学術情報基盤　オープンフォーラム2019
クラウド活用最新事例

クラウドとJupyter Notebookを使ったプログラミング教育

2019年5月29日

室蘭工業大学
桑田喜隆

国立大学法人
室蘭工業大学
MURORAN INSTITUTE OF TECHNOLOGY

概要

1. はじめに
2. プログラミング教育に関する仮説
3. クラウドを利用したプログラミング環境
4. Jupyter Notebookとは
5. 評価実験
6. 考察
7. まとめと今後の課題

1. はじめに

1. プログラミング教育が注目される
   - 新学習指導要領（情報教育・ICT活用教育関係）2017

   - 小・中・高等学校共通のポイント
   - 情報活用能力＝学習の基盤となる資質・能力
   - プログラミング的思考

2. 現状（工学系大学）
   - プログラミング未経験者
   - 2割程度が「苦手」
   - 専門科目の履修に必要なコースもあるが、言語はバラバラ

3. 課題
   大学の一般教養として何を身につけるべきか？

情報教育の強化

- 2019年度より工学部から理工学部へ改組

室蘭工業大学では、大学が改組され、2019年度より理工学部全学科でプログラミングが必修科目となる。

情報教育を厚く

人工知能やIOTなど、情報処理技術における技術革新が進み、これからの技術者には情報技術を身につけていることが求められます。室蘭工業大学は、情報科目を充実し全ての学生が学べるようにカリキュラムを整えました。これにより情報とデータに関わるリテラシー、情報セキュリティー、プログラミングの基本的な能力、統計処理能力を身につけます。

プログラミング入門
(2019後期開講予定)

- シラバス

対象学年	1年
授業科目区分	学科共通科目
必修・選択	必修
授業方法	講義、演習
単位数	2
到達度目標	本講義では、全コースの学生を対象に、プログラミングに必要な概念を理解し、基礎的なプログラムを作成することができるようになることを目標とする。

2. 演習をめぐる仮説

【仮説1】座学＋演習で学生の理解度が向上する

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.

【仮説2】演習が進むほど学生の理解度が低下する

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リスクがある.

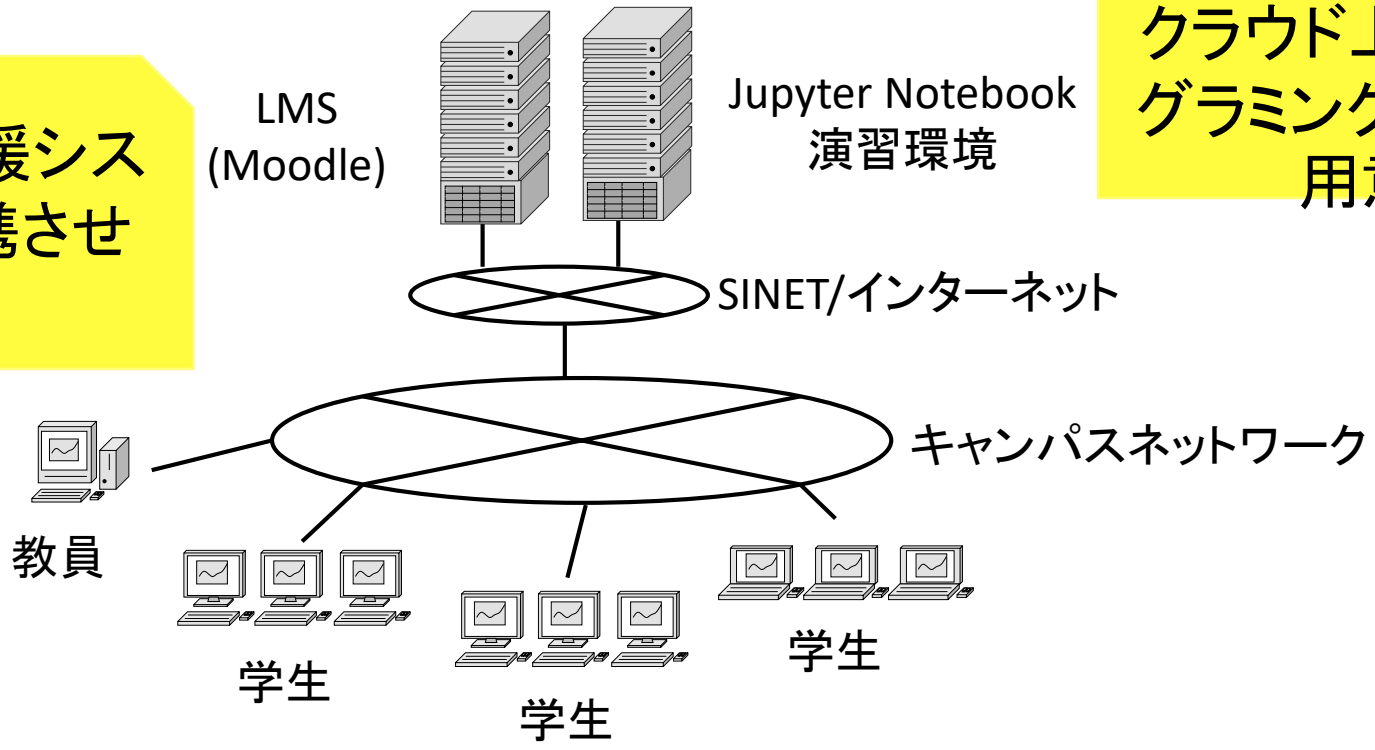
【仮説3】理解度に応じた助言で，理解度低下を防げる

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.

