International Journal of Advanced Culture Technology Vol.9 No.1 203-209 (2021)
DOI https://doi.org/10.17703/IJACT.2021.9.1.203

IJACT 21-3-28

A Study on Coding Education for Non-Computer Majors Using Programming Error List

Hye-Wuk Jung

Assistant Professor, Dept. of College of Liberal Arts and Interdisciplinary Studies, Kyonggi University, Korea
wukj@kyonggi.ac.kr

Abstract

When carrying out computer programming, the process of checking and correcting errors in the source code is essential work for the completion of the program. Non-computer majors who are learning programming for the first time receive feedback from instructors to correct errors that occur when writing the source code. However, in a learning environment where the time for the learner to practice alone is long, such as an online learning environment, the learner starts to feel many difficulties in solving program errors by himself/herself. Therefore, training on how to check and correct errors after writing the program source code is necessary. In this paper, various types of errors that can occur in a Python program were described, the errors were classified into simple errors and complex errors according to the characteristics of the errors, and the distributions of errors by Python grammar category were analyzed. In addition, a coding learning process to refer error lists was designed to present a coding learning method that enables learners to solve program errors by themselves.

Keywords: *Program Error, Coding Education, Non-Computer Majors, Software Education, Error List*

1. INTRODUCTION

As software education has been activated recently to cultivate talent in the Fourth Industrial Revolution era, universities are carrying out coding education for non-computer majors[1]. The learners encounter numerous program errors in the process of learning the basic theory of computer programming and practicing using examples[2]. Program errors can be classified into syntax errors, runtime errors, and logic errors, as shown in Table 1.

Table 1. Types of program errors

Types of errors	Example
Syntax errors	Incorrect keyword, Incorrect punctuation, Undefined variable
Runtime errors	Division by zero, Infinite loop, Incorrect conditional expression, Incorrect operation expression
Logic errors	Incorrect algorithm, Incorrect grammar

Manuscript received: February 3, 2021 / revised: February 10, 2021 / accepted: February 19, 2021
Corresponding Author: wukj@kyonggi.ac.kr
Assistant Professor, Dept. of College of Liberal Arts and Interdisciplinary Studies, Kyonggi University, Korea

Syntax errors are compilation errors that occur during the compilation process due to grammatical errors in the written code and appear when a keyword or punctuation mark is written incorrectly, or an undefined variable is used. Runtime errors occur when a program is not executed normally due to an incorrect conditional expression or operation expression, and logic errors are errors that lead to the inability to obtain desired results from a program created due to a problem with the algorithm or grammar[3].

The various errors that occur when coding a computer program are corrected through processes in which the learner examines the source code to find and solve the errors. These processes are indispensable to identify the results of the written program or to complete program development[4].

Non-computer majors who are not familiar with computer languages receive feedback from instructors to correct program errors because they can hardly solve program errors by themselves[5]. However, since there are limitations for the instructor to teach a large number of students with only his/her feedback for a limited time, programming learning methods that enable the learner to efficiently resolve errors by themselves have been proposed.

J. M. Lee and K. H. Lee designed a mental-model using metaphors, applied it to classes, and analyzed changes in the error patterns of novice learners of programs. He showed that learners who used metaphors made fewer program errors than those who did not[6]. However, since learning effects appear differently according to the levels of difficulty of tasks and only limited conditions were considered, the diversity of learners should be considered.

H. Jang et al. presented a method to construct an error feedback system for beginners in programming in order to provide feedback and examples for code errors, thereby enabling learners to quickly correct code errors[7]. J. W. Nam and I. H. Yoo analyzed the errors generated in robot programming by type and developed an error resolution support tool based on the foregoing analysis so that learners can easily solve code errors[8]. P. Koyya et al. proposed a method to improve code by providing feedback when a learner completes an assignment using software metrics and reference codes[9]. A. J. Ko and B. A. Myers developed a model to classify programming error types and infer, explain, and predict errors to derive design guidelines for new programming tools[10]. S. Krusche et al. built a code review workflow through peer review of programming learners so that it can be used for software development and quality control[11]. These methods provide support systems for learners to resolve code errors but have shortcomings in that beginners who are new to the programs may feel burdened in learning how to use these systems and can hardly develop the ability to resolve errors by themselves because they may become dependent on the systems.

