

# 第4章の代わりにのテキスト

名古屋高校 情報科

2019 年度

## 第4章 問題の解決と処理手順の自動化

### 第4章第1節 はじめに

コンピュータはプログラムを実行することによって作業をしている。それはパソコンだけでなく、iPadのようなタブレットや携帯電話やゲーム機、あるいは自動販売機やデジタル時計でも同様である。我々が普段何気なく使っている機器の多くにコンピュータが搭載されていて、それらを扱うということは、プログラムを実行する（あるいは実行中のプログラムを操作する）ことにほかならない。

この单元では、自分でプログラムを作ってみる。そのことを通じて、身の回りのコンピュータやプログラムがどのように動いているのか、考えを巡らせてみてほしい。

#### 4.1.1 アルゴリズム

アルゴリズムとは、問題を解決するための具体的な手順を記述したものである。たとえば数学の問題を解く手順、理科の実験の手順、駅で切符を買って電車に乗る手順なども一種のアルゴリズムといえる。

アルゴリズムを頭の中だけで考えるのは間違えやすいし、他人に伝えることも出来ない。そこでアルゴリズムを記述する方法として、この授業ではフローチャートを用いる。

#### 4.1.2 PyPEN

アルゴリズムはプログラムではないから、そのままではコンピュータで実行できない。そこでプログラミング言語を用いて、アルゴリズムを元にプログラムを作らなくてはならない。世の中にはたくさんのプログラミング言語があるが、この授業では大学入試センター試験「情報関係基礎」で用いられている DNCL という言語を Python 風アレンジした PyPEN を用いる。

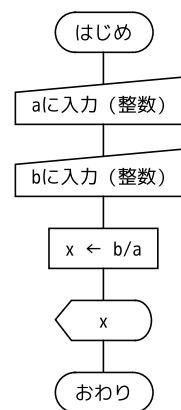
ブラウザを起動して <http://www.nagoya-gakuin.ed.jp> から「PyPEN」を選んで、始めよう。ぜひともブックマークしておいてほしい。

### 第4章第2節 PyPEN の使い方

まず手始めに、一次方程式  $ax = b$  を解くプログラムを作成する。右にフローチャート、下にプログラムを示した。作業を進めることで PyPEN の使い方を覚えることが目的なので、あえて少し問題のあるプログラムになっているが、気づいたとしてもとりあえず我慢してほしい。

なお、今後できるだけプログラムとフローチャートを併記するが、どちらを見てももう片方が思い浮かぶように、両方を理解するようにしてほしい。

a に整数を入力する  
b に整数を入力する  
 $x \leftarrow b/a$   
x を表示する

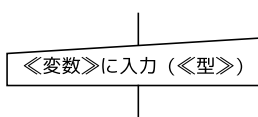


#### 4.2.1 変数

PyPENに限らず多くのプログラミング言語では、値を覚えておくための変数を使える。変数の名前は、英字で始まる英数文字列（半角文字）でなくてはならない。また、大文字と小文字は区別される。

#### 4.2.2 入力

a, b の値は、ユーザがキーボードから入力することにしよう。入力はフローチャートでは右のような記号で表され、PyPEN では次のような構文になる（《型》は「整数」「実数」「文字列」「真偽」のどれか）。



《変数》に《型》を入力する

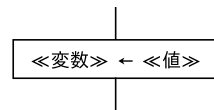
下にある入力支援ボタンの「入力（整数）」を押してみよう。するとプログラム入力欄に「《変数》に整数を入力する」と表示されるので、「《変数》」のところをクリックして a と書き換えよう。

b の入力も同様に入力すればいいのだが、これはフローチャートの画面でやってみよう。まず「コード→フローチャート」ボタンを押すと、既に入力した a の入力に相当するフローチャートが右側に表示される。その下の縦棒のところで右クリックするとメニューが表示されるので「入力」を選ぶと、フローチャートに入力の記号が挿入される。今度はその記号を右クリックして出てくるメニューから「編集」を選ぶと、編集画面が表示されるので変数名を書き換えるなどする。

このように入力支援ボタンを使ってプログラムを入力することもできれば、フローチャートでプログラムを生成することもできるので、好きな方を選んで使ってくればいい。ただし、フローチャートを編集したときは即座にプログラムに反映されるが、プログラムを編集したときは「コード→フローチャート」ボタンを押さないとフローチャートに反映されないことに気をつけてほしい。

#### 4.2.3 代入

変数に値を覚えさせることを代入という。 $x = \frac{b}{a}$  だから、この計算結果を x に代入しよう。代入はフローチャートでは右のような記号で表され、PyPEN では次のような構文になる。



《変数》 ← 《値》

入力支援ボタンでもいいし、フローチャートでもいいのでこれを追加しよう。ただし、 $\frac{b}{a}$  は b/a と記述する。

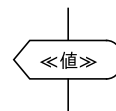
◇入力と代入 入力と代入はどちらも変数に値を覚えさせることではあるのだが、まったく違う動作であることに注意してほしい。

入力 ユーザが入力した値を変数に覚えさせる

代入 コンピュータが計算した値を変数に覚えさせる

#### 4.2.4 出力

計算した x の値を出力すれば目的は達成される。出力はフローチャートでは右のような記号で表され、PyPEN では次のような構文になる。もう説明しなくても何をしたらいいかわかるだろう。さっそくやってみよう。



《値》を表示する

#### 4.2.5 実行

プログラム入力欄の上にある「実行」ボタンを押すとプログラムが実行できる。さっそく実行して、 $3x = 6$  や  $4x = 13$  などが解けることを確認しよう。実行するとすぐ入力待ちの状態になるので、 $ax = b$  の a の値を入力して Enter を押す。また入力待ちになるので、今度は b を入力して Enter を押す。そうすると x の値が表示されるはずだ。

