Title	Pythonスクリプト等によるオンライン課題の採点・コメント返却システムの構築 : 特に論文形式のレポートの採点フィードバックについて
Sub Title	Construction of an efficient scoring-feedback system of students' reports submitted online
Author	上村, 佳孝(Kamimura, Yoshitaka)
Publisher	慶應義塾大学日吉紀要刊行委員会
Publication year	2022
Jtitle	慶應義塾大学日吉紀要. 自然科学 (The Hiyoshi review of natural science). No.69 (2022. 10) ,p.53- 75
JaLC DOI	
Abstract	The COVID-19 pandemic due to spread of SARS-CoV-2 infections caused drastic changes of the college life in Japan. Providing appropriate feedbacks to students' reports is particularly challenging to develop online lectures. A series of Python scripts were written to develop an efficient flow of formatting–scoring–commenting students' reports submitted online. In conjunction with the functions (and extensions) of commercially available software (Adobe Acrobat and Microsoft Excel), they enable us: (1) to format students' answers into a single sheet for scoring, (2) to automatically prepare multiple scoring sheets for each student, (3) to merge/divide PDF files of student reports (to be scored or after scoring), and (4) to return the teacher's comments (in text format or as an attachment file) individually via e-mail. Students' responses to this system are also reviewed briefly.
Notes	教育
Genre	Departmental Bulletin Paper
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=AN10079809-20221030-0053

Hiyoshi Review of Natural Science
Keio University No. 69, 53-75（2022）

Python スクリプト等によるオンライン課題の採点・コメント返却システムの構築
――特に論文形式のレポートの採点フィードバックについて――

上村佳孝

Construction of an Efficient Scoring-Feedback System of Students' Reports Submitted Online

Yoshitaka Kamimura

Summary――The COVID-19 pandemic due to spread of SARS-CoV-2 infections caused drastic changes of the college life in Japan. Providing appropriate feedbacks to students' reports is particularly challenging to develop online lectures. A series of Python scripts were written to develop an efficient flow of formatting–scoring–commenting students' reports submitted online. In conjunction with the functions (and extensions) of commercially available software (Adobe Acrobat and Microsoft Excel), they enable us: (1) to format students' answers into a single sheet for scoring, (2) to automatically prepare multiple scoring sheets for each student, (3) to merge/divide PDF files of student reports (to be scored or after scoring), and (4) to return the teacher's comments (in text format or as an attachment file) individually via e-mail. Students' responses to this system are also reviewed briefly.

1. はじめに

2019 年 12 月に初めての症例が中国・武漢から報告され，その後世界へと広がった SARS コロナウイルス 2 による Covid-19 の世界的パンデミックは，本稿執筆の 2022 年 7 月の時点でまだその終息を見通すことができていない。このパンデミックによる悪影響（以下，「コロナ禍」とする）は当然教育にもおよび，特に 2020 年度は，4 月の年度開始から多くの大学でオ

慶應義塾大学生物学教室（〒 223–8521　横浜市港北区日吉 4–1–1）：Department of Biology, Keio University, 4–1–1 Hiyoshi, Kohoku-ku, Yokohama, 223–8521, Japan. E-mail: kamimura@keio.jp

ンラインでの授業開講を迫られ，筆者も授業用動画の作成に追われることとなった。

オンラインでの授業において特に問題となるのは，その双方向性の確保である。慶應義塾大学日吉キャンパスでは，文系学部（文学部・経済学部・法学部・商学部）の主に1・2年生を対象に，生物学Ⅰ・生物学Ⅱが開講されており，筆者を含め，同学部所属の生物学を専門とする教員が担当している。しかし，文系学部の学生という背景もあり，高等学校卒業までの生物学の学習の程度や，科学全般に対する理解や興味関心の程度には個人差が大きい。そのため，オンラインでも質問し易い環境の提供とともに，課題やレポートの得点やコメントを，効率よく各学生に返却し，適切なフィードバックをおこなうためのシステムの構築が必要となった。

Python は 1991 年の誕生以来，世界中で広く使用されているプログラミング言語である。大規模なシステムの開発にも使用されているが，簡潔な短いスクリプトで様々な業務を効率化することを得意としている（ルバノビック　2015）。スクリプトを毎回コンパイルすることなく実行し，その結果をすぐに確認することができるため，開発の手間が少ない。また，利用者が多い言語であることから，膨大な数のモジュールが開発されている。Python モジュールは，よく使う命令（関数）を集めたものであり，関連したモジュールを集めたパッケージとともにライブラリと呼ばれることもある。PDF ファイルの操作，Microsoft 社の Word・Excel 形式ファイルの操作，E-mail の送受信などに関して，既存のモジュールを活用することで，有益なツールを手軽に開発することができる（ルバノビック　2015；スウェイガード 2017）。

本稿では，上記のような必要性に迫られる中で筆者が構築した，オンラインで提出された課題やレポートを効率よく採点し，得点やコメントを各学生に個別返却するシステムについて解説したい。市販の表計算ソフトや PDF 作成・編集ソフトの機能に依存している部分も多いが，それらに短い Python スクリプトを組み合わせて一連の流れとすることで，作業効率が劇的に改善し，人的エラーも減らすことができた。

具体的には，図 1 に集約されるような，以下の諸操作の実装について解説していく。

＜記号選択や短文などの小問形式の提出課題の場合＞

・個々の学生から提出された課題の解答を，採点しやすい 1 枚のスプレッドシートに自動でまとめる（図 1–①）。

＜報告書形式の提出課題の場合＞

・個々の学生からオンラインで提出されたファイルを PDF に変換し，その各ページに，学籍番号をヘッダーとして付す。さらに，1 つの PDF ファイルにまとめて，採点の便宜を図る（図 1–②）。

＜報告書形式の提出課題を PC 上で採点する場合＞

・採点コメントや得点記入欄を有したスプレッドシートを，課題提出人数分自動で生成する（図 1–③）。採点後，シートから得点のみを抽出し，一覧表を生成する。さらに，学生への返却用に，各シートを個別の PDF ファイルに変換する（図 1–④）。

