E-09

初学者がなぜループ処理でつまずくのか？
―文系女子大におけるプログラミング入門の経験から―
Difficulties in Learning the Loop Process for Beginners
―Case Study of the Introduction to Programming in Women's University―

永井 絵梨奈† 伊地知 咲希† 内田 奈津子††
Erina Nagai Saki Ijichi Natsuko Uchida

1. はじめに

日本経済再生本部が発表した日本再興戦略 2016 [1]において，IT 人材の育成を徹底するため小・中・高校でのプログラミング教育の必修化が発表された．2020 年より順次，小・中・高等学校の学習指導要領 [2]に含まれる．今後の文系学生は理系学生と同様に，プログラミング的思考を身につけなければならない．

本学では，情報の知識に関する講義が複数開かれており，企画型，座学型，プログラミング型などがある．本論文では，著者らが，2019 年度後期に履修したコーディングを含むプロジェクト型のプログラミング入門の講義を取り上げる．本授業は，筆者らを含めて履修者の殆どがプログラミング初学者であった．初回の講義では１年生が多く見られたが，PC 操作に不慣れな生徒が操作についていけない様子が見られ，実際に履修した学生の多くが３年生，次いで２年生，１年生と４年生という割合で構成された．

授業を通じて，多くの学生がループ構文の扱いに苦難していることに気がついた．本論文では，初学者のつまずきを学習者の声と成果物から分析し，初学者に向けたテキストの改善案を提案する．

2. 関連研究

2.1 つまずきの類型化

岡本らは， [3]において指導者の立場からプログラミング学習における初学者のつまずきを調査した．テキストの通りにコードを実行する写経型学習に係るつまずきを 2 つに類型化した．

場合 1　写経型学習を遂行する上で自立的に作業することができない場合

場合 2　写経型学習の過程から目的とする内容を学び取ることができない場合

さらに認知的負荷理論を用いて原因を分類した．

原因 1　学習課題を構成する情報の間に見られる関係性の複雑さ，その課題に対する学習者の熟達の程度

原因 2　学習内容を理解していくためには本質的には関係ない活動を強いることによって学習者に生じるもの

2.2 つまずきの各類型に対応する学習方略

岡本らは，こうしたつまずきの排除として「コンパイル手順やエラー発見の方法に関する手順」に関してテキスト本体とは別の冊子にまとめ，学習効果を得た．

岡本らの研究は理系，かつ指導者というプログラミングにおいて優位な立場の視点に限定されているが，本論文では，文系，かつ履修者の視点からより具体的に初学者のつまずきのプロセスを分析する点で異なる．本論文では，初学者に易しい学習法の提案を目指す．

3. プログラミング入門の状況

3.1 授業の流れと状況

著者らの受講した講義は，図１に示すように前半と後半で二分され，前半はテキストを用い，図形の生成を教材に，制御構造，メソッド等のプログラミングの基礎を学ぶ時間，後半はチームによる課題制作に充てられた．

回数	内容
1 回目	プログラミングの導入
2 回目	分岐と繰り返し
3 回目	制御構造と配列
4 回目	手続き・関数と抽象化
5 回目	２次元配列と画像
6 回目	プログラミングのまとめ
7 回目	グループ作成・課題とゴール設定
8 回目	アイデア出し・ブレスト・仕様決め
9 回目	デザイン・設計・分担・制作
10 回目	試作・中間ドキュメント作成
11 回目	中間発表・仕様見直し
12 回目	制作・単体テスト
13 回目	コードレビュー
14 回目	制作・統合テスト
15 回目	最終成果発表会 (まとめ)

図 1 15 回の授業の構成

1 回目の授業では，初めてコマンドプロンプトに触れて，感覚的に操作できるものでは無いことから難しそうと判断する学生や，指示通りにタイピングができず授業の進行に遅れる学生が見受けられた．文系初学者にとっては，コードを考えることだけではなく，入力すること，実行することを一緒に進めることも困難の要因になっている．

