学習者による書き出しを支援するSlackBotの開発と運用

長 慎也[1,a)]　山中 脩也[1]　北島 茂樹[1]　今野 貴之[1]

概要：筆者はこれまで，プログラミング等の演習中において，理解したことを言語化して書き出し，他の学習者と共有するシステムを開発し，授業で利用してきた．理解した内容を書き出すことにより，学習者が自分自身の考えを整理することができる，他の学習者の異なる考えに触れることで新しい発想を得ることができる，などの効果が期待される．しかし，他の学習者がすでに書き出した内容を事前に見てしまうことで，他の学習者と似たような意見を書いてしまう恐れがある．これを防ぐために，「自分が書き込みをするまでは，他の学習者の書き込みを見ることはできない」という制約が必要であると考え，システムを開発した．ただ，これまで開発したシステムは，テーマのタイトルとその内容がどのような対応になっているかが把握しにくく，過去の書き込みを見直すことが困難であることや，通知の仕組みがないことによって，その後の議論に発展しずらいという問題点が残った．そこで，今年度からは新たなシステムとして，同等の機能をもったシステムをコミュニケーションツールであるSlack上で動作するBotとして実装しなおした．また，テーマの一覧を閲覧・編集可能な管理画面を設置し，Botの動作を制御できるようにした．これにより，Slackの使いやすいインターフェースや通知機能を利用でき，先述したような問題点を解決できると期待される．本稿では，執筆時点で進行中の授業において本システムを使用した途中経過を報告する．

キーワード：プログラミング学習，書き出し，協調学習

Development and operation of a Slack bot to support writing by learners

CHO Shinya[1,a)]　YAMANAKA Naoya[1]　KITAJIMA Shigeki[1]　KONNO Takayuki[1]

1. はじめに

筆者はこれまで，プログラミングの教材において，なるべく説明を教材中に入れず，プログラムの振る舞いを学習者が自ら推論させるような手法を検討してきた．理解した内容を自分の言葉で書き出すこと（言語化）を行うことによって，学習効果が高まると考えられている [1][2][3]．その際，あらかじめ説明した資料などを与えてしまうと，その文章の内容に引っ張られてしまい，書き出しの内容が似通ってしまうため，「学習者が自分の言葉で言語化する」ことの意義が薄れてしまう．

幸い，プログラミングはコンピュータのソースコードと実行結果を見れば，その振る舞いを推測できるため，開発環境の操作方法などの説明は別にして，プログラムに関する詳しい説明がなくてもある程度推測ができると考えられる．

また，他者の書き込みを見ることで，教授者だけでなく受講者同士でも異なる意見や解釈の仕方を複数見ることができ，理解の幅を広げることも期待される．

本手法を取り入れたプログラミング教材は次の4種類のPhaseを導入している [4][5]．

- **Phase1** 教材が示したソースコードを受講者がコンピュータに実行させると，コンピュータは処理系の決めたルールに従って結果（出力）を受講者に出力する．
- **Phase2** 学習者は出力結果を教材に記入した上で，ソースコードと出力結果を見て，そこに含まれるルールを推論する．

[1] 明星大学
Meisei University, Japan
[a)] cho@is.meisei-u.ac.jp

- **Phase3** 教材がソースコードを空欄（場合によっては穴埋め）のまま，結果のみ提示する．受講者は Phase2 で推論したルールを使用して，空欄部分のソースコードを推察する．
- **Phase4** 受講者は推論したソースコードを再びコンピュータに実行させ，結果が提示されたものと同じであれば，推察したルールが正しいという確信を持つ．結果が同じでなければ，コンピュータからの出力（エラーメッセージや提示されたものとは異なる出力結果）を見て，修正を加える．
- **Phase1'** 教材は別のソースコードを提示し，以下同様に繰り返す．

このうち，Phase1 と Phase3 は通常のプログラミングの授業でもよく行われている「プログラムの例を実行する」「題意を満たすプログラムを書く」という演習である．

一方，Phase2 については，ルールが正しく推論ができているかどうか自然言語で書き出す活動をもって達成されると考えられる．また先述した通り，書き出した内容は他者（＝他の学習者）と共有できることが望ましい．

学習者間で書き出しを共有するには，グループワークでよく見られるように，対話を直接行う方法が一般的である．しかし，発話が苦手な受講者には抵抗があることや，グループ内でいつも積極的に発話をするメンバーの意見に同調しがちになる，などの点が問題点と考える．また，LMS や Slack などのコミュニケーションツールを使用して書き出しを行うことが考えらえるが，これらのツールでは，自分が書き込む前に他の学習者の書き込みが見えてしまう．そうすると他の学習者の意見に引っ張られてしまい，内容が似通ってしまうことが危惧される．また，そもそも書き出しを行う活動をせずに，問題を解くこと（Phase3：プログラムを書く）だけを行ってしまう学習者もいると考えられる．