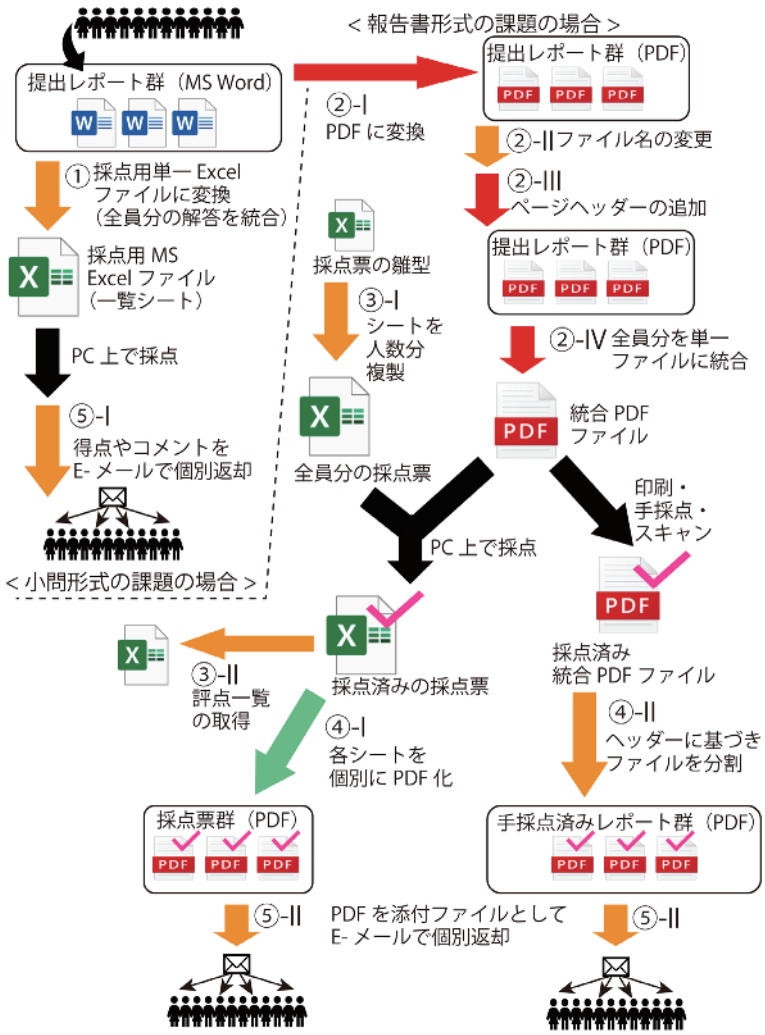

図 1　学生レポートの受付から返却までのフロー。オレンジ色の矢印は本稿で紹介する Python スクリプトによる処理を示す。緑色の矢印は MS Excel のマクロ（本稿で紹介），赤色の矢印は Adobe Acrobat の機能（およびその拡張機能），黒色の矢印は手作業によるプロセスをそれぞれ示す。

＜報告書形式の提出課題を手採点する場合＞

・手採点後の PDF ファイルを，返却用の個別 PDF ファイルに分割する（図 1–④）。

＜課題の採点結果を学生に個別にフィードバックをおこなう場合＞

・各学生へ得点，コメント，もしくはそれらを含む PDF ファイル（添付書類）を，履修者各人のメールアドレス宛に送信する（図 1–⑤）。

実際に上記システムを運用した際のトラブルから学んだ注意点や，学生からの反応も併せて

表１　本稿で紹介するシステムで使用した各ソフトウェア等のバージョン

ソフトウェア等	バージョン
OS	Windows 10
Python	ver. 3.8. 3
Anaconda	ver. 4.9. 2
Jupyter Notebook	ver. 6.0. 3
Microsoft Office	Home and Business 2019
Adobe Acrobat Pro DC	ver. 2022.001.20169

紹介することで，今後の授業運営の参考と資したい。以下，図１中のフローに示す丸数字の番号に基づいて，各プロセスの処理について紹介する。

Python には様々なディストリビューションがあるが，本稿で紹介するシステムの開発には，科学計算に定評のある Anaconda にパッケージされている統合開発環境（IDE）「Jupyter Notebook」を使用した。これらをインストールして，Python の使用環境を整える手順については，本稿のスコープを逸脱するため，ルバノビック（2015）や大澤（2021）を参照いただきたい。使用したソフトウェア等のバージョンは表１に示す通りである。また，標準ライブラリに含まれていない外部モジュール（本稿では docx，numpy，openpyxl，pandas，pdfminor，PyPDF2）のインストールについても，大澤（2021）等を参照していただきたい。

2. 各処理の実際

2-1　提出された Word ファイル（.doc）から採点用統合 Excel ファイル（.xlsx）を作る

一連の記述による報告書（レポート）形式の課題ではなく，小テスト形式で複数の小問への解答を回収する場合の方法である。各大学が採用している教育支援のシステムの中には，このような学生の解答を一覧できるスプレッドシートを自動生成し，ダウンロード可能なものもあるだろう。ここで紹介する Python のスクリプト（“create_book_for_scoring.py”：Box 1）はそのような機能が提供されていない場合に有効であるとともに，解答中に使用されているキーワードの色分け表示，フィルター機能の自動追加など，その後の採点を補助する様々なオプションを自前で追加・拡張できる利点がある。

この機能を利用するためには，各設問の番号を予め記入したテンプレートの Word ファイルを学生に提供し，利用させる（図２）。各設問番号の冒頭には特殊な文字列（図２の例では #_#_#）があり，これを消去，コピー・ペーストしないよう，学生に注意を促しておく。学生から提出されたすべての Word ファイルを含むディレクトリ（＝フォルダ：それ以外のファイルを置かないように注意する）において，スクリプト“create_book_for_scoring.py”（Box 1）を実行すれば，各学生の各問への解答を一行とする一覧形式のスプレッドシートが

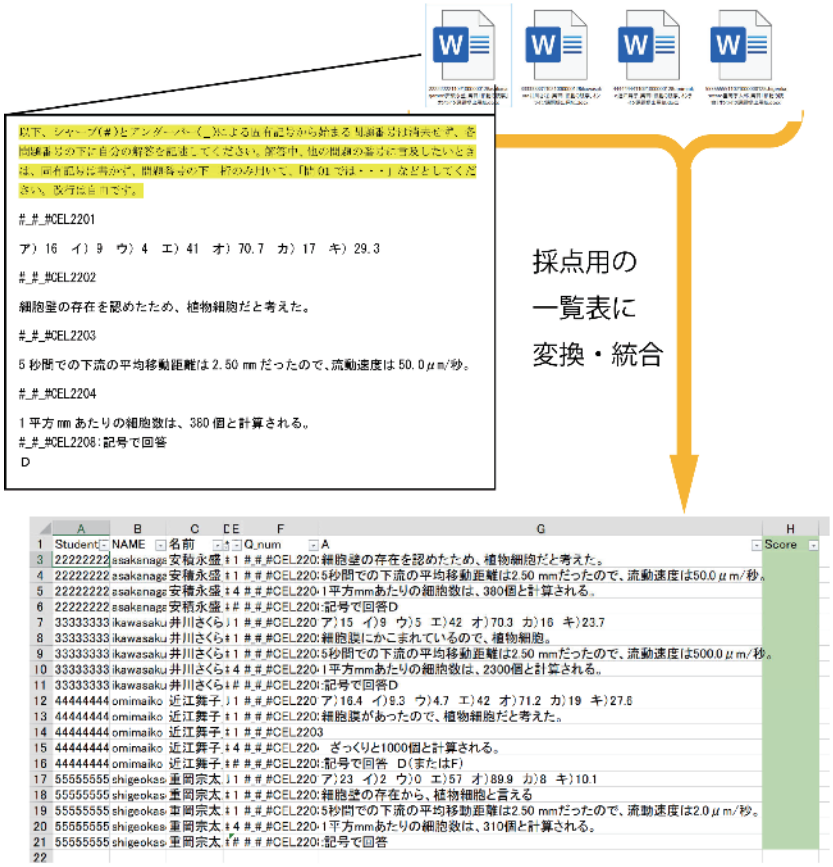

図 2　小問形式の課題提出用のテンプレートを使用して提出された答案の例（左上）と，それに基づいて作成された解答の一覧表（下）。一覧表のシートを含むファイルは“merged_ 作成日 .xlsx”という名称で作成される。本図を含め，本稿で登場する学籍番号や学生氏名は全て架空のものである。