† フェリス女学院大学　国際交流学部
†† フェリス女学院大学　情報センター

2 回目の授業では，分岐，ループ構文を扱ったが，多くの文系学生は予習，復習の習慣が無く，1 回目の内容を十分習得していないため実際には 1 回目の復習をしながら進行する形となってしまった．

3 回目の授業は，制御構造を学んだ．

どの回も図形の生成を教材として進め，方眼紙を用いて擬似コードにあたる設計を行った．図２に，１回目のテキストで扱った例題を示す．

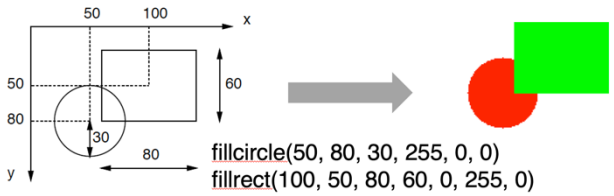

図2　設計図と例題の例

3.2 ループ構文の例題でのつまずき

ループ構文を学ぶとき，使用したテキストでは「ブレーキのかかるボール」という名称の例題（図 2）と初対面する．

```
def brakeball
  x = 20; y = 100; dx = 80
  while dx >= 5
    fillcircle(x, y, 10, 255, 0, 0); x = x + dx
    dx = dx * 0.7
  end
  writeimage("pict2.ppm")
end
```

図 2　テキストの例題

ここでは実際にコードを動かした時に出力される図画が示されており，この次に，下記の応用問題が提示される．

(1) 例題ではホボールが水平に動いていたが，斜めに動くようにしてみなさい．
(2) 水平方向にはブレーキがかかるが，縦方向には一定の速さで
(3) 動き続けるようにしてみなさい．
(4) 水平方向には一定の速さで動き続けるが，縦方向には徐々に速くなるようにして「落ちる」
(5) a～c と似ているが，画像の下端に来たら跳ね返るようにする．跳ね返った後どうするかは 自分で好きに決めてよい．
(6) その他自分がやってみたいと思う好きな「ボールの軌跡」を描く．

第一筆者は，本題を解く際，設計図の作成に困難を感じた．指示される x 座標，y 座標の動きを数式に直すとき，「ブレーキがかかる」や「跳ね返る」という動作が抽象的で，難解に思われた．

また「ブレーキのかかるボール」という題には，図画を描写する際，実際には関わりのない「速度の概念」が加わってくるために，描写コードの組み立てにつまずきやすいボトルネックになっているだろう．

3.3 改善案の提案

プログラミングにおいて設計図を起こすには，目標物を描写するコードの動きをしっかり理解している必要がある．しかし，プログラミング入門において，この例題は３週目である初学者，特に，著者らのような文系の学生にはまだ難易度が高い．さらに，文系学生は X，Y などの数学記号の理解に時間を要する傾向が見られる．

そのつまずきを解消するためには，計算式における文字の関係性の図解を用いたより詳しい解説があると良いだろう．

(1) まず X 点の位置を示すことで，図形のどこを頂点として計算式が組み立てられるのか解りやすくなる（図３）．

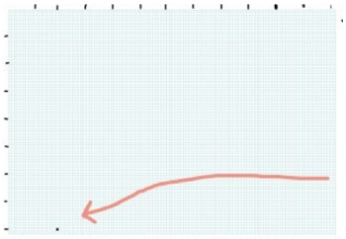

図 3　x の起点

(2) 次に D X の示す数値を図式化（図４）．dx を含んだ数式の変化が想像しやすくなる．

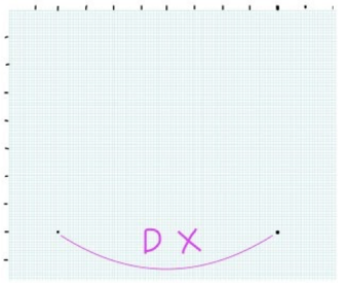

図 4　dx の位置

(3) While の条件が当てはまる間，計算が繰り返され，上書きされていく．x=x+dx なので，どんどん x は増える．dx=dx*0.7 なので，どんどん dx は減る．2 回目の描写(図 5)は x が増えたので右にずれる．3 回目の描写(図 6)は x が増える数(dx)が減ったので，ずれは小さくなる．これが dx が 5 より小さくなるまで続く．