There are limitations in applying existing studies to online classes at universities that have been changing recently due to the effect of COVID-19 because they did not consider the online learning environment. When holding a coding class online, the instructor can provide feedback to learners' questions through e-mail, etc., and present problem-solving methods through real-time video conferencing. However, in an online classroom environment where the time to study alone is long, when a coding error occurs while the learner is practicing by himself/herself, the learner has to review and correct the error by himself/herself and experiences many difficulties in the process. Consequently, developing the ability of learners to learn and solve errors by themselves is a very important issue in the online classroom environment.

Therefore, in order to enable non-computer major learners to solve coding errors by themselves, errors will be classified and analyzed in this paper according to the characteristics of errors occurring in Python programs and a coding learning method with the process applied to refer to error lists will be proposed.

2. CLASSIFICATION AND ANALYSIS OF PYTHON PROGRAM ERRORS

In this paper, errors were classified for the Python language, which can be learned easily even by beginners and is widely used for education[12, 13]. As for error classification, various types of errors were classified into simple errors and complex errors according to the degree of difficulty for learners to solve the errors by themselves so that learners can easily refer to them when studying on their own. Figure 1 shows the process of classifying various types of errors that can appear in Python grammar into simple and complex errors according to their characteristics.

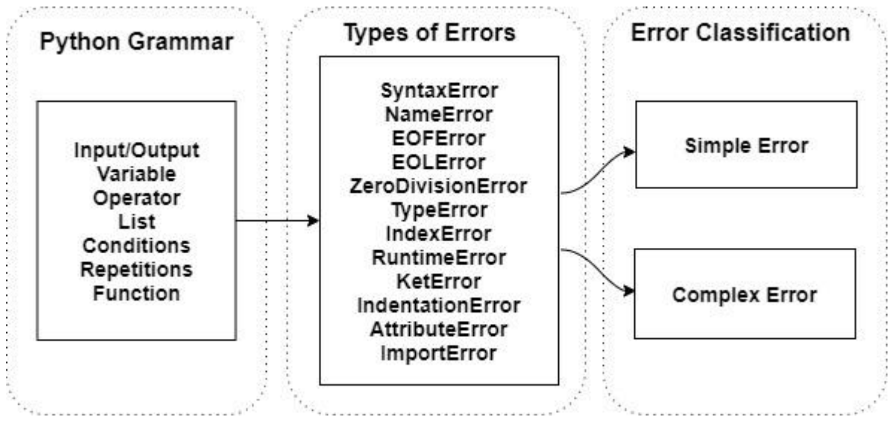


Figure 1. Classification according to Python grammar and types of errors

Examples of errors classified into simple and complex errors are shown in Table 2.

Table 2. Simple error and complex error of Python

Error Classification	Example	Types of errors
Simple Error	Typing mistake	SyntaxError
	Incorrect punctuation marks	SyntaxError
	Undefined variables	NameError
	Typing mistake in function’s bracket	EOFError
	Typing mistake in quotation marks	EOLError
	Division by zero	ZeroDivisionError
	Incorrect data type	TypeError
	List index out of range	IndexError
	Call of non-existent key	KeyError
Complex Error	Incorrect formula declaration	RuntimeError
	Incorrect use of conditional statements	LogicError
	Incorrect use of repetition statements	LogicError
	Incorrect indentation	IndentationError
	Incorrect use of “return” in a function	TypeError
	Incorrect use of parameters in a function	TypeError
	Use of incorrect function names	NameError
	Use of non-existent attribute of a module	AttributeError
	Referring an incorrect function of a module	NameError
	Use of incorrect module names	ImportError

Simple errors mean ones that occur in the basic rules of Python, which happen frequently in Python grammar, and include SyntaxError, NameError, End of File(EOF) Error, End of Line(EOL) Error, ZeroDivisionError, TypeError, IndexError and KeyError. Complex errors are ones that occur when the structured grammar of Python is incorrectly written, which happens when using structural grammar, formulas, or conditions, and these include RuntimeError, LogicError, IndentationError, TypeError, NameError, AttributeError and ImportError. On reviewing the simple and complex errors, it can be seen that there are overlapping errors. This is because the classification into simple and complex errors was not simply based on types of errors, but was also carried out focusing on situations where the errors occur.

