RECODEZ: AN INTELLIGENT AND INTUITIVE ONLINE CODING EDITOR USING MACHINE LEARNING AND AI

Justin Kim[1], Yu Sun[2] and Fangyan Zhang[3]

[1]Los Osos High School, Rancho Cucamonga, CA 91739, USA
[2]California State Polytechnic University, Pomona, CA 91768, USA
[3]ASML, San Jose, CA 95131, USA

ABSTRACT

Recent years have seen a large increase in the number of programmers, especially as more online resources to learn became available. Many beginner coders struggle with bugs in their code, mostly as a result of a lack of knowledge and experience. The common approach is to have plenty of online resources that can address these issues. However, this is inconvenient to the coder, who may not have the time or patience to look for a solution. In this project, we address this problem by integrating the coding and error resolving environment. A website has been developed that examines code and provides simpler error messages that give a more comprehensive understanding of the bug. Once an error has been added to the database, the program can display the error more understandably. Experiments show that given several sample programs, our tool can extract the errors and report a more easily understandable solution.

KEYWORDS

Programming Environment, Python, Server, Database

1. INTRODUCTION

Programming is the act of writing instructions for a computer [1][2]. This often includes several processes, such as writing code and fixing errors (debugging). With the increasing popularity of programming, especially as more online resources become available that teaches beginners how to code, the necessity of making coding a simpler process becomes more apparent. Whereas many resources give help on the writing code aspect of programming, few exist that work on making errors easier to fix. However, fixing bugs is difficult even for the experienced programmer, and therefore this process must be simplified further for beginner programmers.

When a beginner coder writes their first programs, there are several large challenges that they face [4]. One of these is that code must be error-free. When a program is run, if there are errors or bugs, the program will not function as expected. Often, this results in the program failing to complete its task. However, a beginner programmer will often not know how to fix these errors. They lack the knowledge and experience necessary to resolve the bugs.

As another of the challenges that a beginner coder faces are that they lack the motivation to create programs, a simple bug could cause enough frustration for them to quit programming [12][13]. This is not ideal, as programming is an essential skill as more and more jobs become mechanized.

David C. Wyld et al. (Eds): MLNLP, BDIoT, ITCCMA, CSITY, DTMN, AIFZ, SIGPRO - 2020
pp. 157-164, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.101216

Without more programmers learning how to code, many will be left behind by those who have studied some computer science. Thus, it is important to improve the simplicity of errors and make them easier to understand.

Several resources attempt to create simpler errors. Some websites allow a user to ask questions and receive answers from other experienced coders [3][11]. These allow people who need help to reach other people who may have experienced similar issues and were able to solve them. However, this is not a convenient method. If the programmer cannot find another person with the same issue, then they must ask the question themselves. This can result in their question taking time to answer as few have the same problem and fewer have a solution. In the case that a solution exists, they often are not tailored to fit the programmers' needs, which can lead to confusion when trying to implement this fix into their own code.

Another system that creates simpler to understand errors is the programming environment [5][6]. They often have built-in software that checks for errors as you write code, and create error messages that are given to the user. The error messages are usually short and often highlight the location necessary to fix. However, they don't usually describe how to fix the error, and just notify the user that a problem exists. They must go to another resource to find out how to truly fix the error. Additionally, these error messages are often too short to be very descriptive, such as not including the definition of the error. Some error messages have extraneous information that does not help resolve the issue, such as error codes that wouldn't make sense to a beginner programmer.

In this paper, we present ReCodeZ, an online tool to improve the experience of beginner programmers in resolving errors. Similar to the programming environment, both the code and the output areas are part of the same application. This allows for more convenience as the programmer does not have to switch tabs or open another application to see the results of the code. However, our method of making debugging easier for beginners is to reword every error message into a more understandable form. Instead of having extra error codes that do not inform the programmer on what the problem is, we give a better response including pointing out where the bug is and several ideas on how to fix it. This is different from the standard programming environment because instead of keeping error messages in a hard to understand way, ReCodeZ outputs an improved version that helps the beginner programmer understand why they have that bug and what they can do to resolve the issue. Compared to other online resources that allow programmers to ask questions to other experienced programmers, our tool is more convenient. Instead of having to search online for help, a beginner programmer can stay on one site and find out how to debug their code.