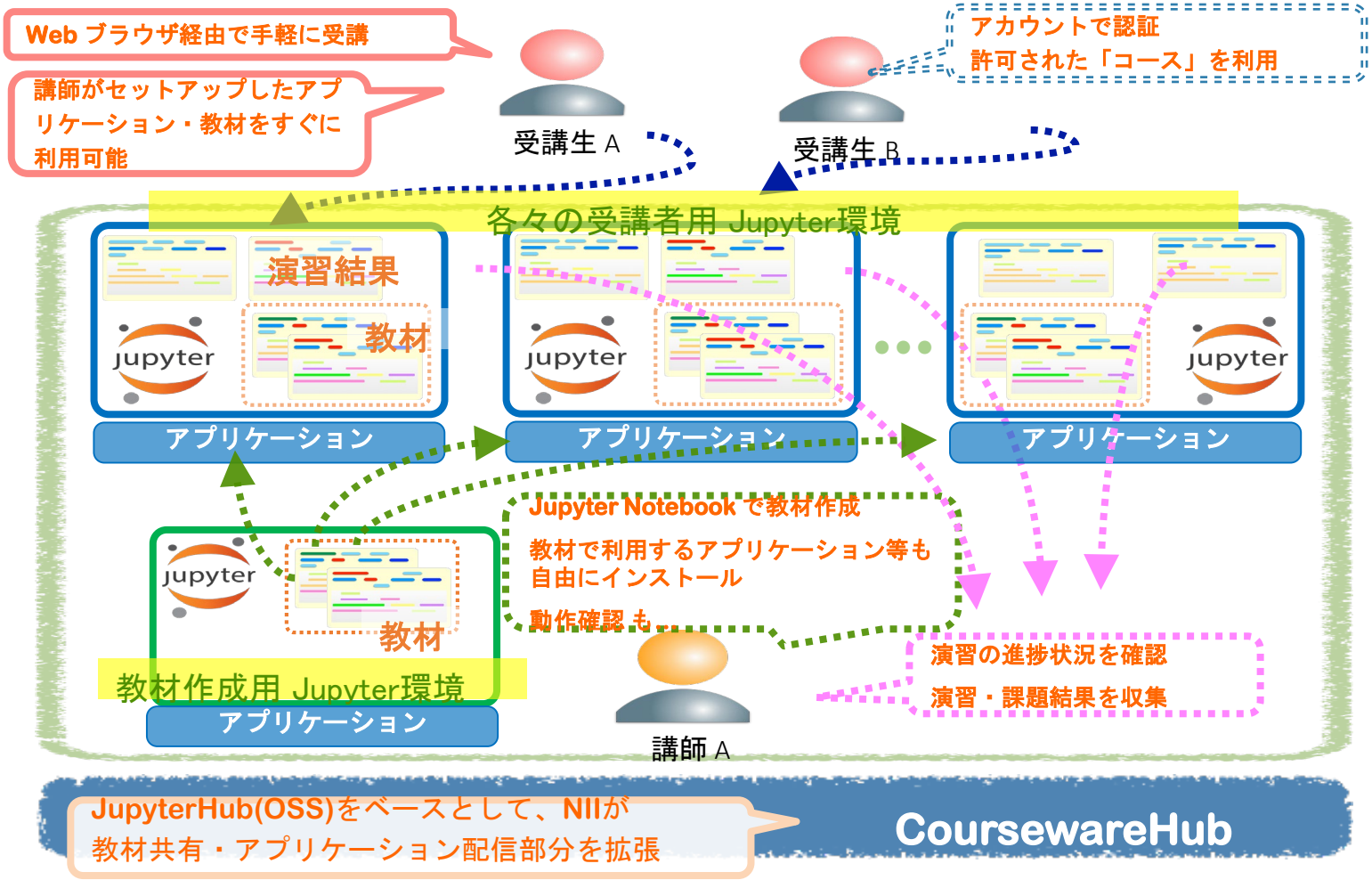
3. クラウドを利用したプログラミング環境

①授業にあわせて
クラウド上にプログラミング環境を用意

②学習支援システムと連携させる



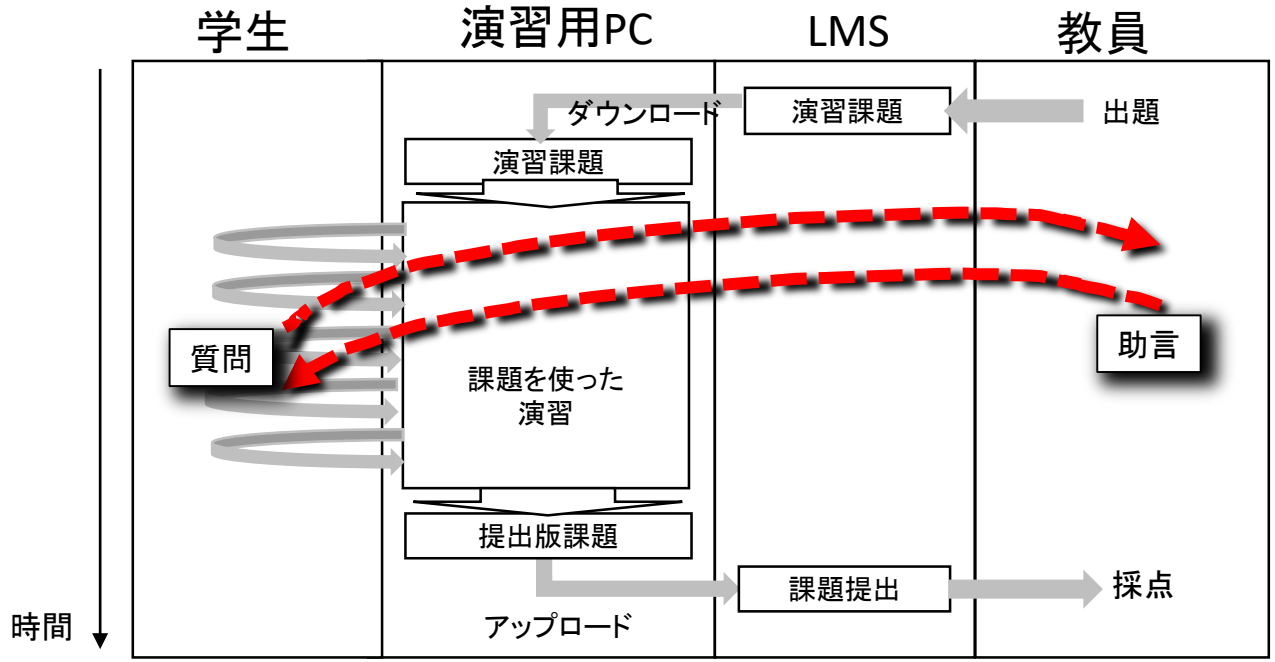
CoursewareHubの活用



長久　勝、政谷 好伸,合田 憲人,「Notebookによる講義・演習環境の開発」IPSJ 研究報告 教育学習支援情報システム（CLE）,2019年3月13日より引用

