まねておぼえるプログラミングのいろは（後編）
―ビットくんをいじりたおせ―

基般

坂本一憲（国立情報学研究所）

前編との関係性

本稿は，「まねておぼえるプログラミングのいろは」の後編にあたる．読者に記事中のソースコードをそのまま入力してもらう写経を通して，プログラミングを体験していただくことを想定している．前編にてプログラミング環境や入力方法を説明しているので，適宜，前編も参照いただきたい．

ループで同じ作業をまとめる

本章では，生卵が1個しか入らないフライパンで目玉焼きを10個作る手順を例に取り，プログラミングにおけるくり返しについて説明する．

前編で説明した目玉焼きを1個作る手順は，以下の3ステップからなる．

1. フライパンを火にかける．
2. 生卵を割って，フライパンに入れる．
3. 焼けた目玉焼きを皿に移す．

上記の手順をもとに考えると，目玉焼きを10個作る手順は以下の21ステップからなる．

1. フライパンを火にかける．
2. 1個目の生卵を割ってフライパンに入れる．
3. 焼けた1個目の目玉焼きを皿に移す．
4. 2個目の生卵を割ってフライパンに入れる．
5. 焼けた2個目の目玉焼きを皿に移す．
6. …（省略）…
20. 10個目の生卵を割ってフライパンに入れる．
21. 焼けた10個目の目玉焼きを皿に移す．

上記の手順で，10個の目玉焼きを作ることはできるが，ステップの数が多すぎて手順の記述が面倒であるし，読み手にとっても分かりにくい．実際に，本稿では6ステップ目から19ステップ目までを省略する形で記載している．もし，読者が上記の手順をほかの人に説明する場合は，多くの読者が本稿と同様に，くり返されるステップについて省略したいと考えるだろう．

そこで，特定のステップをくり返すという記述を活用して，手順を簡略化してみよう．

1. フライパンを火にかける．
2. 3および4ステップ目を10回くり返す．
3. 生卵を割って，フライパンに入れる．
4. 焼けた目玉焼きを皿に移す．

上記の手順では，2ステップ目で3および4ステップ目を10回くり返すことを指示している．くり返しを使うことで，目玉焼きを10個作るという手順の内容を一切変更せずに，21ステップからなる手順を，わずか4ステップの手順で表現することができた．

さらに，10個の目玉焼きを作る手順に加えて，100個の目玉焼きを作る手順を考えてみよう．先ほどのような目玉焼きそれぞれの作り方をすべて指示する手順の場合は，101ステップからなる手順に変更する必要がある．一方で，くり返しを使った手順の場合は，2ステップ目の「10回」と記載した個所を「100回」に置き換えるだけで良い．このように，作る目玉焼きの個数を簡単に変えることができる．

以上のように，くり返しを使えば手順の表現に必要なステップ数を減らすことができ，さらに，くり返す回数を簡単に変更できるようになる．プログラミングの世界では，くり返しのことをループと呼び，プログラムを構成する重要な要素の1つとなっている．

前編と同様に，ビットくんのTwitterページの画面を変更するJavaScriptプログラムを紹介する．

図 -1 で先頭から 5 番目のツイートを赤文字に変えるプログラムを，図 -2 で図 -1 のプログラムを実行した結果の画面を示す．

```
$('li.stream-item').eq(0).css('color', 'red')
$('li.stream-item').eq(1).css('color', 'red')
$('li.stream-item').eq(2).css('color', 'red')
$('li.stream-item').eq(3).css('color', 'red')
$('li.stream-item').eq(4).css('color', 'red')
```

図 -1　先頭から 5 番目までのツイートを赤文字に変えるプログラム

図 -2　図 -1 のプログラムの実行結果

図 -1 のプログラムを Web ブラウザの右側に表示されている「Console」に入力することで ☆1，図 -2 のように 1 番目から 5 番目までの 5 個のツイートを赤文字に変更できる．しかし，図 -1 のようなプログラムでは，赤文字に変更したいツイートの個数が増えると，プログラムを記述する手間が増えがちである．たとえば，100 個のツイートを赤文字に変更したい場合は，100 行のプログラムを入力する必要がある．

