使い方を説明します。「代入」ボタンを押すと「《変数》=《値》」と入力されます。この「《変数》」「《値》」をそれぞれ「n」，「0」にするのですが，「消して書く」のではなく「選択して上書きする」方がやりやすくできています。Ctrl+左右矢印で選択部分やカーソルが移動するので，「《変数》」を指定して「n」を上書き，「《値》」を指定して「0」を上書きします。こうすることで，日本語入力状態にすることなくプログラムが作れます [1)]。

1) 実際のところ「n=0」に限らず代入については入力支援ボタンを使わずに直にキー入力した方が早いので，そのあたりは適宜自分の判断で行ってください。

3.1.2 配列

続いて a ですが，ここには 10 個の景品それぞれについて取ったか取ってないかの記録を残さなくてはいけません。そこで「配列」を使います。これは何個かの値に番号をつけてまとめたものです。ただし番号は 0 から始まることに気をつけてください。a に [0,0,0,0,0,0,0,0,0,0] を代入することで，a[0],a[1],a[2],…,a[9] の値がそれぞれ 0 になります。このプログラムでは k 番目の景品が取得済みかそうでないかを，a[k] の値が 1 か 0 かで表すことにします。だから a[0]〜a[9] は最初すべて 0 にしておいて，k 番目の景品がとれたら a[k] を 1 にする，ということを繰り返します。

3.1.3 繰り返し（ループ）

景品が揃うまでクジ引きを繰り返すので，プログラムも繰り返し（ループ）を使います。入力支援ボタンの「〜の間」を押すと「**《条件》の間：**と入力されるので，条件を具体的に書いていきます。この繰り返しはすべての景品が揃った時点，つまり a の要素がすべて 1 になったら終わるのだから，a が [1,1,1,1,1,1,1,1,1,1] でない間は繰り返しを続けることになります。等しくないことは!=で表しますから，結局「a!=[1,1,1,1,1,1,1,1,1,1] **の間：**」となります。

では，繰り返し行う処理を書いていきます。「**〜の間:**」の行末で Enter を押すと，左側にスペースが 4 つ空いたところにカーソルが移動しますが，このスペースは消してはいけません。繰り返し行う処理がどこからどこまでであるかというのを，このスペースによる文字下げ（「インデント」といいます）で表現するからです。PyPEN ではインデントの増減を Tab と Shift+Tab で行います（スペースキーを何度も押すのでは回数を間違う危険がありますから，このキーを使ってください）。

まず乱数で景品の番号を決めます。PyPEN では random(9) で 0〜9 の整数からランダムに選んだ数が得られますから，これを k に代入します。そして k 番目の景品が取れたことを示すために a[k] に 1 を代入します。さらにクジを引いた回数 n を 1 だけ増やしますが，これは n より 1 大きい値である n+1 を n に代入することで実現します。最後に途中経過として n と a の値を表示しておきましょう。入力支援ボタンの「出力」を押すと「**表示する（《値》）**」と表示されるので，n と a をコンマ区切りで書いておきます。

3.1.4 実行

これでプログラムが完成したので，左上の「実行」ボタンを押して実行してみてください。エラーがでた場合は，どのあたりにエラーと思われる場所があるかが表示されるので，その周辺を見直してください。何百回も繰り返しても終わらないときは「リセット」ボタンで止めて，プログラムを見直してください。特に 2 行目，3 行目の 0 や 1 の個数が間違っていないかとか…。

3.2 何回もやってみる

何回か実行してみると，運のいいときと悪いときでずいぶん回数に差があることがわかります。ではこれの平均はどのくらいでしょうか。だいたいの見当はつきそうですが，それでは不確かなの

で 100 回くらいやってみて平均を取ることにしましょう。

3.2.1 新しい構文と右クリックメニュー

話を進める前に新しい構文をいくつか紹介しておきます。これを使ったプログラムを例 2 に書いておきます。まず [0,0,0,0,0,0,0,0,0,0] ですが，こういうのを回数間違えないように注意して入力するというのはイカさないやり方です。そこで同じ値からなる配列は「《整数》個の《値》」で生成できるようになっています。そしてこれはプログラム入力欄で右クリックして出るメニューで入力することができます。まず [0,0,...] を消して，そのカーソル位置でマウスを右クリックするとメニューが出てきます。「入力補助」の中に「個の」があるのでこれを使ってください。

例 2

```
n=0
a=10 個の 0
a の中に 0 の間：
    k=random(9)
    a[k]=1
    n+=1
    表示する (n,a)
```