そこで，特定のテーマに対して，自分が書き込みを行わない限り，他者の書き込みを見ることができないようなシステム「Note & Key」を作成した [6]．

2. 昨年度使用したシステム

図 1 Web アプリケーションによる書き出し支援システム (Note & Key)

Note & Key の画面を**図 1** に示す．Web アプリケーションとして実装されており，ユーザはノートを作成・閲覧することができる．ノートには任意のタイトルをつけることができる．ノートは非公開（下書き）の状態と公開の状態があり，公開の状態にすると，同一のタイトルをもった他ユーザの書き込みが一覧表示され，内容を確認することができる．

教授者が学習者に対して，ある教材に沿って演習した内容について書き出しを行ってほしい場合には，教授者側でその教材に紐づいたタイトルを指定して学習者にノートを作成してもらう．すると学習者は，他の学習者の内容を事前に見ることなく書き出しを行え，書き出しを行った後で初めて他の学習者の書き出しを確認できる．また，教授者も同一のタイトルでノートを公開し，そこに次のテーマのタイトルと資料を提示しておけば，書き出しをすることなく先の資料に進むことを防止できる．

2022 年度はこのシステムを利用して授業実践を行った．当初は各テーマについて「気づいたことを書いてみよう」という抽象的な投げかけを行っていたが，受講者からの書き出しは明確なものではなく，「あ」など 1 文字だけを書いている受講者もいた．そこで，各テーマについて具体的な質問を設けることで積極的に書き出しを行えるようにした．

一方で，書き出した内容について再度見直したり，他者の書き込みと比較を行ったり，という活動はあまり行われなかった．

その理由の一つが，Note & Key の使い勝手にあった．例えば，通知の機能がなく，他の学習者や教員が返信をしたとしても気づかれないため，その後の議論が進展しないことが多かった．また，過去の議論に遡るためには，それぞれのタイトルと内容を見なおす必要があったが，そのタイトルがどのようなテーマであったかを確認する手段が Note & Key そのものになかった．

3. 新たに作成したシステム

先述のような問題を解決するため，Note & Key のシステムを Slack の Bot として実装した「Note & Key on Slack」を新たに開発した．

Slack を利用した理由としては，受講者が普段から連絡用に使っていること，通知機能があることで議論が継続的に行えること，などが挙げられる．

また，Slack の Bot と連携して，テーマの管理が行える Web 画面も新規に追加した．

今回作成した Note & Key 向けの Bot の構成を**図 2** に示す．教授者および学習者には次のような機能を提供する．

- **テーマ作成・編集（教授者）**
  教授者は，テーマ管理画面を用いて，あるテーマに関連するテキストを作成する．テキストには学習者に学習させたい教材の内容または入手方法（URL など）を記述し，必要に応じて，学習者に書き出してほしい内

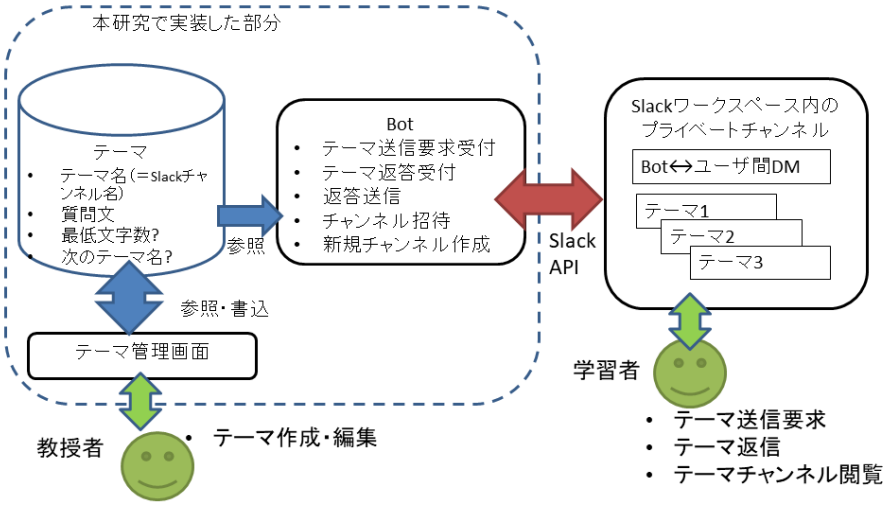

図 2 Note & Key on Slack のシステム構成

容の方向性や，具体的な質問文も追記できる．テキストの他に，学習者が書き出す文章に最低限必要な文字数や，「次のテーマ」（後述）も設定できる．
テーマが作成されると，Bot は自動的にテーマ名に対応したチャンネルを Slack ワークスペースに作成する．作成されたチャンネルはプライベートチャンネルであるため，学習者はテーマが作成された段階ではチャンネルの内容は見られない．

