Jupyter Notebookの実行履歴を活用した プログラミング演習の状況把握

○桑田 喜隆[†1] 石坂 徹[†1] 小川祐紀雄[†1] 政谷好伸[†2]
長久勝[†2] 横山重俊[†2†3] 浜元信州[†3]
[†1] 室蘭工業大学
[†2] 国立情報学研究所
[†3] 群馬大学

Knowing the Progress of Programming Exercises by make use of the History of Jupyter Notebook Environment

Yoshitaka Kuwata[†1], Toru Ishizaka[†1], Yukio Ogawa[†1],Yoshinobu Masatani[†2] ,
Masaru Nagaku[†2], Shigetoshi Yokoyama[†2†3], and Nobukuni Hamamoto[†3]

[†1] Muroran Institute of Technology, Japan
[†2] National Institute for Informatics, Japan
[†3] Gumma University, Japan

概要

Jupyter Notebookは対話的なコンピューティング環境として人気がある．筆者らは，プログラミング演習にJupyter Notebookを利用することを提案している．本稿では，実行履歴を分析する方法を提案する．実行履歴を解析することで，演習の進行状況を把握し，支援に役立てることが可能である．また，試行結果を元に実行履歴の解析例を示す．

Abstract

Jupyter Notebook becomes very popular as an interactive environment for computing. The authors proposed to use Jupyter Notebook environment for programming exercise. In this paper, we propose a method of analyzing the interactive experience of users with Jupyter Notebook. By using execution history of the cells on Notebooks, we can estimate the progress and the status of exercises. By making use of the information, we can give advices for students. We also show examples of the analysis.

1. はじめに

1.1 プログラミング演習における状況把握

近年，ICT 教育の一環として，プログラミング的な思考を身につけることの重要性が再認識されている．プログラミング教育において，座学だけでなく実際にプラグラムを作成する演習を行いプログラミングの概念に慣れることは非常に重要である．

筆者らはプログラミング教育方法の改善のため，大学教育を対象にしたプログラミング演習に，クラウド上に構築した Jupyter Notebook[1)]を活用することを提案している[2)]．

図１に筆者らの提案する，クラウドを利用した演習環境の概要を示す．筆者らの提案している環境において，クラウド上に学習支援システム (Learning Management System, LMS)および演習環境として Jupyter Notebook を用意する．学生や教員はキャンパスネットワークを利用し SINET やインターネット経緯でそれらの環境にアクセスすることができる．必要に応じてリソースを確保することで，効率の良い演習が可能になるとともに，教員にとって演習環境の一元管理が容易になるという利点がある．

演習環境を一元的にクラウドに置くことで，従来のパソコン環境で演習を行う場合に比べ，学生の状況を把握しやすいというメリットもある．本稿では，クラウド環境を利用する利点を活かし，

[1] Yoshitaka Kuwata
室蘭工業大学
北海道室蘭市水元町２７－１
kuwata@mmm.muroran-it.ac.jp

個々の学生の演習の実施状況を把握し，指導に役立てる方法について述べる．

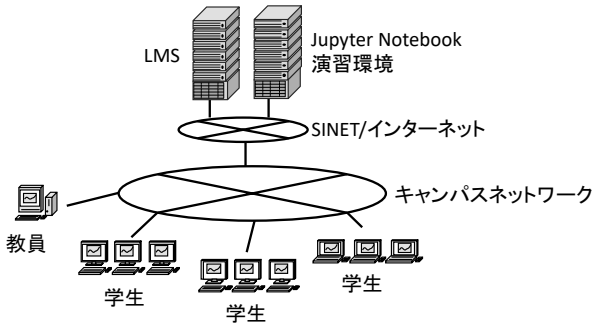

図１ クラウドを利用した演習環境

1.2 プログラミング教育に関する3つの仮説

筆者らは，プログラミング教育の経験から以下の３つの仮説を置き，検証することを計画している．

演習中の試行錯誤でプログラミングに関する理解が進み，知識が深まると考えている．

(1) 座学に加え演習を実施することで学生の理解度が向上する

他方，概念を理解しないまま次の演習に進んでしまい，次の演習ではますます分からなくなるというリスクがある．

(2) 一般的に，演習が進むほど学生の理解度が低下する

そのため，学生の理解度を把握し適切な助言や指導を行うことが重要であると考える．

(3) 理解度に応じた指導を行うことで，理解度の低下を防ぐことができる

(1)-(3)を検証するために，定量的な評価が重要になると考えている．また，定量的に演習の進行状況を把握することで，(3)の理解度に応じた指導を実現することが必要であると考える．