#### 4.2.6 デバッグ

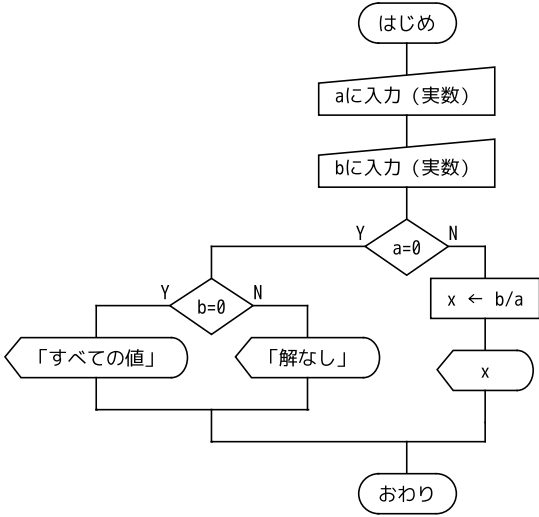
さて、これでプログラムが完成したわけだが、これで完璧だろうか。試しに次の方程式を解いてみよう。

$$3.5x = 7$$

$$0x = 2$$

どちらもエラーになってしまう。このようなことが起きてしまうようでは完璧なプログラムとはいえない。このようなプログラムの誤りをバグといい、バグを取り除いて正しいプログラムにすることをデバッグという。デバッグした結果、できあがったプログラム、フローチャートは以下ようになる。実際に作って実行してみよう。現時点では意味は詳しくわからなくてもいいが... フローチャートとプログラムを比較すれば、どこがどう対応しているかはわかるのではないかな。

a に実数を入力する  
 b に実数を入力する  
 もし a=0 ならば：  
   もし b=0 ならば：  
     「すべての値」を表示する  
   そうでなければ：  
     「解なし」を表示する  
 そうでなければ：  
   x ← b/a  
   x を表示する



4.2.7 課題提出の方法

課題を提出するときは、プログラムをコピーして貼り付ける。プログラムの画面をクリックしてから Ctrl+A を押すと全文選択、これをコピーして (Ctrl+C) 貼り付ければよい (Ctrl+V)。右クリックメニューは使えない場面があるので、これらのキーを覚えること。

第 4 章第 3 節 値・演算子

4.3.1 値

PyPEN で使える値には次の 4 種類の型がある。

- 整数 整数が保存できる。
- 実数 実数が保存できる...ということだが、小数が扱えると考えればよい。
- 文字列 文字列が保存できる。プログラム中では文字列は「」か"で囲む。
- 真偽 条件の真偽が保存できる。TRUE (真) または FALSE (偽) のどちらかの値をとる (この授業では使用しない)。

4.3.2 演算子

計算の記号 (演算子) は次のものが使える。

+	和
-	差
*	積
/	余りを出さない割り算の商 (結果は実数になる)
//	余りを出す割り算の商 (商の整数部分)
%	余りを出す割り算の余り

基本的に左から順に計算すること、+ や - よりも \*、/、//、% が優先されること、カッコがあればそれが最優先であることなどは算数や数学と同じである (\*1)。

次のプログラムで確認してみよう。

14/3 を表示する  
 14//3 を表示する  
 14%3 を表示する  
 1/3\*3 を表示する  
 1//3\*3 を表示する  
 1\*3//3 を表示する

(\*1) 数学ではカッコの外のカッコは中カッコを使ったりするが、ほとんどのプログラミング言語ではカッコはすべて ( ) である。

□ 課題：偶数奇数の判定

ここでは課題の作成・提出に必要な手順を述べる。作るプログラムは次のものとする：

整数を1つ入力し、それが偶数なら「偶数」、奇数なら「奇数」と表示する。

プログラムの作成は次の手順で行なう。

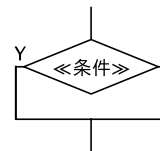
変数 どのような変数が必要か

アルゴリズム どのような順番で処理をしたらいいか

プログラム作成 考えたアルゴリズムの通りにプログラムを作る

動作確認 実行して、正しい結果が得られるかどうか確認しよう（うまくいかなかったら戻って考える）

このプログラムでは偶数のときと奇数のときで別の動作を行なうことになるので、その方法について説明する。このような処理は選択と呼ばれるもので、フローチャートでは右のような記号で表され、PyPEN では下のような構文になる。なお、行末のコロンや命令の前の空白には意味があるので、勝手に削除してはいけない。このプログラムでは入力支援ボタンの「もし～そうでなければ」ボタンを用いるのが良い。



もし《条件》ならば：  
《命令》

もし《条件》ならば：  
《命令》  
そうでなければ：  
《命令》

《条件》は  $a=0$  や  $a>0$  のように、等式や不等式で表す（詳細は後の授業で説明する）。数学だと  $n$  が偶数だといえば「 $n = 2k$  ( $k$  は整数)」と考えるが、それは使えない。そこで、 $n$  を2でわったときの余りを用いる。

完成したらプログラムをコピーして提出する。プログラム入力欄で Ctrl+A を押して全文選択して、コピー (Ctrl+C) する。返信メール (または moodle) の画面で貼り付け (Ctrl+V) すれば、本文にプログラムが挿入されるので、これを送信すればいい。

□ 2020年1月の曜日

2020年の1月は水曜日からはまる。日付の数字を入力したら、1月のその日が何曜日であるかを答えるプログラムを作ろう。ただし、結果は日曜日を0、月曜日を1、...、土曜日を6で表すことにする。

曜日は7日周期なので、7で割った余りを考えればいい。しかしそのまま日付の数字を7で割ったのではうまくいかない。たとえば1日が水曜日だから1を入力したら水曜日を表す3が出力されるように、5日が日曜日だから5を入力したら0が表示されるようにしたいわけだが、だったら日付の数字に何を足してから7で割ったらいいい？

□ 課題：Zellerの公式

特定の月の曜日だけがわかるというだけでは大して面白くもない。どうせなら自分の生まれた日の曜日が分かるものを作ろう。そこで、曜日を計算する方法としてもっともよく知られている Zeller (ツェラー) の公式を紹介する。これで曜日の計算をするプログラムを作って提出するのが今回の課題だ (\*2)

西暦  $y$  年  $m$  月  $d$  日の曜日は次のように計算できる。ただし1月、2月は前の年の13月、14月として計算する (つまり  $m < 3$  のときは  $y$  を1減らして  $m$  を12増やす)。  $y$  の上2桁を  $j$ 、下2桁を  $k$  とし (つまり  $y//100$  が  $j$ 、 $y\%100$  が  $k$ )、

$$h = d + \left\lfloor \frac{13m + 8}{5} \right\rfloor + k + \left\lfloor \frac{k}{4} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor + 5j$$

とするとき、 $h$  を7で割ったときの余りが0,1,...,6であれば、その日は日曜、月曜、...、土曜である。ただし  $\lfloor x \rfloor$  は  $x$  の小数部分を切り捨てた値である (\*3)。つまり割り算の商の整数部分を使いたいから「//」で割り算すればいい。

