

問題の解決と 処理手順の自動化

名古屋高等学校情報科

<pre>1 a=1 2 bを1から100まで1ずつ増やしながらか 3 ... a=a*b 4 ... 表示する(bと"!="とa) 5 . 6 .</pre>	<pre>1!=1 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</pre>								
<input type="button" value="入力 (整数)"/>	<input type="button" value="入力 (実数)"/>	<input type="button" value="入力 (文字列)"/>	<input type="button" value="入力 (真偽)"/>	<input type="button" value="出力"/>	<input type="button" value="改行無出力"/>				
<input type="button" value="代入"/>	<input "="" type="button" value="+="/>	<input "="" type="button" value="-="/>	<input "="" type="button" value="*="/>	<input "="" type="button" value="/="/>	<input "="" type="button" value="//="/>	<input "="" type="button" value="%="/>	<input type="button" value("&=""/>	<input "="" type="button" value=" ="/>	<input "="" type="button" value="^="/>
<input type="button" value="もし"/>	<input type="button" value="もし~そうでなければ"/>	<input type="button" value="そうでなくもし"/>							
<input type="button" value="~の間"/>	<input type="button" value="増やしながらか"/>	<input type="button" value="減らしながらか"/>	<input type="button" value="配列の要素について"/>	<input type="button" value="繰り返しを抜ける"/>					
<input type="button" value="配列に追加"/>	<input type="button" value="配列に連結"/>	<input type="button" value="関数"/>	<input type="button" value="手続き"/>	<input type="button" value="値を返す"/>	<input type="button" value="手続きを抜ける"/>				

第1章 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。

この單元では、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。

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いやすいし、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。

1.2 PyPEN

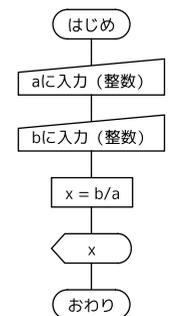
アルゴリズムはプログラムではないから、そのままではコンピュータで実行できない。そこで**プログラミング言語**を用いて、アルゴリズムを元にプログラムを作らなくてはいけない。世の中にはたくさんのプログラミング言語があるが、この授業では大学入学共通テストの「情報関係基礎」で用いられているDNCLという言語を、Python風アレンジしたPyPENを用いる。

ブラウザを起動して、授業用サイトか <https://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2章 PyPENの使い方

```
aに整数を入力する
bに整数を入力する
x=b/a
表示する(x)
```

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。上にプログラム、右にフローチャートを示した。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。



フローチャート

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。

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**が使える。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。

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。なお、《型》は「整数」「実数」「文字列」「真偽」のどれかである。



```
《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などを書き換える。

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。

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

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。



表示する(《値》)

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。

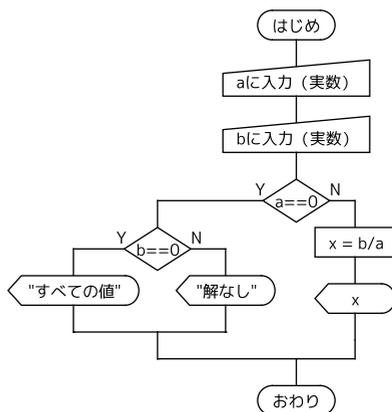
2.6 デバッグ

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。たとえば $3.5x = 7$ や $0x = 2$ を解こうとすると、どちらもエラーになってしまう。

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のようになる。「サンプル1」に入っているので、呼

び出して実際に実行してみよう。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して、どこがどう対応しているかくらいは見てほしい。