1.3 関連研究

これまでにも，プログラミング環境の進捗を行うシステムが提案されている．

例えば，参考文献３で内藤らはプログラミングの進捗把握し，学生の指導に役立てるシステムを提案している．

また，参考文献４で長谷川らはグループ学習を対象に進捗業況を視覚化し，教員や TA による指導を支援するシステムを提案している．どちらも，進捗の把握にために専用のシステムを設計して演習およびその評価に利用している．このため，他の分野などでの検証が難しいという課題がある．

2. Jupyter Notebook を活用したプログラミング演習方法の提案

2.1 従来の演習環境

伝統的なプログラミング演習では，演習課題を LMS や紙などで出題し，学生がパソコンで演習を実施する方法がとられている．図２に従来のプログラミング環境を使った演習方法を示す．

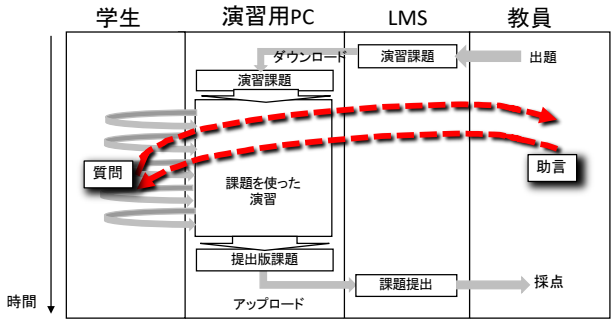

図２ 従来のプログラミング演習環境

演習時間中に発生した不明点は，教員に質問することで解決する．また，教員は教室の中を回って，学生の画面を覗きこむことで進捗の状況を把握していた．演習に利用する PC の利用状態を直接知ることはできないため，質問のあった項目以外でどのような誤りが生じているかを正確に知ることは出来ない．

学生は，課題終了時に作成したプログラムやその出力結果などを教員に提出し，その後，教員が評価を行う．このため，作成したプログラムの誤りは演習中に指摘することができないという課題がある．

2.2 提案する演習環境(Jupyter Notebook)

Jupyter Notebook を利用した演習環境を図３に示す．

演習環境を Jupyter Notebook で一元管理することで，学生の状態を把握することが可能になる．

教員は直接演習環境に演習課題を出題することが可能である．また，学生の提出用に保存した課題を評価することが可能である．同時に，実行履歴が記録されるように設定することで，演習の状況をモニタリングし，必要に応じて助言することが可能となる．

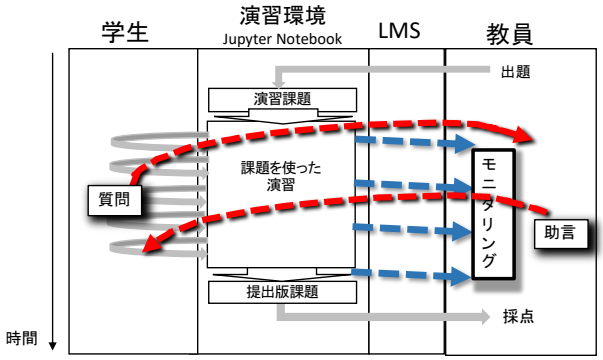

図3　Jupyter Notebook を利用した演習環境

3. Jupyter Notebook による実装方法

3.1 Jupyter Notebook の構造

Jupyter Notebook で扱われる文書は Notebook と呼ばれ，一つ以上のセルから構成される．セルは実行可能な Code セルと，構造化された文書を記述できる Markdown セルに大別される．

図4に Notebook の例を示す．

図4　Notebook の例

(1) Markdown セル

Markdown セルには，TeX のような構造化文書を記述する．Markdown セルは Jupyter Notebook によって整形される．

プログラミング演習では，演習内容の説明や解説を記述する．また，学生による文章の回答などにも利用することがある．

(2) Code セル

Code セルに入力された式は，Python などの言語処理系に渡され，評価が行われる．プログラミング演習では，Code セルにプログラムを記述する．言語処理系から式の評価結果が返された場合にはセルの下に表示される．エラーがあった場合にも評価結果として表示される．