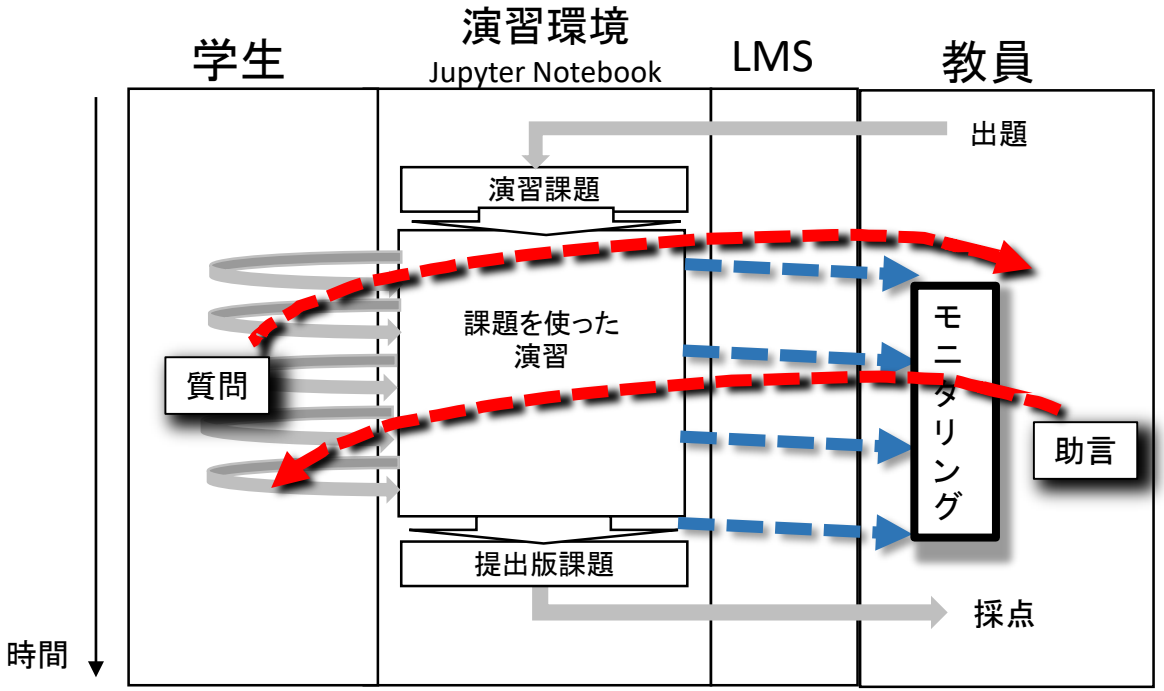
従来の演習環境

- 学生の演習用PCの状況（進捗状況やエラーなど）は分からなかった。



提案する演習環境

- 演習環境を一元管理することで、状況のモニタリングが可能になる。



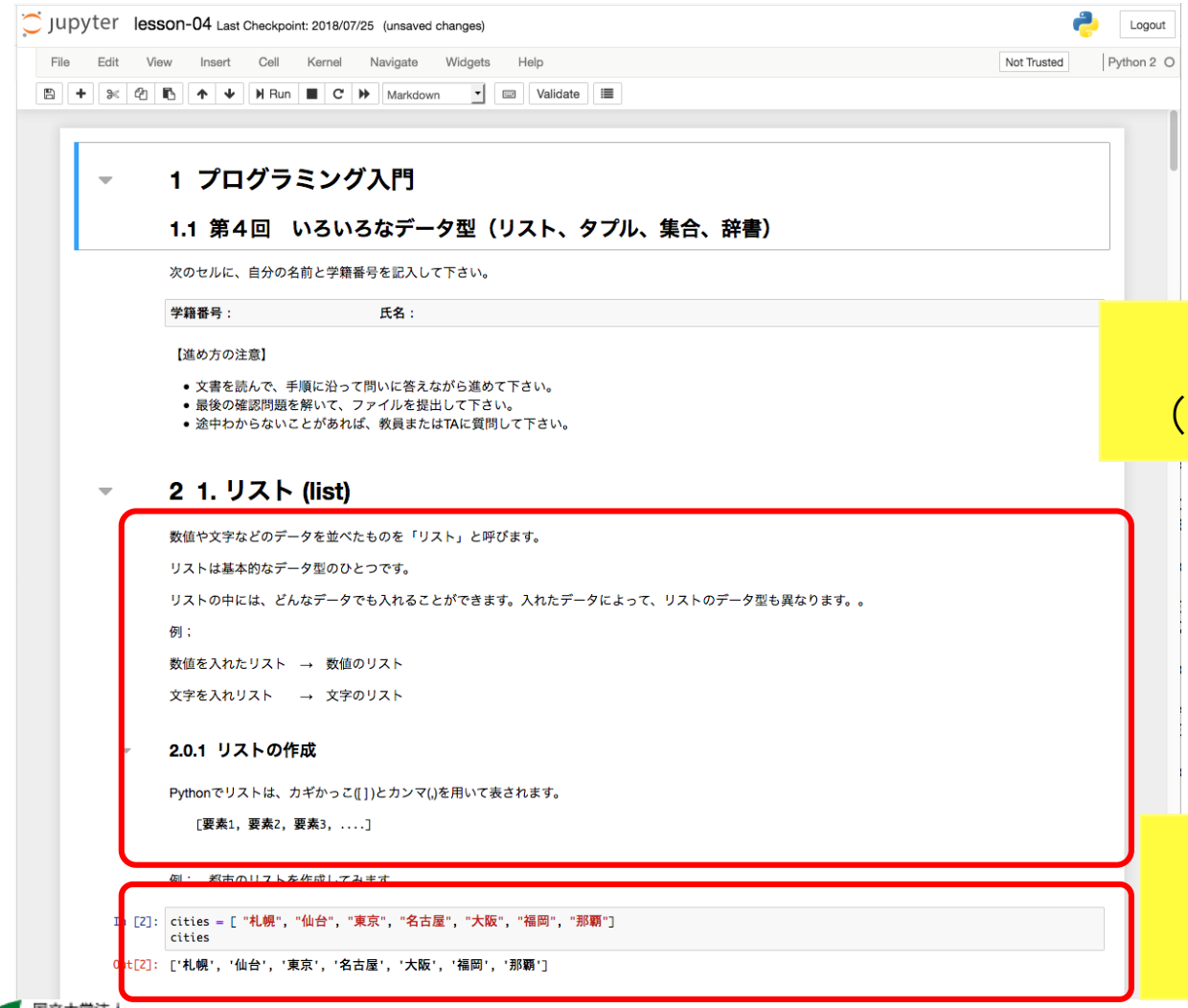
4. Jupyter Notebookとは

Webベースの対話的な計算機環境
科学技術計算やデータ解析，データの可視化で利用が広がっている.

