A Study of Code Modification Problems for Excel Operations in Python Programming Learning Assistant System

San Hay Mar Shwe
Department of Electrical and Communication Engineering
Okayama University
3-1-1, Tsushima-Naka, Kita-Ku, Okayama, 700-8530, Japan
ph8r93wu@s.okayama-u.ac.jp

Nobuo Funabiki
Department of Electrical and Communication Engineering
Okayama University
3-1-1, Tsushima-Naka, Kita-Ku, Okayama, 700-8530, Japan
funabiki@okayama-u.ac.jp

Khaing Hsu Wai
Department of Electrical and Communication Engineering
Okayama University
3-1-1, Tsushima-Naka, Kita-Ku, Okayama, 700-8530, Japan
puag507p@s.okayama-u.ac.jp

Shune Lae Aung
Department of Electrical and Communication Engineering
Okayama University
3-1-1, Tsushima-Naka, Kita-Ku, Okayama, 700-8530, Japan
p1pp6ypa@s.okayama-u.ac.jp

Wen-Chung Kao
Department of Electrical Engineering
National Taiwan Normal University
No. 162, Section 1, Heping East Road, Taipei, Taiwan
jungkao@ntnu.edu.tw

2022 10th International Conference on Information and Education Technology (ICIET) | 978-1-6654-8048-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIET55102.2022.9779025

***Abstract*—*Python programming* has gained popularity in various fields due to rich libraries and short coding features. We have developed *Programming Learning Assistant System (PLAS)* for self-study of *Python Programming*. Previously, we presented *Code Modification Problem (CMP)* for studying *data visualization operations* for data analysis. A CMP instance consists of one source code and two images. One image represents the output of the code, and the other is the output of the answer code from a student and it should be obtained by modifying the given code. The correctness of any answer is verified through *string matching* with the correct one. In this paper, we present CMP for *Excel operations* by extending previous works. A *hint function* is implemented for each CMP instance to assist learners in solving it. We generated 25 CMP instances using Python codes for various Excel operations using *pandas* and confirmed the validity from the application results to students in Okayama University.**

Keywords—python, code modification problem, PLAS, Excel operations, pandas

I. INTRODUCTION

Nowadays, due to rich libraries and short coding features, *Python programming* has become very popular in various applications in both *IT (information technology)* and non-IT fields. Then, low-cost, low-hurdling learning tools have been highly demanded for self-study of introductory *Python programming*, since it can be assumed that many users have no opportunities of taking Python programming courses at their schools.

Currently, we are developing *Programming Learning Assistant System (PLAS)* as self-study tools for various programming languages, including Python. *PLAS* offers several types of exercise problems such as the *grammar-concept understanding problem (GUP)* [1] studying keyword definitions, the *value trace problem (VTP)* [2] studying how to use keywords in the source code, the *element fill-in-blank problem (EFP)* [3] studying partially writing source code, and the *code writing problem (CWP)* studying fully writing source code to cover various students at different learning levels. The correctness of any answer is automatically checked in the system, and the result will be returned to the learner so that he/she can instantly find the mistakes and correct them.

In PLAS, we have studied VTP [4] and CMP (Code Modification Problem) for Python programming. In the VTP instance, one source code and several questions are given. Each question asks to answer the actual values of the important variables or output messages in the given code. The correctness of any answer is checked through *string matching.*

In the CMP instance, one source code and two images are given. The first image represents the output that can be obtained by running the source code. The second image is the output of the answer code that requests students to modify the source code. The correctness of the answer is also checked through string matching. Data analysis or data science is one of the important applications in Python programming. Then, data visualization is necessary to illustrate data with figures such as graphs or charts. CMP has been introduced in PLAS to include figures in a problem instance. For data analysis, *Excel* is often used to manipulate the large amount of data within short durations and generate graphs as a powerful tool. It has become entrenched in business processes worldwide for diverse functions and applications.

In this paper, we study CMP for learning how to use Python programming libraries to manipulate data in *Excel* files.