生成される（図 2）。今回例示のスクリプトの場合，Microsoft Excel（以下，単に Excel）の Book 形式ファイル（.xlsx）として，提出された個々の課題ファイルと同一のディレクトリ内に生成される。なお，各 Box に示すスクリプトの中で，赤字で示す部分は，動作させる PC 環境等に応じて適宜修正すべき部分である。

筆者が所属する慶應義塾大学において，2022 年 7 月現在使用されている教育支援システム K-LMS では，学生からアップロードされたファイルには，8 桁の学籍番号から始まり，学生氏名を含むファイル名が自動的に付与される。（スクリプト中の Python の文法の詳説は本稿では割愛するが）これらの情報と，あらかじめ与えた設問番号を読み込むことで，全学生の解答を 1 枚に納めたシートが生成される仕組みである。

Excel のソート機能やフィルター機能を活用しながら採点することで，設問毎に，採点基準を一定に保ちながら，効率よく採点を進めることができた。紙媒体への手採点と異なり，採点後の得点を手入力する手間が省け，合計点や標準偏差の算出にも便利であった。ただし，学生

```
## create_book_for_scoring.py ##
# 小問形式課題の解答を一覧した採点用シートを作成する
import os  #OS 関係の操作に必要
import re # 正規表現の利用に必要
from datetime import date # 日付の取得・操作に必要
import docx #.docx ファイルの操作に必要
import openpyxl #Excel ファイルの操作に必要
from openpyxl import Workbook #.xlsx ファイルの操作に必要
from openpyxl.styles import PatternFill # セルの塗りスタイルの設定に必要

eng_lower_regex = re.compile('[a-z]+') # 英小文字をあらわす正規表現
not_eng_lower_regex = re.compile('[^a-z]+') # 英小文字「以外 (^)」をあらわす正規表現
target = '#_'  # 問題番号の目印：問題番号の操作に必要な定義
column_width = {'D':1, 'E':2, 'F':12,'G':155} #Excel book の各列の幅の指定用定義

os.chdir("C:\\XXX\\YYY\\student_reports") # 対象ディレクトリ（赤字 : 使用者が調整）へ移動
file_list = os.listdir() # ワーキングディレクトリ内のファイル名のリストを作成

today = date.today() # 実行日の日付を取得
wb = Workbook()
ws = wb.active
ws.title = 'merged'
ws['A1'] = 'StudentID';  ws['B1'] = 'NAME';ws['C1'] = ' 名前 '; ws['D1'] = 'tempA';
ws['E1'] = 'Checker';  ws['F1'] = 'Q_num'; ws['G1'] = 'A'; ws['H1'] = 'Score' # 各列見出しの設定

score_fill = PatternFill(fgColor = 'C6E0B4', fill_type = 'solid') # セルの塗りの指定

n_row = 2 #Excel へのデータ入力は 2 行目からスタート
for k in range(len(file_list)):
    stuIDcandidate = re.match('^\d{8}',file_list[k]) # 先頭 (^) の数字 (\d) を 8 文字抽出
    if stuIDcandidate:
      stuID = stuIDcandidate.group()
```

＊

Box 1　採点用に全学生の解答一覧シートを作成するための Python スクリプト。スウェイガード（2017），リブロワークス（2020）を参考に作成。なお，現在プログラムが動いているディレクトリ（カレントワーキングディレクトリ）は os.getcwd（）命令で得られ，print（os.getcwd（））で知ることができる。ファイル名からの情報抽出に関する部分（＊）は，慶應義塾大学の学修支援システムの特性に依拠している部分が大きく，システムごとに修正を加える必要がある。

```
stuNAMEcandidate = re.search('(?<=\d)\D{2,50}',file_list[k])
#(?<=prev)next は「prev が前にある next」をあらわす正規表現
#(?<=\d)\D{2,50} で「数字（\d）が先行する２～５０字連続の数字以外の文字（\D）」を抽出
if stuNAMEcandidate:
  stuNAME = stuNAMEcandidate.group()
  result = eng_lower_regex.search(stuNAME[0])
  # 抽出されたものの先頭文字が英小文字か確認
  if(result == None): # そうでなければ、クラス名なので先頭の文字は除去
    stuNAME = stuNAME[1:]
  stuNAME = stuNAME[0:len(stuNAME)-1]
  stuNAMEeng = eng_lower_regex.search(stuNAME)
  stuNAMEjap = not_eng_lower_regex.search(stuNAME)

doc = docx.Document(file_list[k])
q_counter = 0
full_text = []
for para in doc.paragraphs:
  qNUMcandidate = re.search('#_#_#',para.text)
  if qNUMcandidate:
    if (q_counter >= 1):
      ws.cell(n_row,1).value = stuID[0:8]
      ws.cell(n_row,2).value = stuNAMEeng[0]
      ws.cell(n_row,3).value = stuNAMEjap[0]
      full_text = '\n'.join(full_text) # リスト状態のままでは書き込めないので、改行で接続
      ws[f'D{n_row}'] = full_text
      if re.search('#_#_#\w{3}\d{4}', full_text):
        ws[f'F{n_row}'] = re.search('#_#_#\w{3}\d{4}', full_text)[0]
      else: ws[f'F{n_row}'] = re.search('#_#_#', full_text)[0]
      full_text = re.sub('\n', '',full_text) # 回答から改行を除去
      idx = full_text.find(target); full_text = full_text[idx+12:] # 問題番号より前を除去
      ws[f'G{n_row}'] = full_text
      full_text = []
      full_text.append(para.text)
      n_row = n_row + 1
      q_counter = q_counter + 1
```

＊

Box 1（続き 1）

```
            else:
                full_text.append(para.text)

        ws.cell(n_row,1).value = stuID[0:8]
        ws.cell(n_row,2).value = stuNAMEeng[0]
        ws.cell(n_row,3).value = stuNAMEjap[0]
        full_text = '\n'.join(full_text)
        ws[f'D{n_row}'] = full_text
        ws[f'F{n_row}'] = re.search('#_#_#\w{3}\d{4}', full_text)[0]
        full_text = re.sub('\n', '',full_text)
        idx = full_text.find(target); full_text = full_text[idx+12:]
        ws[f'G{n_row}'] = full_text
        n_row = n_row + 1

    for i in range(n_row-2):
        ws[f'E{i+2}'] = f'=RIGHT(F{i+3},1) - RIGHT(F{i+2},1)' # 確認用の数式の入力
        ws[f'H{i+2}'].fill = score_fill # 得点入力欄のみセルの塗りを指定し、誤入力を防ぐ

    ws['H1'].fill = score_fill
    ws.auto_filter.ref = ws.auto_filter.ref = "A1:H1" #A ～ H 列にオートフィルターを設定
    ws.freeze_panes = 'A2' #2 行目以降のみスクロールして、タイトル行が固定されるよう設定

    for col, width in column_width.items():
        ws.column_dimensions[col].width = width # 各列の幅を指定したものに調整して見やすくする

    wb.save(f'merged_{today:%Y-%m-%d}.xlsx')
```