```
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：
        表示する("すべての値")
    そうでなければ：
        表示する("解なし")
    そうでなければ：
        x=b/a
    表示する(x)
```



2.7 自動採点

プログラムを作ったら、必ず自分で実行して正しい動作をするかを確かめてもらいたい。しかし問題の読み間違いや思い違いをしていたら、間違っただけを正しいと誤解してしまうこともある。そこでこのテキストのいくつかの演習については、自動採点できる仕組みを用意した。左上にある「問題選択」で「Q1:一次方程式」を選ぶと問題が表示され、「採点」ボタンを押すといくつかの入力に対して正しい出力が得られるかを採点してくれる。

2.8 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全文選択 (Ctrl+A), コピーして (Ctrl+C) 貼り付けられたい (Ctrl+V)。右クリックメニューでのコピーは使えないので、これらのキーを覚えること。

2.9 Pythonでの実行

PyPENで作ったプログラムは、Pythonのプログラムに変換できる。「コー

ド→Python」 ボタンを押すと、出力欄にPythonに変換されたコードが表示される（出力欄を横に広げると読める）。[Google Colaboratory](#)を開いてノートブックを新規作成し、その「コード」の枠にPythonのコードを貼り付けて実行ボタンを押せばいい。

第3章 値・演算子

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いい。

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。

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)

// 余りを出さず割り算の商 (商の整数部分)

÷ 同上

% 余りを出さず割り算の余り

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。ただし、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。/と//の使い分けはPythonと同じになっている。次のプログラムで確認してみよう (「サンプル2」を参照)。

表示する(14/3)
表示する(14//3)
表示する(14%3)
表示する(1/3*3)
表示する(1//3*3)
表示する(1*3//3)

[課題：偶数奇数の判定]

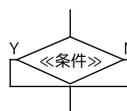
課題の作成・提出に必要な手順を述べる。次のプログラムを作ろう：

整数を1つ入力し、それが偶数なら"even", 奇数なら"odd"と表示する。

プログラムの作成は次の手順で行なう。

1. 変数…どのような変数が必要か
2. アルゴリズム…どのような順番で処理をしたらいいか
3. 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
4. 動作確認…実行して、正しい結果が得られるかどうか確認する（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。



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。なお、行末の「:」や、そこで改行したときの行頭の空白は大事なので消してはいけない。

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が、「等しい」は $=$ 、「等しくない」は \neq で表すことに気をつける）。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 $n\%2$ を用いる。

完成したら「問題選択」の「Q2:偶数・奇数」で答え合わせをして、うまく出来ていたらプログラムをコピーして提出しなさい。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。提出画面で貼り付け (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。

[2022年1月の曜日]

2022年の1月は土曜日から始まる。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日が土曜日だから1を入力したら土曜日を表す6が出力されるように、2日が日曜日だから2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？

[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。せっかくだから任意の日の曜日が分かるようなものを作ろう。そこで、曜日を計算する方法としてみ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。

なお、このプログラムは後の授業で、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、完成したらダウンロードして残しておくこと。

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 $0, 1, \dots, 6$ であれば、その日は日曜、月曜、 \dots 、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 (床関数という。数学では $[x]$ という表記で習い、「ガウスの記号」と呼ぶ)。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 (たとえば ab は1つの変数と解釈されるので、 $a*b$ とは別物である)。

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。また、「問題選択」の「Q3:曜日の計算」でも確認できる。

2022年1月1日	土曜(6)	2024年1月1日	月曜(1)
2022年3月1日	火曜(2)	2024年3月1日	金曜(5)
2022年12月31日	土曜(6)	1966年12月20日	火曜(2)

これなら自分の生まれた日が何曜日であるかもわかるだろう (正しいかどうか、何らかの方法で確認してみよう)。

完成したら保存しておこう。手順は次の通り：

1. 「Save」 ボタン右の欄にファイル名にしたい文字列を入力する
2. 「Save」 ボタンを押す

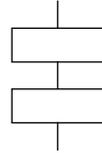
たとえば「曜日」と入力したら、「曜日.PyPEN」という名前のファイルがダウンロードされる。何回か後の授業で、これを「Load」のボタンでアップロードすることになる。

第4章 構造化プログラミング

世の中にある複雑なプログラム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。

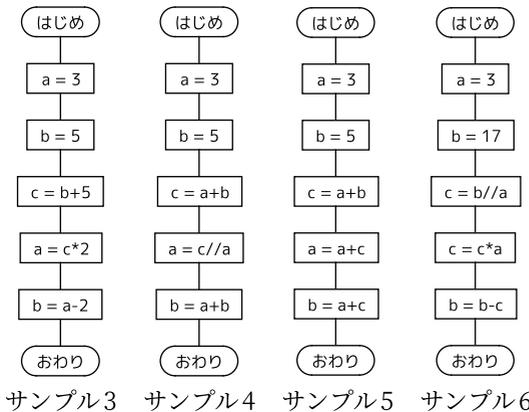
4.1 順次

順次というのは、上から順番に処理を実行していくだけである。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]

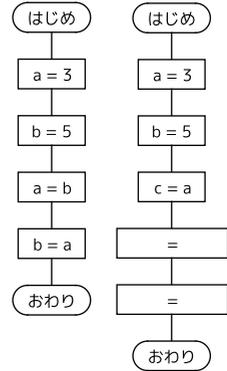
次の各アルゴリズムが終了したときの、各変数の値を答えよ。終わったら「サンプル3」～「サンプル6」で確認しなさい。



【問題2】

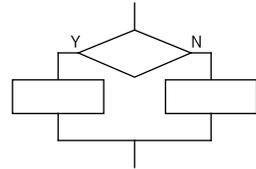
右図の左側はaとbの値を入れ替えようとしたものだが、失敗している（「サンプル7」で確認しなさい）。実は変数の値の入れ替えをするには、もう一つ変数を用意しなくてはならない。aとbの値の入れ替えが成功するように、右側の空欄を埋めなさい（「サンプル8」）。

これは変数の値を入れ替えるときの常套手段である。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。



4.2 選択

選択は、ある条件が満たされるときとそうでないときに別の処理を行なうというものである。すでに偶数・奇数のプログラムで使ったが、あらためてここでまとめておく。



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、どちらが大きいとかいうようにYes/Noで判断できるものである。比較で用いられる演算子は次のとおりである：

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

「ならば：」「そうでなければ：」の末尾にあるコロンや、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いけない。「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:数当てゲーム]

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"Hit"、違っていたら"Boo"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）。

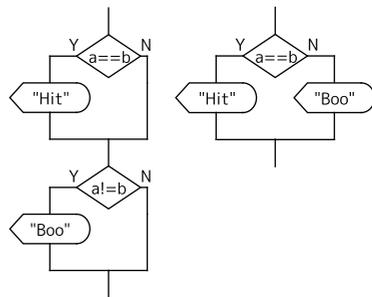
乱数とはランダムに数を発生させる仕組みであり、ゲームや占いには欠かせない（もちろん遊びでない目的にも使われる）。PyPENでは`random(n)`（`n`は自然数）で0以上`n`以下の（整数の）乱数が得られる。たとえばこのプログラムを何度も実行して、0～10の値が表示されることを確認してほしい。