目玉焼きの例と同様に，くり返し（ループ）を使って図 -2 のプログラムを簡略化できる．図 -2 の各行はほぼ同じ内容で，「eq(0)」の 0 の部分に入力する数値だけが異なる．目玉焼きの例ではまったく同じステップをくり返していたが，今回は数値を変化させながらループする必要がある．JavaScript 言語では，for 文という命令を利用することで，数値を変化させながらループするプログラムを記述できる．

図 -3 で for 文を使って 1 番目から 5 番目ツイートを青文字に変更するプログラムを，図 -4 で図 -3 のプログラムを実行した結果の画面を示す．

☆1　Web ブラウザ上でプログラムを入力する方法の詳細は，前編を参照されたい．

```
for (i = 0; i < 5; i++) {
  $('li.stream-item').eq(i).css('color',
'blue')
}
```

図 -3　ループを使って先頭から 5 番目までのツイートを青文字に変えるプログラム

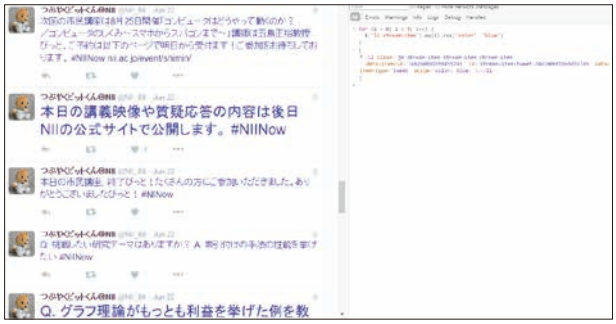

図 -4　図 -3 のプログラムの実行結果

図 -3 のプログラムを実行すると，図 -4 のように 5 個のツイートを青文字に変更できる．図 -2 のプログラムが 5 行であったのに対して，図 -3 のプログラムは 3 行（2 行目の折り返しを含むと 4 行）である．さらに，100 個のツイートを青文字に変更したい場合は，「i < 5」の部分を「i < 100」と書き換えるだけで良い．

「for (i = 0; i < 5; i++) { ... } 」は以下の動作を指示するプログラムであり，for 文がループを意味する命令であることが分かる．

1. 変数 i に数値 0 を代入する（i = 0;）．
2. ... のプログラムを実行する（{ ... }）．
3. 変数 i の数値を 1 増やす（i++）．
4. i < 5 が成り立つ場合は，2 番目のステップに戻り，そうでない場合は，終了する（i < 5）．

なお，変数 i の名前は必ずしも i である必要性はないが，整数を意味する英単語「integer」の頭文字である「i」を用いることが慣わしである．

図 -3 のプログラムでは，{} の中に「i 番目のツイートを青文字に変える」という命令が記述されている．したがって，図 -3 のプログラムは，変数 i の数値が 0 から 5 まで 1 ずつ変化しながら，「i 番目のツイートを青文字に変える」を実行して，変数 i の数値が 5 になったところで，「変数 i が 5 未満ではなくなったので（i < 5 が成り立たないので），ループを終了する」という動作をとる．

図 -5 で 2 番目から 4 番目のツイートを緑文字に変更するプログラムを，図 -6 で図 -5 のプログラムを実行した結果の画面を示す．

```
for (i = 1; i < 4; i++) {
  $('li.stream-item').eq(i).css('color',
'green')
}
```

図 -5　ループを使って 2 番目から 4 番目のツイートを緑文字に変えるプログラム

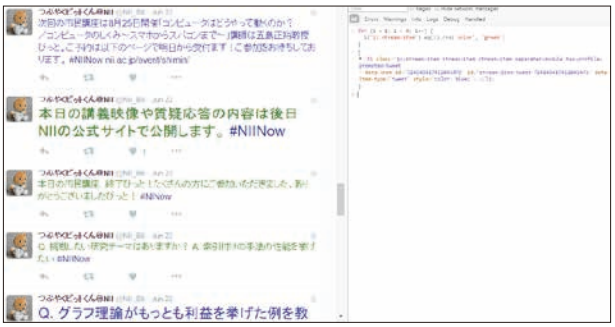

図 -6　図 -5 のプログラムの実行結果

図 -5 のプログラムを実行すると，図 -6 のように 3 個のツイートを緑文字にできる．人間にとっての「2 番目から 4 番目」は，コンピュータにとっての「1 番目から 3 番目」になる．図 -5 の for 文は，変数 i に 1 を代入して，4 未満（3 以下）である限りループを続けるという動作の指示になる．