また、掛け算記号は省略できないことにも注意してほしい。たとえば  $13m + 8$  は  $13m+8$  でなく  $13*m+8$  と表すし、 $5j$  も  $5*j$  と表す。  $ab$  は1つの変数であって、 $a*b$  とは別物である。

(\*2) あとの授業で文字列での表示ができるようにバージョンアップするので、このプログラムは完成したら保存して、それまで残しておくこと。

(\*3) 床関数という。数学では  $\lfloor x \rfloor$  という表記で習い、「ガウスの記号」と呼ぶ。

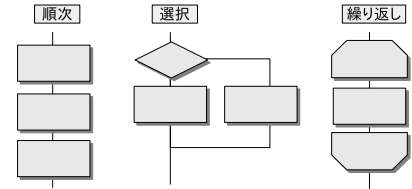
完成したら、いくつかの日付を入力して、正しい曜日が出力できることを確認してから提出しなさい。たとえば次の日付で答え合わせしてみるといい。

2020年1月1日	水曜(3)	2021年1月1日	金曜(5)
2020年3月1日	日曜(0)	1964年10月10日	土曜(6)
2020年12月31日	木曜(4)	2020年7月24日	金曜(5)

自分の生まれた日が何曜日であるかもわかるだろう。

## 第4章第4節 構造化プログラミング

世の中にはとても複雑なプログラムがあり、そのアルゴリズムは当然複雑なものであるが、実は「順次」「選択」「繰り返し」の組み合わせで作られているものが多い。実際 PyPEN では、この3つの構造で構成されたプログラムしか作ることにはできないが、アルゴリズムを表現するにはそれで十分なのである。では、この3つの構造を順に見ていこう。

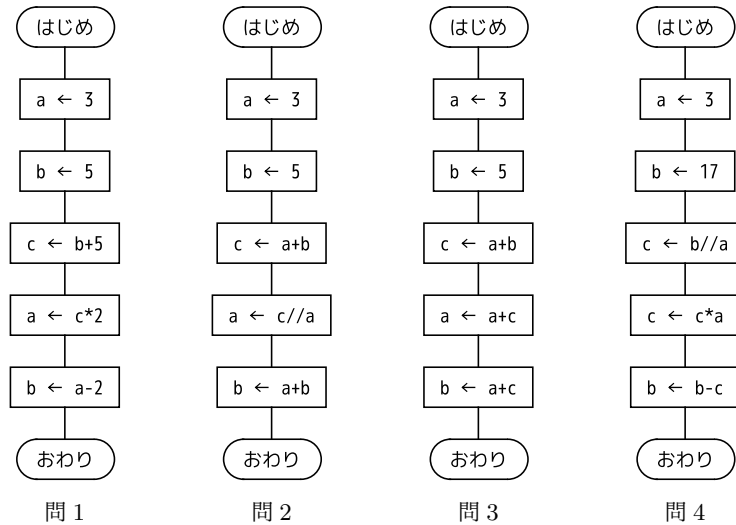


### 4.4.1 順次

順次というのは、上から順番に処理を実行していただくだけである。それは一見簡単なことのように思われる。しかし、たとえば変数に値を代入するという単純な処理でも、順番を入れ替えてしまうと望み通りの結果は得られない。

#### □ 問題1

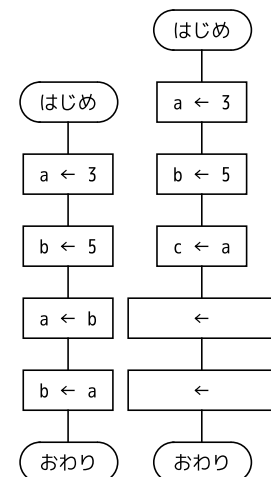
次の各アルゴリズムが終了したときの、各変数の値を答えよ。



#### □ 問題2

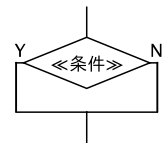
右図の左側は a と b の値を入れ替えようとしたものだが、失敗している（うまくいかないことを確認せよ）。実は変数の値の入れ替えをするには、もう一つ変数を用意する必要がある。a と b の値の入れ替えが成功するように、右側の空欄を埋めなさい。

なお、これは変数の値を入れ替えるときの常套手段である。そのまま丸暗記するようなものではないが「もう一つ変数を用意する」ということだけは覚えておいてほしい。



#### 4.4.2 選択

選択は、ある条件が満たされる時とそうでない時で別の処理を行なうというものである。



もし《条件》ならば：  
《命令》

もし《条件》ならば：  
《命令》  
そうでなければ：  
《命令》

条件は、2つの値を比較して、等しいとか、等しくないとか、どちらが大きいかというように Yes/No で判断できるものである。条件を表すために用いられる演算子は次のとおりである：

=	等しい	>	大なり	>=	大なりイコール (≥ の意味)
!=	等しくない (≠ の意味)	<	小なり	<=	小なりイコール (≤ の意味)

「ならば:」「そうでなければ:」の末尾にあるコロンの、その次の行頭の空白(「インデント」という)は必要であるし、複数の命令を書く場合には空白の数を揃えなくては行けない。「Tab」のキーで空白4つが挿入され、「Backspace」で左側の空白4つを削除するようになっているので、適宜利用すること。

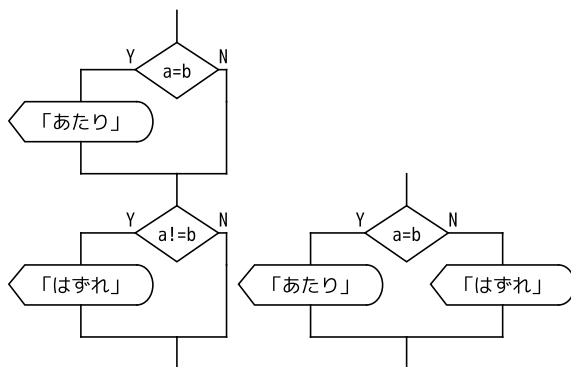
#### □ 例題:数あてゲーム

1桁の乱数(1~9)を発生させ、ユーザが入力した数値がそれと一致したら「あたり」、違っていたら「はずれ」と表示するプログラムを作りなさい(確認のために乱数の値と自分が入力した値も表示すると良い)。

乱数とはランダムに数を発生させる仕組みであり、ゲームや占いには欠かせない(もちろん遊びでない目的にも使われる)。PyPENではrandom(n)(nは自然数)で0以上n以下の(整数の)乱数が得られる。たとえばこのプログラムを何度も実行して、0~10の値が表示されることを確認してほしい。

random(10)を表示する

では、1から9の乱数を作るにはどうしたらいいだろう。random(9)では0~9になってしまうから...