```
表示する(random(10))
```

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いけないからだ。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～そうであれば」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"big", 小さければ"small"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「 $a > 3$ or $b > 3$ 」のように「and」や「or」を用いる（否定を表す「not」もあるのだが、この授業では扱わない。たとえば「not $a == 0$ 」なら「 $a != 0$ 」, 「not $a > 3$ 」なら「 $a <= 3$ 」のように書けるからだ）。

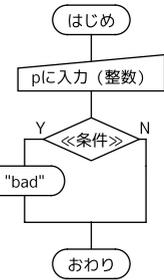
たとえば二次不等式の解は数学では $x < 1, 3 < x$ と書くが、プログラミングではコンマをそのような意味で使うことはできないので「 $x < 1$ or $3 < x$ 」と表す。同様に $1 < x < 3$ も「 $1 < x$ and $x < 3$ 」となる。同じことだが $x = 1, 3$ を「 $x == 1, 3$ 」や「 $x == 1$ or 3 」と書いてはいけない。正しいのは「 $x == 1$ or $x == 3$ 」である。

🔪 注意

ただし PyPEN では「 $a < b < c == d$ 」のような条件を「 $a < b$ and $b < c$ and $c == d$ 」のように解釈するので、 $1 < x < 3$ を「 $1 < x < 3$ 」と書くこともできる。これは Python の文法を真似たものであるが、このような書き方が許されるプログラミング言語は少ないので注意してほしい。たとえば PHP ではエラーになる。C ではエラーにならないが、数学の $1 < x < 3$ とは別の解釈をするのでさらに厄介である。

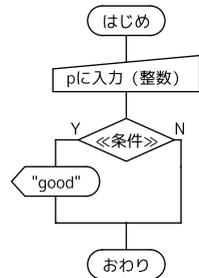
[例題:点数入力]

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"bad"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(「問題選択」「Q4:点数入力」で答え合わせしなさい)。



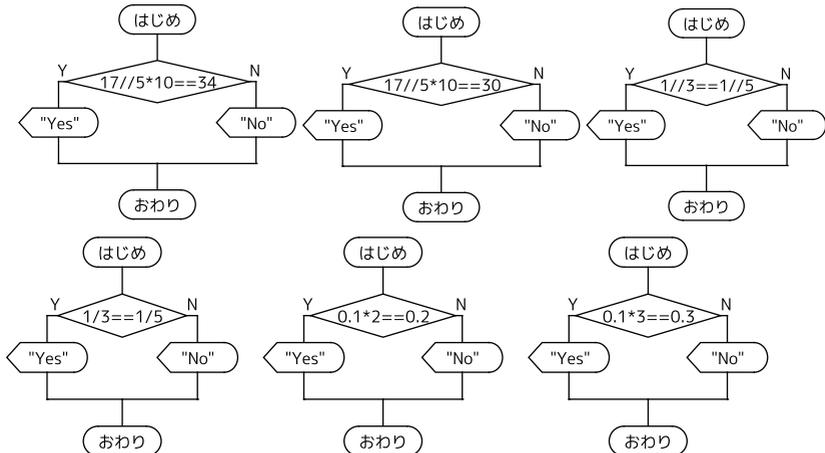
[例題:点数入力(続)]

今度は正しい数値が入力されたときに"good"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(「問題選択」「Q5:点数入力(続)」で答え合わせしなさい)。



[問題3]

次の各アルゴリズムで"Yes"と"No"どちらが表示されるか考えなさい。その後で「サンプル9」に条件式を書きこんで確かめなさい。



🔑最後の2問について

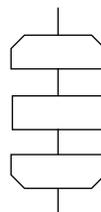
十進法の0.1は有限小数だが二進法だと無限小数になる、コンピュータ内部

の計算では無限の計算は不可能で、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て、このような結果になってしまった。最後の2間はどちらが"Yes"かなどということはどうでもいい。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。だから一致するかどうかの判断を実数（小数）でするのは要注意。
- 整数なら、一致するかどうかの判断は正確にできる。

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5章 繰り返し