- **テーマ送信要求（学習者）**
  学習者は，Slack 上の Bot ユーザにダイレクトメッセージ（DM）を通じてテーマの名前を送信する (**図 3** の最初のメッセージ）と，Bot がそのテーマに対応したをスレッドに返信する（その次のメッセージ）．なお，テーマの名前は，ワークスペース内の公開されているチャンネルに提示するなどの方法で教授者から学習者に提示する．
- **テーマ返信（学習者）**
  学習者は，さきほどの Bot のテキストに対して回答を返信する (図 3 の 3 番目のメッセージ)．すると Bot はそのユーザを対応するテーマのプライベートチャンネルに追加させ (4 番目のメッセージ)，同時に学習者の回答をプライベートチャンネルに匿名で送信する．もしテーマに対して最低文字数が設定されており，回答文が最低文字数に満たない場合「書き出しが不十分です」と返信し，チャンネル招待と回答送信は行わない．また，テーマに「次のテーマ」が指定されている場合，Bot が次のテーマのテキストを自動的に送信する（今のテーマとは別スレッドになる）．
- **テーマチャンネル閲覧（学習者）**
  学習者は追加されたプライベートチャンネルを閲覧し，他の学習者の書き出しを閲覧する．

テーマ管理画面を除けば，教授者や学習者が使用するのはあくまで Slack であるので，上記の機能に加えて Slack の機能をそのまま使うことができる．例えば，各テーマのプライベートチャンネルにさらに返信を行って議論を継続したり，教員がそのテーマの補足資料を置いたりすることができる．

図 3 テーマ送信要求とテーマ返信に用いるスレッド

4. 授業実践

現在 Note & Key on Slack を使った授業を実施中であり，ここではその途中経過を報告する．

4.1 授業の概要

「プログラミング演習 3」は，明星大学情報学部情報学科 2 年生の必修科目であり，習熟度別に 4 クラスに分かれて実施されている．そのうち，習熟度が下位から 2 番目のクラスにおいて Note & Key on Slack を使用した．このク

表 1 「プログラミング演習 3」シラバス

回	学習内容
1	復習 1：変数・入力・制御構造
2	復習 2：配列
3	復習 3：関数
4	値と型（文字列の構築など）
5	OOP：クラス・フィールド・コンストラクタ
6	OOP：メソッドによる値の取得
7	OOP：メソッドによる値の操作
8	OOP：オブジェクト同士の関係・演算子の拡張
9	OOP：継承
10	オブジェクトの配列
11	応用問題
12	ファイル IO
13	統計処理 (pandas)
14	アルゴリズム設計の基礎
15	振り返り

ラスの 2023 年度の受講者は 44 人であり，すべて対面で行う予定である[*1]．

「プログラミング演習 3」のシラバスを**表 1** に示す．Python を用いたオブジェクト指向プログラミング（OOP）の基礎を学ぶが，最初の 3 回は 1 年生で学んだ事項の復習となっている．

4.2 配布資料

配布資料の構成を**図 4** に示す．資料は，主に「あらかじめ書かれているプログラムについて出力結果を記入させる（1 節の Phase1 相当．図中では「例示」）」「指定した出力結果（プログラムによっては指定された入力に対応する出力結果）を得るようなプログラムを記述させる（Phase3 相当．図中では「記述」）」という 2 つのステップを交互に繰り返して進める．プログラムを記述させるステップは，それまでのプログラムの内容と入出力例から推測ができるようにしてある．資料全体がいくつかの質問（Phase2 相当）によって複数のパートに分割されており，授業開始時点では最初のパートの資料だけを入手できるようになっている．各パートの末尾では，そのパートで学習した内容について書き出しを行ってもらう．昨年度と同様「気づいたことを書いてみよう」という投げかけでは思うような書き出しをしてくれないと予想されたので，末尾にはそのパートに登場したプログラムに関する具体的に質問を記述した．学習者はその質問に対する回答[*2]（書き出し）を行うことで次の資料を得ることができるようにした．

Note & Key on Slack を用いて，これを次のように実現した．

- 教員は，各パートの資料を Web サーバにアップロードする．ファイル名は連番にするが，容易に推測できない名前を末尾に追加する．
- 教員は，Note & Key on Slack のテーマ管理画面を用いて，各パートの資料のファイル名と同じ名前のテーマを作成する．質問文は「**【資料の URL】**を見て，末尾の質問に答えよう」とする．また，次のテーマには，次のパートの資料のファイル名を指定する．
- 教員は，学習者に Slack の公開チャンネルを通じて，「今日の資料を入手するには，@Bot 宛に**【最初のパートのチャンネル名】**と話しかけてください」と伝える．
- 受講者は，Bot 宛に話しかけて資料を入手して演習を始める．
- 末尾の質問の答えを Bot に返信すると，対応するテーマのチャンネルに招待され，同時に次の資料の URL が Bot から送られてくる．