A *hint function* is implemented for each CMP instance to assist learners in solving it. We generated 25 CMP instances that cover Python libraries for important Excel operations through three steps. First, we collect the relevant source code in the website [5] and the text book [6]. Second, we find and modify the important parts in the code for understanding the Excel operation. Third, we run both the original and modified codes to obtain the outputs and convert them into images. Finally, we generate the HTML/CSS/JavaScript files for this new instance that will be used in the PLAS answer function using a web browser.

To evaluate the generated instances, we assigned them to 13 students in Okayama University, Japan. Their solution results show that the average correct rate for any instance exceeds 99%. Thus, the difficulty levels of the 25 CMP instances for Excel operations in this paper, are proper for novice learners and confirm its effectiveness as a viable self-study tool.

The rest of this paper is organized as follows: Section II discusses related literature reviews. Section III presents CMP instance generations for learning Excel operations. Section IV describes evaluation results. Finally, Section V concludes this paper with future works.

II. Related works In literature

In this section, we briefly discuss related works in literature.

In [7], Cai et al. studied performances of scientific applications with Python programming. They investigated several techniques for improving computational efficiencies of serial Python codes and discussed basic programming techniques for parallelizing serial scientific applications.

In [8], Elenbogen et al. described a set of developed interactive web exercises and a development environment designed to facilitate language acquisition in a beginning course in C++.

In [9], Rowe et al. implemented VINCE-an online tutorial tool for teaching introductory programming to allow the execution of a C program to be graphically displayed. The tool is written in Java, allowing it to be used on the web so that the student can enter their own C code, or select from a menu of pre-written tutorials, each illustrating a particular aspect of programming.

In [10], Sodhi et al. explored basic concepts related to the machine learning and attempted to implement its applications using *Python*. The author majorly used Scikit-Learn library for implementing the applications.

In [11], Guo presented online Python tutor so that the teachers and students can write Python programs directly in the web browser (without installing any plugins), step forwards and backwards through execution to view the run-time state of data structure, and share their program visualizations on the web.

In [12], Sajaniemi et al. introduced program animation system, PlanAni, that is based on the concept of the role of variables, which represents schematic uses of variables that occur in programs over and over again, and a set of nine roles cover practically all variables in novice-level programs.

In [13], Stasko presented TANGO: a framework and system for algorithm animation by developing a conceptual framework with formal models and precise semantics for algorithm animation. The framework contains facilities for defining operations in an algorithm, designing animations, and mapping the algorithm operations to their corresponding animations.

In [14], Levy et al. developed the Jeliot 2000 program animation system intending for teaching introductory computer science to high school students with the goal of helping novices understand basic concepts of algorithms and programming such as assignment, I/O and control flow, whose dynamic aspects are not easily grasped just by looking at the static representation of an algorithm in a programming language.

III. CMP for EXCEL OPERATIONS

In this section, we present *code modification problems* for learning Python programming libraries for Excel operations**.**

A. CMP Instance Definition

In a CMP instance for Excel operations in this paper, generally, one source code and three images showing Excel files are given to the learner. The first image shows the input data file for the source code to generate the output file illustrated by the second image. The third image does another output file that will be generated by the answer source code obtained by a learner. The answer code should be made by modifying the given source code after carefully checking the difference between the second and third images. By solving the CMP instances, the learner can master how to use Python libraries for common Excel operations. The correctness of the answer code is verified through string matching of each statement with the original code.

B. CMP Generation Procedure

A CMP instance can be generated through the following procedure:

1) To select one relevant Python source code from a website or a text book that contains the target library for this instance,

2) To prepare an input Excel file to the code and modify it if necessary,

3) To find the important parts (functions, variables, or parameters) in the code that should be modified by the learner to learn this library,

4) To make the answer code by replacing the modified parts in the original code,

5) To run the original and modified codes to obtain the corresponding output Excel files,

6) To put together the source code, the modified code, and the correct answers into one text file,

7) To run the program with this text file to generate the HTML/CSS/JavaScript files for the PLAS answer function,

8) To modify the HTML file to add the images, and,

9) To assign the generated CMP instance to learners,

C. *Answer Interface*

The answer interface for a CMP instance is implemented using a web browser. It allows a learner to solve the instance both on online and offline, since the answer marking is processed by running the JavaScript program on the web browser. Since the correct answers need to be distributed to the learners, they are encrypted using *SHA256* to avoid cheating**.**

Fig. 1 and 2 illustrate the answer interface for an example CMP instance. In this instance, actually two lines (line 8 and line 10) in "Source Code" are requested to be modified in by a learner who needs to modify the corresponding lines in "Output".