プログラムによる印字やグラフ出力などがある場合には，セルの下にインライン出力される．

3.2 Jupyter Notebook の拡張

以下の Jupyter Notebook の拡張機能を導入した．

(1) セルの ID 管理

演習に使う Notebook のセルにユニークな ID を付与して，どのセルを評価したかがわかるようにする．この機能により，学生に配布する Notebook の，同一課題の同一セルを一意に特定できるようになる．

(2) セルの実行記録

セルの入力式および式の評価結果，出力結果を，セル評価のたびにファイルに記録する．

また，後の解析の目的で Notebook 自身にも実行の記録が残され，Notebook の一部として保存される．

ソフトウェアの詳細については，4.3 節に記載した．

4. 評価実験の方法

4.1 評価実験の条件

表 1 に評価実験の条件を示す．

概念検証を目的とした予備実験の位置付けとした．混乱を防ぐため，プログラミングの知識を持った教員を被験者とした．

表 1　実験の条件

項目	設定	数
課題	プログラミング入門の課題 (課題 1-課題 3)	3 回分
対象言語	Python	–
被験者	プログラミングに関する知識を持つ教員 (User02-User05)	4 名
実施方法	ブラウザ経由で演習課題へ回答 途中の質問や説明はなし 各自ばらばらに実施	–
分析対象データ	セルの評価ログ，出力結果 完了後の課題ファイル	–
集計項目	セルの評価時刻および回数 セルの評価結果 (エラー有無) 演習課題の回答	–

4.2 出題した課題内容

表2に課題の概要を示す．

実際の演習で想定している課題に沿ったものを出題した．

表2　課題の概要

番号	表題	Code セル数
課題1	イントロダクション	28
課題2	プログラムの構造	23
課題3	リストと繰り返し	30

演習課題の例として，図5に課題1の最初の部分を

示す．

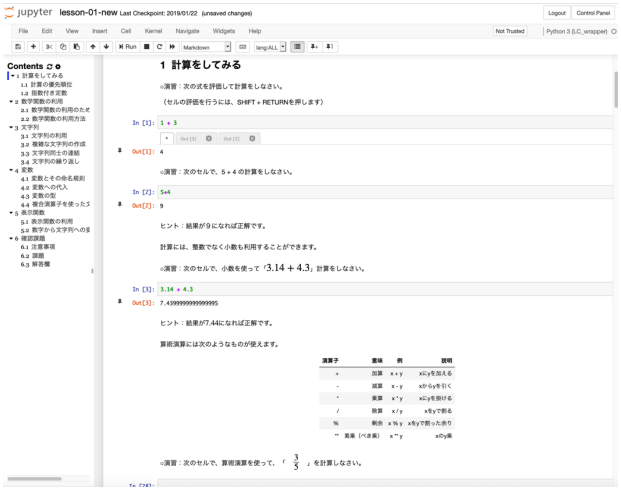

図 5 演習課題例(課題 1)

4.3 実験環境

表 2 に実験に利用した演習環境の詳細を示す．

本実験でデータを取得するために利用した Jupyter Notebook の拡張機能は，クラウドの運用管理などの応用向け（参考文献 5）に開発されたものである．一般向けに公開されており，Github のレポジトリから取得可能である．

表 2 実験に利用した演習環境

ソフトウェア項目	名称（バージョン）	備考
演習環境	Jupyter Notebook (4.4.0)	
言語処理系	Python (3.6.7)	
主な Jupyter Notebook の拡張機能	JupyterHub (0.9.4)	複数ユーザの管理
	Jupyter-LC_wrapper (Dec 28, 2018)[2]	セル評価の記録
	Jupyter-LC_nblineage (Mar 17, 2018)[3]	セルの ID 管理
	Jupyter-multi_outputs (Oct 16, 2018)[4]	複数出力結果の管理
	Jupyter-LC_run_through (Jun 15, 2018)[5]	一括実行の管理
ハードウェア項目	スペック	備考
CPU	Core-i3 8300	
メモリ	40GB	
ディスク	500GB SSD	
OS	Ubuntu 18.04.1	