繰り返しの条件も直しましょう。「a!=10 個の 1」でもいいのですが，配列の中にある値があるかどうかは「《配列》の中に《値》」で判定できます。a の中に 0 が残っている間繰り返せばいいのだから，「a の中に 0 の間：」とすればいいわけです。これも右クリックメニューにあるので，それを使って書き直してみてください。

最後にもう 1 つ書き換えたところがあります。例 1 では n=n+1 と書いてあったところを例 2 では n+=1 としてありますが，これはまったく同じ意味です。同様に-=や*=なども使えます。

3.2.2 関数

ではいよいよシミュレーションを繰り返してみます。1 回分のプログラムは例 2 にできているのでこれを有効に使いましょう。そこでこの例 2 の手順を「関数」にしてしまいましょう。ここでは一連の手順をまとめたものを関数といいます [2]。途中経過の表示はいらなくて，何回で揃ったかという回数だけがほしいので，その値を返すような関数にします。

例 3

```
関数 gacha()：
    n=0
    a=10 個の 0
    a の中に 0 の間：
        k=random(9)
        a[k]=1
        n+=1
    n を返す
```

やり方を次に示します。最後の「n を返す」のインデントを減らすのは，繰り返しが終わってから行う作業だからです。出来上がりは例 3 のようになります。

1. 最初の行の前に「関数 gacha()：」と入力する（1 行空けてから入力支援ボタンの「関数」）。
2. 関数の中身をマウスで選択して，Tab のキーでインデントを 1 段増やす。
3. 関数最後の行を「n を返す」に書き換える。ただしインデントを 1 段減らす（Shift+Tab）。

2） 数学でいう関数は同じ入力に対して同じ出力を返すものだと定義されていますが，プログラミングでいう関数はそうとは限りません。

この関数を 100 回呼び出して平均をとることにしましょう。回数の決まった繰り返しでは「増やしながら」「減らしながら」の構文を使うと便利です。例 3 の関数の外に例 4 のプログラムを作って実行してみましょう。平均はどのくらいになったでしょうか。なお，ここで使われている n は関数 gacha() の中で使われている n とは別物です。

例 4

```
s=0
n を 1 から 100 まで 1 ずつ増やしながら：
    s+=gacha()
表示する (s/100)
```

実はこの問題は「クーポンコレクター問題」と呼ばれており，一般に n 個の景品を集め終わるまでの回数の期待値は $n(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n})$ であることが知られています。景品が 10 種類の場合は $n=10$ なので $10(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{10})=29.3$ 回です。

3.3 バリエーション

このプログラムはいろいろなバリエーションが考えられます。たとえば日本語 Wikipedia のクーポンコレクター問題のページには「全 50 種類のクーポンを収集するには，平均で約 225 回の試行が必要となる」と書かれています。これを検証する場合にこのプログラムの 10 を 50 に書き換えただけではダメです（random(9) → random(49) も必要）。プログラムの修正にはこういうところのミスがつきものです。そこで関数 gacha の引数で景品の個数を指定できるようにしてみたのが例 5 です。こうしておけば安心していろいろな回数を試すことができます。

また，クーポンコレクター問題の期待値の計算は比較的簡単なのですが，確率分布についてはかなり難しいようです [3]。そこで回数のヒストグラムを書いてみたのが例 6 です。

例 5

```
関数 gacha(N)：
    n=0
    a=N 個の 0
    a の中に 0 の間：
        k=random(N-1)
        a[k]=1
        n+=1
    n を返す
s=0
n を 1 から 100 まで 1 ずつ増やしながら：
    s+=gacha(10)
表示する (s/100)
```

例 6

```
関数 gacha(N)：
    n=0
    a=N 個の 0
    a の中に 0 の間：
        k=random(N-1)
        a[k]=1
        n+=1
    n を返す
a=[]
n を 1 から 100 まで 1 ずつ増やしながら：
    a に gacha(10) を追加する
layout={"title":"ガチャの回数"}
data={"type":"histogram", "x":a}
グラフ描画 (layout,[data])
```

またここまではすべて等確率でシミュレーションしてきましたが，実際のガチャはレアアイテムなどの要素があるものがよくあります。これはどのように実装すればいいでしょうか。たとえば 10 種類の景品で 1 つだけ確率 1%，他の 9 つは 11%とするのであれば