【特徴】

- ブラウザから計算を行う
- Python, R, Julia, Scalaなど複数の言語処理系がサポートされている
- プログラムを作らなくても，大規模計算や機械学習などのライブラリを利用して計算をえる.
- 結果の可視化
  - 説明をマークアップ言語で記述
  - グラフや画像イメージ，ビデオなどをWebページ上にインラインで挿入する
  - 計算結果を直感的に理解しやすい形式で記録することが可能である.
- 作成したコンテンツ(Notebook)は，さまざまな形式で公開し，第三者と共有可能である.

Jupyter Notebookの画面例



説明用
（マークダウンセル）

プログラム、式
(コードセル)

Jupyter Notebookを使った教材の構成

1. 説明（マークダウンセル）
   - 演習内容やその前提知識の説明を，マークダウンセルに記述する．
   - 学生の理解補助のため，数式や図を併用する．
2. 例題（記述済みのコードセル）
   - 例題として，記述済みのコードを含むセルを用意する．
   - 学生はコードをその場で実行して，結果を確かめることができる．
   - 故意にエラーの出るコードを用意しておき，学生に原因を考えさせる応用も考えられる．
3. 穴埋め問題（部分記述済みのコードセル）
   - コードの一部のみを記述しておき，未記入の部分を学生が記述することでプログラムを完成する．
   - 複数のステップ（セル）に分割してコードを作成する．
4. 自由回答（未記入のコードセル）
   - 自由にコードを書けるように，未記入のセルを用意する．
   - これまでの説明を理解したかどうかを確認するために利用すると効果的である．
   - 単元ごとに設ける提出課題は未記入のコードセルを利用する．

5. 評価実験

【位置付け】

・本格評価の前の事前評価

【目的】

- Jupyter Notebookの有効性評価
- 提案手法の有効性評価
- 教材の評価

実験に利用した環境

ソフトウェア項目	名称（バージョン）	用途
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)	セル評価の記録
	Jupyter-LC_nblineage (Mar 17, 2018)	セルのID管理
	Jupyter-multi_outputs (Oct 16, 2018)	複数出力結果の管理
	Jupyter-LC_run_through (Jun 15, 2018)	一括実行の管理