Box 1（続き 2）

の解答文の中に，図やハイパーリンクされたオブジェクトなど，通常のテキスト以外のものが含まれていた場合，それらは取り除かれた形で取り込まれるので注意が必要である。

2–2　提出された報告書形式の課題レポートから採点用統合 PDF ファイル（.pdf）を作る

自身の調査に基づいて，その調査の背景，手法，得られた結果やそれに対する考察を第三者に報告する。このような報告書の執筆スキルは，卒業論文執筆にも必要となり，文系・理系を問わず，大学生に必要なリテラシーと言えるだろう（佐藤ら 2012）。生物学Ⅰ・生物学Ⅱでは，講義と実習（実験）が隔週で実施されており，実習（実験）の結果を報告する作業は，報告書執筆の訓練の場となり得る。報告書形式の課題レポートの採点には，上記のようなスプレッド

シート形式はそぐわないが，学生の個々のレポートファイルを PC 上で逐一開いて採点する作業は煩雑であり，単一の PDF ファイルにまとめるのが便利である。

一部もしくは全てのファイルが Microsoft Word 形式で提出されている場合，まずそれらを PDF に変換する必要がある（図 1 の②-Ⅰ）が，その機能は Adobe Acrobat Pro で提供されており，容易に利用できるのでここでは解説を省略する。全提出レポートを個々の PDF ファイルに変換したのち，その後の処理に便利なよう，ファイル名を学籍番号のみのシンプルなものに変換する（図 1 の②-Ⅱ）。それは，個々の PDF ファイルがすべて含まれるディレクトリにおいてスクリプト“change_file_names.py”（Box 2）を実行することで実現される。

個々のファイルの開閉の手間や間違いを減らすため，最終的に単一の PDF に統合するが，採点中のページの作成者がわからないと，採点の効率が悪いことを経験した。また，学生が提出したファイルの中身自体には学籍番号等が記入されていないことや，記入されていたとしても入力ミスがあり得る。そこで，学籍番号をレポート各ページのヘッダーとして追加することとした。Adobe Acrobat Pro の拡張機能 File Name Stamper（https://acrobatusers.com/actions-exchange から導入可能）を，「ツール」－「アクションウィザード」から実行することで，ファイル名をヘッダーとして記入することができた（図 1 の②-Ⅲ）。今回の場合，ファイル名は学籍番号に変換してあるため，最後に，Adobe Acrobat Pro の「ツール」－「ファイルを結合」で，全部のレポートを一つのファイルにまとめた後も（図 1 の②-Ⅳ），ヘッダーを見れば，常にどの学生のレポートを採点中なのかを把握することができた。Python スクリプトでも同様な処理は可能だと思われるが，Python での PDF ファイルへの追記は，ページの重ね合わせという煩雑な処理をともない（スウェイガート 2017），トラブルが起こりやすいと思われる。

```
## change_file_names.py ##
# フォルダ中の全 PDF ファイルのファイル名を最初の 8 文字（学籍番号）のみに短縮
# 専用のディレクトリ（例示では "student_reports"）を用意し、対象ファイルを置いておく
import os  #OS 関係の操作に必要
import shutil # ファイル・ディレクトリー操作に必要

os.chdir("C:\\XXX\\YYY\\student_reports") # 対象ディレクトリ（赤字：使用者が調整）に移動

for original_filename in os.listdir("."):
    new_filename = original_filename[0:8] + '.pdf'
    shutil.move(original_filename,new_filename)
```

Box 2　対象ディレクトリ中の PDF ファイル名を，学籍番号のみに短縮するための Python スクリプト。これによりファイル名をヘッダーとして各ページに書き込む等，後の作業が簡略化される。ルバノビック（2015）を参考に作成。

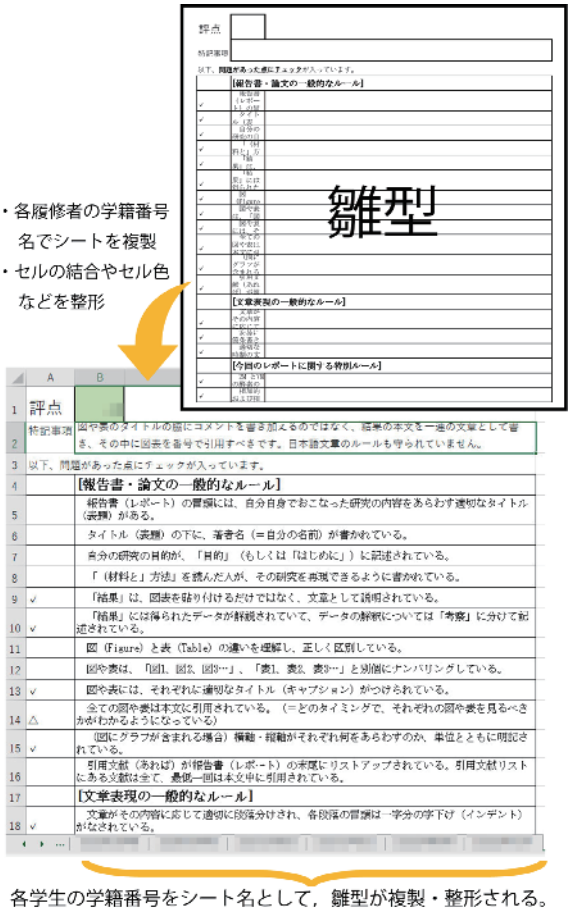

図 3　報告書（レポート）形式の課題採点用の雛型シート（テンプレート：上）と，自動生成された複数シート（各学生につき 1 シート）を含む Excel Book（下）。

2–3　報告書形式の課題レポートの採点票 Book を作る

報告書形式のレポート課題が提出された場合，その内容面のみならず，報告書としてのスタイルに関する指摘も，卒業論文等執筆の訓練として必要な作業といえるだろう。しかしそれは，時間と労力を要する作業である。特に学部 1 ～ 2 年生の場合，報告書形式の執筆そのものに不慣れなため，表（Table）と図（Figure）の区別や，それらの本文中への引用ルール，文章の適切な段落分けや文章の時制など，指摘すべき点は数多く存在する。同様な指摘を赤字で記入することを，多くの学生レポートに対して繰り返す必要が生じる。