3) 第 2 種スターリング数とか出てきたり…。

- random(9) を (random(99)+10)//11 にする。
- 0 がでたらもう一度乱数を引き直す。

などの方法があります（いずれも 0 がレアアイテムになる）。

4 モンティ・ホール問題

有名な問題なのでご存知の人も多いとは思いますが，「モンティ・ホール問題」を取り上げてみます。アメリカのテレビ番組の演出で，次のようなルールのゲームがありました。

1. プレイヤーの前に 3 つのドアがあり，一つだけに「アタリ」が入っている。
2. プレイヤーは 1 つのドアを選ぶ。
3. 司会者はプレイヤーが選んでない「ハズレ」のドアを開けて見せる。そしてプレイヤーに「ドアを選び直しますか」と質問する。プレイヤーは選んだドアを変えてもいいし変えなくてもいい。

プレイヤーは選んだドアを変えたほうが有利なのか，変えないほうが有利なのか，それとも変えても変えなくても同じなのか…というのがモンティ・ホール問題です。

数学で考えるとすぐに結論が出てしまうので，どうするのが有利なのかをプログラミングで検証してみましょう。

4.1 メイン部分

アタリかハズレかを判定する関数 play を後で作ります。アタリなら 1，ハズレなら 0 を返す関数です。これを 1000 回呼び出してアタリの確率を求めることにしましょう。そうするとプログラムのメイン部分は例 7 のようになります。w はアタリを引いた回数なので，最初は 0 で初期化しておいて，アタリを引くたびに 1 が加えられます。

例 7

```
N=1000
w=0
n を 1 から N まで 1 ずつ増やしながら：
    w+=play()
表示する (w/N)
```

4.2 変えない場合

選んだドアを変えない場合の play 関数を作ってみましょう。例 7 のプログラムの前でも後でもいいので，入力支援ボタンの「関数」を押して関数名を play に書き換えてください。

ドアの番号は 0〜2 とします。0〜2 の乱数は random(2) で生成できます。アタリの番号を x，プレイヤーが選ぶドアの番号を y とし，それぞれを乱数で決定します。x と y が一致したときがアタリなのでそのときは 1 を，そうでないときは 0 を返すので，関数は例 8 のようになります。実行して確率を確認してください。

例 8

```
関数 play()：
    x=random(2)
    y=random(2)
    もし x==y ならば：
        1 を返す
    そうでなければ：
        0 を返す
```

4.3 変える場合

今度は選んだドアを必ず変える場合について考えます。例 8 で y を選んだ後に選び直す処理をするのですが，これは割とややこしいので関数にしてしまいましょう。選ぶドアを変えるためには司会者が開けるドアを決めなくてはいけないわけですが，そのためにはアタリの番号がわかっている必要があります。そこで x と y の値を受け取って，新しく選ぶドアの番号を返す関数 reselect を作ることにします。

例 9 の reselect(x,y) を実際に作ってみてください。ヒントをいくつか述べます。

例 9

```
関数 play()：
    x=random(2)
    y=random(2)
    y=reselect(x,y)
    もし x==y ならば：
        1 を返す
    そうでなければ：
        0 を返す

関数 reselect(x,y)：
    ???
```

- まず司会者が開けるドアの番号を決めます（これを z とする）。ただしこれは x と y が一致する場合とそうでない場合で決め方が違ってきます。
  - x と y が一致する場合は，z は残り 2 つのどちらかで決めます。
  - x と y が異なる場合は，z は 1 つに決まります。その番号は x でも y でもないのだから，z は 3-x-y になります（なぜかわかりますか）。
- z が決まったら，選び直すドアの番号を決めるわけですが，それは y でも z でもない番号なので 3-y-z になります。

例 10

```
関数 reselect(x,y)：
    もし x==y ならば：
        z=x
        z==x の間：
            z=random(2)
    そうでなければ：
        z=3-x-y
    3-y-z を返す
```

完成したら実行して確率を調べてください。モンティ・ホール問題の結論はどうだったでしょうか。参考までに筆者が作成したプログラム例を例 10 に示しておきます。

実は結論がこうなるのは当たり前なのです—reselect でアタリを引く確率は，それをしなかった場合にハズレを引く確率と同じなのですから。プログラムを作成するためには問題を正確に考える必要がありますから，その過程でそういったことに気がつくこともあるのではないかと考えています。

ちなみに，このプログラムはもっと簡単にすることができます。たとえば x と y の両方を乱数にしましたが，片方を定数にしても確率は同じです。x を 0 に固定した場合，reselect はもっと簡単な関数になります。

5 PyPEN の使用に関して

5.1 使わなかった機能

すべての機能について説明することはできませんが，よく使う機能のうち今日の講習の中で使わなかったものについて簡単に説明を加えておきます。詳細はマニュアルをご覧ください。

5.1.1 セーブ・ロード

PyPEN のプログラムのセーブやロードは Download や Upload のリンクやボタンで行います。拡張子は.PyPEN になりますが，実際のところはテキストファイルです。プログラムをドラッグ&ドロップすることで読み込ませることもできます。

