

プログラミングの基礎的知識の理解深化を目指した 相互コメント付与システムの設計

布施 泉¹⁾, 平林 義治¹⁾, 山本 裕一¹⁾, 中原 敬広²⁾

1) 北海道大学 情報基盤センター

2) 合同会社 三玄舎

ifuse@iic.hokudai.ac.jp

Designing a mutual comment system aimed at deepening understanding of basic programming knowledge

Izumi Fuse¹⁾, Yoshiharu Hirabayashi¹⁾, Yuichi Yamamoto¹⁾, Takahiro Nakahara²⁾

1) Information Initiative Center, Hokkaido Univ.

2) Sangensha LLC

概要

本稿では、プログラミング授業における正解到達型の課題実施において、目標創出型の要素を学習者間の相互コメント付与により加えることを目指したシステムの実装とその試行実践について報告する。大学の一般教育における選択科目のプログラミング授業を対象とし、基礎的課題に関連した設問にコメントを求めることで、各学習者の知識レベルに応じたコメントが収集される。その結果を学習者が確認し、相互にコメントすることで、当該課題の学習内容をより深く理解させることが可能である。本実践は、2025年度の大学入学においては新旧学習指導要領を経た学生が混在することを想定し、知識レベルに幅がある学習者への適用可能性を視野に入れた試行でもある。

1 はじめに

新学習指導要領[1]を経た学生が2025年度に大学に入学する。初等中等教育段階でのプログラミング教育が強化されており、全学生が「プログラミング的思考」を習得して大学に入学することが期待される。しかし、学習者の入学時における知識やスキルの幅は大きいことが予想され、大学入学時におけるプログラミングに対するレディネスへの懸念もある[2]。また、旧学習指導要領による学生が混在することに対する配慮も必要となる。

著者らは、所属大学において、選択の一般教育科目として、2つのプログラミングの授業を継続して開講している（後期2単位）。プログラミング言語はPythonとRubyをそれぞれ取り上げ、指導が行き届くように履修上限を設定し、それぞれ70名、50名程度が履修する。1年生が多いが、1年の前期には、7週程Pythonを必修で学習している。後期の本授業は選択科目であり、プログラミングに興味を持った学生が履修するものの、その習熟度には幅がある。授業開始時に行う簡単なチェックテスト（前期に学習した中でも簡単な数値演算、

条件分岐、繰り返し、リスト処理等）では、平均正解率は8割5分程度であるが、4割～6割台の正解者が14%程度、満点が16%程度含まれる。

本授業では、前半に基礎課題、後半に応用課題を課す。前半は、前期の授業で行ったことを含む基礎的内容の課題を課し、プログラミングの作成・実行と結果の提出を求める。後半は教員が提示する多数の応用課題の中から興味関心に応じた課題を学習者自身に選択させ、コードの作成及び、それに関するレポートを課す。また、レポート内容について、簡単な質疑を個別に行い、学習者の理解度を確認する。必要に応じて、さらに調査するとよい点をコメントすることもある。

Scardamaliaらは、授業設計の2つのアプローチとして、目標創出型の「前向きアプローチ」と、正解到達型の「後ろ向きアプローチ」とに整理している[3]が、本授業は、前半は正解到達型、後半は目標創出型での設計で実施していることに対応する。後半の応用課題は、複数提出を認めており、多数提出する者も少なくない。一方で、一題も完遂しない学習者も存在する。

前述したように、今後もプログラミングの知識

習熟度の差が大きい学習者集団を対象とした授業になることが想定される。そこで、前半で行う正解到達型の授業に、他の学習者と協調学習する機会を加え、目標創出型の要素を組み込むために、学習者間の相互コメント付与システムを設計した。本稿では、その設計内容と試行実践を報告する。

2 授業の概要と実践環境

2.1 授業の概要

本実践の対象とする授業は、第一著者から第三著者が所属する大学の一般教育として、後期に開講している2つの選択科目である。Ruby 言語の授業は、総合科目として2006年度から継続的に開講してきた。Python 言語の授業は、2020年度から共通科目の情報学の一環として開始され、第一著者と第二著者が担当している。共に15回2単位の授業である。両科目を履修する者が若干含まれるが、応用課題は別課題を提出させるなど、成績に影響しないように配慮している。2020年度以降は、前

表1 授業の前半実施内容の概要と対応課題
(表内の py/rb はそれぞれ Python/Ruby を示す)