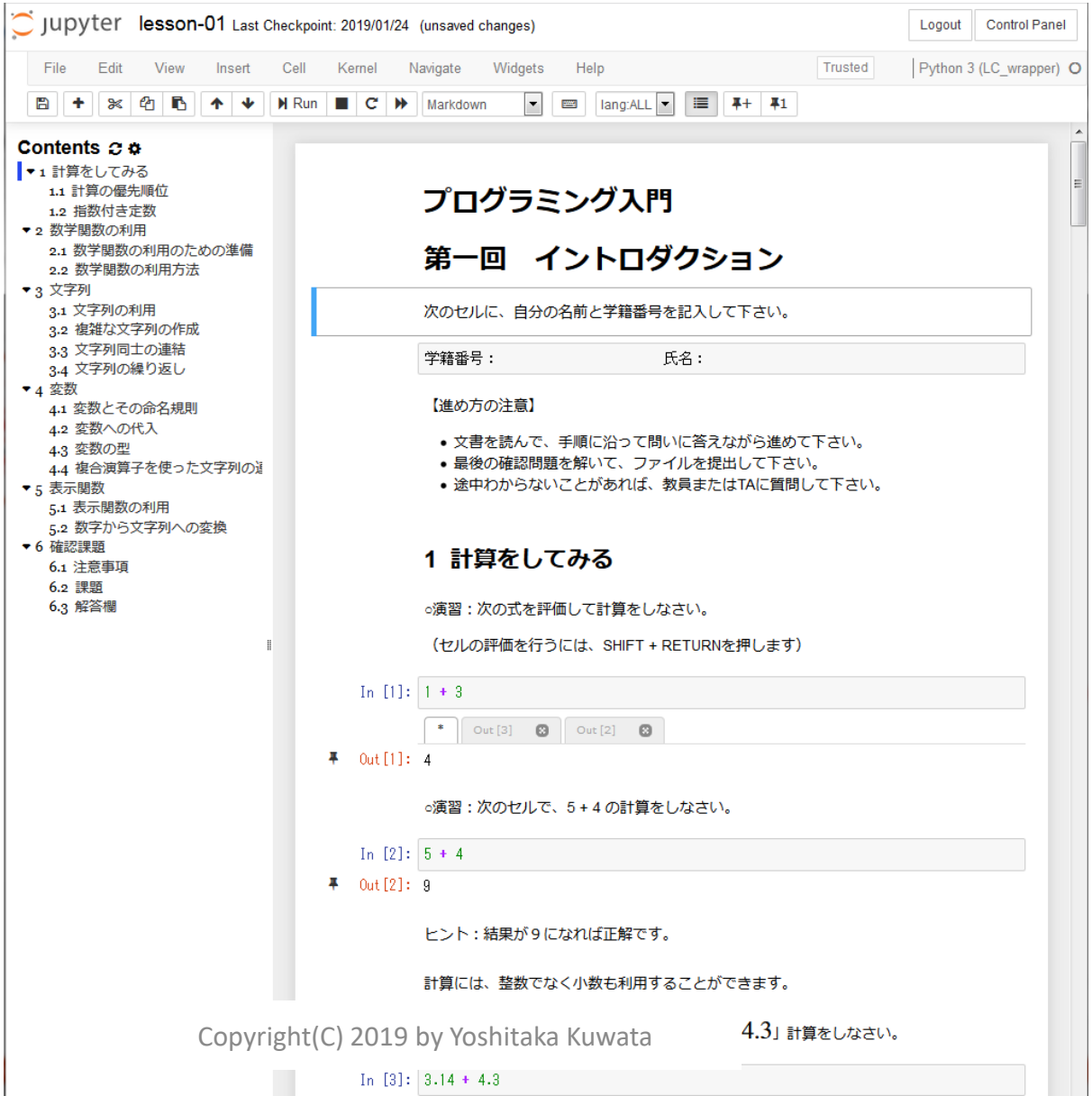
試行実験の概要

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	–

実験に利用した課題の概要