◇注意 このプログラムを作る場合、左図の左側のアルゴリズムでも正しく動作はするが、こういう作り方は良くない。判定が2回になってしまって効率が悪いし、もし当たりの条件を変えることになったら2箇所を修正しなくては行けないからだ。うっかり片方だけ直してしまうと、見つけにくいバグを作ってしまうことになる(直すのは人間だから、こういうミスはよく起きる)。だから右側のようにするのが良い。入力支援ボタンの「もし」ではなく「もし~そうでなければ」を使う。

#### □ 課題:数あてゲーム(発展)

上の数あてゲームを、違っていたときにユーザが入力した数値が乱数の値よりも大きければ「大きい」、小さければ「小さい」と表示するように改造しなさい。

◇複雑な条件 条件を複数組み合わせることもできる。その場合には、たとえば「a>3 または b>3」のように「または」や「かつ」を用いる(\*4)。

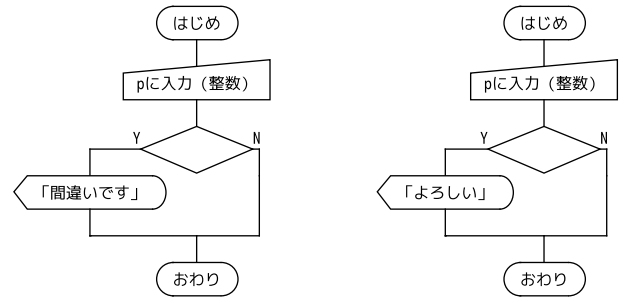
数学では  $1 < x < 3$  というような表し方ができるが、PyPENでは  $1 < x < 3$  という表現はできない。これは  $1 < x$  と  $x < 3$  がともに成り立つということだから「 $1 < x$  かつ  $x < 3$ 」と表す。同様に、 $x < 1, 3 < x$  も「 $x < 1$  または  $3 < x$ 」と表す。

同じことではあるが、例えば  $x = 1, 3$  を「 $x=1$  または  $3$ 」と書いては行けない。「3」では条件にならないからだ。この場合は面倒でも「 $x=1$  または  $x=3$ 」と書く。

(\*4) 否定を表す「でない」もあるのだが、この授業では扱わない。たとえば「 $a=0$ でない」なら「 $a!=0$ 」, 「 $a>3$ でない」なら「 $a<=3$ 」のように書けるからだ。

□ 例題:点数入力

テストの点数は 0 点以上 100 点以下である。この条件に合わない (0 より小さいか、100 より大きい) 数値が入力されたときには「間違いです」と表示するプログラムを作る。どのような条件にすればよいか。

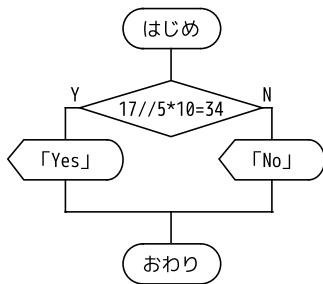


□ 例題:点数入力 (続)

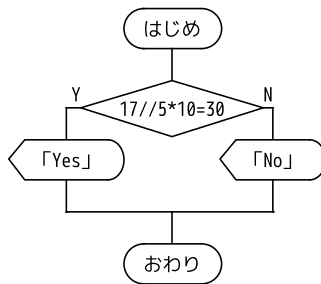
今度は正しい数値が入力されたときに「よろしい」と表示するようにしたい。どのような条件にすればよいか。

□ 問題 3

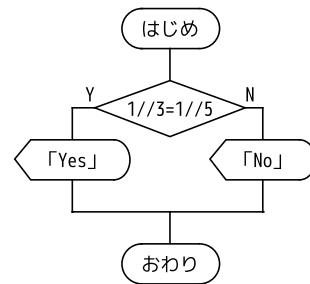
次のアルゴリズムで「Yes」「No」どちらが表示されるか考えなさい。その後で、実際にプログラムを作って確かめなさい。



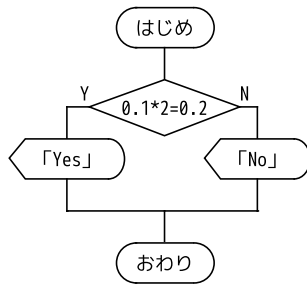
問 1



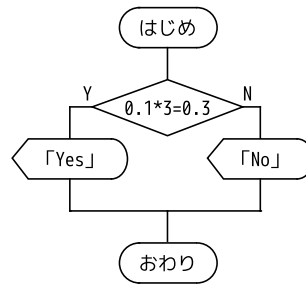
問 2



問 3



問 4



問 5

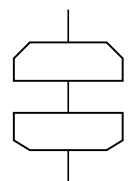
◇問 4, 問 5 について : 十進法の 0.1 は有限小数だが、二進法だと無限小数になり、コンピュータ内部の計算では有限の適当な桁数で打ち切ってしまう。そのため、最後の桁には誤差が含まれるので、このような結果になってしまった。問 4 と問 5 のどちらが「Yes」かなどということはどうでもいいのであって、次のことを覚えておいてほしい。

- 実数 (小数) の計算には誤差が含まれる。
- 数値が一致するかどうかの判断を実数 (小数) としてはいけない。  
差の絶対値が十分小さいかどうかで判断するのがよい。
- 整数なら、一致するかどうかの判断は正確にできる。

4.4.3 繰り返し

「繰り返し」はある条件が満たされるまで (あるいは満たしている間) 同じ処理を繰り返すというものである。フローチャートでは図のような記号で表し、上端か下端に繰り返しの終了または継続に関する条件を書く。本当は JIS で決まっている正式な書き方があるのだが、この授業では (規格の上では不正確ではあるが) 直観的にわかりやすいと思われる表記を用いる。

繰り返しに関しては内容が多いので、節を改めて説明する。



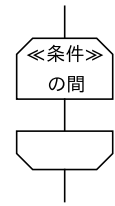


## 第 4 章第 5 節 繰り返し

### 4.5.1 ~の間繰り返す

次の構文で繰り返しが表現できる。見ての通り、上端にきたときに繰り返しを続ける（繰り返しの中にある命令を実行する）か、終了するかを決めるというものだ。

```
《条件》の間繰り返す：  
《命令》
```



◇注意 次の 2 つの例題は、入力する整数を記憶する変数のほかに、繰り返しのための変数が 1 つ必要になる。こういったことは課題の文章には表れないので、自分で判断して追加しなくてはならない。