回	Python	課題	Ruby	備考
1	四則演算、数学関数	1,2	同左	Ruby クラス分け調査
2	文字列、変数の使いまわし	3	同左	Python 進捗確認開始(1回/3週)
3	真偽値、条件分岐、繰り返し、例外処理	py 4-7 rb 4-5	比較演算、繰り返し	Ruby 進捗確認開始(1回/2週)
4	リスト(多重含む)	py 8 rb 6-7	多重繰り返し、例外処理	
5	HTML 出力等	py 9,10 rb 8	配列	進捗確認(2回目)
6	メソッド、変数の有効範囲	py 11,12 rb 9,10	HTML 出力等	
7	ビット演算	py 13 rb 11,12	メソッド	
8	正規表現	py 14 rb 13	ビット演算	
9	応用課題	rb 14	正規表現、変数の有効範囲等	Python はローカルPC利用可

半8回は著者らが独自開発したプログラミング環境[4]上でプログラムを実践させる。編集・実行ログ等を収集・確認できるようにしており、学習者にもその旨を周知している。これは、Covid-19でオンライン授業を中心に実施せざるを得なかったことに端を発するが、学習者のプログラム実行状況を逐次確認し支援可能であること[5]、また欠席か否かの判断材料にもできることがその長所として挙げられ、継続的に利用している。

Ruby の授業は、教員3名(第一著者から第三著者)で履修者のスキルと希望によりクラス分けをして実施する。Python の授業は、前述の通り教員2名が共同で行い、レベルによるクラス分けは行っていない。表1に、前半の基礎課題における各回での進捗の目安を示す。基礎課題は全14題を設定している。表2にその内容の概要を示す。各自が目安より早く到達した場合には、応用課題に随時取り組ませるが、前半の期間は、あくまで独自サーバ上で実行可能な応用課題から選択させ、サーバ上でプログラムを編集・実行させる。Python の授業は、クラス分けを行わないこと、ならびに前期の学習内容との重複が多いこと等から、Ruby に比べると一週前倒しに進める目安としている。

表2 14題の基礎課題の概要

	テーマ	課題
1	基本数値演算	加減乗除、べき乗、余り等
2	実数の絶対値の上限・下限	指数形式での表記、10のべき乗での表記でそれぞれ確認
3	変数	人口減少を指数関数で表現
4	論理演算	not/and/or 演算子の振る舞い
5	素数判定	繰り返し/条件分岐を使う穴埋め問題
6	複数の素数を求める	多重ループの単純な繰り返しによる穴埋め問題
7	平方根の逐次近似	ループ脱出の設問であるが、他の考え方で解いてもよい
8	2次元のリスト・配列	指定された演算結果を2次元リスト/配列の要素として格納。行・列・対角和を求める
9	ファイル出力	HTML ファイルの出力
10	ファイルの読み込みと出力	身長・体重・氏名データを外部ファイルから読み込み、処理をしてファイル出力
11	関数の定義と引数の追加	定義された関数に、指定する利用に応じた引数を追加
12	再帰	パスカルの三角形の表示
13	ビット演算	ビット列の演算結果を図示
14	正規表現	Python と Ruby では別課題

後半の授業では、各自の PC 上でプログラムを編集・実行することを認めている（Python の授業では第 9 回以降、Ruby の授業では、第 10 回以降）。Life Game のアニメーションなど、サーバでは実行できない応用課題に取り組むことが可能となる。

授業では学習者支援として、教員と学習者個々が、2-3 週に 1 度、5 分程の進捗確認を行う。Zoom のブレイクアウトルーム上で、他学生に声の影響が出ないように、実習室の調整をしながら実施する。独自サーバ上で、実行ログ等を確認できるため、当該学習者がどのような状況にあるかを教員は確認できる。予定を決めておくことで、事前確認が可能となり、5 分という短時間であっても、適切なコメントと質問を受けることが可能である。

2.2 独自学習環境での補助資料の確認と実行

授業の前半は、図 1 に示す著者らが独自開発したプログラミングの実行環境を用い、必要に応じた個別支援をしながら基礎課題を進めていく。授業で用いる学習支援システム上には、基礎課題のテキストを掲載しているが、その他に、各回の学習内容に応じた補助資料を与えている。学習者には、基礎課題を実施するだけでなく、テキスト掲載例、補助資料上のプログラム例を含めて実行し、内容を理解することを求めている。