なお，資料の最後のパートには，詳細な解説を最後に載せている．これによって，学習者が自らプログラムの構成要素について予め推論を行い，その上で解説を読むことで，知識を確実に定着させることを狙った．

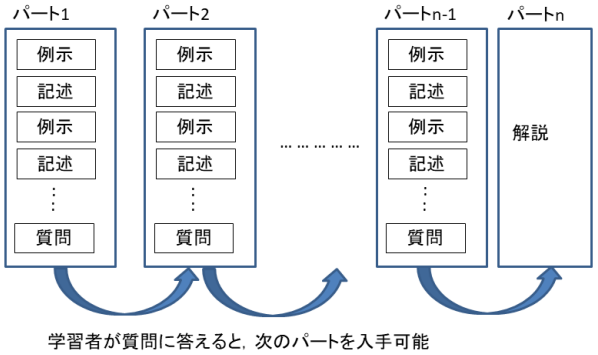

図 4 授業資料の構成

4.3 本授業に提案システムに導入することで期待される効果

今回報告するのは，プログラミングの基礎を学ぶ授業であり，すべての学習者が同じプログラムを用いて演習を行う．また，書き出す際に具体的な問いかけを行っているため，学習者の書き出す内容は似通ってくると考えらえるが，それでも，次のような効果は期待されると考えられる．

- 言語化を行うことで，単純に説明を受けるだけよりも理解が深まる．
- 説明されても理解が難しい内容について，表現が異なる他者の書き込みを見ることで，自分なりの考えをまとめることができる．
- Phase1（提示されたプログラムの出力結果を確認する）の段階で，どのようなことを推論したかを明文化する (Phase2) ことで，Phase3（提示された出力結果になるようなプログラムを記述する）を解く上でのヒントになる．

[*1] 荒天のため第 7 回を非対面で実施
[*2] 正解を求めているわけではなく，書き出すこと自体が重要なので「解答」ではなく「回答」である

5. 実践結果

現在，第 13 回までの授業が終わっている．ここまでの各回に提示した教材のパートごとのチャンネル名と質問内容を**表 3** と**表 4** に示す[*3]．ここに出てくる質問に加えて，各回の最後に「最後に，今回の授業で気づいたことや疑問に思ったことを書こう．解説が読めます.」という質問があり，13 回で計 81 件の質問をしている．それぞれのテーマは最低 10 文字以上の書き込みをすることを要求している．

表 3 と表 4 に示した質問には，最後の解説を読めばおおむね適切に回答ができるようになっているが，最後の解説を読む前に自分の言葉で説明ができていれば，その回に学習すべき内容を深く理解していると考えられる．そこで，受講者の書き込まれた内容と，解説に掲載された内容とに共通点があるかどうかを評価の観点としたい．ただ，まだ授業が完了していない状況から，現状は速報として，書き込みの総数，文字数と，主な書き込みについてピックアップしたものを報告する．

提示した質問の総数 81 件に対して，第 13 回終了時点で 3489 件の回答があり，平均で 43 件の回答があった．ここには教員 1 名と 3 名の授業補助者の書き込みも含まれているが，受講者が 44 名であることを考えると，ほぼすべての学生が何等かの書き込みをしていることがわかる．また，1 件あたりの平均文字数は 36 文字であった．さらに詳しく，テーマごとの文字数の分布と，ユーザごとの文字数の分布（平均文字数順）を**図 5** と**図 6** に示す[*4]．

書き込みの長さが長いが意味のない書き込み（例：「あああ」など）は目視で見た限り見受けられなかった[*5]．このことから，文字数によって書き込みの質をある程度推し量ることが可能である．

各テーマごとに文字数にはばらつきはあるが，ユーザごとの分布を見ると，平均文字数が少ないユーザでも長い書き込みをする場合もあるので，特定のユーザばかりが長い（短い）書き込みをしているわけではなく，すべてのユーザが何らかの形で書き出しに参加していることがわかる．

書かれた内容についての精査はこれから行っていくが，ここでは例としてチャンネル名 「week7-1func」の書き込みについて取り上げる．このテーマでは，リスト 1 のプログラムを提示し，出力結果を記述させたうえで「(破壊的にフィールドのを書き換える）move と（非破壊的にオブジェクトを作成して返す）moved の動作の違いは何か」という質問を行っている．もちろん，オブジェクトの破壊的・非破壊的な書き換えという用語や意味については事前に何も説明をしていない．このテーマは，プログラムの振る舞いから新しい概念を類推させるという色合いが強いものであり，かつオブジェクトならではの概念を学ぶという点から，特にここでピックアップした．