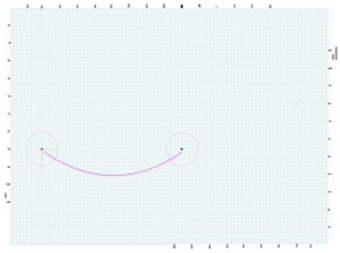

図 5　2 回目の描写

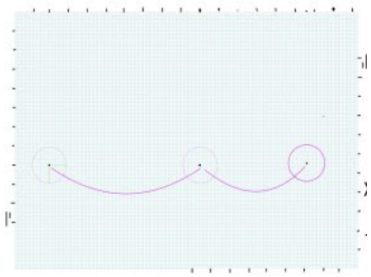

図 6 3 回目の描写

題目にも課題があり，例題「ブレーキのかかるボール」には，描写の計算式に関係のない速度の概念が混じり，理解を妨げる恐れがある．そのほか応用問題に見られる「斜めに」「跳ね返る」「落ちる」といった動作の題目は数学的な観点からみると抽象的で，読み手ごとに動き方の解釈が分かれる懸念を持つ．これらを「間隔を変えてオブジェクトを描写するメソッド」とモデル化して改めると，コードの働きがよく伝わり，応用する場面でのつまずきが減ると思われる．

3.4 式とつまずきの関係

ループ構文には，いくつか種類があり，条件づけの if 文，～の間を示す while 文，～回繰り返す times do 文を学んだ．

学生の進行状況を見ると，times do の活用に問題はなく，if，特に while 文の活用に悩んでいた．

Times do は「～回繰り返す」という単純な繰り返しの指示だ．対して while 文は「x が～以下の場合に～する」というように複雑な条件つきの指示になり，条件は数式で構成する．

文系学生は，数学への苦手意識から，数式の構築につまずき易い．

これまでの算数・数学の学習を振り返ると，算数はテキストが読めれば解けていたが，数学は活用する思考能力を要する．多くの文系学生が，算数は理解はしているが，数学的思考でコードを活用できてないという事がプログラミングに触れてわかった．活用するにはインプットの量が重要なことから，初学者には例題を多く提示し，そこから数学的思考で設計の手順を読み解くプロセスが必要だ．そうした訓練を経て，“わかる”から“できる”に変わる経験が大事であると考える．

4. チーム課題に見るループ構文のつまずき

4.1 つまずきの例

本制作では，ループ構文の活用が義務付けられ，各々「観覧車の回転」や「背景の空模様の変化」など多様な動作で使用された．しかし，ループ構文の活用につまずくグループも存在した．その一例であるグループ A のアニメーションを，図５に示す．

図 7 アニメーションの様子

パンダの頭上を肉まんが移動する様子を描こうとしたらしい．だが，「オブジェクトを横に移動する」単純な指示が上手に働かず，斜めに動いている．

4.2 つまずきと制作状況の関係

ここでは 4 人 1 組となってコードの合作を目指した．1 枚のアニメーションを描写することが目標で，役割分担や設計は学生の采配に委ねられた．グループごとに進行は異なり，下記はグループ A の進行である．

(1) 1 人が設計図を作成
(2) パーツごとに分解し，コーディングの担当を分配
(3) コードを合体させ，一本のプログラムにする

対して，アニメーションを予定通りに動かす事ができたグループ B にヒアリングを行った．進行を比較してみると設計図の作成方法(1)に違いが現れた．

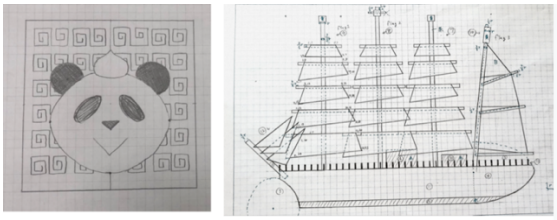

図 8 グループ A(左)、グループ B(右)の設計図

グループ B は全員で設計図の作成に取り組んだ．表現方法を相談しながら，オブジェクトごとに座標と X，Y の起点を明示した（図６右）．座標は，後から調整できるので誤差があっても大きな問題ではないが，起点はサイズ変更やアニメーションを行う際，動きを計算するために必要な情報である．