Various situations where simple errors happen can be explained as follows. A SyntaxError occurs when any Python reserved word or punctuation mark is used incorrectly, and a NameError occurs in relation to variable names. An EOFError, which appears when the unexpected end of a file is recognized, happens because a parenthesis was not opened and closed when using a function, and an EOLError is caused in the case where the quotes used are an odd pair of marks when entering a string. In addition, a ZeroDivisionError happens when a number is incorrectly divided by 0, a TypeError occurs when the data type is incorrectly changed in an input statement, an IndexError occurs when the index range is exceeded when using a list, and a KeyError occurs when a key that does not exist is called.

Various situations where complex errors happen can be explained as follows. A RuntimeError occurs when a formula is declared incorrectly because the command is not executed normally. When a formula is used incorrectly in a conditional statement or a repetitive statement, a LogicError occurs so that the program is abnormally terminated. An IndentationError occurs when the indentation spacing between sentences is not set properly, and a TypeError occurs when a parameter or return statement is written incorrectly when using a function. In addition, a NameError occurs when there is an error in the name of a function or an error happens when referring to the function in a module, and in a module, an AttributeError and an ImportError occur due to wrong attributes and a module name, respectively.

In this paper, the distributions of error types by Python grammar category were measured to classify the errors described above, and the contents of feedback between learners and teachers in basic software basic course held from 2018 to 2019 at K University in Gyeonggi-do, South Korea were used. All 554 learners who participated in the classes were non-computer majors, and the Python classes were given for sequential learning of input/output functions, variables, operators, lists, conditional statements, repetitive statements, functions, and modules[14]. As for the error data, the contents of those errors that were corrected by learners through feedback from the instructor while they were practicing coding during class hours were used.

The rates of error occurrence by Python grammar category were measured as shown in Equation (1).

$$\mathrm{p}(\%) = \left(\frac{1}{M}\sum_{i=1}^{M}\frac{n_i}{N}\right) \times 100 \qquad (1)$$

where p(%) denotes error occurrence rate, M denotes the number of students surveyed, and N denotes the total sum of detailed errors for each error. ni denotes the number of detailed errors in each grammar in the case of the ith student.

The distributions of errors obtained when the errors occurring in input/output functions, variables, operators, lists, conditional statements, repetitive statements, functions, and modules input/output functions, variables, operators, lists, conditional statements, loop statements, functions, and modules were classified into simple and complex errors are shown in Figure 2.