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

課題の例

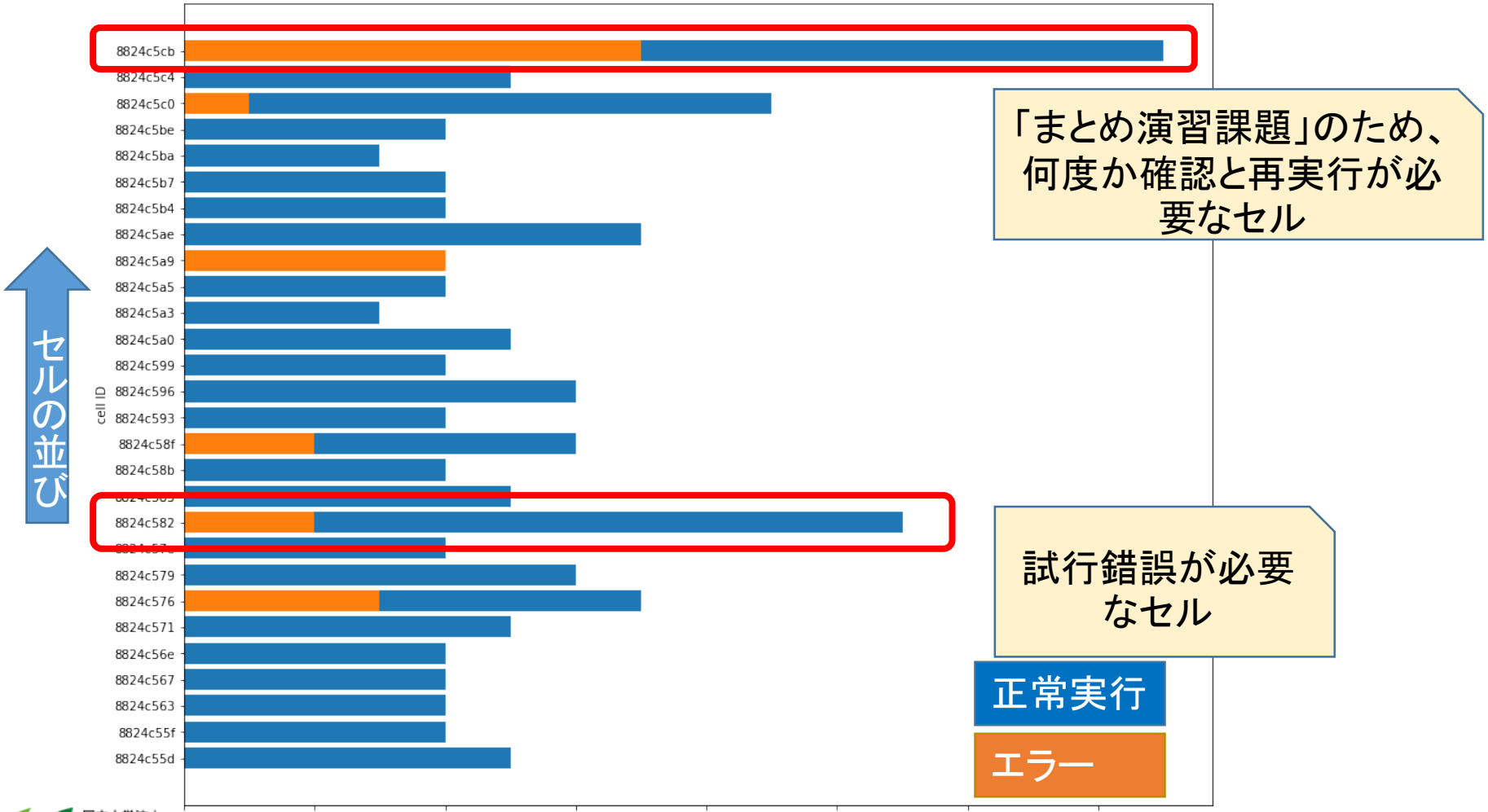


結果の概要（課題１）