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、上端に来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（1から順に数えるもの）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いけない。

[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（「問題選択」「Q6:カウントアップ」で確認する）。

[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（「問題選択」「Q7:約数」で確認する）。

[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（「問題選択」「Q8:最大公約数」で確認する）。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。

🔪 大事な余談

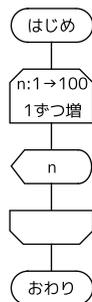
一つ前の例題で作った「約数をすべて求めるプログラム」を使えば公約数を求めることもできるが、このような大きい数だとすごく時間がかかる。実際、素因数分解は「時間がかかる」処理であり、2学期に扱ったRSA暗号は「時間がかかる」からまともな時間では解けないと考えられている。だからもし画期的に速い素因数分解のアルゴリズムが開発されたら、RSA暗号は役に立たなくなるだろう。

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 :
  表示する(n)
  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（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）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。



nを1から100まで1ずつ増やしながらか繰り返す：
表示する(n)

では、これを使って以下の例題や課題をやってみよう。

[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（「問題選択」の「Q9:2桁の奇数」で確認しなさい）。

[例題]

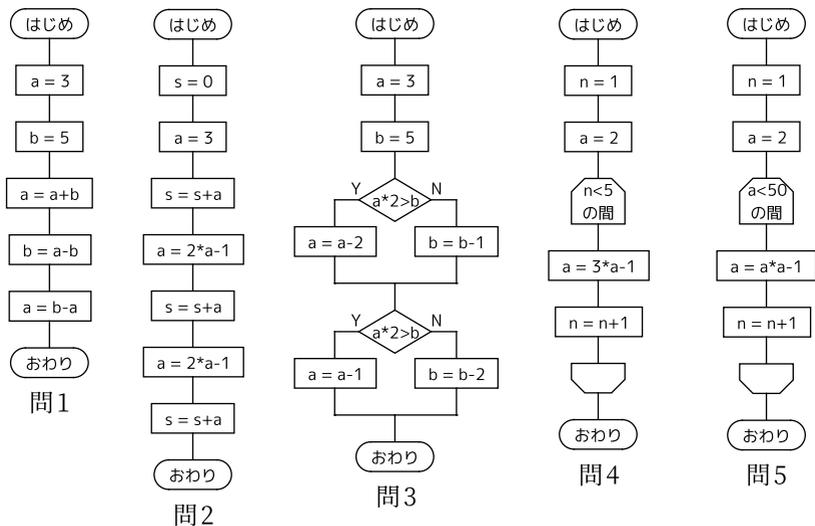
整数を1つ入力し、1から入力した整数までカウントアップするプログラムを「増やしながらか」を使って作りなさい。

なお、ここでも変数は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]

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]



上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。ちなみに正解は

問1: $a = -5, b = 3$

問2: $a = 9, s = 17$

問3: $a = 1, b = 3$

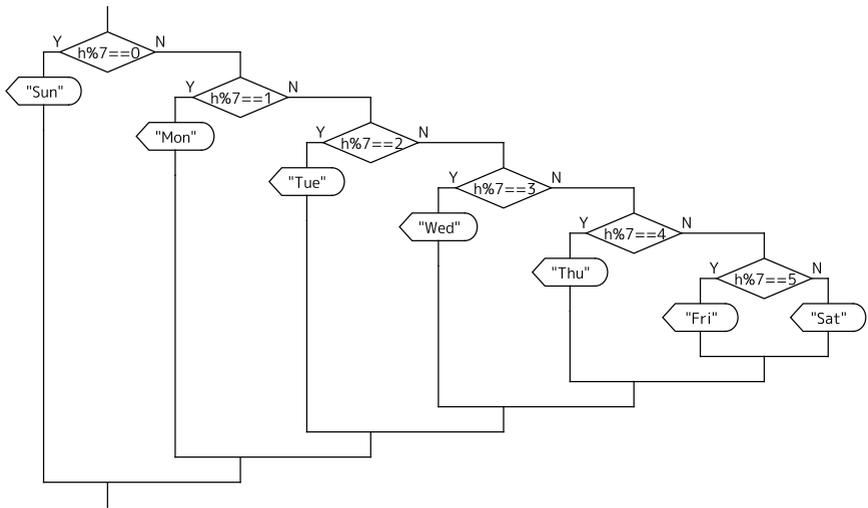
問4: $a = 122, n = 5$

問5: $a = 63, n = 4$

第6章 配列

6.1 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「Sun」「Mon」…「Sat」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 :

表示する("Sun")

そうでなければ :

もし $h\%7==1$ ならば :

表示する("Mon")

そうでなければ :

もし $h\%7==2$ ならば :

表示する("Tue")

そうでなければ :

もし $h\%7==3$ ならば :

表示する("Wed")

そうでなければ :