The first image in Fig. 1 represents the common input Excel file for the given code and the modified code. The second image represents the output Excel file that is generated by running the given code. It sorts the lines in the input file in ascending order of 'Code_no' where the tie is resolved by 'Name'. The third image does the output file that is obtained by sorting the lines in descending order of 'No' instead. Actually, it requests to change 'No' and 'ascending = True' to 'No' and 'ascending = False' in line 8, and 'Sortingvalues1.xlsx' to 'Sortingvalues2.xlsx'in line 10. Fig. 2 shows that the learner still needs to modify line 10**.**

After the learner modifies the source code in the input forms, he/she should click the "Answer". If the answer is not correct, the background color of the corresponding input form becomes red. Otherwise, it is white. It is possible to submit answers repeatedly until all the answers become correct. Moreover, the 'Hint' button is implemented for helping the learner by explaining the running steps of the modified source code.

D. *Generated CMP Instances*

In this paper we generated 25 CMP instances using source codes in the website [5] and the text book [6]. Table I shows the instance number, the topic, the number of blanks for each CMP instance. Here, the average correct rate of the students in the next section is also included.

As the topics, we select important Excel operations such as operations on multiple sheets, replacing missing values, sorting data values. These operations are essential in data analysis by cleaning, transforming, or reducing data.

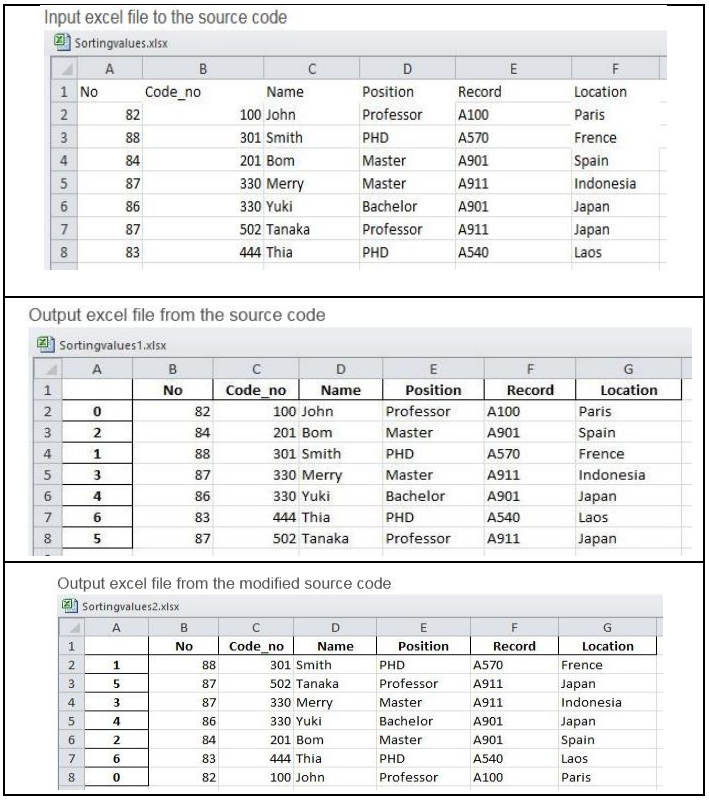
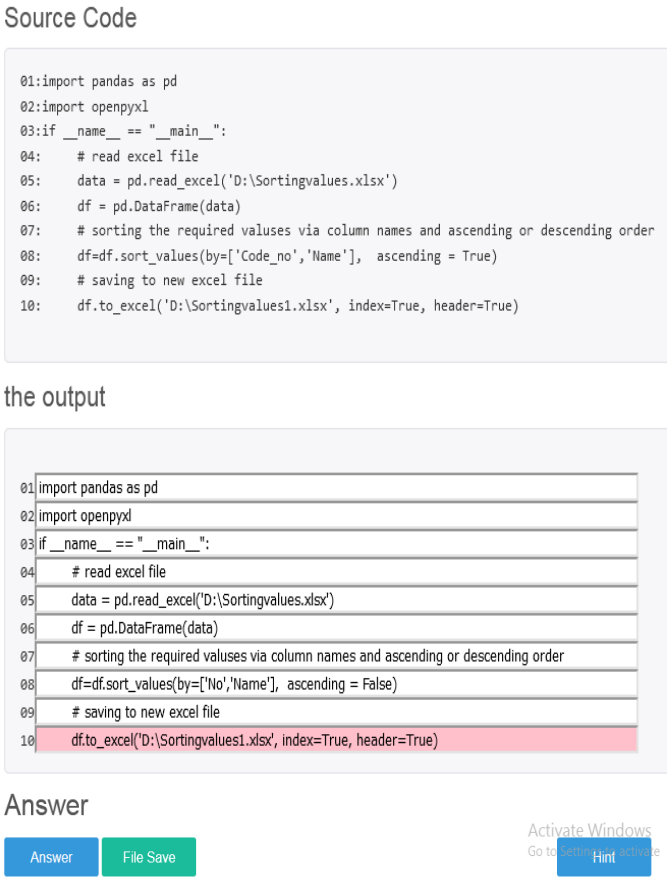