[2] https://github.com/NII-cloud-operation/Jupyter-LC_wrapper
[3] https://github.com/NII-cloud-operation/Jupyter-LC_nblineage
[4] https://github.com/NII-cloud-operation/Jupyter-multi_outputs
[5] https://github.com/NII-cloud-operation/Jupyter-LC_run_through

5. 実験結果

5.1 Code セルの総評価回数

演習が完了するまでに，Code セルの評価状況を利用者ごとに分析した．課題 1 の集計結果を表 3 に示す．

表 3 利用者ごとの集計（課題 1）

利用者	評価回数	回答セル数	回答率	エラー
User02	34	27	96%	2
User03	48	26	93%	14
User04	36	28	100%	2
User05	32	28	100%	1

全体としては評価回数に極端に大きなばらつきは認められなかった．回答が難しい Code セルがあった場合には，試行錯誤により評価回数が増加することが予測される．本評価実験では，被験者がプログラミングに関する知識を持っていたため，スムーズに回答を進められたと考えられる．他方，同じ Code セルの値を変更して何度か評価して試させる出題の場合には，当該の Code セルの評価回数が増加する．

また，Code セルの回答率は 100%になっておらず，一部に未評価の Code セルがあった．

より多くの利用者の分析を行うためには，上記の指標をヒストグラムなどで表示することが必要になると考えられる．

5.2 Code セルごとの評価数

Code セルごとの評価回数を集計した．結果を図 6 に示す．各バーは Code セルの 4 名による評価数を示しており，結果にエラー含まれる場合にはオレンジ色で示している．また，バーは下から ID の若い順番に並んでおり，下のバーほど課題の上位に位置する Code セルを示している．最上位のバーは，課題最後の Code セルの評価回数を示している．

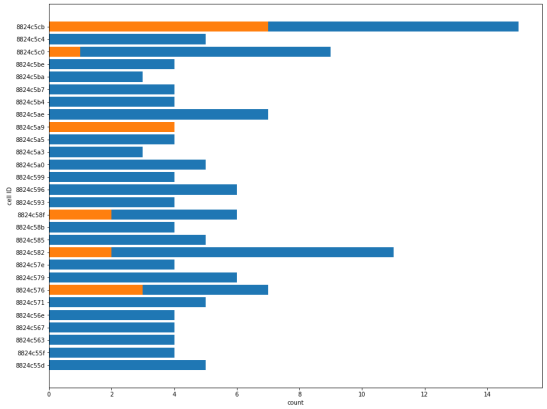

図 6 Code セルごとの評価数とエラー数(課題 1)

3 回しか評価されていない Code セルは，被験者の少なくとも一人が回答しなかったことを意味する．これは回答率が 100%に達していない事実と符合する．

最後の Code セルの評価回数が多い原因としては，まとめ課題を出題しているためである．正しい回答をするためには試行錯誤が必要になり，当該 Code セルの評価回数およびエラーが多くなったと考えられる．

5.3 利用者ごとの Code セル評価履歴

利用者ごとに Code セルの評価履歴をプロットした．図 7 に課題 1 の分析結果を示す．

縦軸は Code セルの ID を順番にプロットしたものであり，横軸は評価した時刻を示している．開始時刻から終了時刻までを正規化して表示しているため，横軸の表示時刻および時間間隔は利用者ごとに異なる点に注意されたい．