プログラミング環境では、図 1 の画面例のように、右上の領域に教員の補助資料を掲載し、その内容を左の編集画面に打ち込むこと、補助資料内容をそのまま貼り付けることも可能な仕様としている。実行ボタンを押下することで、プログラムの実行結果が右下に表示される。

このように、補助資料の内容を含め、プログラムを逐次実行し、結果を吟味することを求めているものの、補助資料をおろそかにし、基礎課題をこなすことが中心となる学習者は少なくない。そのような学習者は、授業後半に自身でコードを書く際に、その難易度の高さに大きなギャップを感じがちであるように見受けられる。

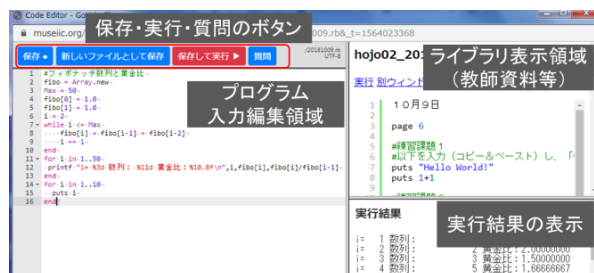


図 1 授業で用いるプログラミング環境

2.3 基礎課題の提出内容と補助資料との関係性

本稿執筆時は、多くの学習者が基礎課題 3 までは完了した状況にある。以下に、2024 年度の基礎課題 3 に関する補助資料の例を示す。

基礎課題 3 は、30 年で人口が半減する仮定を課し、100 億の世界人口が 500 年後に何人になるかを計算させる問題である。テキストでは、預金の複利計算を例としており、基本的には指数の減少関数を想定した立式となる。整数同士の割り算の演算子「/」は、Python では浮動小数点数に、Ruby では小数点以下切り捨ての整数になることから、図 2 に示すように補助資料には異なる内容を提示している。2024 年度実施の 2 科目における補助資料上の主な違いは、以下の 5 点である。

- Ruby では、想定される答え（2 種）を例示
- Ruby では、500 年後のみならず、30 年後、60 年後に想定値となるかを検算することを推奨（Python では、書式例のバリエーションを提示）
- Python では具体的な立式は学習者が記述、Ruby では、a, r, n という変数を用いることを指示
- Python の変数名は内容が分かる用語を、Ruby の変数名は内容を気にしないものを用いている
- Python は初期人口、半減期間（30）、目標年を、Ruby は初期人口、減少率（0.5）、目標年を変数として提示

一方、共通事項としては、単に 500 年後の人口の値だけを表示させるのではなく、書式付きの print 文を使い、「500 年後の人口は***人」と表示させるようにしていることが挙げられる。

学習者が実際に補助資料をどの程度確認しているかは、最終的な基礎課題 3 の提出内容にも反映される。例えば、書式付きの print 文でないものを提出した場合は、進捗確認時に補助資料の内容を説明し、補助資料の実行を求める等の対応を行う。

245	# 基礎課題3
246	PRINCIPAL, INTEREST, TERM = 100_000, 0.05, 10
247	total = PRINCIPAL * (1 + INTEREST) ** TERM
248	print('元利合計 =', total)
249	
250	# プログラム例
251	INITIAL_POPULATION, HALF_LIFE, TERM = 100_0000_0000, 30, 500
252	total = 計算式を記述する
253	# 以下の出力は同じ意味
254	print(f'500年後の人口は{total:.0f}人')
255	print('500年後の人口は{:.0f}人'.format(total))
256	print('500年後の人口は%.0f人' % total)

151	#基礎課題3
152	#答えは、152587人、または、96124人。どちらが正しい？
153	#プログラム例
154	a, r, n = 100_0000_0000, 0.5, 500
155	t = 計算式 (a, r, n を用いて)
156	printf "500年後の人口は %d 人", t
157	#検算
158	#例えば n=30 (30年後)の時は人口は半分(50_0000_0000人)のはずである
159	#同様に n=60 (60年後)の時は人口は25_0000_0000人である。確認してみる

図 2 基礎課題 3 に関する補助資料例（抜粋）
（上が Python の授業、下が Ruby の授業のもの）

2.4 コメント付与による基礎的知識の理解深化

基礎課題3までの初期の学習範囲においてさえ、理解が不十分な学習者が散見される。2023年度までは個々の学生の進捗確認時にコメント対応していたが、その場合、全員への対応は最大3週間を要する。2024年度は学習者のコメント付与活動により、理解深化が図れるかを検討することとした。具体的には、基礎課題2において、実数の絶対値の上限・下限値をきちんと調べず、適当な値を実行して終えている学生、基礎課題3において、自らの式の振る舞いへの理解が不十分な学生らへの対応等が挙げられる。