We demonstrate how the above usage of both the coding and debugging environment on the same website can be both convenient and useful to the beginner programmer. To do so, we conducted a case study on several beginner programmers. We created several sample Python programs that all had an error. These were simple errors that a beginner would be expected to make. Then, by putting these errors into our tool, we determined whether our tool worked as expected. Additionally, we gave the ReCodeZ output to the beginners to see if they were more able to understand what the error was, comparing it to several of the other methods discussed previously.
The remainder of the paper is as follows. Section 2 gives the details on the challenges that we met in designing and testing our tool. Section 3 focuses on the details of our solution corresponding to the challenges stated in Section 2. Section 4 presents relevant details in the experimentation to analyze our solution. Section 5 gives an analysis of related work. Finally, Section 6 concludes and provides insight into future work related to this project.

2. CHALLENGES

2.1 Challenge 1: Detecting Errors

There are several types of errors that can be present in a piece of code, such as syntax errors and arithmetic errors. With all these different kinds of errors, detection becomes more difficult. In order to detect an error, we had to figure out what can cause that error. We then have to create an algorithm that can determine if the code has a certain piece that causes an error. We also have to determine other details about the bug, such as the specific location of the error. This process had to be repeated for every type of error that the Python programming language could possibly give.

2.2. Challenge 2: Describing Errors

For our tool to reach its goal of making error messages more understandable, the errors have to be described in a way that makes sense to a beginner. Some beginners may understand more about coding than others, and creating a one-size-fits-all description of the error is difficult. One person may be able to understand a certain description, but others may not. We had to find the balance between a long explanation and a summary of the error, creating an error message that is clear to any beginner who uses our tool.

For example, using the same error in Figure 1, we had to figure out how to describe that parentheses were missing. Some beginners may not understand that parentheses have to be balanced, so we had to explain what was missing from the code. However, we had to keep the messages short to appease the more experienced programmers, who would not want to read through a long error message.

2.3. Challenge 3: Possible Solutions to the Error

While there are a few errors that only have one solution, the vast majority of them are determined by the programmer's intent. Depending on what the user wanted to happen, the method to fix the error changes. This is a task that is difficult for humans, which means that many algorithms on a computer would not be able to predict what the user wants to happen. Because there can be many solutions to each error, a certain fix of the error may not be the one that results in their code behaving as they expected.

For example, continuing with Figure 1, there are several possible solutions to making the parentheses balanced. However, the possible solution displayed in the error message may not be the exact one that the programmer wanted. While this example is simple, in a more complex piece of code, such as an equation, the correct placement of parentheses depends on the desired function. In such cases, it may be best to leave it to the programmer to figure out what they need, but helping them understand the error as much as possible.

3. SOLUTION

ReCodeZ is an online Python programming environment that clarifies errors for beginner programmers [7]. ReCodeZ was designed to be flexible enough to handle all types of errors. Many Python errors can be handled by ReCodeZ, which has a modular method of implementing additional error messages. Error messages may be added, removed, or edited, allowing ReCodeZ to keep up with the improvement of programming languages. Additionally, other programming languages can be added to our tool. Therefore, it can be used for learning a variety of different programming languages and determining how to fix bugs in each language.

ReCodeZ implements many of the features of a standard programming environment while being online. Users can edit code, run programs, and see the outputs on one page. This allows more people to learn to program and improves convenience for people who do not have access to a computer.

An overview of the entire system is given in Figure 1. Our tool consists of four main parts: the website, the server, the interpreter, and the database. The website handles the user input and displays the output. The server is the main controller of ReCodeZ and combines the functions of the other components. The interpreter runs the code and gives output and an error message. The database holds all of the errors and can retrieve them. When a user runs a program on our website, it sends their code to our server. The server then runs that code using a Python interpreter and returns the output. However, if there is an error, the output is intercepted and an error message from our database is retrieved to replace it.