リスト 1 week7-1func(オブジェクトの破壊的書き換えと，非破壊的書き換えの違いを見つけるプログラム)

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def move(self,dx,dy):
        self.x+=dx
        self.y+=dy
    def moved(self,dx,dy):
        x=self.x+dx
        y=self.y+dy
        p=Point(x,y)
        return p
    def __str__(self):
        return "({},{})".format(self.x, self.y)
p1=Point(2,3)
p2=Point(4,5)
print("p1=",p1)
print("p2=",p2)
p1.move(1,10)
p3=p2.moved(1,10)
print("p1=",p1)
print("p2=",p2)
print("p3=",p3)
```

この質問に対して，38 件の回答を得た．そのうち 10 件を列挙する．

- move は座標 (self.x, self.y) に dx と dy を加えて新しい座標を表示するが、moved は座標 (self.x, self.y) に dx と dy を加えて、p=Point(x,y) を作ってから、これを返している
- move は移動前の点のオブジェクトを移動させた先の点に書き換えていて、moved は移動前の点のオブジェクトを残して 新しい移動後の点オブジェクトを作っている。
- move ではポインタの値そのものを変えているのに対し、moved ではポインタの値は変えずに、x,y の値を返している。
- move はオブジェクト自体の座標を変更します。moved は現在の座標を変更せずに、与えられた値 (dx と dy) を加算した新しい Point オブジェクトを作成して返します。
- move は例えば p1.move(1,10) ならｐ１の（２，３）のやつと足し算されちゃうが、moved なら p1.move(1,10) ならｐ１の（２，３）の部分を出す

[*3] 表中の”ch” はチャンネル名だが，実際にはもっと長く，推測しずれらいものになっている

[*4] テストユーザや誤登録で複数アカウントを使用した受講者を含んでいる

[*5] 書き込みが正しいか誤っているかは成績には影響しないが，意味のない書き込みを頻繁に行った場合は減点の対象とする予定である．このことはまだ受講者から問い合わせがなく，該当する受講者もいないのであえて告知していない．

表 2 アンケート結果（4：あてはまる，1:あてはまらない）

質問＼尺度	4	3	2	1
Bot に対して書き出しをしたことで問題を解くヒントを得ることができた.	10	14	3	4
Bot から招待されたチャンネルで他の学生が書いた内容を参考にした.	15	10	3	3

- move はどんどん動いていくのに対し、moved は一回動かすのみである。連続してできるかどうかが違いである。
- ‘move‘は、‘Point‘クラスのメソッドとして定義されているが、‘moved‘は、‘Point‘クラスには存在しないメソッド。
- move はその数の分だけ動かして、moved はその数の分だけ動かされたものに変える
- Python において、”move”と”moved”の動作の違いは、通常、過去形と現在形の違いにあります。
- 働きとしては同じではあるが moved を move に変更したところ、プログラムが上手くいかなかったので役割を分けていると予想する

このように，破壊的，非破壊的という概念について用語を使わずともきちんと書き出しをしているものもあれば，表現がうまくいっていないが違いについて書こうとしているもの，単語の意味だけを表層的に捉えたもの，あるいはプログラムの写し間違いで異なる結果を得たもの，など多様な書き込みが得られていることがわかる．学習者にとって重要なのは，これらの中から正しいと思われるものを選びとる力や，正しいものの中にも表現の異なるものがあり，その中から「しっくりくる」ものを選ぶ力であると考える．このような過程を通して，教員が与える説明だけでは理解できない部分まで理解を深めることが期待できる．

