DNCL

中西渉

watayan@meigaku.ac.jp
名古屋高等学校

2022.07.31
2022 年度高等学校情報科教員研修

（このスライドのデータは https://watayan.net/doc/ にあります）

1. はじめに

DNCL とは…

- 大学入試センター試験・大学入学共通テスト「情報関係基礎」プログラミング問題で用いられる擬似言語
- 日本語ベース
- Daigaku Nyuushi Center (shiken) Language（と言われている）

これまでの経緯（と予想）

- 1997 「情報関係基礎」
  BASIC，COBOL，Pascal で出題
- 1998 日本語ベースの擬似言語
  「読めばわかる」
- 2002 仕様公開
- 2003 DNCL と命名
- 2011 仕様を事前公開
- 2021 共通テストにあわせて仕様調整
- 2025 「情報 I」共通テストで新 DNCL（？）

コードの例

旧 DNCL

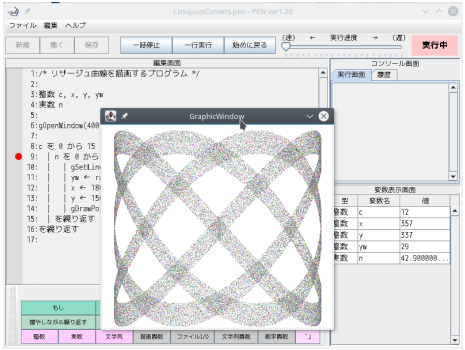
```
n ← 100
n > 1 の間，
 │ もし n % 2 = 0 ならば
 │  │ n ← n ÷ 2
 │ を実行し，そうでなければ
 │  │ n ← 3 * n + 1
 │ を実行する
 │ n を表示する
を繰り返す
```

新 DNCL（一部想像）

```
n = 100
n > 1 の間：
 │ もし n % 2 == 0 ならば：
 │  └ n = n / 2
 │ そうでなければ：
 │  └ n = 3 * n + 1
 └ 表示する(n)
```

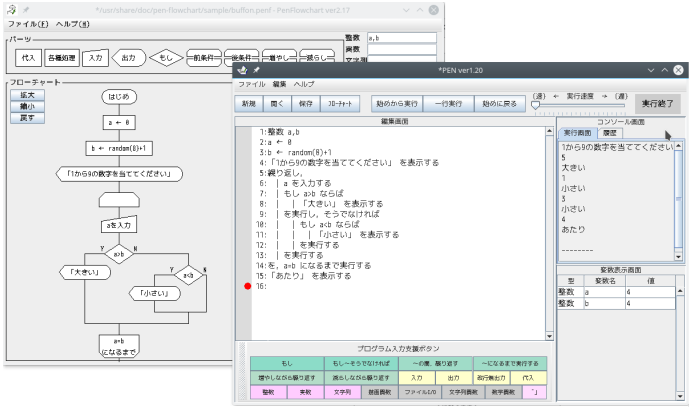
PEN(Programming Environment for Novice)