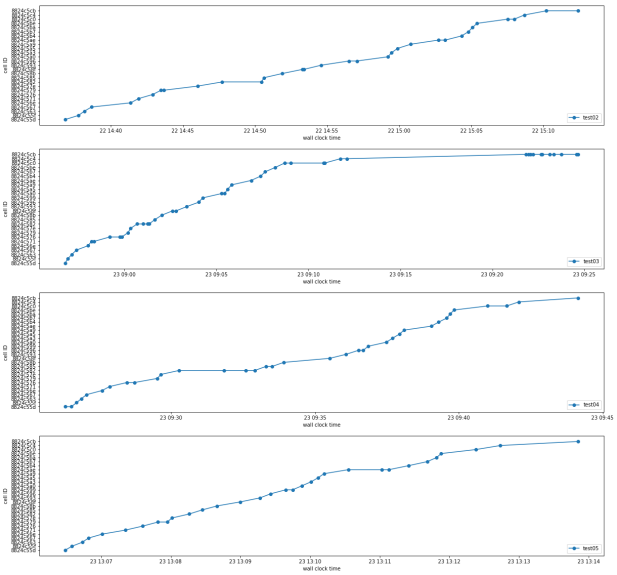

図 7 利用者ごとの Code セル評価履歴（課題 1）

本グラフを見ると，利用者の評価順序よび評価時間の間隔が分かる．この例では，全てのグラフが右上がりになっており，全員が課題に記された Code セルの順番に評価していることが分かる．仮に，課題の途中から開始する利用者がいた場合には，グラフは途中から始まる形状となる．また，最後の課題で評価間隔が長くなっており，回答に時間をかけていることが分かる．

図 8 は，図 7 と同じデータを，開始時刻からの相対時刻として同じ時間軸上で示したものである．授業時間内に，一斉に演習を行う場合には，このようなデータが得られる．

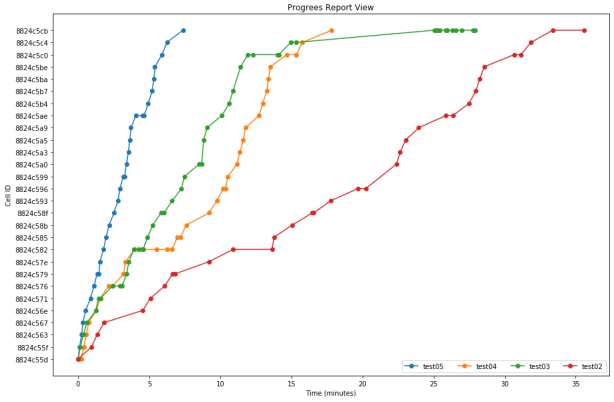

図 8 利用者ごとの Code セル評価履歴(課題 1,相対時刻表示)

グラフの傾きは演習実施の速さを示しており，利用者ごとに大きく異なっていることが分かる．User05 が最も速く，開始から 7 分程で完了しているのに対し，最も遅い User02 は 26 分程かかっている．

グラフの下から 10 番目の Code セル(8824C582)で 3 名の利用者が複数回の評価を行なっており，そのため時間も要している．当該セルは，回答の難しい Code セルであることが伺える．

6. 考察

6.1 リアルタイムの進捗把握への利用

図 9 は演習後にデータをプロットしたものであるが，演習中にデータを収集し，このグラフを描画することで，教員が演習の実施状況をリアルタイムに知ることが可能である．

また特定のセルによって進捗が遅くなっている場合には，演習中に補足説明を行うなどのフォローをすることができる．

学生間の相対的な演習速度の違いも分かるため，遅い学生に個別に指導するなどの方策も考えられる．但し，演習が遅いからといって必ずしも理解ができていないわけではないため，扱いに注意することが必要である．逆に，速すぎる学生は何か不正な行為を行なったり，誤った答えを適当に入れている等の可能性もあるため，注意が必要である．

6.2 課題評価への応用

今回のように，予め教員自ら教材の演習をやってみることで，問題点が見つかる場合がある．

例として, 図 9 に課題 2 の進捗評価結果を示す．

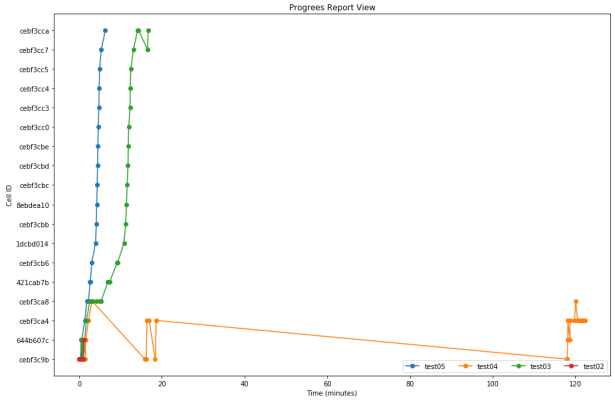

図 9 利用者ごとの Code セル評価履歴(課題2，相対時間表示)