#### □ 例題：カウントアップ

整数を 1 つ入力し、1 からその整数までカウントアップするプログラムを作りなさい。

#### □ 例題：約数

整数を 1 つ入力し、その整数の約数を小さい順にすべて表示するプログラムを作りなさい。

#### □ 課題：ユークリッドの互除法

2 つの整数の最大公約数を求めるときに、ユークリッドの互除法という方法が用いられる。その手順は次の通りだ。

1.  $a, b$  をその 2 数とする。
2.  $b$  が 0 でない間、以下の (a)~(c) を繰り返す
  - (a)  $c$  に  $a \% b$  を代入する。
  - (b)  $a$  に  $b$  を代入する。
  - (c)  $b$  に  $c$  を代入する。
3.  $a$  が最大公約数である。

これを用いて、整数を 2 つ入力し、その最大公約数を表示するプログラムを作って提出しなさい。

なお、最初は簡単な値 (35 と 21 とか) で正しい答えが出ることを確認すること。うまくいったら 839835391 と 527215277 の最大公約数を求めてみなさい (7 桁の数になる)。

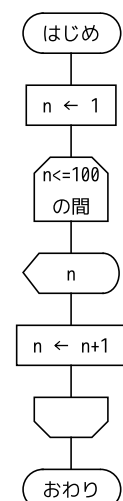
◇大事な余談 一つ前の例題で作った「約数をすべて求めるプログラム」を使えば公約数を求めることもできるが、このような大きい数だとすごく時間がかかる。実際、素因数分解は「時間がかかる」処理であり、2 学期に習った RSA 暗号は「時間がかかる」からまともな時間では解けないと考えられている。だからもし画期的に速い素因数分解のアルゴリズムが開発されたら、RSA 暗号は役に立たなくなるだろう。

### 4.5.2 カウントアップ、カウントダウン

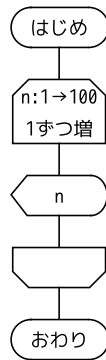
1 から 100 までの整数を小さい順に表示するプログラムは次のようになる。

```
n ← 1  
n <= 100 の間繰り返す：  
  n を表示する  
  n ← n+1
```

しかし、ある範囲の数を順に扱うケースはよくあるので、PyPEN ではこれを簡単に記述する方法が与えられている。それが次のフローチャート (\*5) とプログラムだ。入力支援ボタンでは「増やしながらか」「減らしながらか」となっている。



(\*5) この表記も正式なものではないが、直感的にわかりやすいのでこの授業ではこれを用いる。



n を 1 から 100 まで 1 ずつ増やしながらか繰り返す：  
n を表示する

□ 例題:2桁の奇数

2桁の奇数を小さい順に表示するプログラムを作りなさい。

□ 例題:カウントアップ

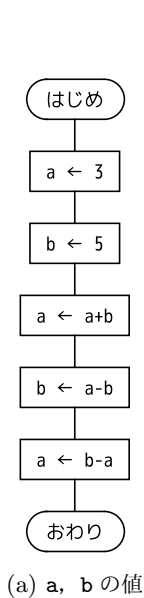
整数を1つ入力し、1から入力した整数までカウントアップするプログラムを「増やしながらか」を使って作りなさい。なお、ここでは変数を2つ用意する。入力した整数（つまりカウントアップのゴール）を表すものと、カウントアップしていく数を表すものを別に扱うためだ（次の課題でも同様）。

□ 課題:約数

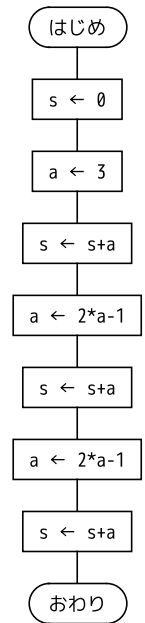
整数を1つ入力し、その整数の約数を小さい順にすべて表示するプログラムを作って提出しなさい。前は「～の間繰り返す」を使ってこれを作ったが、今回は「増やしながらか」を使って作ること。

□ 問題4

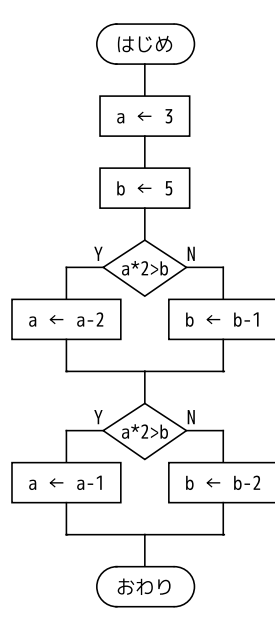
次の各アルゴリズムが完了した後、各変数の値はいくつになっているか。



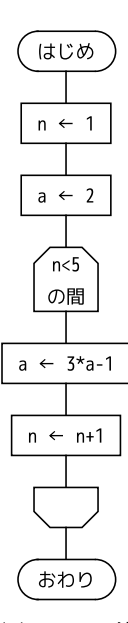
(a) a, b の値



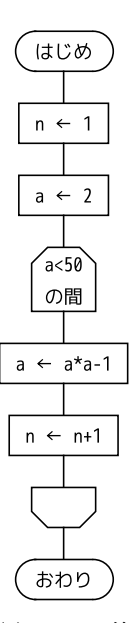
(b) a, s の値



(c) a, b の値



(d) a, n の値



(e) a, n の値

正解は (a)  $a = -5, b = 3$  (b)  $a = 9, s = 17$  (c)  $a = 1, b = 3$  (d)  $a = 122, n = 5$  (e)  $a = 63, n = 4$

## 第4章第6節 文法まとめ

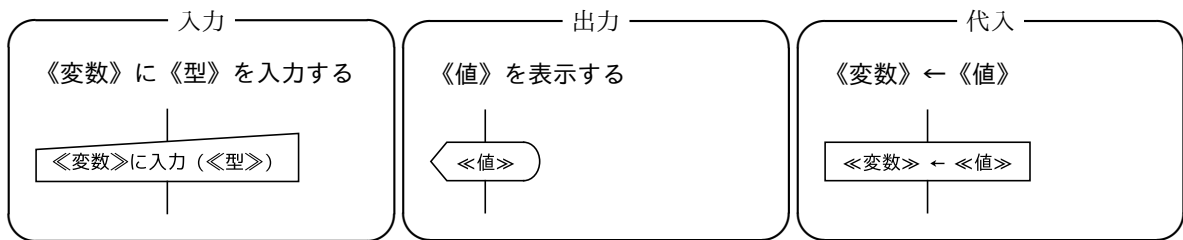
PyPENの文法のうち、この授業で必要となるものをまとめておく。

◇値の型 値には、整数、実数、文字列、真偽の型がある。文字列は「」か"で囲む。

◇計算 +, -, \*, /, //, %が使える。\*は掛け算, /は余りを出さない割り算の商(結果は実数になる), //は余りを出す割り算の商(商の整数部分)を表し, %は余りを出す割り算の余りを表す。