- 大阪市立大学・大阪学院大学で開発
- 変数宣言などを追加（xDNCL）
- 入力支援ボタン



PenFlowchart

- PEN にフローチャートを付加
- フローチャート ↔ プログラム
- 実行部分は PEN そのまま



Java アプリケーション→ Web アプリケーション

- WaPEN
- どんくり
- wPEN
- Tetra, XTetra
- …

新 DNCL への対応も

- PyPEN
- つちのこ
- …

PyPEN

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 p=1
2 nを1から100まで1ずつ増やしながら：
3 ····p=p*n
4 ····表示する(nと"!="とp)
```

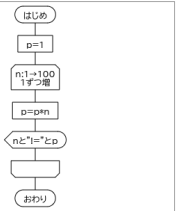
```
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
11!=39916800
12!=479001600
13!=6227020800
14!=87178291200
15!=1307674368000
16!=20922789888000
17!=355687428096000
18!=6402373705728000
実行時エラーです
3行目:整数で表される
範囲を越えました
```

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
〜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

マニュアル

\+ 0 −



アップロード ダウンロード 削除 全削除
--空--

PyPEN のインストール

1. Git を使う

`git clone https://github.com/watayan/PyPEN`
`answer.js-dist` を `answer.js` にコピー
`sample.js-dist` を `sample.js` にコピー

2. Git を使わない

`https://watayan.net/prog/pypen.html` から `PyPEN.zip` をダウンロード
適当なフォルダに展開

本研修のためにセットアップした PyPEN は
`https://watayan.net/misc/ipsj2022/`
にあります

PyPEN での新 DNCL の文法（構文）

入力

《変数》に《型》を入力する

出力

表示する（《値》）

代入

《変数》=《値》

選択

もし《条件》ならば：
　　《処理》
そうでなければ：
　　《処理》

繰り返し

《条件》の間：
　　《処理》

回数を決めた繰り返し

《変数》を《値》から《値》まで《値》ずつ増やしながら：
　　《処理》

PyPEN での新 DNCL の文法（演算子）

値の演算					
+	加算	&	ビット AND	and	論理 AND
-	減算	|	ビット OR	or	論理 OR
*	乗算	^	ビット XOR	not	論理 NOT
/	余りを出さない除算	~	ビット反転		
//	余りを出す除算	<<	ビット左シフト		
%	除算の余り	>>	ビット右シフト		
**	べき乗				

値の比較		その他	
==	等しい	《配列》の中に《値》	配列の中に値があるか
!=	等しくない	《整数》個の《値》	値を整数個並べた配列
>	大なり	《配列》に《値》を追加する	
<	小なり	《配列》に《配列》を連結する	
>=	大なりイコール		
<=	小なりイコール		

2. プログラムの例

ガチャのプログラムを作ってみる

- 景品は 10 個（等確率）
- 全部揃うまでクジを引き続ける
- 回数とどの景品を取ったかを，毎回表示する

方針：

- `random(9)` で 0〜9 の整数の乱数が得られる
- 配列に 10 個の 0 を入れておく
- 取った景品番号に対応する値を 1 にする
- 配列が全部 1 になったら終わり

ガチャのプログラム　（サンプル 1）

```
n = 0
a = 10 個の 0
a != 10 個の 1 の間：
    n = n + 1
    b = random(9)
    もし a[b] == 0 ならば：
        表示する (b, "ゲット！")
        a[b] = 1
    表示する (n, a)
```

（参考）クーポンコレクター問題

n 個の景品が揃うまでの回数の期待値は次の式で表される：

$$n\left(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}\right)$$

$n=10$ のときは約 29.3 回

どんな改造が考えられるか

- 景品数を増やしてみる
- レアアイテムを考える
- 何度も繰り返して平均回数を求める
- …

景品数を増やしてみる　（サンプル 2）

```
n = 0
a = 100 個の 0
a != 100 個の 1 の間：
    n = n + 1
    b = random(99)
    もし a[b] == 0 ならば：
        表示する (b, "ゲット！")
        a[b] = 1
    表示する (n, a)
```

景品数を増やしてみる（改良版）　（サンプル 3）

```
N = 100
n = 0
a = N 個の 0
a != N 個の 1 の間：
    n = n + 1
    b = random(N - 1)
    もし a[b] == 0 ならば：
        表示する (b, "ゲット！")
        a[b] = 1
    表示する (n, a)
```

レアアイテムを考える　（サンプル 4）

```
n = 0
a = 10 個の 0
a != 10 個の 1 の間：
    n = n + 1
    b = random(9)
    もし b == 0 ならば：
        b = random(9)
    もし a[b] == 0 ならば：
        表示する (b, "ゲット！")
        a[b] = 1
    表示する (n, a)
```

何度も繰り返して平均回数を求める　（サンプル 5）