オンラインでの課題採点・返却システムは，このような作業負担も軽減し得る。著者の場合，指摘が頻発するポイントを予めチェック形式で準備し，得点と特記事項（コメント）の記入欄を備えた採点票雛型ファイル（Excel Book 形式：.xslx）を作成・使用した（図 3）。これを，上記の「個別 PDF」がすべて含まれるディレクトリ（例示では“student_reports”）とともにワーキングディレクトリに置き，スクリプト“multiply_score_sheets.py”（Box 3）を実行すれば，各学生の学籍番号をシート名とした Excel Book が自動生成される（図 1 の③ - I，図 3）。

```
## multiply_score_sheets.py ##
# 提出者の人数分、学籍番号のシート名で Excel の採点票をコピーする
# 事前に採点シートのひな形（” Score_Template.xlsx”）を準備
# ひな形は個々の学生レポート PDF 含むディレクトリ ('student_reports') と同じ場所に置く
import os  #OS 関係の操作に必要
import re # 正規表現の使用に必要
from openpyxl import load_workbook # 既存のワークブックに対する処理に必要
from openpyxl.styles import Alignment, Border, PatternFill, Side # 各種スタイルの設定に必要

score_fill = PatternFill(fgColor = 'C6E0B4', fill_type = 'solid') # セルの塗りの指定
centertop_align = Alignment(horizontal = 'center', vertical = 'top') # 文字配置の指定
print_area = 'A1:C23' # シートの印刷範囲の指定。雛型に併せて調整

os.chdir("C:\\XXX\\YYY) # 雛型をおいたディレクトリに移動。赤字は使用者が調整
number_list = os.listdir('student_reports') #PDF のファイル名から学籍番号一覧を取得
for k in range(len(number_list)):
   number_list[k] = re.sub('.pdf','',number_list[k])
wb = load_workbook(' Score_Template.xlsx') # 採点シートの雛型を読み込む

for ws in wb.worksheets:
   ws.sheet_view.tabSelected = None
ws_sheet1 = wb['Sheet1'] #Score_Template.xlsx の Sheet1 を ws_sheet1 とする
for i in range(len(number_list)):
   ws_copy = wb.copy_worksheet(ws_sheet1) #ws_sheet1 をコピー
   ws_copy.title = number_list[i] # コピーされたシートの名前を学籍番号とする
   wb.active = i+1
   ws = wb.active
   ws.print_area = print_area # 印刷範囲の設定
   ws.cell(1,2).fill = score_fill # 点数入力欄だけは色塗りする
   ws.cell(2,1).alignment = centertop_align #「コメント（特記事項）」を上中央配置
   ws.merge_cells(start_row = 2, start_column = 2, end_row = 2, end_column = 3)
   for j in range(20): # セルの結合範囲を指定
      ws.merge_cells(start_row = 4 + j, start_column = 2, end_row = 4 + j, end_column = 3)
   wb.active = 0
wb.save('ForScoring.xlsx') # 完成した Book は別の名前で保存
```

Box 3　採点票の雛型シートを，採点対象の学生人数分複製した Excel Book を生成するための Python スクリプト。作成された各シートの名称は学籍番号となる。スウェイガード（2017），リブロワークス（2020）を参考に作成。

あとはステップ②で作成した統合 PDF を見ながら，指摘する必要のない点のチェックを消していけば良いので，採点がスムーズであった。

雛型のシートに「セルを結合する」の指定が含まれている場合，上記処理時にエラーが出ることがある。雛型段階では書式指定せず，プログラム中で指定するのがよい。また，採点作業が全て終了したら，スクリプト“extract_data_from_Excel_sheets.py”（Box 4）を実行する

```
## extract_data_from_Excel_sheets.py ##
# 指定した Excel book (.xlsx) の各シート名と指定したセルの値を一覧表にする
import os  #OS 関係の操作に必要
import openpyxl #Excel ファイル操作に必要
from openpyxl import load_workbook, Workbook ###.xlsx ファイル操作に必要

os.chdir("C:\\Users\\XXX\\YYY\\student_reports") # 対象ディレクトリに移動

Source_book = \
input(" 得点を抽出したい Excel Book の名前を拡張子も含めて入力してください [ 例 ]Score.xlsx:")
Target_cell = \
input(" 得点が入力されているセルの番地を入力してください [ 例 ]B1:")

wb_new = Workbook()
ws_new = wb_new.active
ws_new.title = 'scores'
ws_new['A1'] = 'Student ID'
ws_new['B1'] = 'Score'

wb = load_workbook(Source_book, read_only = True)
#read-only 指定（＝上書きされない）でマルチシートの採点票 Book（採点済み）を開く

for i, ws in enumerate(wb.worksheets): # 以下、全ワークシートに関して繰り返す
    row_no = i + 1
    if (row_no == 1): row_no = row_no + 1
    ws_new[f'A{row_no}'] = ws.title
    # 各シートのタイトル（＝学籍番号）を A 列に格納
    ws_new[f'B{row_no}'] = ws[Target_cell].value
    # 評点は各シートの B1 セルに入力されている。それを B 列に格納

wb_new.save('Scores_extracted.xlsx') # 新規の Excel book として別の名前で保存
```

Box 4　採点済みの採点票の束（Excel Book）から，学籍番号と得点だけを抽出して一覧表にするための Python スクリプト。学籍番号はシート名から抽出される。リブロワークス（2020）を参考に作成。

ことで，学籍番号と得点の一覧表を抽出した。これにより，転記ミスを避けることができた（図1の③-Ⅱ）。

2–4　コメント・得点返却用の個別 PDF ファイルを作る

報告書（レポート）形式の課題については，コメントや指摘を各学生に返却することなしには，その学習効果は限定的なものになるだろう。上記ステップ③で紹介した採点票 Book の各学生のシートを，採点後の返却用に個別の PDF に変換したい場合は，Excel のマクロを利用するのが便利であった。具体的には，以下の手法によって実施した。

採点を終えた採点票 Book の「開発」メニュー－「Visual Basic」サブメニューより，Microsoft Visual Basic for Applications のウィンドウを開く。「挿入」－「標準モジュール」を選び，現れた Module1 の画面に Box 5 に記載の VBA コードを記述する。この Excel ファイルをマクロ有効 Book（例えば Scored.xlsm）として別名保存しておく。「開発」メニュー

```
'Sub SheetSavePDF
'Excel Book の各シートを個別に PDF ファイルとする
Sub SheetSavePDF()

Dim i As Integer
i = 1

For Each ws In Worksheets

With Worksheets(i).PageSetup
   .Zoom = False ‘拡大・縮小を無効にする
   .FitToPagesWide = 1  ‘シートの幅が紙幅におさまるよう設定
   .CenterHorizontally = True ‘水平方向中央揃えを有効にする
End With

Worksheets(i).ExportAsFixedFormat Type:=xlTypePDF,_
Filename:=ThisWorkbook.Path & "\" & Worksheets(i).Name,_
IgnorePrintAreas:=False, OpenAfterPublish:=False

i = i + 1
Next

End Sub
```

Box 5　Excel Book の各シートを，個別の PDF ファイルとして印刷するための VBA スクリプト（マクロ）。国本（2019）を参考に作成。

―「マクロ」サブメニューで，上記で作成したマクロ（“SheetSavePDF”）を実行すると，マクロ有効 Book が存在するのと同一のディレクトリ中に，各学生のシートを個別に PDF 化したものが作成される。

報告書形式のレポートでは，その内容次第では手作業による採点が現実的なことも多い。例えば，手描きのスケッチを含む場合である。しかし，そのような場合も，オンラインでのレポート整理・返却が便利な点が多かった。提出されたレポートに，あらかじめ上記ステップ②-Ⅲの方法により学籍番号のヘッダーを含めておけば，手採点を終えたレポートをすべてまとめてスキャンしたのち，容易に学生個別への PDF へと分割することができた（図 1 の④-Ⅰ）。