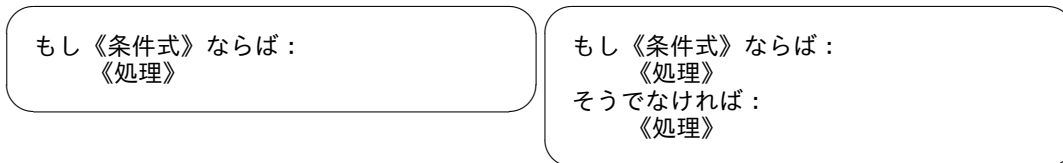
◇条件 =, !=, >, <, >=, <=によって二つの値を比較する(!=は≠, >=は≥, <=は≤をそれぞれ表す)。条件と条件を‘かつ’や‘または’で組み合わせることもできる。

◇入力・出力・代入 出力するときに、複数の値や文字列を‘と’でつなぐことができる(例:「xは」とxを表示する)。

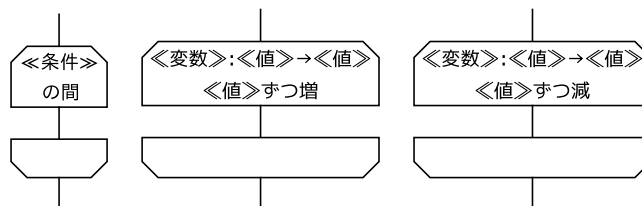
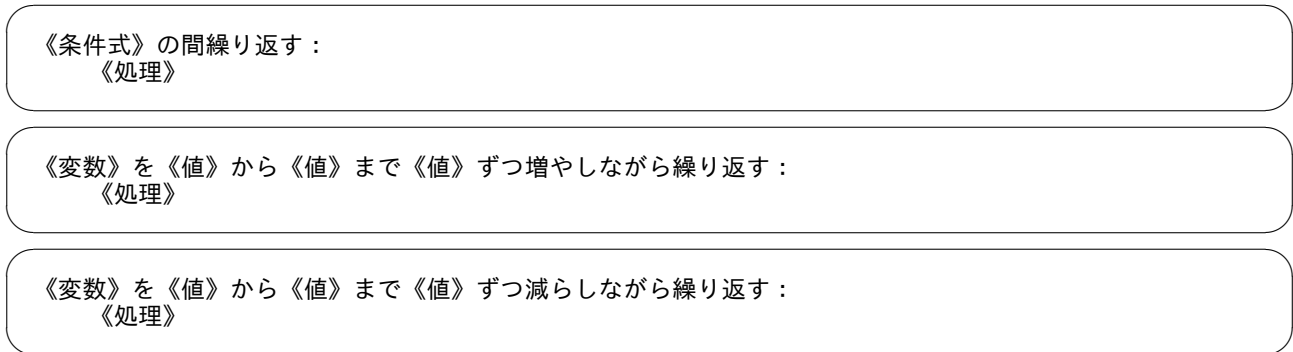


入力の《型》は「整数」「実数」「文字列」「真偽」のどれかである。

◇選択 条件が満たされるときと、そうでないときで行う処理を変える。



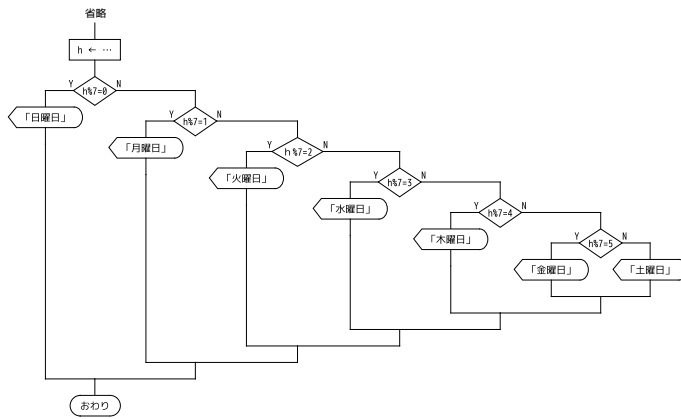
◇繰り返し 一定の条件のもとで同じ処理を繰り返す。



## 第4章第7節 配列

### 4.7.1 配列

前に、Zellerの公式を使って、日付を入力すると曜日を0~6の数字で表示するプログラムを作った。これを「日曜日」「月曜日」...「土曜日」と表示したい。次のようなアルゴリズムでもできるが、決してスマートな方法とはいえない。



```
h ← ...
もし h%7=0 ならば：
    「日曜日」を表示する
そうでなければ：
    もし h%7=1 ならば：
        「月曜日」を表示する
    そうでなければ：
        もし h%7=2 ならば：
            「火曜日」を表示する
        そうでなければ：
            もし h%7=3 ならば：
                「水曜日」を表示する
            そうでなければ：
                もし h%7=4 ならば：
                    「木曜日」を表示する
                そうでなければ：
                    もし h%7=5 ならば：
                        「金曜日」を表示する
                    そうでなければ：
                        「土曜日」を表示する
```

PyPENでは番号付きの変数である配列を使うことができる(番号は0から始まる)。これを使うことによって、プログラムがすっきりすることがよくある。たとえばこの曜日のプログラムであれば、

```
a[0] ← 「日曜日」
a[1] ← 「月曜日」
a[2] ← 「火曜日」
a[3] ← 「水曜日」
a[4] ← 「木曜日」
a[5] ← 「金曜日」
a[6] ← 「土曜日」
(中略)
a[h%7] を表示する
```

とすればよい。a[0]~a[6]に「日曜日」~「土曜日」を代入するのが多少面倒ではあるが、それでも最初に書いた「もし」の連続に比べれば、行数は半分以下だ。

実は次のようにまとめて代入することもできる。[ ]とコンマを忘れないように(カッコは[ ]でも{ }でもよい)。

```
a ← [「日曜日」, 「月曜日」, 「火曜日」, 「水曜日」, 「木曜日」, 「金曜日」, 「土曜日」]
```