グループ A は，作図を一任したことで，コーディング上の再現方法について議論がなかった．それによりオブジェクトの起点を共有しないまま制作が進行してしまい，アニメーションをつける際，想定外の動きを示したのだ（図５）．

4.3 チーム課題の利点

チームで制作を行う中で，学生はループ構文を用いたグラデーションやアニメーションの表現ができた。例題を学んだ際に苦戦していたことから，書きたい絵の設計図→ストーリーが浮かぶ→式がイメージできる，という思考プロセスが重要であったのでは無いかと推測する．

また一人の力で再現できなくても，チームで協力が得られるため，前半の知識理解で終わらずに，活用できるところまで経験できる．この事により，反復練習，相互学習の効果が得られた．

以上のことから，チーム課題におけるループ構文の活用には，チームである利点を活かした学び合いや教え合いの姿勢が重要だと思われる．

その関係性を構築するキーの 1 つに，設計図がある．グループ A の設計図は，設計者と下請けという関係を構築した．グループ B の設計図は，全員が監修者となった．

前者は，製作中のコミュニケーションが下請けと設計者の間で 1 対 1 に留まってしまった。後者は，設計図から全員で意見し合うことで全員が当事者意識を持ち，制作中の議論やアドバイスが自由に行われた．

5. 改善提案

5.1 係数ループを用いた例題

3.2 で取り上げた例題「ブレーキのかかるボール」は，数式によって条件付けをする while 構文で制作されている．条件部分と指示部分の両部に計算式があることで，座標の変化と計算の回数が不規則になり，文系女子大においては多くの初学者がつまずく原因となった．算数の不得意な学生には，考える要素が過多であるため，例題の内容に工夫があると考える．

最初に取り組む例題では，変化する座標を 1 つに絞り，ループの動き方を説くことが重要であると考える．そのため，まずは構造が単純化された times do 構文(図 5 参照)を用いることを提案する．いわゆる係数ループだが，これはカウンターを用いて実行されるため，while 構文に比べると何回計算が繰り返されるかが明確である．

```
i = 0           # i はカウンタ
while i < n do  #  「n 未満の間」繰り返し
  ...           #   ここでループ内側の動作
  i = i + 1     # カウンタを 1 増やす
end
```

図 5 テキストの係数ループの説明

提案として，例題「球を１０個並べるメソッド」(図 6)は，　例題「ブレーキのかかるボール」の後に出てくる例題であるが，記号が最小限に抑えられており，文系学生も図と数字から感覚的にコードの仕組みを追う事ができる．そのため，順番を入れ替えて，こちらを先に学ぶことを提案する．

```
def fixedrepeat
  10.times do |i|
    fillcircle(20*(i+1), 40, 10, 255, 0, 0);
  end
  writeimage("pict2.ppm")
end
```

図 6 球を１０個並べる例題

また，応用問題では，係数ループを用いて，2 つの座標が変化する問題を出題すると段階的に理解を進める事ができるだろう．(応用問題の図) では y と r の値が変化する回数が 3 回に抑えられているため，図から数値の変化を逆算し易い．「ブレーキのかかるボール」のように変化の回数が多く，変化の計算が複雑な例題は，これらの次の段階で出題するとつまずきが減少すると思われる．