以上のように，for 文を使うことで，目玉焼きの例と同様にプログラムの行数を減らすことができ，さらに，ループ回数を簡単に変更できる．

ところで，コンピュータの計算速度は非常に高速である．たとえ，1,000 個のツイートであっても，コンピュータは一瞬で文字の色を変えることができる．このコンピュータの性能を有効活用する上で，ループは非常に重要である．ループを使うことで，コンピュータに対して，人間には到底できないような回数でくり返す命令を指示できる．プログラミングを上手く使いこなすことで，コンピュータに難しい仕事を依頼して，私たち人間の生活をより豊かにすることができる．

条件分岐で動作を切り替える

私たちの日常生活において，将来に起こり得る状況を複数想像できるケースがよくある．たとえば，目玉焼きを作ろうとして卵を割ったときに，黄身が崩れずにフライパンに乗る場合と，黄身が崩れてフライパンに乗る場合の両方のケースを想像できる．目玉焼きを作る手順としては，黄身が崩れても目玉焼きを作るという手順でも良いのだが，黄身が崩れた場合は料理を変えて，スクランブルエッグを作るという手順の方が，好ましい場合があるだろう．今回は，黄身が崩れた場合はスクランブルエッグを作る手順を考えてみよう．

1. フライパンを火にかける．
2. 生卵を割って，フライパンに入れる．
3. 黄身が崩れなかった場合は 4 ステップ目へ，崩れた場合は 5 ステップ目へ移る．
4. 卵を焼いて目玉焼きを作り，お皿に移して料理を終える．
5. 卵を牛乳と混ぜてスクランブルエッグを作り，お皿に移して料理を終える．

上述の手順の 3 ステップ目では，黄身が崩れなかった場合と黄身が崩れた場合の 2 種類のケースで場合分けを行っている．このように状況に応じて手順の内容を変化させることで，さまざまな状況に対応した柔軟な手順を作ることができる．

上述の場合分けは，プログラミングの世界にも存在する概念であり，条件分岐と呼ばれる命令で実現される．条件を満たすか満たさないかに応じて，次に実行する命令を変えるために分岐する．そのため，条件分岐と呼ぶ．

さっそく，条件分岐を使ったプログラムを紹介する．ビットくんの Twitter ページには人気のツイートもあれば，そうでないツイートも存在する．ゆっくり Twitter ページを閲覧する時間がない場合は，人気のツイートを優先して読みたいと思うだろう．図 -7 で，「いいね」の数が 0 個の人気のないツイートを半透明にして，見えにくくするプログラムを，図 -8 で図 -7 のプログラムを実行した結果の画面を示す．

```
for (i = 0; i < 10; i++) {
  e = $('li.stream-item').eq(i)
  if (e.find('.IconTextContainer:last')
       .text() == 0) {
    e.css('opacity', 0.2)
  }
}
```

図 -7　人気のないツイートを半透明にするプログラム

図 -8　図 -7 のプログラムの実行結果

図 -7 のプログラムは，前章で説明したループを使って，上から 10 個のツイートに対して，「『いいね』の数が 0 個だったら，そのツイートを半透明にする」とコンピュータに指示する．

図 -7 の 3 行目では条件分岐の命令である if 文が使われている．if 文は「if(条件){ 条件が成立したときに実行する命令 }」と表記する．図 -7 の if 文の条件に該当する個所の記述内容は，「e.find('.IconTextContainer:last').text() == 0」だが，この条件の意味は「いいね」の数が 0 個であるかどうかを示す．条件が満たされたときに実行する命令は，「e.css('opacity', 0.2)」だが，この命令の意味はツイートの文字を半透明にすることである．なお，条件が満たされないときは何も行われないので，画面上のツイートは変化しない．

図 -7 のプログラムの理解を助けるために，図 -9 で，命令を日本語で書き換えた版を示す．

```
for （変数iを0から9まで1つずつ増やす）{
  変数eに変数i番目のツイートを代入する
  if （変数eのツイートの「いいね」が0個かどうか）{
    変数eのツイートを半透明にする
  }
}
```