```
関数 gacha(N)：
    n = 0
    a = N 個の 0
    a != N 個の 1 の間：
        n = n + 1
        b = random(N - 1)
    a[b] = 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 = s + gacha(10)
表示する (s / 100)
```

3. センター試験問題より

よくある誤解
「出題のコードはそのまま実行できる」

データの初期化部分は出題のコードにはない
→補わないと動くコードにならない

例：2018 年度問題（迷路）

第 3 問　**（選択問題）**　次の文章を読み，下の問い（**問 1** ～ **3**）に答えよ。（配点　35）

白マス（道）と黒マス（壁）で構成された図 1 のような迷路を解きたい。このような迷路では，すべての袋小路を黒く塗ることによって，スタート（S）からゴール（G）までの道が，図 2 のように白いままのマスとして浮き出る。なお，図 2 では，わかりやすくなるように，塗ったマスを灰色で示している。

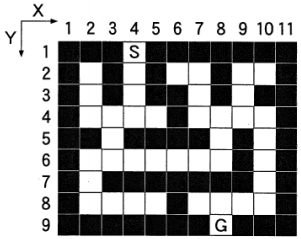


図 1　迷路の例題

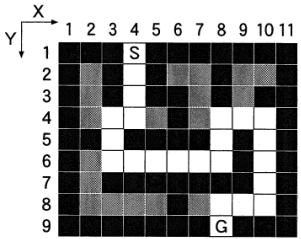


図 2　袋小路を塗った迷路

問題文

手順 1 に従って袋小路を塗る手続きを，次ページの図 5 のように作成した。

変数 **tate** と変数 **yoko** には，迷路の行数と列数がそれぞれあらかじめ格納されている。

迷路は 2 次元配列 **Masu** に格納されている。図 4 に示すように，配列の要素 **Masu[x,y]** には，マス (x, y) が壁ならば 1 が，道であれば 0 が，またスタートやゴールならば 9 があらかじめ格納されている。マスを塗ることは，対応する配列の要素の値を 1 にすることである。

Y＼X	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	9	1	1	1	1	1	1	1
2	1	0	1	0	1	0	0	1	0	0	1
3	1	0	1	0	1	1	0	1	0	1	1
4	1	0	0	0	0	1	0	0	0	0	1
5	1	1	0	1	1	1	1	0	1	0	1
6	1	0	0	0	0	0	0	0	1	0	1
7	1	0	1	1	1	1	1	1	1	0	1
8	1	0	0	0	0	1	0	0	0	0	1
9	1	1	1	1	1	1	1	9	1	1	1

図 4　数値による迷路の表現

コード

```
(01) 繰り返し，
(02)     nutta ← [コ]
(03)     y を 2 から [サ] まで 1 ずつ増やしながら，
(04)         x を 2 から [シ] まで 1 ずつ増やしながら，
(05)             s ← Masu[x − 1,y] + Masu[x + 1,y] +
                     Masu[x,y − 1] + Masu[x,y + 1]
(06)             もし [ス] かつ s = [セ] ならば
(07)                 Masu[x,y] ← [ソ] ，nutta ← 1
(08)             を実行する
(09)         を繰り返す
(10)     を繰り返す
(11) を，nutta = 0 になるまで実行する
(12) y を 1 から tate まで 1 ずつ増やしながら，
(13)     x を 1 から yoko まで 1 ずつ増やしながら，
(14)         もし Masu[x,y] = 1 ならば
(15)             "■"を改行なしで表示する
(16)         を実行し，そうでなければ
(17)             "□"を改行なしで表示する
(18)         を実行する
(19)     を繰り返す
(20)     改行を表示する
(21) を繰り返す
```

新 DNCL 版のコード　（サンプル 8）

```
nutta = 1
nutta != 0 の間：
    nutta = 0
    y を 1 から tate-2 まで 1 ずつ増やしながら：
        x を 2 から yoko-2 まで 1 ずつ増やしながら：
            s = Masu[x-1,y]+Masu[x+1,y]（次の行に続く）
              +Masu[x,y-1]+Masu[x,y+1]
            もし Masu[x,y] == 0 and s == 3 ならば：
                Masu[x,y] = 1
                nutta = 1
y を 0 から tate-1 まで 1 ずつ増やしながら：
    x を 0 から yoko-1 まで 1 ずつ増やしながら：
        もし Masu[x,y] == 1 ならば：
            改行なしで表示する ("■")
        そうでなければ：
            改行なしで表示する ("□")
    改行する
```

冒頭に必要なコード