筆者はスキャナーとして，FUJITU ScanSnapS1500 を使用しているが，両面印刷されたものを含めて，レポートの束を一気に読み，単一の PDF ファイルにまとめることができるので，採点結果の保存に便利である。これに Adobe Acrobat DC 提供の文字認識（OCR 機能）を適用し，文書内容を検索可能な PDF にしておく。これを置いたディレクトリにおいて，スクリプト“PDF_divider.py”（Box 6）を実行すれば，各学生の学籍番号をファイル名とした個別 PDF に自動で分割することが可能であった（図 1 の④-Ⅱ，図 4）。ただし，各ページのヘッダーより上部の領域にコメントを書き込んだ場合，分割プロセスに干渉する可能性があるので注意が必要である。

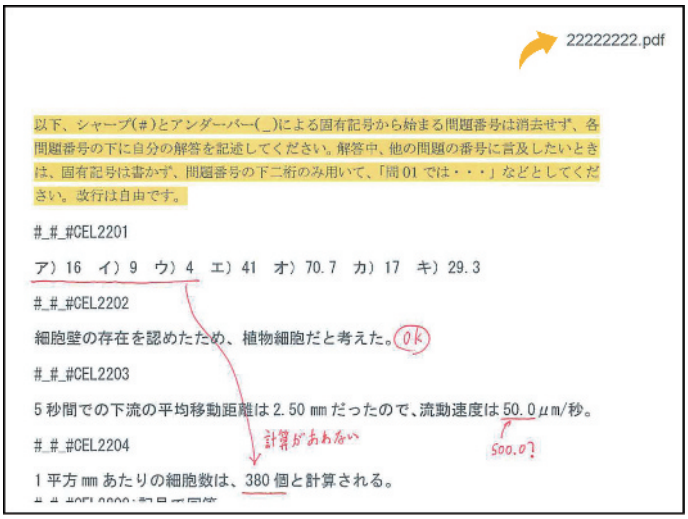

図 4　学籍番号のヘッダー（矢印）が追加された課題を手採点し，スキャンの後に分割した場合の例。ここでは小問形式の提出課題の仮想例を図示しているが，報告書形式の場合も有効である。

```
##PDF_divider.py##
# 学籍番号を含むヘッダーを読み取り、PDF を個別ファイルに分割
from io import StringIO # 文字列バッファの操作に必要
import os  #OS 関係の操作に必要
import re # 正規表現の利用に必要
from pdfminer.pdfinterp import PDFResourceManager # 以下、PDF ファイルの操作に必要
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from PyPDF2 import PdfFileReader
from PyPDF2 import PdfFileWriter

os.chdir("C:\\XXX\\YYY\\student_reports") # 対象ディレクトリ（赤字：使用者が調整）に移動
rmgr = PDFResourceManager()
st_num_regex = re.compile(r'\d{8}.\s{0,1}pdf')
# サーチ対象として「8 桁の数字 (= 学籍番号）.pdf」を定義
# 学籍番号の後ろに 0 ～ 1 個の空白文字を許容 ( 印刷、スキャン、OCR した場合の対応 )

cum_pg_counter = 0; each_pg_counter = 0; prev_st_number = "00000000";
target_pdf_file = \
input(" 分割したい PDF ファイルの名前を拡張子も含めて入力してください [ 例 ]Merged.pdf:")
fp = open(target_pdf_file,'rb') # 分割対象の PDF ファイルを読取専用モードで開く
input = PdfFileReader(fp)
pgnum = input.getNumPages() # 確認用のページ数の取得

output = PdfFileWriter()
for page in PDFPage.get_pages(fp):
  outfp = StringIO()
  device = TextConverter(rmgr, outfp)
  PDFPageInterpreter(rmgr, device).process_page(page)
  mo = st_num_regex.search(outfp.getvalue())
  st_number = mo.group()[0:8]
  # 最後の 12 文字から先頭 8 文字の数値を文字列として拾う
```

Box 6　手採点後，スキャナーで一括して読み込んだレポート束を，各学生への返却用に分割するための Python スクリプト。全ページに学籍番号がヘッダーとして印刷されていて，文字認識がなされていることを前提として，「学籍番号 _checked.pdf」が学生人数分作成される。スウェイガード（2017），日経ソフトウェア［編］（2020）を参考に作成。

```
    if (prev_st_number == "00000000"):
        prev_st_number = st_number # 最初の 1 ページ目の処理
        each_pg_counter = each_pg_counter + 1
        cum_pg_counter = cum_pg_counter + 1
    else:
        if (st_number == prev_st_number):
            each_pg_counter = each_pg_counter + 1
            cum_pg_counter = cum_pg_counter + 1
        else:
            output = PdfFileWriter()
            for i in range(cum_pg_counter-each_pg_counter,cum_pg_counter):
                output.addPage(input.getPage(i))

            outputfile = open(str(prev_st_number) + '_checked.pdf', 'wb')
            output.write(outputfile)
            outputfile.close()

            prev_st_number = st_number

            each_pg_counter = 1; cum_pg_counter = cum_pg_counter + 1

output = PdfFileWriter()
for i in range(cum_pg_counter-each_pg_counter,cum_pg_counter):
    output.addPage(input.getPage(i))

    outputfile = open(str(prev_st_number) + '_checked.pdf', 'wb')

output.write(outputfile)
outputfile.close()

outfp.close(); device.close(); fp.close()

# 分割前のページ数と分割後の合計ページ数の一致を確認し、メッセージを出力
if (cum_pg_counter == pgnum):
    print("Process successfully finished")
else:
    print("Process failed")
```

Box 6（続き）

2-5　各学生に得点，コメント，採点済みレポートを返却する

コロナ禍により，急遽オンラインでの授業展開が要求された際，多くの教員にとって問題になったのは，採点結果（得点やコメント）の返却方法だろう。多くの大学においては，学生各人に E メールアドレスが付与されていると思われるが，これを教員が利用できる場合，各学生に得点やコメントのテキストのみや，添付ファイルを正確かつ簡単に返却できる。学生ごとに違った本文や添付ファイルの E メールを書くことは，手作業では現実的ではないが，それぞれスクリプト“score_return.py”（Box 7）とスクリプト“attachments_return.py”（Box 8）により，準備を含めても数分での返却が可能となった（図 1 の⑤-Ⅰ，⑤-Ⅱ）。

具体的には，それぞれ図 5 に示すような成績管理用の Excel ファイル上のデータをクリップボードから読み込んだ状態で，スクリプトを実行することで，得点等を返却した。筆者の場合，得点やコメントの返却は希望者に対してのみ行っているため，Box 7，8 のスクリプトも Excel の「返却希望」の列が“1”の受講生だけに送信されるようにデザインされているが，

A. 得点のみを返却する場合

	A	B	C	D
1	氏名	E-mail	得点返却希望	細胞の観察
2	上村佳孝	[redacted]	1	1000
3	安積永盛	[redacted]	0	7.5
4	井川さくら	[redacted]	0	6
5	近江舞子	[redacted]	1	9.5
6	重岡宗太郎	[redacted]	1	10
7	上村佳孝	[redacted]	1	1001
8				