5.1.2 コード→フローチャート

「フローチャート」のチェックを入れて，「コード→フローチャート」ボタンを押すとコードに対応したフローチャートを生成することができます。ただし関数や手続きを定義しているプログラムには対応していません。

また逆に，フローチャートの部品や縦線を右クリックしてフローチャートに変更を加えると，プログラムもそれに対応したものに書き換えられます。

5.1.3 コード→ Python

「コード→ Python」ボタンを押すと，Python で実行できるコードが出力されます。ただしグラフィックを使用しているプログラムには対応していません。

PyPEN はプログラムの実行速度が遅いので，大量の計算をさせるには向いていません。そこでこのボタンで Python のコードを出力して，そのコードを Python の実行環境 [4] にコピーして実行するなどの利用方法が考えられます。

5.1.4 URL 生成

このプリントのプログラム例のリンクは「URL 生成」ボタンで生成したものです。これで生成される URL を開くと，プログラムが入った状態で PyPEN の環境が開きます。

5.2 授業等で使う場合

PyPEN は自由にお使いください。筆者のサイトにあるものを使っていただくこともできますが，Web サーバには何の設定もいらないので，ファイルをコピーするだけで違うサーバなどに置くこともできます。そのようにすることで，サンプルプログラムや問題を自由に変えることができます。サンプルプログラムは sample.js，問題は answer.js というファイルなのですが，これらを生成するプログラムも用意してあります [10]。また，Web サーバ上でなく PC 上にファイルを置いて，index.html を Web ブラウザで開くことでも使えます。

自分で PyPEN の環境を用意するには 2 つの方法があります。

1. git を使う
2. 筆者のサイトにある ZIP ファイルを展開する

後者については筆者のサイトにある PyPEN のページから PyPEN.zip をダウンロードして展開する

4) たとえばこの後で話がある Google Colaboratory とか。

だけなので，説明は省略します。前者については git をインストールした環境で

```
git clone git@github.com:watayan/PyPEN.git
```

または

```
git clone https://github.com/watayan/PyPEN.git
```

を実行すると，PyPEN というフォルダが作られ，中に必要なファイルがコピーされます。そこでそのフォルダの中で sample.js-dist，answer.js-dist というファイルをそれぞれ sample.js，answer.js という名前にコピーしてください [5]。この方法のメリットは，今後 PyPEN がバージョンアップしたときに

```
git pull
```

を行うだけで最新版に更新できるということです。

PyPEN は開発途上のプログラムです。不具合等を感じることがありましたら，筆者までお知らせいただけると幸いです。

参考文献

[1]文部科学省. **令和 7 年度大学入学者選抜改革に係る大学入学共通テスト実施大綱の予告**. July 2021. URL: https://www.mext.go.jp/content/20210729-mxt_daigakuc02-100001207_2.pdf.

[2]情報処理学会. **大学入学共通テストへの「情報」の出題について**. Dec. 2020. URL: https://www.ipsj.or.jp/education/edu202012.html.

[3]大学入試センター. **令和 7 年度以降の試験**. Mar. 2021. URL: https://www.dnc.ac.jp/kyotsu/shiken_jouhou/r7ikou.html.

[4]大学入試センター. **センター試験用手順記述標準言語 (DNCL) の説明**. 2011. URL: https://www.dnc.ac.jp/albums/abm00004841.pdf.

[5]中村亮太，西田知博，松浦敏雄. **プログラミング入門教育用学習環境 PEN**. 情報処理学会研究報告　コンピュータと教育 (CE) 2005-CE-081. No.104 (2005). URL: https://www.media.osaka-cu.ac.jp/PEN/.

[6]本多祐希，漆原宏丞，兼宗 進. **試験問題記述言語 DNCL の改定に合わせた「どんくり」システムの修正と検討**. 情報処理学会　研究報告コンピュータと教育 (CE) 2021-CE-159. No.2 (2021).

[7]大門 巧. **つちのこ**. URL: https://t-daimon.jp/tsuchinoko/.

[8]伊藤一成. **Pictogramming – Code yourself**. URL: https://www.pictogramming.org.

[9]中西 渉. **PyPEN**. URL: https://watayan.net/prog/pypen.html.

[10]中西 渉. **WaPEN Tools**. URL: https://watayan.net/prog/wapentools.html.

5) そのまま使えないのは少し面倒ですが，この 2 つのファイルは上述したように自分で書き換えたものを使うことがあるので，更新対象からはずしておくためにこのようにしてあります。