```
tate = 9
yoko = 11
Masu = [
    [1,1,1,1,1,1,1,1,1,1,1],
    [1,0,0,0,1,0,0,0,1],
    [1,1,1,0,0,0,1,0,1],
    [9,0,0,0,1,0,1,0,1],
    [1,1,1,0,1,0,1,0,1],
    [1,0,1,1,1,0,1,1,1],
    [1,0,0,0,1,0,1,0,1],
    [1,1,1,0,0,0,1,0,9],
    [1,0,0,0,1,1,1,0,1],
    [1,0,1,0,0,0,0,0,1],
    [1,1,1,1,1,1,1,1,1,1,1]
    ]
```

（参考）改良版のコード　（サンプル 9）

```
y を 1 から tate-2 まで 1 ずつ増やしながら：
    x を 1 から yoko-2 まで 1 ずつ増やしながら：
        i = x
        j = y
        Masu[i,j]== 0 and（次の行に続く）
                Masu[i+1,j]+Masu[i-1,j]+Masu[i,j+1]+Masu[i,j-1]==3 の間：
            Masu[i,j] = 1
            di = Masu[i-1,j]-Masu[i+1,j]
            dj = Masu[i,j-1]-Masu[i,j+1]
            i = i + di
            j = j + dj
…表示部分は同じ…
```

（参考）改良版のコード…の更に書き換えた版　（サンプル 10）

```
y を 1 から tate-2 まで 1 ずつ増やしながら：
    x を 1 から yoko-2 まで 1 ずつ増やしながら：
        i = x
        j = y
        s = Masu[i+1,j]+Masu[i-1,j]+Masu[i,j+1]+Masu[i,j-1]
        Masu[i,j]== 0 and s == 3 の間：
            Masu[i,j] = 1
            di = Masu[i-1,j]-Masu[i+1,j]
            dj = Masu[i,j-1]-Masu[i,j+1]
            i = i + di
            j = j + dj
            s = Masu[i+1,j]+Masu[i-1,j]+Masu[i,j+1]+Masu[i,j-1]
…表示部分は同じ…
```

4. 試作・サンプル問題より

大学入学共通テストに「情報」が追加される
→問題イメージの公開

- 試作問題（検討用イメージ）
- サンプル問題

試作問題のプログラミング問題

課題　英文をシフト暗号で暗号化した以下の暗号文を解読しなさい。ただし，英文は全て小文字でアルファベット以外のスペースや数字，「'」「,」「.」「?」などは変換されていません。

（省略）····nonsmkdo k zybdsyx yp drkd psovn, kc k psxkv bocdsxq zvkmo pyb dryco gry robo qkfo drosb vsfoc drkd dro xkdsyx wsqrd vsfo. sd sc kvdyqodrob psddsxq kxn zbyzob drkd go cryevn ny drsc. led, sx k vkbqob coxco, go mkx xyd nonsmkdo - go mkx xyd myxcombkdo - go mkx xyd rkvvyg - drsc qbyexn. dro lbkfo wox, vsfsxq kxn nokn, gry cdbeqqvon robo, rkfo myxcombkdon sd, pkb klyfo yeb zyyb zygob dy knn yb nodbkmd. dro gybvn gsvv vsddvo xydo, xyb vyxq bowowlob grkd go cki robo, led sd mkx xofob pybqod grkd droi nsn robo. sd ···· （省略）

図1　先生が出した課題

文字の出現頻度を調べるプログラム…の前に

要素数(値)…配列の要素数を返す。
例：`Data=["M","i","s","s","i","s","s","i","p","p","i"]`の時
要素数(`Data`)は 11 を返す

差分(値)…アルファベットの「a」との位置の差分を返す
値がアルファベット以外の文字であれば－1を返す
例：**差分**(`"e"`)は 4 を，**差分**(`"x"`)は 23 を返す
差分(`"5"`)や**差分**(`","`)は－1を返す

文字(値)…番号の値に対するアルファベットの文字を返す。
値が 0 以上 25 以下でなければ「アルファベットでない」を返す
例：**文字**(4)は「e」を，**文字**(23)は「x」を返す
文字(-1)や**文字**(27)は「アルファベットでない」を返す

関数の実装　（サンプル 11）