B. 添付ファイルを返却する場合

	A	B	C	D
1	氏名	E-mail	得点返却希望	学籍番号
2	上村佳孝	[redacted]	1	11111111
3	安積永盛	[redacted]	1	22222222
4	井川さくら	[redacted]	1	33333333
5	近江舞子	[redacted]	1	44444444
6	重岡宗太郎	[redacted]	1	55555555
7	上村佳孝	[redacted]	1	66666666
8				

図 5　E-メールを用いたフィードバックの際に，クリップボードに読み込むデータの例。上（A）は得点のみ返却する場合，下（B）は添付ファイルとして採点済みレポートを返却する際のもの。いずれもメール送信状況の確認のため，先頭および末尾の行に自分自身の氏名・メールアドレスを含め，A では（平均点等の算出の際，誤って含めてしまっても気が付くよう）得点として桁の大きな数字を与えている。A は得点のみを返却するケースを表示しているが，短文のコメントテキストも併せて返却したい場合は，E 列にコメントを追加し，メール本文中にコピーされるよう，スクリプトを修正すればよい。

```
## score_return.py ##
# 得点のみを個別で E-mail で返却する
import smtplib #SMTP プロトコルの使用に必要
from email.mime.multipart import MIMEMultipart # 以下 5 行、E-mail の送付に必要
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.utils import COMMASPACE, formatdate
from email import encoders
import pandas as pd # クリップボードからのデータ読み込みに必要
import numpy as np # 多次元配列データの操作に必要

charset = 'iso-2022-jp'

def send_mail(send_from, send_to, subject, message, reply_to, \
        server="localhost", port=587, username='', password=''):
  msg = MIMEMultipart()
  msg['From'] = send_from
  msg['To'] = send_to
  msg['Date'] = formatdate(localtime=True)
  msg['Subject'] = subject
  msg.add_header('reply-to', reply_to)
  msg.attach(MIMEText(message))
  smtp_obj = smtplib.SMTP(server, port, timeout = 10)
  smtp_obj.login(username, password)
  smtp_obj.sendmail(send_from, send_to, msg.as_string())
  smtp_obj.quit()

score_data = pd.read_clipboard()
# ここでクリップボードにコピーしたデータ（タイトル行を含む）を読み込み。
data = score_data.to_numpy()
#to_numpy メソッドで numpy の多次元配列に変換。タイトル行はここで除去される。
print(len(data))
print(data) # いったんここまで実行して、確認をおこなうと良い
```

Box 7　希望者各人に E-メール本文で得点を返却するための Python スクリプト。必要なデータをクリップボードにコピーした状態で実行する。また，前半をまず実行し，データを確認したのち，実際のメール送信に関わる後半部分を実行するとよい。スウェイガード（2017）を参考に作成。

```
PW = input("PW:") # パスワードは保存を避け、使用の都度入力を求める

for i in range(len(data)):
  Student_Name = data[i,0]
  Student_E_mail = data[i,1]
  Return_Request = data[i,2]
  Score = data[i,3]

  if Return_Request == 1:
    Main_Message = Student_Name + " さま。今回の点数は " + str(Score) +\
    " 点でした (10 点満点 )。今回の平均点は 7.8 点、標準偏差は 1.6 点でした。"
    send_mail(' 自身の送信元アドレス ', Student_E_mail, \
          ' 生物学実習「細胞の観察」採点結果 ', Main_Message,\
          ' 返信先（reply_to）アドレス ', server=' 送信サーバ ', \
          port=587, username=' ユーザ名 ',\
          password=PW)
    print(Student_Name) # 確認用にメールを送った相手の氏名を出力
```

Box 7（続き）

全員に返却したい場合はこの列の値を全て“1”にするか，スクリプト中の場合分け処理を取り除けばよい。

紙媒体でのレポートは，採点結果を学生に返却した際，全てをコピーしておかない限り手元にはコメントが残らない。また，各学生を呼んで個別に返却することは時間と手間がかかり，教室に並べておけば，盗用の危険やプライバシーの問題もある。E-メールを利用した得点やコメントの返却は，対面授業においてもメリットが大きいと考えられる。なお，G-mail など，セキュリティの強化されているシステムでは，ここで紹介した方法そのままではうまく送受信できないことがあり（スウェイガード 2017），十分な予備実験の上で運用する必要がある。

3. 履修学生の反応

コロナ禍で全面的にオンライン授業となった 2020 年度以来，筆者の生物学 I・生物学 II では，授業が対面実施の回も含めて，課題の提出や得点・コメントの返却は全てオンライン形式で実施している。後者は返却希望を表明した履修者に対してのみ実施しているが，記録の残っている直近 3 学期分のデータでは，平均 88.9％の履修者（分母の N は 369）が得点・コメントの返却を希望しており，フィードバックに対する要望は無視できないものがある。

また，各授業の最終回に実施した自由記述形式のアンケートでも，「全クラスの平均点が掲

```
## attachments_return.py ##
# 添付ファイルで採点結果を返却
import os #OS 関係の操作に必要
import os.path as op # 添付ファイルの取得などに必要
fimport smtplib #SMTP プロトコルの使用に必要
from email.mime.multipart import MIMEMultipart # 以下 5 行、E-mail の送付に必要
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.utils import COMMASPACE, formatdate
from email import encoders
import pandas as pd # クリップボードからのデータ読み込みに必要
import numpy as np # 多次元配列データの操作に必要

charset = 'iso-2022-jp'
os.chdir("C:\\XXX\\YYY") # 対象ディレクトリ（例示では student_reports）の直上に移動
# 赤字部は使用者が調整

def send_mail(send_from, send_to, subject, message, reply_to, files=[], \
        server="localhost", port=587, username='', password=''):
  msg = MIMEMultipart()
  msg['From'] = send_from
  msg['To'] = send_to
  msg['Date'] = formatdate(localtime=True)
  msg['Subject'] = subject
  msg.add_header('reply-to', reply_to)
  msg.attach(MIMEText(message))

  for path in files:
    part = MIMEBase('application', "octet-stream")
    with open(path, 'rb') as file:
      part.set_payload(file.read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition','attachment;\
    filename="{}"'.format(op.basename(path)))
    msg.attach(part)
```