5.1 アンケート調査の結果

授業日程の終わりに，アンケート調査を記名式で実施し，31 件の回答を得た．4 段階のリッカート尺度で回答する質問の結果を**表 5.1** に示す．質問項目には，4.3 に示したような効果が実感できたかどうかを問うべく，表のような質問を行っている．多くの学生が，他の学生の書き込みを参考にしたこと，問題を解く上でのヒントを得ることができていたと回答している．

また，自由記述では「他の人の意見と照らし合わせて自分の考えを考察を確認することが出来る」「自分のペースで演習を進められる」などの利点を挙げたものがある一方，「学習が進むにつれ，チャンネルの数が多くなってくると復習の際にどこを見てよいかわからなくなる」といった問題点も指摘された．また，Bot からプライベートチャンネルへの投稿は匿名で行われていたが，実際にはチャンネルに受講者が招待された旨のメッセージが名前つきでプライベートチャンネルに自動送信されてしまうため，「匿名で投稿できるはずだが，招待順で誰が投稿したかわかってしまうので，改善されると投稿しやすくなる」という，匿名性を保ってほしい旨の意見が多くみられた．また「Bot 自身からヒントや注意点をもらえるようにしてほしい」という機能追加の要望があった．

6. 考察

Note & Key on Slack を使用した結果，書き出された文字数やその内容を見ても，それぞれの学習者が何等かの意味のある書き出しを行わせ，学習者がプログラムの内容と出力結果を通じて体験したことを自分の言葉を使って書き出し（言語化）させることができたと考えられる．

一方で，授業の実践方法やシステムとしての課題もあり，次に述べていく．

6.1 授業実践としての課題

受講者が他の人の書き出しをあまり読んでいない印象がある．特に，「次のテーマ」を設定してしまうと，Bot が次のテーマを提示するので，招待されたプライベートチャンネルを訪問せずに次の活動に入ってしまうことがあった．そのため，第 2 回以降は「次のテーマ」を設定するのをやめ，招待されたプライベートチャンネル内に教員が次のテーマの名称を投稿することで，チャンネルの内容（他の受講者の書き出し）を読ませるようにした．

しかしそれでも解説文を読んでいない状態で質問をしてくる受講者もいるなど，他の受講者の書き出しに関心をもたない受講者が多いのが現状である．そこで，書き出しについて議論させる（例えば，よかった書き出しにスタンプをつけされる程度もよい.）機会を積極的に推奨する必要がある．実際「よかった書き込みにスタンプをしよう」と教員が書き込んだが，スタンプをつける受講者は少数であり，スタンプをつけないと先に進めない，くらいの仕組みが必要と考えられる．

6.2 システムとしての課題

上記に関連して，スタンプや返信などのリアクションがあまり行われないのは，Bot がプライベートチャンネルに対して匿名で書き出しを送信する（これは，匿名のほうが書くことに抵抗がないとの判断でそうしている）ため，その後書き込みに対して返信がついたとしても，投稿したユーザに通知が伝わらないという問題があるためである．

そこで，スタンプや返信がついたことをそのユーザに通知する仕組みがあると，議論がより深まるのではないかと考えている．どのユーザが投稿を行ったかは，Slack には送信しないものの Bot のログには残っているため，投稿したユーザだけに通知を行うことは可能であるため，今後実装する予定である．

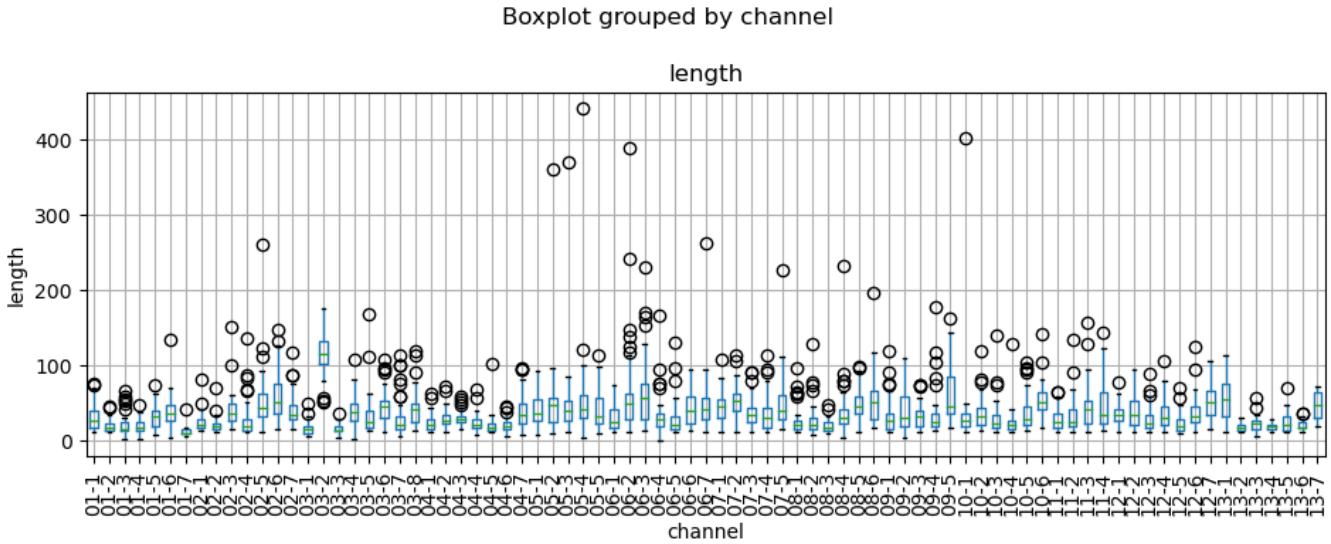

図 5 チャンネルごとの文字数の分布

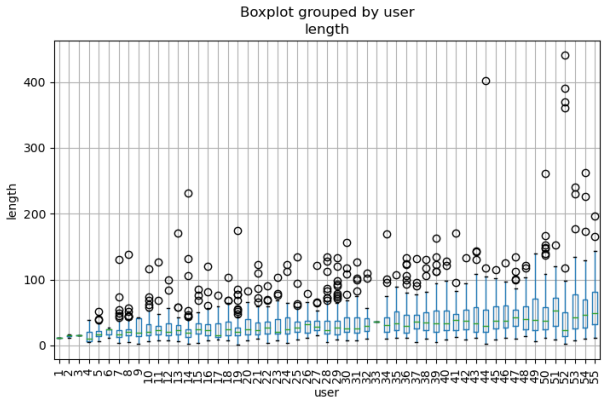

図 6 ユーザごとの文字数の分布

7. まとめ

プログラミング等の演習中において，気づいたことを書き出し，他の学習者と共有するシステムを開発し，授業で運用中の経過を報告した．学習した内容を書き出すことにより，学習者が自分自身の考えを整理することができる，他の学習者の異なる考えに触れることで新しい発想を得ることができる，などの効果が期待される．しかし，他の学習者がすでに書き出した内容を事前に見てしまうことで，他の学習者と似たような意見を書いてしまう恐れがある．これを防ぐために，「自分が書き込みをするまでは，他の学習者の書き込みを見ることはできない」という制約が必要であると考え，システムを開発した．