図 -9　図 -7 のプログラムを日本語で書き換えた版

Twitter には，他人のツイートを引用するリツイートという機能がある．時間のないビットくんファンの読者であれば，ビットくん自身がつぶやいたツイートだけを優先して読みたいと思うだろう．

図 -10 で，前章で説明したループを使って，上から 10 個のツイートに対して，投稿者の名前に「ビット」と含まれていたら，ツイートを赤文字に変えて，そうでなければ，ツイートを緑文字に変えるというプログラムを示す．図 -11 で，図 -10 のプログラムを実行した結果を示す．

```
for (i = 0; i < 10; i++) {
  e = $('li.stream-item').eq(i)
  if (e.find('.fullname').text().indexOf('ビット') >= 0) {
    e.css('color', 'red')
  } else {
    e.css('color', 'green')
  }
}
```

図 -10　投稿者がビットくんかどうかに応じてツイートの文字色を変えるプログラム

図 -11　図 -10 のプログラムの実行結果

図 -10 の 3 行目では条件分岐の命令である if 文が使われている．if 文では，「if(条件){ 条件が成立したときに実行する命令 } else { 条件が成立しないときに実行する命令 }」と else 節を記述することで，条件が成立しないときの命令も指示できる．図 -10 の if 文の条件は，「e.find('.fullname').text().indexOf('ビット') >= 0」だが，この条件の意味は，投稿者の名前に「ビット」が含まれているかどうかを示す．条件が満たされたときに実行する命令は，「e.css('color', 'red')」で，そうでないときに実行する命令は，「e.css('color', 'green')」であり，それぞれ，ツイートを赤文字に変える，ツイートを緑文字に変えるという意味である．図 -10 のプ

ログラムの理解を助けるために，図-12で，命令を日本語で書き換えた版を示す．

```
for（変数iを0から9まで1つずつ増やす）{
  変数eに変数i番目のツイートを代入する
  if（変数eのツイートの投稿者の名前にビットが含まれているかどうか）{
    変数eのツイートを赤文字に変える
  } else {
    変数eのツイートを緑文字に変える
  }
}
```

図-12　図-10のプログラムを日本語で書き換えた版

以上のように，if文を使うことで条件分岐，つまり，場合分けを行うことができる．条件分岐を活用することで，コンピュータに対して，状況に応じて内容が変化する複雑な命令を指示することができる．たとえば，人間が気温に応じてリモコンを押して，「暑いから冷房をつける」や「寒いから暖房をつける」とエアコンを動かすように，コンピュータに対して，「もし室温が28度以上であれば，冷房をつける」や「もし室温が18度以下であれば，暖房をつける」のように，気温に応じてエアコンを動かす命令を指示できる．

人間が条件分岐の仕組みを活用することで，あたかもコンピュータが考えて行動するように，コンピュータに複雑な仕事を依頼できる．人間がプログラミングを使いこなせば，コンピュータは強力な相棒になる．

関数で簡単に命令を伝える

読者は誰かに目玉焼きの調理を依頼するときに，どうするだろうか？　毎回，依頼する人に対して，「まず，フライパンを火にかけて，次に，生卵を割って，フライパンに入れて，……」というように調理の手順をすべて説明するだろうか？　日本で育った人の多くは，目玉焼きという料理の内容も作り方の手順も知っている．だから，「目玉焼きを作ってください」とお願いすれば，それだけで調理の手順も伝わる．このように，私たちの日常生活では，料理名を伝えるだけで，ほかの人に料理を作る仕事を依頼できるので，簡単に依頼ができる．

ただし，「目玉焼きを作ってください」というお願いが通用するのは，依頼する人が目玉焼きの内容と作り方を知っているからである．もし，目玉焼きを知らない人であれば，手順の説明を省いてお願いすることは難しいだろう☆2．

料理名だけで調理の手順を説明するような仕組みは，プログラミングの世界にも存在する．それが，関数である．コンピュータに前もって関数という形で命令の内容を与えておくことで，関数の名前を与えるだけで，該当する命令を指示できる．