```
関数 yousosuu(a)：
    length(a) を返す

関数 sabun(a)：
    s = "abcdefghijklmnopqrstuvwxyz"
    i を 0 から 25 まで 1 ずつ増やしながら：
        もし s[i]==a ならば：
            i を返す
    -1 を返す

関数 moji(a)：
    s = "abcdefghijklmnopqrstuvwxyz"
    もし a>=0 and a<=25 ならば：
        s[a] を返す
    そうでなければ：
        "アルファベットでない"を返す
```

いよいよプログラム本体…と思ったら

```
(01)Angoubun = ["p","y","e","b",…(省略)…"k","b","d","r","."]
(02)配列 Hindo のすべての要素に 0 を代入する
(03)i を 0 から 要素数(Angoubun)-1 まで 1 ずつ増やしながら:
(04)  bangou = 差分( ケ )
(05)  もし bangou != −1 ならば:
(06)    コ = コ + 1
(07)表示する(Hindo)
```

ケ ⓪Angoubun[i]

コ ④Hindo[bangou]

どれだけ省略されているかというと…

pyeb cmybo kxn cofox iokbc kqy yeb pkdrobc lbyeqrd pybdr yx drsc myxdsxoxd, k xog xkdsyx, myxmosfon sx vslobdi, kxn nonsmkdon dy dro zbyzycsdsyx drkd kvv wox kbo mbokdon oaekv. xyg go kbo oxqkqon sx k qbokd msfsv gkb, docdsxq grodrob drkd xkdsyx, yb kxi xkdsyx cy myxmosfon kxn cy nonsmkdon, mkx vyxq oxnebo. go kbo wod yx k qbokd lkddvo-psovn yp drkd gkb. go rkfo mywo dy nonsmkdo k zybdsyx yp drkd psovn, kc k psxkv bocdsxq zvkmo pyb dryco gry robo qkfo drosb vsfoc drkd drkd xkdsyx wsqrd vsfo. sd sc kvdyqodrob psddsxq kxn zbyzob drkd go cryevn ny drsc. led, sx k vkbqob coxco, go mkx xyd nonsmkdo -- go mkx xyd myxcombkdo -- go mkx xyd rkvvyg -- drsc qbyexn. dro lbkfo wox, vsfsxq kxn nokn, gry cdbeqqvon robo, rkfo myxcombkdon sd, pkb klyfo yeb zyyb zygob dy knn yb nodbkmd. dro gybvn gsvv vsddvo xydo, xyb vyxq bowowlob grkd go cki robo, led sd mkx xofob pybqod grkd droi nsn robo. sd sc pyb ec dro vsfsxq, bkdrob, dy lo nonsmkdon robo dy dro expsxscron gybu grsmr droi gry pyeqrd robo rkfo drec pkb cy xylvi knfkxmon. sd sc bkdrob pyb ec dy lo robo nonsmkdon dy dro qbokd dkcu bowksxsxq lopybo ec -- drkd pbyw droco ryxybon nokn go dkuo sxmbokcon nofydsyx dy drkd mkeco pyb grsmr droi qkfo dro vkcd pevv wokcebo yp nofydsyx -- drkd go robo rsqrvi bocyvfo drkd droco nokn crkvv xyd rkfo nson sx fksx -- drkd drsc xkdsyx, exnob qyn, crkvv rkfo k xog lsbdr yp pboonyw -- kxn drkd qyfobxwoxd yp dro zoyzvo, li dro zoyzvo, pyb dro zoyzvo, crkvv xyd zobscr pbyw dro okbdr.

本当にその通りのプログラムにすると…　（サンプル 12）

…関数定義…