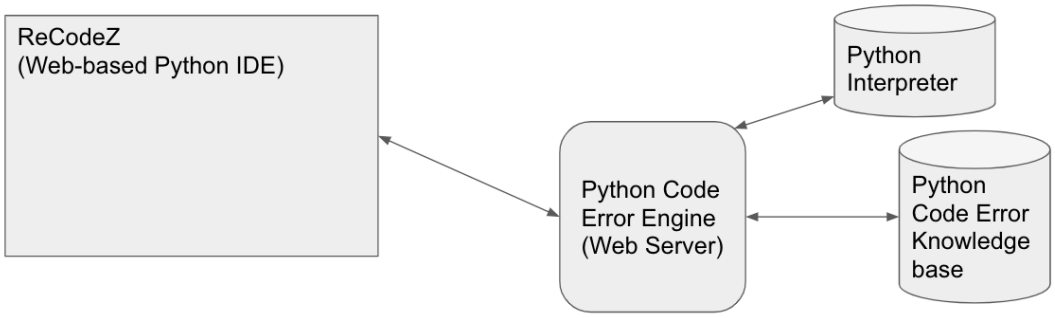

Figure 1. An overview of the system

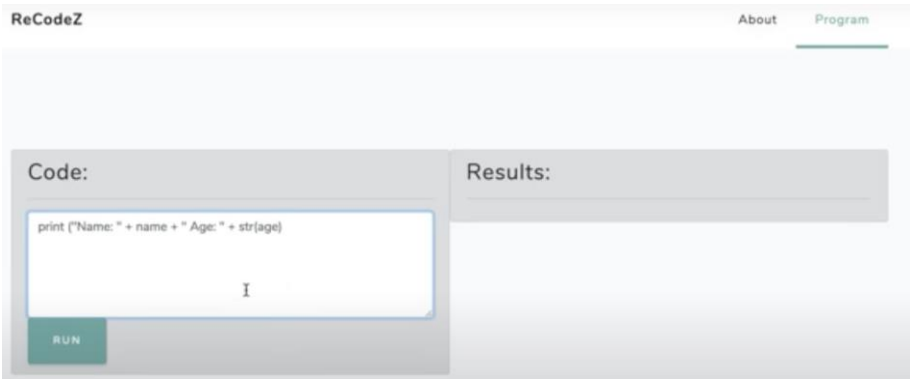

Figure 2. ReCodeZ interface

Our website is where the user can write code, submit it, and see the results. This is done through a text box located on the page that allows the user to type in their code or copy and paste it in. When the user clicks the run button, a JavaScript script submits the code to the server to check and run [15]. Once the server finishes processing the code and returns the output, the website updates the page and displays the output or error message.

The server takes the code passed to it by the website and passes it to the interpreter. After receiving the output, it creates a chain of error handlers. These error handlers are pulled from the database and linked together in a way that allows any errors to be handled in sequence.

As seen in Figure 3, there are three errors checked as a proof of concept. While more errors could be handled, ReCodeZ in the example checks if the code contains unbalanced parentheses, missing colons, or an undeclared variable. These error checkers traverse the code and return if they find an error or pass the code on to the next error checker. Finally, after all errors have been checked, the program gives the output back to the website. The server passes the user's code to the algorithm displayed. This runs another Python file, displayed that will compile and run the user's code. Finally, it splits the error messages into a separate output, allowing our program to capture and replace the errors.

```
private Result checkForErrors(String code) {
    TerminalAccess terminal = new TerminalAccess();
    TerminalOutput overallOutput = terminal.getTerminalOutput(code);

    // executed
    if (overallOutput.getOutput() != null) {
        return new Result(overallOutput.getOutput());
    }

    CodeFileTraverser fileTraverser = new CodeFileTraverser(file);
    ErrorChecker parentheses = new ParenthesesErrorChecker();
    ErrorChecker colons = new ColonErrorChecker();
    ErrorChecker varName = new VariableNameErrorChecker();

    parentheses.setSuccessor(colons);
    colons.setSuccessor(varName);

    fileTraverser.traverse(parentheses);

    int size = ErrorChecker.getMessages().size();
    String[] messages = new String[size];

    for (int i = 0; i < size; i++) {
        messages[i] = ErrorChecker.getMessages().remove();
    }

    return new Result(messages);
}
```