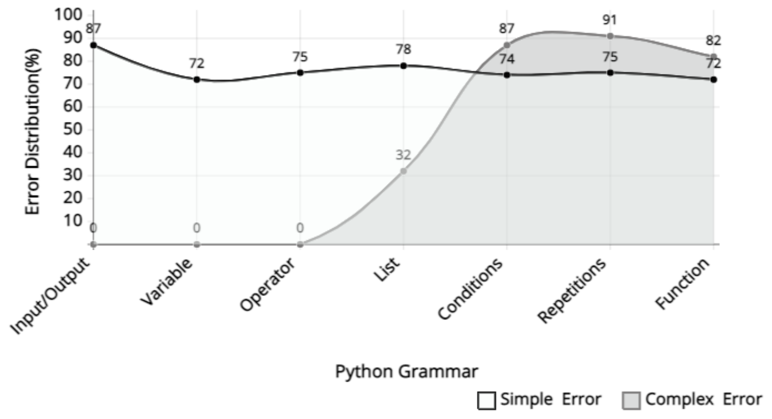


Figure 2. Error distribution of Python grammar

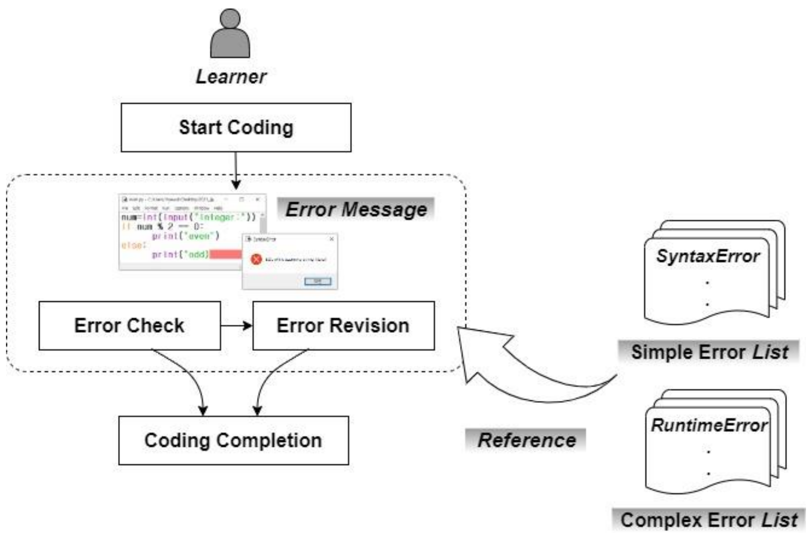
Simple errors occurred frequently in general in each grammar and showed an average distribution ratio of 76.14%. The input/output portion showed an error distribution ratio of 87%, which was the highest among simple errors. This is judged to be a phenomenon that takes place because the learners were unfamiliar with the Python language in the part of using input/output functions at the beginning of learning coding. In addition, since simple errors happened continuously and frequently even though the content of the grammar used for coding increased, it can be seen that care should be used when writing source codes and checking errors.

Complex errors appeared in those lists, conditional statements, repetitive statements, functions, and modules that use structural grammar, formulas, or conditions. A high complex error distribution ratio of 91% was shown in the repetitive statement part, while there were no complex errors in inputs/outputs, variables, and operators. These complex errors are related to the necessity to write a formula correctly or use functions to fit their purposes so that the results wanted by the programmer can be output. Therefore, concrete, and systematic algorithms should be designed in the program planning stage before writing the source code. In addition, as the contents of the grammar become more complex, dividing blocks of code by function, or using functions efficiently to fit their purposes is also important.

3. CODE LEARNING METHOD BASED ON REFERRING ERROR LISTS

Non-computer majors learn the basics of coding while practicing. During this time, they make various errors, and the source code is corrected through feedback from the instructor. However, in cases where the learner practices alone for long periods of time, there are limitations in receiving feedback from the instructor to resolve errors. Therefore, a method for the learner to develop the ability to correct code errors by himself/herself is desperately necessary.

Therefore, in this paper, a learning process was designed through which the learner resolves errors made when he/she writes and executes the source code by himself/herself, referring to the lists of simple errors and complex errors, as shown in Figure 3.



```
num=int(input("integer:"))
if num % 2 == 0:
    print("even")
else:
    print("odd)
```

Figure 3. Code learning process based on referring error lists

The designed learning process progresses as follows. The learner writes the source code for the given problem and executes it. If there is a problem with the written source code, a code error occurs and the result is not displayed normally, and the Python editor shows an error notification message. Although a solution to this error can be obtained right away in cases where there is an instructor, it is difficult for a non-major learner to find a solution when they are studying by themselves. In such cases, the learner finds the relevant error by comparing the error contents in the simple error list and the complex error list with the content of the error notification message. Because these error lists are based on data from previous learners, a student can easily find out the error. The learner corrects the code based on the error information found, and if the source code is executed normally, the learner outputs the result and completes the code design. If other error notification messages are generated consecutively, the learner should repeatedly carry out the error list referral method mentioned out earlier. The learning procedure for practicing coding by referring to the error lists is as follows.

Step 1. The learner writes and runs the program.
Step 2. The learner checks the error messages generated when the code is executed.
Step 3. The learner corrects the errors generated by referring to the error lists.
Step 4. The learner completes the code design.

The learner repeats steps 1 through 4 in the coding practice process, becomes familiar with the error lists along with program learning, and learns to match and refer to error notification messages and error lists. In particular, learners can develop their error correction ability by themselves through the process of matching and referring to the simple and complex errors lists classified earlier with the error notification messages from the Python editor.

In this paper, a learning process to find and correct errors by referring to a list of errors classified into simple and complex errors was designed. The coding learning method, which enables learners to resolve errors by themselves, has the characteristics of eliminating the fear of coding errors felt by non-computer majors and facilitating coding practice. In addition, unlike existing studies, the method can be useful for learning coding in an online environment.

4. CONCLUSION

When an error happens in the coding process, non-computer majors who are learning computer programs for the first time receive feedback from instructors to modify the program. However, in situations where there are long periods of time to learn online due to the recent pandemic, there are limitations in receiving feedback from the instructor to resolve errors. Therefore, a learning method that would enable learners to check and correct program errors by themselves is very much needed.