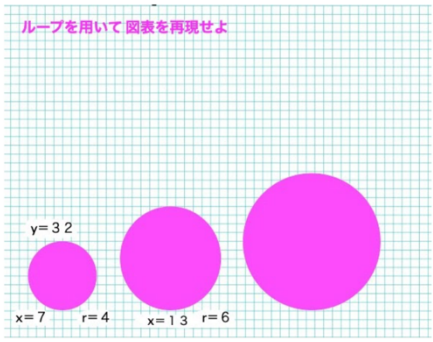

図 9 段々大きくするメソッド

5.2 グラデーションを用いた例題

ループ構文を理解する上で，グラデーションを用いた空を作成する方法を提案する．なぜなら，後半のプロジェクトにおいて，多くのチームが活用した実績と女子学生が色に興味関心があり，楽しみながら学べる教材になると考えるからである．この方法を用いることで，より分かりやすく且つ，自由に学習することが可能であると期待する．例として，第二筆者のチームで作成したものを図 7，図 8 で示す．

```
def sky
200.times do |i|
fillrect(400, 0+2*i, 800, 2, 242, 97, 0+i, 0.0);
end
writeimagef("sky.ppm")
end
```

図 7 ループを用いた空のプログラム

図 8 図 7 から生成された空の背景

図 7 で示すように，四角のみを用いてそれらを繰り返す．そして，色を少しずつ変えるように設定することで，図 8 のようなグラデーションを表現している．最小限の数字で表現することが可能であるため，この方法は基礎学習でのループ構文の理解には適切であると考える．また，ループを使用することによって，小さな情報の入力で一度に多くの情報を処理することが可能である．これらの情報を誤りなく，確実に処理できる制御構造やアルゴリズムの理解にも繋がった．

加えて，色の設定の際に，カラーコードの使い方を交えるとより応用出来ると考える．例えば図9の様に基準となるオレンジ色を指定し，入力することで夕焼けを表現することができる．

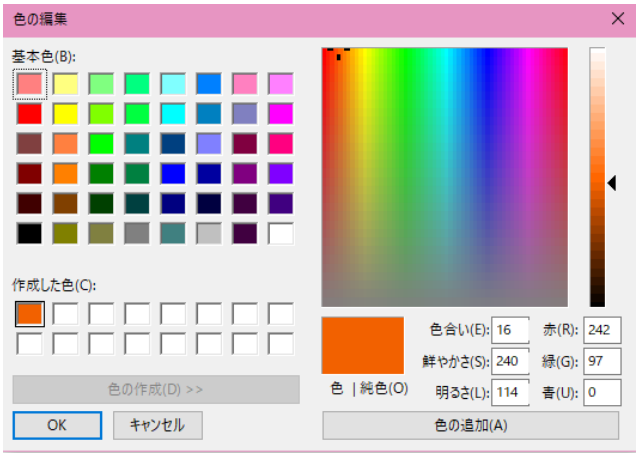

図 9 カラーコード

カラーコードで自分が表現したい色を選択すると，赤色(r)，緑色(g)，青色(b)がそれぞれ数字で表記されるため，r，g，b の数字を簡単に入力することが可能である．また，イメージする色の表現が容易であるため，海や草原など様々に応用が可能である．このように実際にあるものを表現することで，イメージしやすいため，より深い理解に繋がるだろう．また，前半で背景を作成することにより，後半のプロジェクトでのアニメーションやステッカー作成の際にも活用することができ，作業の軽減や時間短縮にも繋がる．

6. まとめ

本稿では，プログラミング入門を通して，ループ構文における初学者のつまずきを履修者の観点から分析し，つまずきの回避方法を提案した．文系学生にとって初めて触れるツールや知識が多いため，情報量を調節しながら学習を進める事がつまずきを減らすための要因である．5 で提案した例題 2 つは材料，条件をなるべくシンプルにする事で，理解を促している．

今後の展望として，初学者がその他のメソッドではどのようなつまずきをするのか，学びを通し調査・分析していく．

参考文献

[1] 首相官邸，第二次安倍内閣， “日本再興戦略 2016 —第 4 次産業革命に向けて—，” 2 June 2016. [オンライン]. Available: https://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/2016_zentaihombun.pdf. [アクセス日: 10 June 2020].

[2] 未来の学びコンソーシアム， 21 May 2019. [オンライン]. Available: https://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2019/05/21/1416331_001.pdf. [アクセス日: 10 June 2020].

[3] 岡 雅子， 喜多一， “プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察，” *パイデイア：滋賀大学教育学部附属教育実践総合センター紀要*， 2014-03-24.