基礎課題2では、正しく解答している学習者も多い。そこで、各自が課題提出した結果をまずは提出させ、結果が収集された時点で、提出者全員分の提出内容を公開し、相互コメントすることを考えた。なお、提出内容を公開する際には、解答した学習者の氏名等はわからないような仕組みが必要である。基礎課題3では、まずはどのような変数名を付したか、また、年次変化でどのように人口が変化するかを学習者に確認することとした。

3 コメントの相互付与システムの設計

3.1 多段階相互評価機能 Moodle プラグイン

著者らの用いるプログラミング実行環境は、Moodle上に構築している。そのため、相互コメントの付与についてもMoodle上で実施することが望ましい。そこで、多段階相互評価機能を有するMoodleプラグインを新規開発・実装し、本授業の相互コメント付与システムとして用いることとした。

開発したMoodleプラグインの主な機能は以下の通りである。

- ・一般のMoodle小テスト機能(作文問題)による任意の設問を選択し、それに対する学習者の解答を第一段階として、他者に評価させることができる。この相互評価の付与を第二段階とする。
- ・相互評価の人数は、教授者が任意に設定できる
- ・相互評価の視点は、教授者が複数設定することができる。但し、相互評価の解答形式は、ラジオボタン、ならびに自由記述内容に限定する
- ・相互評価結果は、第一段階の学習者に返却することができる(第三段階A)。また、第二段階で同じ解答を評価した評価者全体に対し、評価コメントを共有できる(第三段階B)
- ・第一段階の解答者氏名、第二段階の相互評価者

の氏名については、開示・非開示を選択することができる

- ・第二段階(相互評価)の解答開始・終了ならびに、第三段階(相互評価に対するコメント)の解答開始・終了の日時を設定できる。
- ・相互評価のフェーズを区切ることができる。本機能により、相互評価結果の内容を踏まえて、自身の第一段階の解答を更新し、それをさらに相互評価するといった、連続的な学習者間の相互評価が可能となる。

基本的には、著者らが2005年に大学LMSに構築設計した内容[6]と同機能を有するようにMoodle上に実装したが、「フェーズ区切り」機能は、以前に構築したシステムに改良を加えたものである。

図3に多段階相互評価システム画面を示す。第一段階の解答者名を「表示」し、3名の解答をレビューするように設定した例である。「レビュー依頼」の項目にある「レビュー」ボタンより、第二段階の相互評価を行うことができる。また、「他のレビューを表示」ボタンより、第三段階Bによる「他者のコメント」を確認し、コメントを付与することができる。また、「あなたへのレビュー」にある「レビューを表示」ボタンより、第三段階Aによる自身の解答に対する相互評価結果を受け取ることができ、またそれに対するコメントの付与もできる。図3は、相互評価と相互評価に対するコメント付与の両方の入力可能な設定の場合を例示しているが、相互評価の入力期間外であれば、「レビュー」ボタンではなく、「レビュー期間外」の文字が表示される。また、相互評価の入力のみが許可されている期間であれば、「レビュー」ボタンのみが表示される。

3.2 本授業における相互評価システムの利用

本授業の基礎課題2および基礎課題3(共にPython授業)における画面例を、図4、5に示す。



図3 多段階相互評価システムの画面例

レビューする解答

問題 1

完了

最大得点 1.00

▼ 問題にフラグを付ける

基礎課題 2 で実数の絶対値の上限・下限を調べた結果、どうなりましたか？

・実数の絶対値の上限値を超えた場合
1.0e+??? の表示形式の場合、限界を超えると実行結果は、inf となった。
10 ** ??? の表示形式の場合、限界を超えると実行時に、エラーが出た。

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
OverflowError: (34, 'Numerical result out of range')