以降で，ビットくんのツイートの文字色を変える関数と，ツイートの「いいね」の個数を数える関数を紹介して，関数について説明をする．

図-13で，ビットくんのツイートを赤文字に変えるchangeColor関数を作るプログラムを示す．

```
function changeColor(n) {
  $('li.stream-item').eq(n).css('color',
'red')
}
```

図-13　changeColor関数を定義するプログラム

図-13の1行目の「function changeColor(n)」では，定義する関数の名前（changeColor）と関数が受け取るパラメータを代入する変数名（n）を指定している．なお，関数の名前を関数名，関数が受け取る値を代入する変数のことを仮引数，関数を実行するときに与える値を実引数と呼ぶ．2～3行目はn+1番目のツイートを赤文字に変えるという命令を意味する．このように，関数名と仮引数，関数の内容を指定することで，関数を作ることができる．なお，関数を作ることを，「関数を定義する」と呼び，定義した関数を使うことを，「関数を呼び出す」と呼ぶ．

図-13のプログラムをWebブラウザのConsoleに打ち込んで「Enter」キーを押しても，何も起こらないことに注意していただきたい．図-13のプログラムは関数を定義するだけである．料理の例で説明すると，料理の名前と調理の手順を教えただけであり，料理の名前を伝えて調理を依頼する（関数を呼び出す）ことを指示していないため，何も起こらない．

☆2　ただし，人間に限って言えば，レシピ本やインターネットで調理の手順を調べて，作ることもできる．

図 -14 と図 -15 で，changeColor 関数を使ってツイートの色を変えるプログラムを，図 -16 で，図 -14 と図 -15 両方のプログラムを実行した結果の図を示す．

```
changeColor(0)
```

図 -14　1 番目のツイートの色を変えるプログラム

```
changeColor(2)
```

図 - 15　3 番目のツイートの色を変えるプログラム

図 -16　図 -13 と 14 と 15 のプログラムを実行した結果

図 -14 のプログラムを実行すると，1 番目のツイートが赤色に，図 -15 のプログラムを実行すると，3 番目のツイートが赤色に変わる．

図 -14 と図 -15 のように，関数を呼び出す際に，括弧でくくってパラメータを渡すことができる．図 -14 では，数値 0 の実引数，図 -15 では，数値 2 の実引数を changeColor 関数に渡している．changeColor 関数は 1 つ目の実引数を仮引数である変数 n で受け取るという定義になっているため，図 -14 では変数 n に 0 を代入してから，図 -15 では変数 n に 2 を代入してから，changeColor 関数の内容である「`$('li.stream-item').eq(n).css('color', 'red')`」を実行する．

図 -17 で，先頭から n 番目のツイートに付けられた「いいね」の個数を数える countLike 関数を定義するプログラムを示す．

```
function countLike(n) {
  return Number($('li.stream-item').eq(n)
          .find('.IconTextContainer:last')
          .text());
}
```

図 -17　「いいね」の個数を数える countLike 関数を定義するプログラム

図 -17 のプログラムでは，関数名に countLike，実引数に変数 n，関数の内容に「`return Number($('li.stream-item').eq(n).find('.IconTextContainer:last').text());`」を指定して，関数を定義している．「`Number($('li.stream-item').eq(n).find('.IconTextContainer:last').text());`」は「いいね」の個数を数える命令である．「`return`」は関数が出力する結果を指定する命令である．なお，関数の出力する結果を「戻り値」と呼ぶ．「`a = countLike(0)`」などのように，左辺に変数，右辺に関数呼び出しを記述して，変数に代入する式を書くことで，関数の戻り値を変数に記憶させることができる．このように，関数呼び出しが，あたかも関数の戻り値に置き換わったかのようにプログラムが動作する．

図 -18, 19, 20 で，countLike 関数を呼び出す 3 種類のプログラムを，図 -21 で各プログラムを実行した結果を示す．

```
countLike(0)
```

図 -18　1 番目のツイートの「いいね」の個数を数えるプログラム

```
countLike(2)
```

図 -19　3 番目のツイートの「いいね」の個数を数えるプログラム

```
countLike(0) + countLike(2)
```

図 -20　1 番目と 3 番目のツイートの「いいね」の合計個数を数えるプログラム

図 -18，19，20 を実行すると，それぞれ，1 番目のツイートの「いいね」の個数，4 番目のツイートの「いいね」の個数，1 番目のツイートの「いいね」と 4 番目のツイートの「いいね」の合計個数が表示される．