この例では，(A) 2 人の利用者が全部の Code セルを評価していない．(B) User04 は最初の 3 セル間を行き来している．

そこで，利用者ごとの進捗を個別に図 10 で分析した．

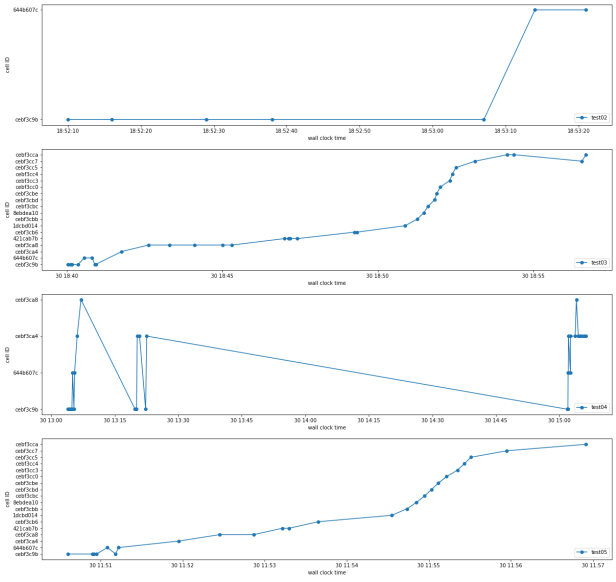

図 10 利用者ごとの Code セル評価履歴(課題 2)

課題の最初の部分で User02 および User04 に何かトラブルが生じたことが分かる．User02 はその後のセルの評価記録が残っていない．User04 はしばらくたってから評価を再開しているが，同じセルで問題が発生していることが分かるため．

後に調査したところ，Python の input()関数を利用する課題の場合，利用者が入力せずに別のセルの評価を行うと，Jupyter Notebook が反応しなくなるということが判明した．演習の場面では，問題が発生しやすい箇所であり，対応方法を事前に検討しておくこととした．

このように，事前に評価を行うことで課題自身の不具合や注意点を発見するためのツールとして利用可能であることが分かった．可能であれば，事前に少数の学生に実施してもらうことでより実際に近い問題点が発見できると考える．

また，意図して後のセルほど難しい課題を出題した場合，意図した結果となったかどうかの確認を行うという利用方法も考えられる．

7. まとめと今後の課題

本稿では， Jupyter Notebook を使った演習教材を利用し，予備実験として教員が演習を行いその履歴を分析した．その結果，演習の進捗状況をリアルタイムに把握する方法として，Code セルの評価履歴が役に立つことが分かった．また，事前に課題を評価することで，課題自体の問題点を発見する手法としても有効であることが確認できた．

今後の課題として，プログラミングの知識を持たない学生による演習の履歴を用いて，提案した分析手法の有効性を確認することが必要であると考える．

また，実際の授業では 160 人以上の学生が同時に演習を実施するため，個別の状況を確認することが難しくなる．このため，クラス全体の進捗状況を示す指標を新たに導入することが必要であると考える．

本研究は JSPS 科研費 (JP18K11561)の「クラウドを活用したプログラミング演習環境に関する研究」の助成を受けたものである．

A. 参考文献

1. Project Jupyter, Project Jupyter Homepage, http://jupyter.org/ (2019/2/14 参照)
2. 桑田喜隆，石坂 徹,政谷好伸，長久勝，横山重俊，浜元信州，クラウドを利用した対話的なプログラミング教育環境とその評価手法の提案，第 2 3 回 人工知能学会 知識流通ネットワーク研究会，2018 年 9 月 28 日
3. 内藤広志，斉藤隆，プログラミング演習のための進捗モニタリングシステム，情報処理学会研究報告 2008-CE-93 (5)， 2008 年 2 月 16 日
4. 長谷川伸，松田承一，高野辰之，宮川治，プログラミング入門教育を対象としたリアルタイム授業支援システム，情報処理学会論文誌, Vol. 52 No.12 Pp. 3235-3149, 2011 年 12 月
5. Literate Computing for Reproducible Infrastructure, https://literate-computing.github.io/ (2019/2/18 参照)