Box 8　希望者各人に E-メールの添付ファイルとして，採点済みレポートを返却するための Python スクリプト。基本的な仕様は Box 7 のスクリプトに準じる。採点済みの PDF ファイル名が“_checked.pdf”として区別されている場合は，添付ファイルの選択（Attach_File_Name =）において，# でコメントアウトされている側の行を代わりに用いればよい

```
    smtp_obj = smtplib.SMTP(server, port, timeout = 10)
    smtp_obj.login(username, password)
    smtp_obj.sendmail(send_from, send_to, msg.as_string())
    smtp_obj.quit()

score_data = pd.read_clipboard()
data = score_data.to_numpy()
print(len(data))
print(data)  # いったんここまで実行して、確認をおこなうと良い

PW = input("PW:") # パスワードは保存を避け、使用の都度入力を求める

for i in range(len(data)):
    Student_Name = data[i,0]
    Student_E_mail = data[i,1]
    Return_Request = data[i,2]
    StudentID = data[i,3]

    if (Return_Request == 1):
        try:
            Attach_File_Name = 'student_reports\\' + str(StudentID) + '.pdf'
            Main_Message = Student_Name + " さま。コメントと得点の返却です（添付 PDF）。"
            Attach_File_Name = 'student_reports\\' + str(StudentID) + '.pdf'
            #Attach_File_Name = 'student_reports\\' + str(StudentID) + '_checked.pdf'
            send_mail(' 自身の送信元アドレス ', Student_E_mail,\
                    ' 生物学実習「レポートの書き方」採点結果 ', Main_Message,\
                    ' 返信先（reply_to）アドレス ', files=[Attach_File_Name], \
                    server=' 送信サーバ ', port=587, username=' ユーザ名 ', password=PW)
            print(Student_Name)
        except FileNotFoundError: # 採点済みファイルが見つからない時の処理
            Main_Message = Student_Name + " さま。今回のレポートは提出されていないようです。"
            send_mail(' 自身の送信元アドレス ', Student_E_mail, \
                    ' 生物学実習「レポートの書き方」採点結果 ', Main_Message, \
                    ' 返信先（reply_to）アドレス ', server=' 送信サーバ ', \
                    port=587, username=' ユーザ名 ', password=PW)
            print(Student_Name + "( 未提出 )")
```

Box 8（続き）

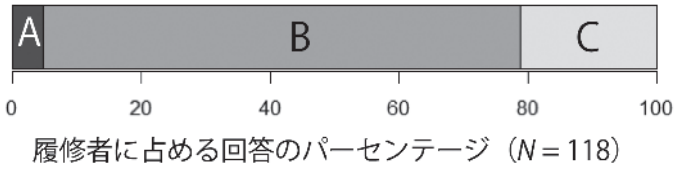

図6　授業形態に関する希望調査の結果。春学期の受講生に秋学期の開講形態の希望をとる形でアンケート調査をおこない，118人の回答を得た。Aは「講義・実習ともに対面実施」，Bは「講義はオンライン（オンディマンド配信），実習は対面実施」，Cは「講義・実習ともにオンライン（オンディマンド配信）」を示す。AまたはBと回答した学生1名については，A，Bにそれぞれ0.5票として集計している。

示されたこと（2021年秋学期）」，「自分の提出した解答が採点され，また平均点と標準偏差も出される点（2021年秋学期）」，「レポートの点数を，すぐに返してもらえたところ（2022年春学期）」といった点を支持する意見があり，フィードバックへの関心の高さがうかがえる。

図6は2022年春学期の生物学Ⅰの履修者（実習は主に対面で実施，講義は主にオンラインで実施）に対して，秋学期に開講される生物学Ⅱの授業形態の希望を，アンケート調査した結果である。実際に実物に触れる実習に対しては対面実施の希望が多いが，講義部分に関してはオンライン形式への要望が強いことがわかる。教室人数制限により，対面実習の実施回数がより制限された2021年度についても，同様の回答（実習のみ対面実施を支持が64.5%，全てオンラインを支持が18.2%）を得ている。適切なフィードバックをおこなえば，オンライン授業に起こりがちな弊害を避けることができ，履修者のオンライン授業に対する抵抗や不満を抑制できるものと思われる。そして，そこで生まれたノウハウの中には，対面形式の授業でも活用できるものが多い。学生側にとっても教員側にとっても，より効率の良い授業運営のシステム構築のチャンスとして，コロナ禍をとらえるべきだと考える。

4. おわりに

同姓同名や，（システムの分解能上）全く同時刻に課題を提出する学生は存在する可能性があるが，各大学において学籍番号は重複なく与えられているはずである。学籍番号を用いてファイル名を管理することで，単一の課題に対して同一ファイル名が複数存在することによる混乱を避けることができる。一方で学籍番号は，採点教員が入力しても，当該学生自身が入力しても間違い易いものである。システム上でのログイン情報に基づいて提出課題のファイルが生成されたならば，そのファイル名内に自動生成された学籍番号には間違いがない。それをそのまま継承し，学籍番号を手入力するプロセスを介在させないことが，効率の良い，トラブルのない採点システムの構築につながると考えた。そこで今回の一連の流れも，「学籍番号は学生に手入力させない・採点者も手入力しない」という原則に基づいて構築をおこなった。

本稿で紹介した各自動処理の実行は，数百人程度の学生に対してならば，どれも1秒以内〜

10 秒程度で終了し，日常業務の効率化の観点からは十分だと考えている。各スクリプトについては，より短く簡潔に書ける別のモジュールの使用や，より処理速度が速いアルゴリズムがあるだろう。しかし，年に数回〜数十回程度の使用頻度のものであり，最適化は求めずに，開発の時間を最小限にとどめているため，その点はご了承いただきたい。より標準的なマナーにしたがった表記法（PEP8 スタイルガイド：例えば，スラットキン 2020 に紹介されている）についても，筆者の怠慢や紙面の都合から犠牲にしている。

本稿の内容に関して運用した結果の影響については，筆者は責任を負いかねる。本稿掲載のプログラムをそのまま，もしくは改変してご利用の際は，（特にメール送信に関わる運用等については）十分な試用のうえで，読者の責任でご活用いただきたい。

謝辞

E-メールでの返却プログラムに関しては，神戸和雄教授（慶應義塾大学・商学部）に有益なアドバイスをいただきました。また，同返却システムの試運用に関しては，生物学教室・助教の墨谷暢子さん（2020 年度時点）・田中正敦さんの両氏と，ティーチングアシスタントを担当いただいていた理工学研究科の大学院生の皆さんのご協力を得ました。様々なフィードバックや，不具合の指摘を寄せてくれた筆者の生物学 I（実験を含む）・生物学 II（実験を含む）の受講者（2020 〜 2022 年度）にもお礼を申し上げます。

引用文献

大澤文孝（2021）Jupyter Notebook レシピ．工学社．

国本温子（2019）Excel マクロ & VBA［実践ビジネス入門講座］［完全版］．SB クリエイティブ．

佐藤望［編著］・湯川武・横山千晶・近藤明彦（2012）アカデミック・スキルズ―大学生のための知的技法入門　第 2 版．慶應義塾大学出版会．

スウェイガード，アル（2017）［相川愛三訳］退屈なことは Python にやらせよう―ノンプログラマーにもできる自動化処理プログラミング．オライリー・ジャパン．

スラットキン，ブレット（2020）［黒川利明訳］Effective Python 第 2 版―Python プログラムを改良する 90 項目．オライリー・ジャパン．

日経ソフトウェア編（2020）仕事と遊びに役立つ Python 活用術．日経 BP．

リブロワークス（2020）できる　仕事がはかどる　Python&Excel 自動処理　全部入り．インプレス．

ルバノビック，ビル（2015）［斎藤康毅監訳・長尾高弘訳］入門 Python3．オライリー・ジャパン．