図 - 21　図 -17, 18, 19, 20 のプログラムをすべて実行した結果

続いて，先頭から 10 番目までのツイートの「い

いね」の合計個数を数えるプログラムを考えてみよう．図 -22 のように，countLike 関数を 10 回呼び出して，表示された個数を足し合わせれば，合計個数を計算できる．

10 個のツイート程度であれば，図 -22 のアプローチでも良いが，1,000 個などツイート数が増えると，人間がコンピュータに命令することも大変である．本稿で説明したループを関数呼び出しと組み合わせることで，ツイートの「いいね」の合計数を計算するプログラムを簡単に書ける．図 -23 でループを使ったプログラムを示す．

```
countLike(0)
countLike(1)
……
countLike(9)
```

図 -22　countLike 関数を 10 回呼び出すプログラム

```
sum = 0
for (i = 0; i < 10; i++) {
  sum += countLike(i)
}
```

図 -23　ループで countLike 関数を 10 回呼び出すプログラム

図 -23 の 1 行目で，変数 sum に数値 0 を代入している．続いて，2 ～ 4 行目で，1 番目から 10 番目のツイートについて，それぞれ「いいね」の個数を数えて，変数 sum に足し合わせている．

図 -23 のプログラムを実行したときの様子を考えてみよう．まず，ループの実行前の変数 sum を 0 に初期化する．次に，1 番目のツイートの「いいね」の個数を数えて，変数 sum に足し合わせて，変数 sum に 1 番目のツイートの「いいね」の個数を代入する．続いて，1 番目のツイートの「いいね」の個数を数えて，変数 sum に足し合わせて，変数 sum に 1 番目と 2 番目のツイートの「いいね」の合計個数が代入する．以上の手順を，10 番目のツイートまで繰り返すことで，1 番目から 10 番目のツイートの「いいね」の合計個数を計算できる．

以上のように，前もって関数を定義しておき，関数を呼び出すことで，複雑な命令を何度も指示したり，複雑な命令を組み合わせたより複雑な命令を指示したりできる．また，仮引数と実引数を活用することで，たとえば，1 番目のツイートを赤文字に変える仕事と，3 番目のツイートを赤文字に変える仕事のように，ほとんど同じだが微妙に異なる命令を簡単に指示できる．

本稿では，自身で定義した関数を呼び出すプログラムを紹介したが，ほかの人が定義した関数を呼び出すこともできる．ほかの人が定義した関数を使うことで，自分で関数を定義する手間を省くことができる．逆に，ほかの人にとっても便利な関数の定義を作成および公開することで，ほかの人がコンピュータに仕事を依頼することを助けることができる．

近年，インターネットの発達に伴い，作ったプログラムを皆で共有するオープンソース・ソフトウェアの文化が広まりつつある．会ったことも話したこともない世界中の誰かと，手を取り合い協力しながらプログラミングできる時代となった．関数という仕組みを使って，世界中の人とプログラムの部品を共有することで，創造的で巨大なプログラムの開発が容易になりつつある．ぜひ，読者には本稿をきっかけとして，ほかの人の関数を活用してプログラムを作ったり，自身が作成した関数の定義をほかの人に公開したりして，より深くプログラミングの世界を楽しんでいただきたい．

前編・後編の振り返り

本稿では目玉焼きを調理する例を通して，プログラミングの基礎的な概念について説明した．前編では，命令の順序について，後編では，くり返しと条件分岐，関数について取り上げた．本稿を通じて，プログラミングは面白く，思ったよりは難しくなく，日常生活の役に立ちそうだと思っていただければ幸いである．本稿が今後のプログラミング学習のきっかけとなることを祈っている．

(2016 年 5 月 9 日受付)

坂本一憲（正会員）■ exkazuu@nii.ac.jp
国立情報学研究所 助教／さきがけ研究員（兼任）．ソフトウェア工学やプログラミング教育の研究に携わる過程で，「やる気」に興味を持つ．現在，目標を達成するために，行動の継続を支援するソフトウェアの研究開発に従事している．