Fig. 1. Three image for Code Modification Problem, CMP instance.



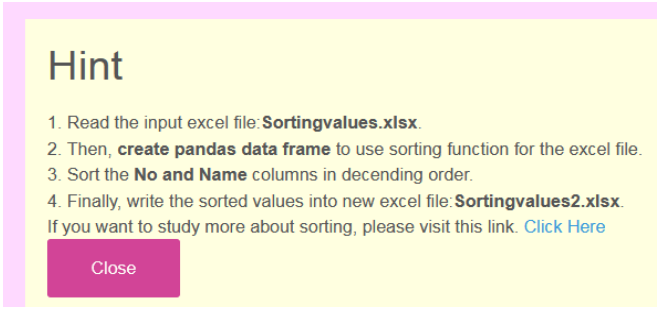


Fig. 2. Answer interface for Code Modification Problem, CMP instance.

TABLE I. GENERATED CMP INSTANCES

inst. #	topic	# of blanks	average correct rate
1	reading and writing excel files	5	99%
2	removing columns	4	100%
3	removing rows	5	100%
4	finding duplicate rows	5	100%
5	replacing null value	3	99%
6	calculating current age	3	100%
7	finding maximum and minimum date	4	100%
8	finding day of week	2	99%
9	operations on given date	3	100%
10	assigning new columns	4	100%
11	adding columns based on condition	10	100%
12	renaming column names	9	100%
13	replacing multiple values	13	100%
14	sorting data values	11	100%
15	searching value	10	100%
16	separating files	10	100%
17	pivot table creation	7	100%
18	merge one file to another file	5	100%
19	merge on specific column	4	100%
20	concatenation of multiple sheets	5	100%
21	operations on multiple sheets	6	100%
22	writing data into csv file	4	100%
23	creating multiple csv files	5	100%
24	converting text file into csv file	5	100%
25	appending a new row to an existing csv file	6	100%
	Average	5.92	99.89%
	Total (SD)	148	(1.1%)

TABLE II. CORRECT ANSWER RATE DISTRIBUTION.

correct rate	# of students
100%	10
99%	3

TABLE III. SUBMISSION TIMES DISTRIBUTION.

submission times range	average # of submissions (SD)	average # of submissions for one instance	# of students
25-50	47 (0)	1.88	1
51-76	69.5 (4.9)	2.78	2
77-102	97.5 (3.9)	3.9	4
103-128	113 (9.5)	4.52	3
129-154	149 (3.6)	5.96	3

IV. APPLICATION RESULTS

In this section, we evaluate the 25 CMP instances through the applications to 13 undergraduate students in Okayama University, Japan, who have not studied Python programming.

A. *Correct Rate for Each Instance*

Table I indicates that the average correct rate among all the students is 100% for the instances except for #1, #5, and #8, where the rate of the three instances is 99%. The overall average rate for all the instances is 99.89% and the standard deviation (SD) is 1.1%. Therefore, the generated CMP instances are not difficult for the students.