このエラーより、1行目に、数値が範囲外となり、オーバーフローエラーが出たことがわかった。

・実数の絶対値の下限値を超えた場合
1.0e-??? の表示形式でも、10 ** ??? の表示形式でも、下限を超えると
実行結果は、0.0 となった。

レビュー

設問 1

この設問は、どの範囲で実数を使えることを確認する問題である。ギリギリの限界を調べたことを述べている。

次の各場合で、ギリギリ限界を超えたべき値の値を具体的に答える（課題への解答結果を記入する）。

(1) 1.0e+??? の形式で、??? 部分に該当する答えを書く。

(2) 10.0 ** ??? の形式で、??? 部分に該当する答えを書く。

(3) 1.0e-??? の表示形式で、「-???」部分に該当する答え（マイナスの値）を書く。

(4) 10.0 ** -??? の形式で、「-???」部分に該当する答え（マイナスの値）を書く。

図 4 基礎課題 2 に関するコメント付与内容

レビューする解答

問題 1

完了

最大得点 1.00

▼ 問題にフラグを付ける

基礎課題 3 は変数に関する設問です。単位時間（30年）当たり半減する計算を行うこととしました。皆さんはどのような変数を具体的に思い、また、どのようなイメージで、半減について考え、式を立てましたか（図のリンクで、どちらのイメージが近いですか）。この設問では、条件として、30年で半減ということしか与えていませんので、（参考の式は提示していますが）それをどのように立式するかは各自に任されています。図のリンク

補助資料にある変数や定数はそのまま使いました。
補助資料では現在の人口を INITIAL_POPULATION、半減期を HALF_LIFE、年数を TERM という「定数」で与えられています。
また、変数としては、
・ x 年後の人口 : total
・ 減少比率 : decrease_ratio（最初に 0.5 を代入しました）
・ 年数 : years
としました。
半減していくイメージは、青い線のような指数的に減少していくイメージで式を作りました。オレンジのような階段関数で減少していくイメージでも式を作ってみようと思います。

レビュー

設問 1

あなたが基礎課題 3 で用いた変数の変数名を具体的に記載してください。

設問 2

あなたがこの設問で考えていた減少イメージは、以下のどれが近いでしょうか。

※ 未選択

☐ 図の青い線のような減少（スームズな指数の減少関数）

☐ 図のオレンジの線のような階段関数的減少（階段関数）

☐ その他（次の設問に具体的に記載）

設問 3

（後日解答可）

設問 2 で、青い線の減少イメージと答えた人は、オレンジ線のような推移となる式のために、どのようにプログラムの式を変えるかといひでしようか。また、オレンジ線の減少イメージと答えた人は、青い線のような推移となる式のために、どのように式を変えるかといひでしようか。さらに、設問 2 でその他を選択した人は、具体的な減少イメージと、プログラム上の式を記述してください。

※ この設問に最終的に解答できなくても、本設問が成績に影響することはありませんが、できるだけ考えて記述してください。

設問 4

（任意解答）

基礎課題 3 では、人口、減少比率、年を変数としたプログラムを求めました。本課題では、便宜上に減少比率は 0.5 の定数、また、500 年後の一点のみを確認する課題としています。しかし、プログラムにおける変数は、通常、その値を何度も使ったり、あるいは変化させながら使うことを想定します。

そこで、本課題を少し膨らませ、年次推移の途中で減少比率を変えたり、毎年の年次変化を求め表示していく例を考えることとします。例えば、ある年に極端に人口が減少するイベントが発生したり、時代毎に人口の減少比率を変えるようなことで、人口の推移を計算するような作問例を自由に考えてください。空想上の自由な発想の記述で全く構いません。（プログラムで具体的な定式化のイメージが持てることより良いですがここまで考慮して記載する必要はありません）

※ 解答例：現在の減少率は 30 年で 0.2 減少する値とする。また、その減少率は 10 年毎に 0.01 ずつ増えていく。さらに、現在から 30 年後に崩石がぶつかりその時点の全人口が半減する。その後の減少率は変わらずに 500 年毎に減る。等

終了

図 5 基礎課題 3 に関するコメント付与内容

図 6 図 5 の設問文「図のリンク」の押下画面

本授業では、まず、第一段階で評価させる解答は、教授者による解答例一つのみとした。従って、相互評価システムの第二段階では、学習者には教授者の解答例のみを確認させ、レビューコメントを付与とさせることができる。図 4 では、基礎課題 2 における（誤りを含む）解答例とレビュー項目の設問を、図 5 では基礎課題 3 における解答例とレビュー項目の設問（全 4 問）を示す。なお、基礎課題 3 では、課題提出済みの人のみがコメント付与できるように制限設定をしている。