たとえば、40人のクラスで点数入力をするなら次のようにすればいい。

```
k を 0 から 39 まで 1 ずつ増やしながら繰り返す：
    a[k] を入力する
```

### 4.7.2 モデル化とシミュレーション

サイコロはそれぞれの目が $\frac{1}{6}$ の確率で出る。だったら60回サイコロを振れば、それぞれの目が10回ずつ出るだろうか。60回くらいなら実際に試してみてもいいのだが、後で600回、6000回ならどうなるかということについても実験してみたい。そうなるとう作業で数えるのは大変だ。そこで、サイコロの実物を使うのではなく、プログラムでその状況を作り出して実験してみるとしよう。このように、実物を使うことが困難であるようなときに、代用品で同じような状況を作って試してみることをシミュレーション(\*6)という。

(\*6)「シミュレーション」という間違いは非常に恥ずかしいのでしないように。

シミュレーションは必ずしもコンピュータを使うものばかりではない。部屋の模様替えをするときに、家具の配置を適当な縮尺の方眼紙の上で考えるのもシミュレーションだし、車や飛行機の開発現場で部品の模型を風洞に入れて空気の流れを計測するのもそうだ。また、物を使うものばかりではなく、計算で行なうものもある（これについてはコンピュータが活躍する場面も多い）。その意味では数学の文章題や理科の計算問題は、実はシミュレーションの一種でもある。

シミュレーションを行なうためには、問題となる状況から必要な要素を抽出してモデル化を行なう。この場合だと「サイコロは6つの目が等確率で出る」「前に出た目は次に出る目に影響を及ぼさない」「回数を重ねても各目の確率は変化しない」などといったことになるだろう。

#### □ 課題：サイコロのシミュレーション

サイコロを60回振って各目が何回ずつ出るかを調べるプログラムを作りなさい。プログラムを簡単にするために、サイコロは0~5の目が出るものとする。

アルゴリズムは次のようになるだろう。2.では繰り返し用の変数が1つ必要になることに注意しよう。60回繰り返すためには「cを1から60まで1ずつ増やしながらかえり返す：」というようにすればいい。

1.  $a[0] \sim a[5]$  の値をすべて0にする。具体的には「 $a \leftarrow [0,0,0,0,0,0]$ 」とすればいい。
2. 次の (a)(b) を60回繰り返す。
  - (a) 0~5の乱数を  $b$  に代入する。
  - (b)  $a[b]$  の値を1増やす。
3.  $a[0] \sim a[5]$  の値を順に表示する（あるいは「 $a$ を表示する」でまとめて表示する）。

## 第4章第8節 集計

データの集計をするプログラムをいろいろ作ってみよう。集計というからにはたくさんの数値を扱いたいのだが、それを毎回入力するのも、代入をたくさん繰り返すのも面倒だ。そこで、ここでは決まった値を先に代入しておくことにしよう。

```
a ← [59, 34, 24, 99, 38, 1, 29, 89, 81, 6]
```

この節では集計をするプログラムをこの続きとして作っていく。

### 4.8.1 合計の計算

$a[0] \sim a[9]$  の合計を求めるプログラムを考える。もちろん

```
a[0]+a[1]+a[2]+a[3]+a[4]+a[5]+a[6]+a[7]+a[8]+a[9] を表示する
```

とすれば一応できるが、これだとデータが10個でないときにはプログラムを書きなおさなくてはならないから、こんなやり方では話にならない。繰り返しを使って合計を求めるのが適切なやり方だ。ここでは合計を  $s$  で表すことにする。まず  $s$  の値を0にし、順次  $a[0]$ ,  $a[1]$ , ...,  $a[9]$  を加算していく。これをそのまま書くと

```
s ← 0
s ← s + a[0]
s ← s + a[1]
(中略)
s ← s + a[9]
```

となるが、これをそのまま入力するのではなく、繰り返しを使おう。

$s \leftarrow s + a[k]$  を、 $k$  を0~9の値にして行なえばいい。つまり「 $k$ を0から9まで1ずつ増やしながらか『 $s \leftarrow s + a[k]$ 』を繰り返す」わけだ<sup>(\*)</sup>。また最後に結果の表示を追加しよう（この節の後の処理でも同じ）。

(\*) 数学だと  $\sum_{k=0}^9 a_k$  のような式になる。

#### 4.8.2 最大値, 最小値

$a[0] \sim a[9]$  の最大値を求める方法を考えよう。ここでは最大値を  $m$  としよう。アルゴリズムは次のようになる。

1.  $m$  の初期値を適切に定める。
2. 次の (a) を,  $k$  を  $0 \sim 9$  の値にして行なう。  
(a)  $m < a[k]$  なら  $m \leftarrow a[k]$ 。

やはり「 $k$  を  $0 \sim 9$  の値にして…」というのが繰り返した。最小値は (a) の不等号を逆向きにすればいい。ところで、1. で定める初期値はどのようにするといいたろうか。

#### 4.8.3 ソート

値を順番に並べることをソートという。 $a[0] \sim a[9]$  を小さい順に並べるプログラムを作ってみよう。なお、小さい順を「昇順」、大きい順を「降順」という。アルゴリズムの基本的な考えは次のようになる。

1.  $a[0] \sim a[9]$  の最小値が  $a[0]$  になるように値を入れ替える。
2.  $a[1] \sim a[9]$  の最小値が  $a[1]$  になるように値を入れ替える。
3.  $a[2] \sim a[9]$  の最小値が  $a[2]$  になるように値を入れ替える。  
(以下、最後まで繰り返す)

これをもう少し詳しくいうと次のようになるだろう。

1.  $a[1] \sim a[9]$  の中で  $a[0]$  よりも小さいものがあれば、それと  $a[0]$  を入れ替える。
2.  $a[2] \sim a[9]$  の中で  $a[1]$  よりも小さいものがあれば、それと  $a[1]$  を入れ替える。
3.  $a[3] \sim a[9]$  の中で  $a[2]$  よりも小さいものがあれば、それと  $a[2]$  を入れ替える。  
(以下、最後まで繰り返す)

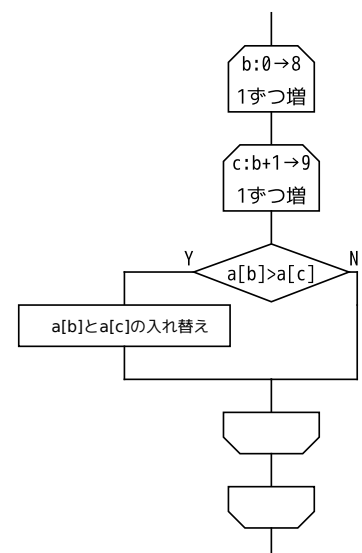
1., 2., ... をまとめて言うと次のようになる (右上図の「入れ替え」と書かれている部分がこれ)。 $b$  を  $0 \sim 8$  にしてこれを行えばいい:

$a[b+1] \sim a[9]$  の中で  $a[b]$  よりも小さいものがあれば、それと  $a[b]$  を入れ替える。

この「入れ替え」をさらに言い換えると次のようになる。

1.  $c$  を  $b+1 \sim 9$  にして, (a) を行なう。  
(a) もし  $a[b] > a[c]$  なら,  $a[b]$  と  $a[c]$  を入れ替える。

では、実際にこれをプログラムにしてみよう。なお、変数の入れ替えにはもう一つ変数を用意する必要があることを、ずっと前に学習した (5 ページの問題)。



## 第4章第9節 グラフィック

### 4.9.1 グラフィックの基本

PyPENにはグラフィックの命令があるので、それを使ってみよう。プログラム入力欄で右クリックして出てくるメニューの「各種処理」から「描画領域開く」を選んで、幅と高さを指定すると、実行したときにここで指定したサイズの描画用ウィンドウが開かれる。その他の描画用命令もすべて「各種処理」にあるので自由に探してみたい。座標は左上が(0,0)で、 $x$ 軸が右、 $y$ 軸が下に伸びていると考えればいい。

次のプログラムでは縦の直線の色を微妙に変えつつ引かれることで、緑から青へのグラデーションになる。

```
描画領域開く (255,255)
x を 0 から 255 まで 1 ずつ増やしながら繰り返す :
  線色設定 (0,255-x,x)
  線描画 (x,0,x,255)
```

次のプログラムでは、自由な位置に自由な大きさの円を100個描く。「線色設定」を使えば色を変えるなどの発展も考えられるだろう。その場合は、色の赤、青、緑の数字はそれぞれ0~255の範囲にすること。

```
描画領域開く (200,200)
i を 1 から 100 まで 1 ずつ増やしながら繰り返す :
  x ← random(200)
  y ← random(200)
  r ← random(50)
  円描画 (x,y,r)
```

これを応用して、いろいろな模様を描いてみよう。気に入った模様ができたなら画像を右クリックすれば、名前をつけてファイルに保存することもできる。

### 4.9.2 サンプルプログラム

次のプログラムは同心円の色を徐々に変えてグラデーションを表現してみたものである。

```
描画領域開く (200,200)
i を 100 から 1 まで 1 ずつ減らしながら繰り返す :
  線色設定 (2*i,0,200-2*i)
  円描画 (100,100,i)
```

次のプログラムは直線だけをたくさん描画しているのだが、曲線が現れる。これがなんという曲線であるかは...

```
描画領域開く (200,200)
i を 1 から 200 まで 1 ずつ増やしながら繰り返す :
  線色設定 (0,i,200-i)
  線描画 (i,0,0,200-i)
```

1=1 は常に成り立つので、「1=1の間」というのは無限ループになる。止めるには「リセット」のボタンを押す。

```
描画領域開く (200, 200)
1=1 の間繰り返す :
  i を 1 から 100 まで 1 ずつ増やしながら繰り返す :
    x ← random(200)
    y ← random(200)
    w ← random(50)+1
    h ← random(30)+1
    矩形描画 (x, y, w, h)
  描画領域全消去 ()
```

ボールが壁や床ではねかえる様子を表現してみるとこんな感じだろうか（はねかえりの処理はかなりいい加減である）。なお、`t` の範囲を広げる場合は、左端ではねかえりも考慮する必要がある。

```
描画領域開く (200,200)
vx ← 1.0
vy ← 5.0
x ← 0.0
y ← 200.0
t を 0 から 400 まで 1 ずつ増やしながら繰り返す :
  円塗描画 (x,y,1)
  x ← x+vx
  y ← y-vy
  もし x>=200 かつ vx>0 ならば :
    vx ← -0.8*vx
  もし y>=200 かつ vy<0 ならば :
    vy ← -0.8*vy
  vy ← vy-0.1
```

次のプログラムは配列 `a` に代入した値を棒グラフで表す。`length(a)` は配列 `a` の要素の個数を表す。最初 `max` に 0 でなく 1 を代入しているのは、`a` の値が全部 0 であってもエラーにならないようにするためである。

```
h ← 400
w ← 400
a ← [100,240,220,150,180]
l ← length(a)
max ← 1
i を 0 から l-1 まで 1 ずつ増やしながら繰り返す :
  もし a[i]>max ならば :
    max ← a[i]
描画領域開く (w,h)
x ← w*0.1
y ← h*0.9
w ← w*0.8
h ← h*0.8
wb ← w/l
線描画 (x,y,x+w,y)
線描画 (x,y,x,y-h)
文字サイズ設定 (16)
i を 0 から l-1 まで 1 ずつ増やしながら繰り返す :
  塗色設定 (0,255,0)
  hb ← a[i]*h/max
  矩形塗描画 (x+wb*0.1,y-hb,wb*0.8,hb)
  塗色設定 (0,0,0)
  文字描画 (a[i],x+wb*0.1,y-hb-2)
  文字描画 (i+1+「番目」,x+wb*0.1,y+16)
  x ← x+wb
```

## □ 最後の課題

自分で考えたグラフィックのプログラムを 1 つ提出しなさい。

## 最後に

この授業では PyPEN という見慣れないプログラミング環境で勉強したが、グラフィックの部分以外はほとんどそのまま Python に書き直すことができる。「もし～ならば:」は `if ~:`、「～の間繰り返す」は `while ~:`、「`a` を `b` から `c` まで `d` ずつ増やしながら繰り返す:」は `for a in range(b,c+1,d):` といった具合だ。代入に `=` を使ったり、逆に値の比較には `=` でなく `==` を使ったりという違いはあるが、基本的な構文はほとんど同じになるように作ったつもりだ。Python は最近人工知能などの分野で脚光を浴びている言語なので、将来使うことになる人も多いただろう。

C などの言語でも、インデントされている部分が `{ }` で囲まれるだけの違いと思えばそんなに大差はない。この授業の内容が、そういった本格的な言語への足がかりになれば幸いである。



名古屋高等学校情報科が著作者である本テキストは、クリエイティブ・コモンズ 表示 4.0 国際 ライセンスで提供する。