B. *Correct Rate for Each Student*

Table II shows the distribution of the correct answer rates of the students. It suggests that 10 students among 13 correctly solved all the CMP instances and three students achieved the 99% correct rate.

C. *Submission Times for each Student*

Table III shows the distribution of the number of answer submission times by the students. This table suggests that for one instance, the minimum average number is 1.88 and the maximum one is 5.96. Therefore, the students can easily solve all the CMP instances. Thus, our proposal is generally proper to study Python programming libraries for Excel operations.

V. CONCLUSION

This paper presented the study of the code modification problem (CMP) for learning Python programming libraries on Excel operations in Programming Learning Assistant System (PLAS). 25 CMP instances were generated using source codes in a textbook and a website that cover common Excel operations. The application results to 13 students in Okayama University confirmed the validity of the proposal with almost 100% correct rate for all instances. In future, we will generate CMP instances for other popular libraries for IoT (Internet of Things), machine learning, and multimedia.

ACKNOWLEDGMENT

We are very grateful to our laboratory members for fruitful discussions of advancing this research. We would like to thank to all the students who participated in this research.

REFERENCES

[1] S. T. Aung, N. Funabiki, Y. W. Syaifudin , H. H. S. Kyaw, S. L. Aung, N, K. Dim, and W.-C. Kao, "A proposal of grammar-concept understanding problem in Java programming learning assistant system," J. Adv. Inform. Tech. (JAIT), vol. 12, no. 4, pp. 342-350, Nov. 2021.

[2] K. K. Zaw, N. Funabiki, and W.-C. Kao, "A proposal of value trace problem for algorithm code reading in Java programming learning assistant system," Inform. Eng. Exp., vol. 1, no. 3, pp. 9-18, Sep. 2015.

[3] N. Funabiki, H. Masaoka, N. Ishihara, I-W. Lai, and W.-C. Kao, "Offline answering function for fill-in-blank problems in Java programming learning assistant system," in Proc. ICCE-TW, pp. 324-325, May 2016.

[4] S. H. M. Shwe., N. Funabiki, Y. W. Syaifudin, P. P. Tar, H. H. S.Kyaw, H. A. Than, W.-C. Kao, N. W. Min, T. Myint, and E. E. Htet, "Value trace problems with assisting references for Python programming self-study", Int. J. Web Inform. Syst., June 2021.

[5] Excel Tutorial, https:// https://www.w3schools.com/EXCEL/index.php.

[6] Working with Excel spreadsheets, https://automatetheboringstuff.com/chapter12.pdf.

[7] X. Cai, H. P. Langtangen, and H.Moe, "On the performance of the Python, programming language for serial and parallel scientific computations," Sci. Programm., vol. 13, no. 1, pp. 31-56, Jan. 2005.

[8] B. S. Elenbogen, B. R. Maxim, and C. McDonald, "Yet, more web exercises for learning C++," in Proc. ACM SIGCSE Bulletin, vol. 32, no. 1, pp. 290-294, May. 2000.

[9] G. Rowe, and G. Thorburn, "VINCE - an on-line tutorial tool for teaching introductory programming," in Proc. ACM SIGCSE Bulletin, vol. 30, no. 3, Sept. 1998.

[10] P. Sodhi, N. Awasthi, and V. Sharma, "Introduction to machine learning and its basic application in Python," in Proc. Int. Conf. Digital Strat. Org. Success, pp. 1354-1375, Apr. 2019.

[11] P. J. Guo, "Online Python tutor: embeddable web-based program visualization for CS education," in Proc. ACM Tech. Symp. Comput. Sci. Edu., pp. 579-584, Mar. 2013.

[12] J. Sajaniemi, and M. Kuittinen, " Program animation based on the roles of variables," in Proc. ACM Symp. Soft. Visual., pp. 7-16, Jun. 2003.

[13] J. T. Stasko, "Tango: a framework and system for algorithm annimation," IEEE Computer, vol. 23, no. 9, pp. 27-39, Sept. 1990.

[14] R.B.Levy, M.Ben-Ari, and P.A.Uronen, "The jeliot 2000 program animation system", Comput. Edu., vol. 40, no. 1, pp. 1-15, Jan. 2003.