基礎課題 2 については、学習者によるレビュー（学習者が課題提出した値）が概ね揃った段階で、第三段階 B により、レビューコメントを共有化し、相互コメントの付与を実施した。1 つの解答例に対し、全学習者が全レビューコメントを確認できる設定のため、相互コメントの付与は 1 つ以上でよいこととし、以下のように学習者に指示した。「面白いと思う解答、不十分と思われる解答に 1 つ以上コメントする（例えば、教員側の設問にミスがあり、整数の上限（サーバ上での限界値）をチェックした学生もいる）。特に不十分と思われる解答に対しては、その誤りを修正できそうな丁寧なコメントを心掛ける。」

図 7 に基礎課題 2 における相互コメント（第三段階 B によるコメント）例を示す。基礎課題 2 で、10.0 のべき乗による限界値を計算する際に、指数の値に小数を与えたレビューコメントに対する他の学習者からのコメントである。他の学習者の発想に驚いている様子が見て取れる。但し、厳密には小数点第 3 位は、実は限界値とは微妙に異なっており、その指摘がない点は少々残念である。他に、整数の上限値を確認したコメントに対し、「他のコメントにもありますが、(2)に関して、10** ??? と 10.0 ** ??? ではこの問いの回答も変わるようです。実際に試してみたところ、同様の結

レビューのコメント

「こんなに調べたのですか 凄いです」

「私は整数値でしか調べなかったので、エラー表示が出るようになるのが小数点以下の数字を追加したところだと思ってなくて、面白いと思いました。また、eを使うか、**を使うかの形式の違いでも差が出るのは面白いと思いました。」

「小数点以下まで調査しており、他の方より精度の高い結果が得られていて素晴らしいと感じました。」

「指数に整数だけではなく小数点も調べている点が良いと思いました。僕は小数を調べるという発想に至らなかったのでもこの結果はとても面白いと思いました。」

「小数点以下まで調べるという発想に至りませんでした。探求心に脱帽です。」

「小数を調べようという発想が抜けていて非常に面白いです。小数での結果が小数3位で精度が高く素晴らしいです。」

図 7 基礎課題 2 におけるレビューコメントに対する他学習者からの相互コメントの例

果になりました。」とのコメント付与があり、当該学習者は、コメント内容に関し、実際に実行して、結果を確認したことが分かる。

基礎課題 3 は、原稿執筆時点においては、まだ相互評価のコメント付与段階に留まる。相互評価の視点としては、図 5 に示す通り、(1)自身が用いた変数名の記述、(2)図 6 に示した青（スムーズな減少関数）とオレンジ（階段関数）のどちらのイメージに自身の計算が近いかの選択、(3)自身のイメージと異なるタイプのイメージとするために式のどこを変更するとよいかの説明、(4)任意解答ではあるが、変数の使用に関する考え方を示した上で、基礎課題 3 を膨らませる作問について求めるものとした。

4 考察

曜日による授業進行の違いにより、現状では Python の授業にのみ、相互コメントの付与を課して試行している。基礎課題 2 については、2023 年度までは各学習者への個別コメントが必要であったが、2024 年度は図 4 に示すコメント付与をさせることで、課題の意図に対する理解の向上が見受けられた。図 4 の解答例を確認し、課題意図を取り違えていたことを知った学習者は、自主的に課題の再提出を行っていた。

また、他の学習者のコメントを確認することによる理解の深化が見受けられた。例えば、10.0 のべき乗での表示の場合は、指数部の小数が可能であるが、指数形式の表示の場合は、整数しか許されないことを確認した学習者がいる。

しかし一方で、他の学習者のコメントをそのま

ま鵜呑みにし、相互コメントを付与する学習者がいることが分かった。そのような学習者に対しては、進捗確認時に自身での確認を推奨することで、さらに理解を深めることができると考えられる。例えば、前述の小数点以下第 3 位の数値が実際には異なることは、自身で実行してみるとわかることである。さらにその限界値は、Python における浮動小数点数が、IEEE 754 の 2 進数の浮動小数点演算を用いていることに起因する。このように、定型的な課題であっても、学習者のコメントに端を発して、正解到達型の学習に留まらない学習者の理解深化を計れる可能性がある。また、整数のべき乗と浮動小数点数のべき乗における限界が異なることについて、「どうして 10**?? と 10.0**?? で違うのか不思議に思いました。」という相互コメントをする学習者がいた。テキストに記述している整数と浮動小数点数の限界についての説明を確認していない、もしくは整数と浮動小数点数との違いが分かっていない可能性がある。このように、相互コメント内容から、課題提出のみでは確認できなかった学習者の総合的な理解度が推察できる可能性があると考えられる。