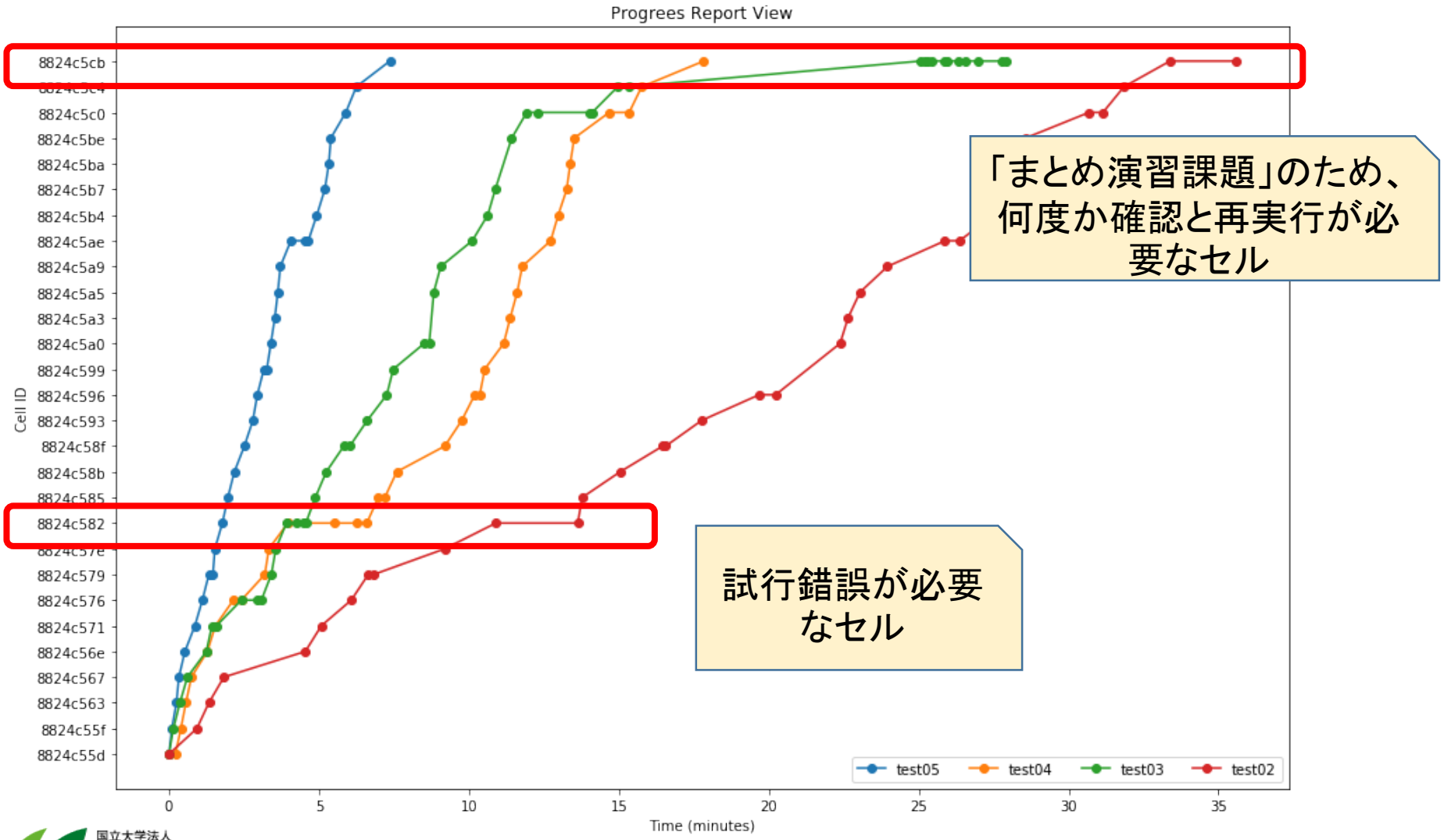
利用者	評価回数	回答セル数	回答率	エラー
User02	34	27	96%	2
User03	48	26	93%	14
User04	36	28	100%	2
User05	32	28	100%	1