Figure 3. ReCodeZ error check

The database of ReCodeZ consists of several classes that each handle a different error. This allows for a modular design, as more classes can be added without needing to change the other classes. When the server constructs the chain of error handlers, each class can be added to the chain either on its own or with a reference to the next handler. When the error from running does not match the error ReCodeZ is checking, it moves on to the next error. This allows for the abstraction of how errors are checked. Each of the errors is a different class that takes in the output from running the code and checks it against a general form of the error. If the error matches, the program then gives the ReCodeZ error message back to the server to update the website for the user.

4. EXPERIMENT

To test ReCodeZ, we created 20 sample pieces of code. All of these samples were short with only a couple lines, and each had an error. This is so that we could test our tool in creating error messages. We submitted them through our website to get the 20 error messages corresponding to each error. To compare our solution to other programming environments, we also ran each piece of code in a different programming environment, PyCharm, to get 20 more error messages [14]. We used a small sample of 10 beginner programmers in the Southern Californian area. These are

programmers who have only been learning coding for less than a year. While this may not represent all beginner programmers, due to a lack of funding and access to these beginners, this small sample space was all of the volunteers for the study. Each of the participants received all of the pieces of code along with their corresponding error messages from ReCodeZ and PyCharm. However, the order of their list of the pieces was randomized, resulting in that any specific ordering would largely not affect the result. There were 81 total votes for PyCharm error messages and 119 total votes for ReCodeZ error messages. This results in a significant difference of 38 votes. Of the 20 samples, 12 had more people vote for ReCodeZ than for PyCharm error messages, with four being tied and four with the opposite.

The experiment results show that ReCodeZ probably creates better error messages than PyCharm for those 20 samples of code. With 16 of them having at least as good if not better error messages in ReCodeZ, it is likely that ReCodeZ is better for beginner programmers, in terms of errors. We did try to create representative code samples, many taken from common online questions, so the results may be applied to a more general population of errors often experienced by beginner programmers. Our research can also be generalized to beginner programmers similar to those in our study, with less than a year of experience.

5. RELATED WORK

PyLint is a Python helper tool that analyzes code [8]. The main features of PyLint include improving coding quality and error detection. This means that our work and PyLint are relatively similar. PyLint is robust and can handle most errors. However, ReCodeZ is a programming environment, while PyLint is a tool that can only be used on existing code. PyLint cannot be used to create new programs. Additionally, ReCodeZ is more convenient as it is online, while PyLint must be downloaded. However, a strength of PyLint is that it can improve the quality of code. It can format the code to fit programming standards.

Another related work is a StackOverflow integration called Seahawk [9]. StackOverflow is an online resource that allows programmers to submit questions and get answers from other programmers. Seahawk imports this into a Java programming environment. This combines the power of the online resource and the programming environment, making it useful for getting specific answers for beginners. However, it is somewhat not as convenient as ReCodeZ as it is not online. Additionally, Seahawk takes some time to receive answers, as it is based on having other people respond to a user's question. ReCodeZ does not have this issue as all of the error messages are preloaded.

Repl.it, an online programming environment, which allows users to program in many different programming languages [10]. Both ReCodeZ and Repl.it are online environments, making programming more convenient. However, Repl.it has the ability to save code and share files online, which ReCodeZ is currently incapable of doing. However, it does not change error messages like ReCodeZ. ReCodeZ is slightly more convenient as the error messages better explain the errors from a user's code. Users of Repl.it may still have to use another online resource to understand errors, while ReCodeZ has error explanations on the same site.