```
Angoubun=["p","y","e","b"," ","c","m","y","b","o",
" ","k","x","n"," ","c","o","f","o","x"," ","i","o",
"k","b","c"," ","k","q","y"," ","y","e","b"," ","p",
"k","d","r","o","b","c"," ","l","b","y","e","q","r",
"d"," ","p","y","b","d","r"," ","y","x"," ","d","r",
"s","c"," ","m","y","x","d","s","x","o","x","d"," ",
"k"," ","x","o","g"," ","x","k","d","s","y","x"," ",
```

…100 行以上省略…

```
"z","o","b","s","c","r"," ","p","b","y","w"," ","d",
"r","o"," ","o","k","b","d","r","."]
Hindo = 26 個の 0
i を 0 から yousosuu(Angoubun)-1 まで 1 ずつ増やしながら：
    bangou = sabun(Angoubun[i])
    もし bangou != -1 ならば：
        Hindo[bangou] = Hindo[bangou] + 1
表示する (Hindo)
```

File I/O もどきを使ってみる　（サンプル 13）

```
…関数定義…
Angoubun=[]
f = openr("crypt.txt")
c = getchar(f)
c != ""の間：
    Angoubun に c を追加する
    c = getchar(f)
close(f)
Hindo = 26 個の 0
i を 0 から yousosuu(Angoubun)-1 まで 1 ずつ増やしながら：
    bangou = sabun(Angoubun[i])
    もし bangou != -1 ならば：
        Hindo[bangou] = Hindo[bangou] + 1
表示する (Hindo)
```

復号のプログラム　（サンプル 14）

…関数定義…
…ファイル読み込み…

```
Hirabun = yousosuu(Angoubun) 個の""
hukugousuu = 26 - 10
i を 0 から yousosuu(Angoubun)-1 まで 1 ずつ増やしながら：
    bangou = sabun(Angoubun[i])
    もし bangou != -1 ならば：
        Hirabun[i] = moji((bangou + hukugousuu) % 26)
    そうでなければ：
        Hirabun[i] = Angoubun[i]
表示する (Hirabun)
```

サンプル問題のプログラミング問題
比例代表選挙の当選者数

Mさん：表1に，最近行われた選挙結果のうち，ある地域のブロックについて，各政党の得票数を書いてみたよ。

表1　各政党の得票数

	A党	B党	C党	D党
得票数	1200	660	1440	180

Kさん：今回の議席数は6人だったね。得票の総数を議席数で割ると580人なので，これを基準得票数と呼ぶのがいいかな。平均して1議席が何票分の重みがあるかを表す数ということで。そうすると，各政党の得票数が何議席分に相当するかは，各政党の得票数をこの基準得票数で割れば求められるね。

Mさん：その考え方に沿って政党ごとの当選者数を計算するプログラムを書いてみよう。ま

```
(01) Tomei = ["A党","B党","C党","D党"]
(02) Tokuhyo = [1200,660,1440,180]
(03) sousuu = 0
(04) giseki = 6
(05) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(06)     sousuu = sousuu + Tokuhyo[m]
(07) kizyunsuu = sousuu / giseki
(08) 表示する("基準得票数:",kizyunsuu )
(09) 表示する("比例配分")
(10) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(11)     表示する(Tomei[m],":",[イ] / [ウ] )
```

```
基準得票数:580
比例配分
A党:2.068966
B党:1.137931
C党:2.482759
D党:0.310345
```

前ページのプログラム　（サンプル 15）