これまで開発したシステムを改良し，コミュニケーションツールである Slack 上で動作する Bot として実装しなおした．また，テーマの一覧を閲覧・編集可能な管理画面を設置し，Bot の動作を制御できるようにした．これにより，Slack の使いやすいインターフェースや通知機能を利用でき，先述したような問題点を解決できると期待される．

現時点では授業の中盤まで進んでおり，ここまで各受講者がなんらかの書き出しに参加しており，教員からの説明がなくても自ら学習内容について理解を進めることができているとみなせるが，受講者同士で書き出しを見て相互に評価する段階までは至っておらず，システムや授業の進め方に改善が必要と考えられる．なお，発表時点では授業が終了しているため，授業全体を通した使用結果を報告できる予定である．

謝辞 本研究は JSPS 科研費 19K03153 の助成を受けたものです。

参考文献

[1] 小林敬一：他の学習者に教えることによる学習はなぜ効果的なのか？，教育心理学研究，Vol. 68, No. 4, pp. 401–414（オンライン），DOI: 10.5926/jjep.68.401 (2020).

[2] 大林史明，下田 宏，吉川榮和：仮想生徒へ「教えることで学習する」CAI システムの構築と評価，情報処理学会論文誌，Vol. 41, No. 12, pp. 3386–3393 (2000).

[3] 白水 始，伴 峰生，辻 真吾，飯窪真也，齊藤萌木：協調学習の授業づくり支援のための「学譜システム」開発，情報処理学会論文誌，Vol. 60, No. 5, pp. 1201–1211 (2019).

[4] 長 慎也，山中脩也，北島茂樹，今野貴之：情報系初年次のプログラミング演習における，コンピュータとの対話を重視したコースデザインと支援システム，技術報告 20，明星大学, 明星大学, 明星大学, 明星大学 (2019).

[5] 長 慎也，山中脩也，北島茂樹，今野貴之：C.S. Peirce の探究過程と SECI モデルに基づくプログラミング演習の実践とその支援システム，しごと能力研究，Vol. 2020 特別号, pp. 84–97 (2020).

[6] Cho, S., Yamanaka, N., Kitajima, S. and o, T. K.: The Tools which Assist ”Creating Questions througn Interaction with a Computer(Abstract only), *WCCE 2022 Book of Abstracts* (2022). `https://wcce2022.org/WCCE_2022_Book_of_Abstracts.pdf`(2023-06-14).

表 3 チャンネル（テーマ）名と質問の内容 (1〜7 回)

ch	質問の内容（ねらい）
1-1	大小比較における「より大きい」「以上」の使い分け
1-2	（FizzBuzz 問題における）条件式の意味
1-3	ネストになった for 文の実行回数.
1-4	while 文の終了後のループ変数の値
1-5	（素数判定プログラムにおける） ループ終了後の変数の状態
2-1	配列のインデックスの有効範囲.
2-2	配列の append メソッドの働き.
2-3	（入力値が負の数になったら繰り返しが終了する プログラムを提示し）終了条件は何か．break 文の働き.
2-4	（配列要素の削除後にインデックスがずれること を意識して）指定された複数の要素を削除する方法.
2-5	配列の参照代入とコピーの振る舞いの違い.
2-6	a.sort() と sorted(a) の違い.
3-1	関数定義の数学とプログラムにおける記述方式の違い.
3-2	提示したプログラムに含まれる関数の振る舞い.
3-3	提示したプログラムにおける関数定義内の各行の実行回数.
3-4	（名前が同一のグローバル変数とローカル変数を 使用したプログラムにおける）変数の値が異なる理由.
3-5	関数呼出を含むプログラムのトレース
3-6	print 関数の使用方法（改行の扱い）
3-7	関数呼出を含むプログラムのトレース