基礎課題 3 では、相互コメントの付与により、まずは変数の利用に関する基礎的理解を求めたいと考えている。基礎課題 3 では、減少比率は 0.5 の定率のため、それを変数とする意味は殆どない。また、500 年後の人口のみを計算する場合は、年数を変数とすることもない。一般に、プログラムにおける変数は、その値を何度も使う場合、あるいは変化させながら使う場合を想定する。そのため、年次推移の途中で減少比率を変えること、毎

年の年次変化を求め図示していくことを考えた例が適切である。学習者に様々な人口推移例を考えさせた上で、実際の世界人口の状況を示し、モデル化するような応用課題の設定等も考えられる。

また、Ruby では、整数同士の割り算は切り捨てになることから、何も考えない場合には、階段関数型の人口減少の式としている可能性が高い。そのため、補助資料では、「答えは、152587 人、または、96124 人。どちらが正しい？」と付与しているが、正しい答えは、各自がどのような人口減少をイメージしているかによって異なる。そこで、図 5 の最初のレビューする設問においては、「この設問では、条件として、30 年で半減ということしか与えていませんので、(参考の式は提示していますが) それをどのように立式するかは各自に任されています。」と記述した。実際に、各自の立式と想定した人口減少モデルが一致しているか否か、今後、調査と確認を進め、必要に応じてフォローアップしていくことを予定している。

4 まとめ

本稿では、プログラミング授業における正解到達型の課題を実施する際に、学習者のコメントを相互に確認し、コメントを付与することにより、基礎的内容の理解深化を計り、目標創出型の要素を組み込む可能性について検討した。学習者間の相互コメント付与のために、多段階相互評価システムを構築・実装し、実際に試行実践を行った。現在はごく一部の設問を課した段階ではあるが、基礎的課題に関連した設問にコメントを求めることで、各学習者の知識レベルに応じたコメントが収集される。その結果を学習者が確認し、相互にコメントすることで、当該課題の学習内容をより深く理解させることが可能であると考え。まず、最初のコメント付与の際に、当該課題の意図の確認を行うことができ、自らの誤りに気付くことが可能である。次に、他の学習者のコメントを確認し、コメントを付与することで、当該学習者の理解状況を総合的に確認できる可能性が示された。さらに、他の学習者の解答による理解深化が見受けられた。

2025 年度に大学に入学する学生は、新旧学習指導要領を経た者が混在することが想定される。知識レベルに幅がある学習者が混在する中で、授業を行う必要性がさらに高まることが予想されるため、本実践で試行しているような相互評価を用い

ての理解深化手法について、その適用の可能性と限界についても引き続き、調査していきたいと考える。

謝辞

本研究の一部は、JSPS 科研費 23K22310 の助成を受けた。また、本授業実践における基礎課題の内容、ならびに多段階相互評価の機能実装の仕様は、故・岡部成玄名誉教授の高い寄与によるものを継承しており、ここに深く感謝する。

参考文献

- [1] 文部科学省、平成 29・30・31 年改訂学習指導要領（本文，解説），
https://www.mext.go.jp/a_menu/shotou/new-cs/1384661.htm (2024 年 10 月 19 日確認)
- [2] みんなのコード、2022 年度プログラミング教育・高校「情報 I」実施実態報告書、p 50、2023 年 7 月、
<https://speakerdeck.com/codeforeveryone/2022nian-du-hurokuraminkujiao-yu-gao-xiao-qing-bao-i-shi-tai-diao-cha-bao-gao-shu> (2024 年 10 月 19 日確認)。
- [3] 大島純、益川弘如、学びのデザイン：学習科学、pp.46-48、ミネルヴァ書房、2016。
- [4] 布施泉、中原敬広、岡部成玄、プログラムの相互利用と相互評価が可能な初学者用プログラミング授業支援環境の構築、教育システム情報学会誌 Vol35、No.2、pp.221-226、2018。
- [5] 布施泉、増本広和、山本裕一、平林義治、オンライン授業での初学者用プログラミング学習環境の検討、大学 ICT 推進協議会年次大会、TA1-3、2020。
- [6] 布施泉、岡部成玄、多段階相互評価法による学習の実践と効果、日本教育工学会論文誌 Vol33、No.3、pp.287-298、2010。