- 評価数に極端な偏りはないものの、２割程度のばらつきがある。
- 回答率（セルのカバレッジ）は１００%になっていない

Codeセルの実行回数とエラー数（課題1）



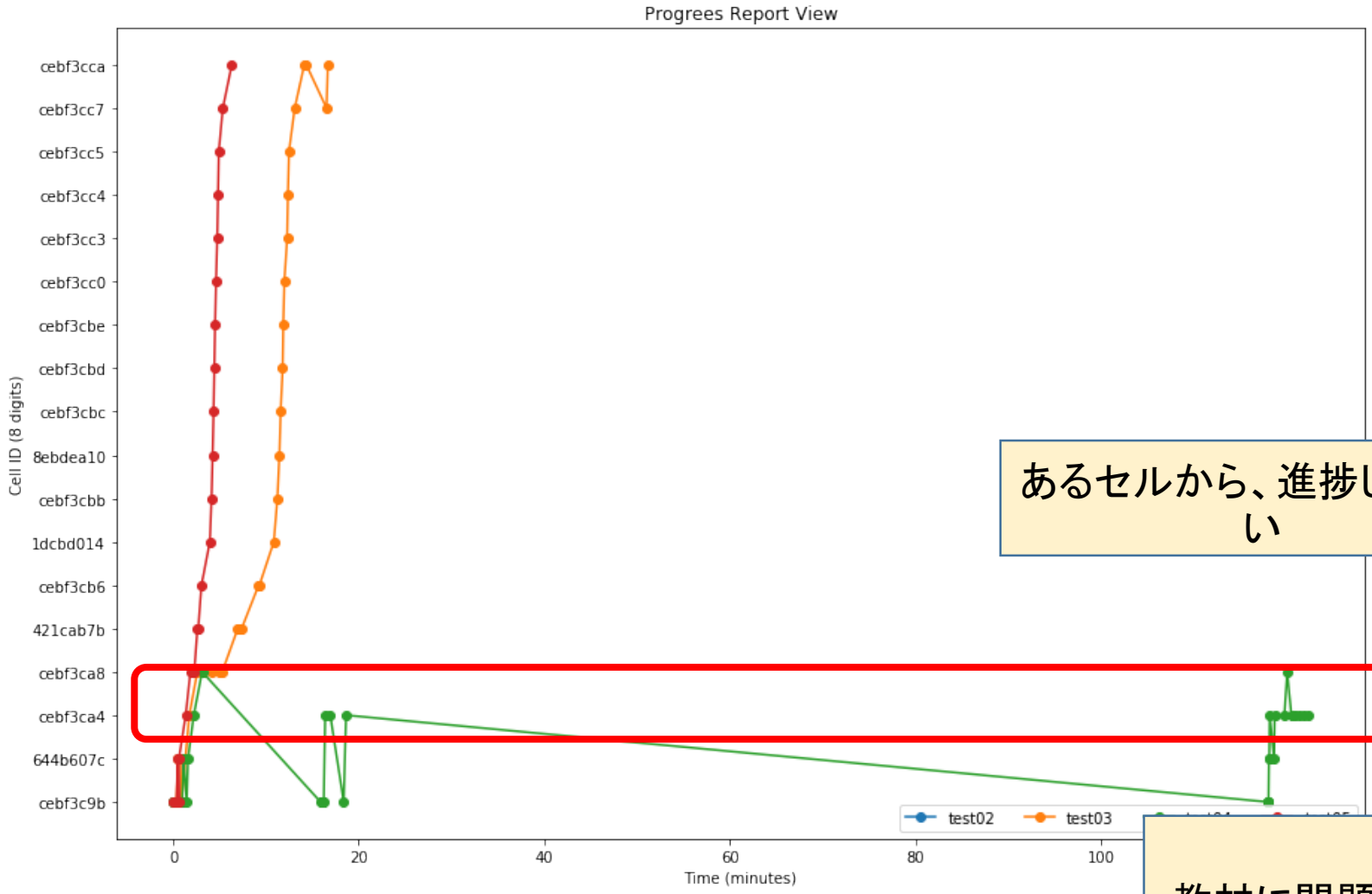
利用者ごとのCodeセルの評価履歴（課題1）



6. 考察1：フィードバックの例

ケース	想定される原因	授業内対応	授業後対応
特定のセルを繰り返して評価	• 該当部分が理解できずに試行錯誤している • 理解できずに先に進めない	• 全員への該当箇所の補足説明 • 個別指導	教材該当箇所の改善（説明追加）
エラーが多い			
セル評価の間隔が長い			教材該当箇所の改善（問題分割など）
想定より進捗が遅い	分量が多い	•残り部分を宿題とする •個別指導	教材量の見直し

考察2：利用者ごとのCodeセルの評価履歴（課題2）



あるセルから、進捗していない

教材に問題がある

7. まとめと今後の課題

【まとめ】

- Jupyter Notebookによるプログラミング演習の提案
- Courceware Hubなどのクラウド環境の活用
- Jupyter Notebookを使った演習教材の予備実験の解析

→課題自体の問題点を発見する手法としても有効.

【今後の課題】

- プログラミングの知識を持たない学生による試行
- 大規模授業の際の支援方法の確立

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