6. CONCLUSION AND FUTURE WORK

In this project, we proposed a method to improve the experience of debugging for beginner programmers through the use of explanatory error messages. A website has been developed for users to submit Python code and see the output as well as any errors that their code receives. Instead of having error messages that have little information or possible misinformation,

ReCodeZ clarifies error messages and gives possible solutions to help the beginner to debug their code. To test our solution, we conducted a study of 10 beginner programmers. We gave them each 20 pieces of code, along with error messages from ReCodeZ and PyCharm, and asked them to vote on which one they thought had a better description. The results showed that ReCodeZ had more pieces of code with the majority vote and more votes than PyCharm. This means that ReCodeZ was able to give clear error messages that allowed the beginner programmer to understand how to debug their code. Regarding the aforementioned challenges, ReCodeZ solves all three. ReCodeZ is able to detect errors using the Python interpreter in conjunction with our server. ReCodeZ can describe the errors using our database of error handlers. Additionally, while fixing errors to make the program do what the user wants it to do is impossible without knowing what the user wants to happen, ReCodeZ gives possible solutions that may help the user to fix their error themselves.

The limitations of ReCodeZ lie in error detection. There are many different errors even for one programming language. Implementing all of the errors would take a large number of resources. Each error has to have an algorithm to detect it and a template to describe the error. The error may also require another algorithm to create possible solutions. Additionally, handling all of the errors in a chain will result in slow response times and lead to the programmer using a different resource. Our solution would lose its convenience if we added more error handling linked by a chain structure.

Another limitation is that many programming languages, including Python, allow users to create errors. ReCodeZ currently does not have the capability to fix these user-created errors as it does not know what the error was made to do. Similarly, as mentioned in Challenge 3, ReCodeZ cannot always predict the correct solution as there are often multiple solutions that each result in different outputs.

In future works, these limitations can be solved. A possible way to implement all of the errors is to use machine learning to generate text to describe errors. Another method is to use online resource integration, like Seahawk with StackOverflow, but with preset questions and answers. The chain structure of error handling can be replaced with a lookup table that can access errors that would be farther down the chain with higher efficiency. Additionally, default error messages can be added so that user-created errors can still be described. Finally, with machine learning, a solution that fits the user may be more likely to be predicted.

REFERENCES

[1] Dahl, Ole-Johan, Edsger Wybe Dijkstra, and Charles Antony Richard Hoare, eds. Structured programming. Academic Press Ltd., 1972.
[2] Baker, Catherine M., Lauren R. Milne, and Richard E. Ladner. "Structjumper: A tool to help blind programmers navigate and understand the structure of code." Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. 2015.
[3] Where Developers Learn, Share, & Build Careers. stackoverflow.com/.
[4] Hartmann, Björn, et al. "What would other programmers do: suggesting solutions to error messages." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2010.
[5] Guzdial, Mark. "Programming environments for novices." Computer science education research 2004 (2004): 127-154.
[6] Ramadhan, Haider, and Benedict du Boulay. "Programming environments for novices." Cognitive models and intelligent environments for learning programming. Springer, Berlin, Heidelberg, 1993. 125-134.
[7] Lutz, Mark. Programming python. " O'Reilly Media, Inc.", 2001.
[8] Thenault, Sylvain. "Pylint—Code Analysis for Python." (2006).

[9] Ponzanelli, Luca, Alberto Bacchelli, and Michele Lanza. "Seahawk: Stack overflow in the ide." 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013.
[10] Repl.it - The collaborative browser based IDE. https://repl.it
[11] CodeProject - For those who code. https://www.codeproject.com/.
[12] Khaleel, Firas Layth, et al. "Programming Learning Requirements Based on Multi Perspectives." International Journal of Electrical and Computer Engineering 7.3 (2017): 1299.
[13] Butler, Matthew, and Michael Morgan. "Learning challenges faced by novice programming students studying high level and low feedback concepts." Proceedings ascilite Singapore 99-107 (2007).
[14] Brains, Jet. "PyCharm." URL https://www. jetbrains. com/pycharm (2018).
[15] Sullivan, Bryan. "Server-side JavaScript injection." Black Hat USA (2011).