```
もし h%7==4 ならば :
    表示する("Thu")
そうでなければ :
    もし h%7==5 ならば :
        表示する("Fri")
    そうでなければ :
        表示する("Sat")
```

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 = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
(中略)
表示する(a[h%7])
```

とすればよい。ここではa[0],a[1],…に順に"Sun","Mon",…を代入している。

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([0]*40というのは、0を40回繰り返した要素を持つ配列を表す)。合計点などの計算も、配列を使うととても楽だ（これは次の項で扱う）。

```
a=[0]*40
kを0から39まで1ずつ増やしながら繰り返す :
    a[k]に整数を入力する
```

[例題：おみくじ]

乱数を用いて「大吉」「吉」「小吉」「凶」「大凶」のどれかがでるようなおみくじのプログラムを作りなさい。余裕があれば、出るメッセージの種類を増やしてみなさい。

6.2 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。そうすると手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**という。「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

やしながら繰り返す：」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「 $a=[0,0,0,0,0,0]$ 」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「表示する(a)」でまとめて表示する）。

6.3 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。この節では集計をするプログラムを「サンプル10」の続きとして作っていく。

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

6.4 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

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

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いけないから、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 $a[0], a[1], \dots, a[9]$ 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。 $s = s + a[k]$ 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『 $s = s + a[k]$ 』を繰り返す」わけだ。数学だと $\sum_{k=0}^9 a_k$ のような式になるが、まったく同じイメージである。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）。

6.5 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. 次の作業を、 k を0~9の値にして行なう。

$$(a) m < a[k] \text{ なら } m = a[k].$$

やはり「 k を0~9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だろうか。

6.6 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。

(略)

4. $a[8] \sim a[9]$ の最小値が $a[8]$ になるように値を入れ替える。

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、 $a[0]$ と入れ替える。
2. $a[2] \sim a[9]$ の中で $a[1]$ よりも小さいものがあれば、 $a[1]$ と入れ替える。
3. $a[3] \sim a[9]$ の中で $a[2]$ よりも小さいものがあれば、 $a[2]$ と入れ替える。

(略)

4. $a[9] \sim a[9]$ の中で $a[8]$ よりも小さいものがあれば、 $a[8]$ と入れ替える。

これを繰り返しを使って言うと次のようになる。

b を0から8まで1ずつ増やしながらか繰り返す：

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

この「 $a[b+1] \sim a[9]$ の中で…入れ替える」をさらに言い換えると次のようになる。

c を $b+1$ から9まで1ずつ増やしながらか繰り返す：

もし $a[c] < a[b]$ ならば、 $a[b]$ 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。

第7章 グラフィック

7.1 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してほしい。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が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。画像を右クリックすれば画像をファイルに保存できる。

7.2 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。

```
描画領域開く(200,200)
iを1から200まで1ずつ増やしながらか繰り返す：
  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」は無限ループになる。「リセット」のボタンを押すと止まる。

```
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ずつ増やしながらか繰り返す：
    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 and vx>0ならば：
        vx=-0.8*vx
    もしy>=200 and 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。

第8章 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 ~:, そんな風に、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だろう。

Cなどの言語でも、PythonやPyPENで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。

本テキストは名古屋高等学校情報科が著作者であり、[クリエイティブ・コモンズ表示4.0国際ライセンス](https://creativecommons.org/licenses/by/4.0/)で提供するものとする。