4-1	（小数点つきの値を int に変換するプログラムを提示し） エラーの理由
4-2	（文字列と数値を足し算するプログラムを提示し） エラーの理由
4-3	（入力値を数値に変換し忘れたプログラムを提示し） エラーの理由
4-4	文字列.format の働き
4-5	文字列に対する `x+=y` の働き
4-6	関数に渡される引数の型の確認
5-1	オブジェクトと配列の違い
5-2	オブジェクトを引数にとる関数の働き
5-3	オブジェクトと数値を引数にとる関数の働き
5-4	2 つのオブジェクトを引数にとる関数の働き
6-1	メソッドと通常の関数の違い.
6-2	（点オブジェクトの要素を k 倍するメソッドにおける） メソッド呼び出しの振る舞い．self の働き
6-3	（2 つの点オブジェクトの減算メソッドにおける） メソッド呼び出しの振る舞いについて．self の働き
6-4	（メソッド呼出を通常の関数呼出と同じ書き方で 行っているプログラムにおける）エラーの原因
6-5	`__str__`メソッドの働き
6-6	真偽値を返すメソッドを if 文の 条件式内で呼び出したときの働き.
7-1	フィールドを破壊的に書き換えるメソッドとフィールドを 書き換えた新しいオブジェクトを返すメソッドの違い
7-2	別のメソッドで上記と同じ質問.
7-3	上記とは別のメソッドを提示し， 引数を明確にして振る舞いを説明させる.
7-4	（フィールドの仕様を変更したプログラムを見せてから） getter メソッドを使うメリット.

表 4 チャンネル（テーマ）名と質問の内容 (8〜13 回)

ch	質問の内容（ねらい）
8-1	（2 つの点オブジェクトの要素をそれぞれ足し算する メソッドにおける）動作の説明
8-2	（足し算するメソッドの引数の仕様を，各要素を渡す 方式から，オブジェクトを 1 つ渡す方式に変更した プログラムを見せて）変更前と後で変化した部分
8-3	（上記のメソッドを`__add__`メソッドとして 実装しなおし）`__add__`メソッドの動作
8-4	（点オブジェクトを中心点として使用する円クラスを 定義し）円クラスのフィールドの名前と型一覧
8-5	（円クラスの中心点を破壊的に移動させるメソッドと， 中心点を移動させた新しいオブジェクトを返す メソッドを定義し）2 つのメソッドの違い
9-1	必要なメソッドが定義されていない クラスを提示して、エラーの原因
9-2	継承を使ったクラス定義を示して、 継承を使っていないコードと比較
9-3	親子クラスで同名のメソッドがあるクラスを 提示し、どちらのクラスのメソッドが呼ばれるか
9-4	super() を含むメソッドがどんな動作をしているか
10-1	オブジェクトを格納した配列から 特定の値を取り出す式を書かせる
10-2	オブジェクトを格納した配列に対する for 文の働き
10-3	「オブジェクトの配列を渡してその 中身を表示する」関数をメソッドにしない理由
10-4	オブジェクトを格納した配列に対する len の働き
10-5	配列の並べ替え（sort）における key の働き
11-1	長方形クラスを定義し、長方形オブジェクトが表す x,y 各座標の範囲を示す
11-2	（長方形オブジェクトの左端の座標を表示する プログラムを提示し）表示された値の意味
11-3	点オブジェクトが円オブジェクトに含まれるか どうかの判定条件
12-1	(ファイルに書き込むプログラムを提示し） open と write の働き
12-2	（ファイルから読み出すプログラムを提示し） open の引数の変化
12-3	ファイルを書き込みモードで開き、 すぐ閉じたときにファイルがどうなるか
12-4	（ファイルに追記するプログラムを提示し） open の引数の変化
12-5	（csv ファイルを読む場合に使用する） split の働き
12-6	(csv ファイルから文字列を読み込むが 数値型にすべきところを変換していない プログラムにおける）エラーの原因
13-1	データフレームから列，要素を取り出す方法
13-2	データフレームの行数を取り出す方法
13-3	データフレームの行同士の演算の振る舞い
13-4	データフレームの行の絞り込みの方法
13-5	データフレームの並び替えの方法
13-6	データフレームの先頭の数行を取り出す方法