In this paper, the types of errors in Python, which is widely used as an educational computer language, were described, and the distributions of error occurrences by Python grammar category were analyzed. Based on the results of the analysis, a learning method was designed focusing on situations where errors happen so that learners can easily refer to the errors when they learn programming by themselves. Also, various types of errors were classified into simple errors and complex errors according to the degree of difficulty felt by learners when they solve the errors by themselves. In addition, a learning process in which learners can find and correct errors and carry out coding by referring to the error lists classified was presented. Through this learning process, learners can become familiar with the program error lists and develop the ability to correct errors by themselves in situations where there is no feedback from instructors, such as online coding classes.

Hereafter, based on the learning process presented in this paper, concrete guidelines for solving coding errors will be designed and applied to actual coding classes to identify the practical utility of the guidelines.

Acknowledgement

This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2015R1D1A1A01061064 and NRF-2020S1A5A8042850).

REFERENCES

[1] H. W. Jung, “A Study on the Current State of Artificial Intelligence Based Coding Technologies and the Direction of Future Coding Education,” International Journal of Advanced Culture Technology, Vol. 8,

No. 3, pp. 186-191, 2020.
DOI: https://doi.org/10.17703/IJACT.2020.8.3.186

[2] J. Kim and Y. Kim, “The Analysis of Relationship between Academic Achievement Level of Concept Learning and Error Type in Online Programming Course,” The Journal of Korean Association of Computer Education, Vol. 17, No. 5, pp. 43-51, 2014.

[3] G. Samara, “A Practical Approach for Detecting Logical Error in Object Oriented Environment ,” World of Computer Science and Information Technology Journal (WCSIT), Vol. 7, No. 2, pp. 10-19, 2017.

[4] W. R. Murray, "Automatic program debugging for intelligent tutoring systems," Computational Intelligence, Vol. 3, No. 1, pp. 1–6, February 1987.
DOI: https://doi.org/10.1111/j.1467-8640.1987.tb00169.x

[5] S. K. Kummerfeld and J. Kay, "The neglected battle fields of syntax errors," in Proc. ACE '03: Proceedings of the fifth Australasian conference on Computing education, pp. 105–111, 2003.

[6] J. M. Lee and K. H. Lee, “The Effects of Metaphoric Instruction on Novice’s Learnnig of C Language Programming,” Korean Journal of Cognitive Science, Vol. 9, No. 4, pp. 75-93, 1998.

[7] H. Jang, S. Choi, S. Jun, Y. Yeom and W. Lee, “Programming Learning Supporting System based on Error Feedback for Novices,” The Journal of Korean Association of Computer Education, Vol. 10, No. 6, pp. 1-10, 2007.

[8] J. W. Nam and I. H. Yoo, “Development of NXC Robot Programming Supporting System Based on Types of Programming Error,” Journal of The Korean Association of Information Education, Vol. 15, No. 3, pp. 375 - 385, 2011.

[9] P. Koyya, Y. Lee and J. Yang, "Feedback for Programming Assignments Using Software-Metrics and Reference Code", International Scholarly Research Notices, vol. 2013, Article ID 805963, 8 pages, 2013.
DOI: https://doi.org/10.1155/2013/805963

[10] A. J. Ko and B. A. Myers, "Development and evaluation of a model of programming errors," in Proc. IEEE Symposium on Human Centric Computing Languages and Environments, pp. 7-14, 2003.
DOI: https://doi.org/10.1109/HCC.2003.1260196

[11] S. Krusche, M. Berisha and B. Bruegge, "Teaching Code Review Management Using Branch Based Workflows," in Proc. 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), pp. 384-393, 2016.
DOI: https://doi.org/10.1145/2889160.2889191

[12] I. Sanders and S. Langford, "Students' perceptions of python as a first programming language at wits," ACM Sigcse Bulletin, Vol. 40, No. 3, 2008.
DOI: https://doi.org/10.1145/1597849.1384407

[13] Python. https://www.python.org.

[14] H. W. Jung, “Exploring the Convergence Possibility of Learner's Major Field in Software Project - Focus on the Non-Computer Major Freshmen -,” The Journal of the Convergence on Culture Technology (JCCT), Vol. 6, No. 2, pp. 263-270, 2020.
DOI: https://doi.org/10.17703/JCCT.2020.6.2.263