```
Tomei = ["A 党","B 党","C 党","D 党"]
Tokuhyo = [1200,660,1440,180]
sousuu = 0
giseki = 6
m を 0 から 3 まで 1 ずつ増やしながら：
    sousuu = sousuu + Tokuhyo[m]
kizyunsuu = sousuu / giseki
表示する ("基準得票数：", kizyunsuu)
表示する ("比例配分")
m を 0 から 3 まで 1 ずつ増やしながら：
    表示する (Tomei[m], ":", Tokuhyo[m] / kizyunsuu)
```

```
(01) Tomei = ["A党","B党","C党","D党"]
(02) Tokuhyo = [1200,660,1440,180]
(03) Tosen = [0,0,0,0]
(04) tosenkei = 0
(05) giseki = 6
(06) m を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(07)     Hikaku[m] = Tokuhyo[m]
(08) [セ] < giseki の間繰り返す:
(09)     max = 0
(10)     i を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(11)         もし max < Hikaku[i]ならば:
(12)             [ソ]
(13)             maxi = i
(14)     Tosen[maxi] = Tosen[maxi] + 1
(15)     tosenkei = tosenkei + 1
(16)     Hikaku[maxi] = 切り捨て( [タ] / [チ] )
(17) k を 0 から [ア] まで 1 ずつ増やしながら繰り返す:
(18)     表示する(Tomei[k],"：",Tosen[k],"名")
```

A党:2 名
B党:1 名
C党:3 名
D党:0 名

前ページのプログラム　（サンプル 16）

```
Tomei = ["A 党","B 党","C 党","D 党"]
Tokuhyo = [1200,660,1440,180]
Tosen = [0,0,0,0]
Hikaku = [0,0,0,0]
tosenkei = 0
giseki = 6
m を 0 から 3 まで 1 ずつ増やしながら：
    Hikaku[m] = Tokuhyo[m]
tosenkei < giseki の間：
    max = 0
    i を 0 から 3 まで 1 ずつ増やしながら：
        もし max < Hikaku[i] ならば：
            max = Hikaku[i]
            maxi = i
    Tosen[maxi] = Tosen[maxi] + 1
    tosenkei = tosenkei + 1
    Hikaku[maxi] = Tokuhyo[maxi] // (Tosen[maxi] + 1)
k を 0 から 3 まで 1 ずつ増やしながら：
    表示する (Tomei[k],":","Tosen[k],"名")
```

5. 新 DNCL の文法について思うこと

新 DNCL は文法が固まっていない

- ループは「繰り返す」が必要？
- !=はあるが==はまだ見てない
- 割り算の区別
- 論理式の解釈順
  旧 DNCL では「左から」なので「A かつ B でない」は「(A かつ B) でない」
- a<b<c を許すか

（個人的に）できてほしいこと

- 配列を添え字を使わずになめる
  Python でいう「`for`《変数》`in`《リスト》」
  PyPEN では「《配列》の要素《変数》について」で実装
- 配列にある値が含まれるかの判定
  Python でいう「《値》`in`《リスト》」
  PyPEN では「《配列》の中に《値》」で実装

PyPEN で「あえて」実装していないこと

- `sum` などの「よく使う」関数
- `sort` などの「よく使う」手続き

6. おわりに

新 DNCL でありそうな話

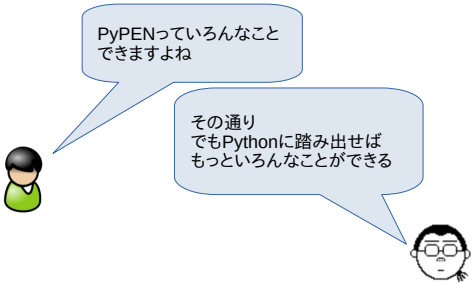
- 共通テストでは新 DNCL が使われるだろう
- 新 DNCL には実行環境が存在する

→ 新 DNCL で「情報 I」の授業をすればいいのでは？

※筆者は賛成しない

筆者の思う DNCL

- DNCL「だけ」の授業ではいけない
  「情報 I」の学習活動に活かせるプログラミング
  →実用的な言語を使えるようにしたい
- 導入・演習だけなら有効か



ある日の授業後…