

Enhancing Python Compiler Error Messages

Abhigya Koirala
akoirala@lakeheadu.ca
Lakehead University
Thunder Bay, Ontario, Canada

Archana Devi Ramesh
aramesh3@lakeheadu.ca
Lakehead University
Thunder Bay, Ontario, Canada

Md Meher Hassan Chowdhury
mchowdh8@lakeheadu.ca
Lakehead University
Thunder Bay, Ontario, Canada

ABSTRACT

Every programmer's common struggle is to decode and understand the confusing error messages produced by compilers. Thanks to search engines, programmers immediately rely on programming-related questions and answer websites like Stack Overflow, Quora, CodeProject, and so forth to check for existing error threads for the right solution or end up starting a new thread on the problem they have confronted. To overcome this time-consuming process, this paper proposes a solution that discusses three packages for Jupyter notebook platform viz. Jupyter Python Compiler Error Enhancer (**JUPYCEE**), Jupyter Python Compiler Error Enhancer Documentation (**JUPYCEEDOC**) and Enhanced Jupyter Python Compiler Error Enhancer (**EJUPYCEE**). **JUPYCEE** automatically queries StackOverflow for the threads that discuss the compiler error message, and it returns whatever it has collected as a whole customized package to the programmer within the IDE. **JUPYCEEDOC** also has the same functionality where it automatically queries Python Documentation. **EJUPYCEE** queries both StackOverflow and Python Documentation and returns the combined results to the user. We also propose an empirical study program where the participants will be assigned Python programming tasks. When the programmers tackle any compiler errors, all three **JUPYCEE** variants will return them an enhanced error message, which would help them to solve the errors easily.

KEYWORDS

Compiler, error messages, Stackoverflow, Python Documentation, **JUPYCEE**, **JUPYCEEDOC**, **EJUPYCEE**

1 INTRODUCTION

Compiler error messages are sometimes very hard to understand and resolve [10] [12]. Programmers often find those error messages produced by the Integrated Development Environment (**IDE**) to be very cryptic, which furthermore hinders the overall development and progress [2]. Programmers often refer to these messages produced by the IDE to lack accurate information [6].

For example, consider the image given in Figure 1. It is the default error message produced by the compiler of Jupyter Notebook Integrated Development Environment(**IDE**). The code shows a simple code segment in Python where the programmer is trying to append an element to an existing list and printing it. The programmer encountered an error for which the message produced by the compiler says 'TypeError : Nonetype Object is not iterable'. A programmer who has not experimented much with Python, would find this error to be highly cryptic and immediately rely on Google looking for a suitable solution for this error.

Novice programmers are the ones who mostly face difficulties in the decryption of such cryptic error messages [5]. As they are new

to programming languages, they are particularly affected by such messages since they have little experience in the field [2]. Debugging is one of the first and foremost skills that novice programmers thrive to master. However, such compiler error messages hinder their development. The compiler messages are the primary guidance for error correction, and thus there is a need for enhancing the inadequately produced error messages.

To solve this issue of ineffectual error messages, we propose an empirical study method to enhance the python compiler error messages. Python programming language was chosen because of its fast-growing popularity and its edge over Java programming language, which is stated in the result published by the 2019 Stack Overflow Developer Survey.¹ From among the common question-and-answer forums, we have chosen Stack Overflow for querying the compiler error messages as they have become a popular source of software documentation with tons of questions tagged with compiler-error and a lot more other questions seeking help to understand a compiler error message [10]. We have also chosen the official Python Documentation website in search for python errors.

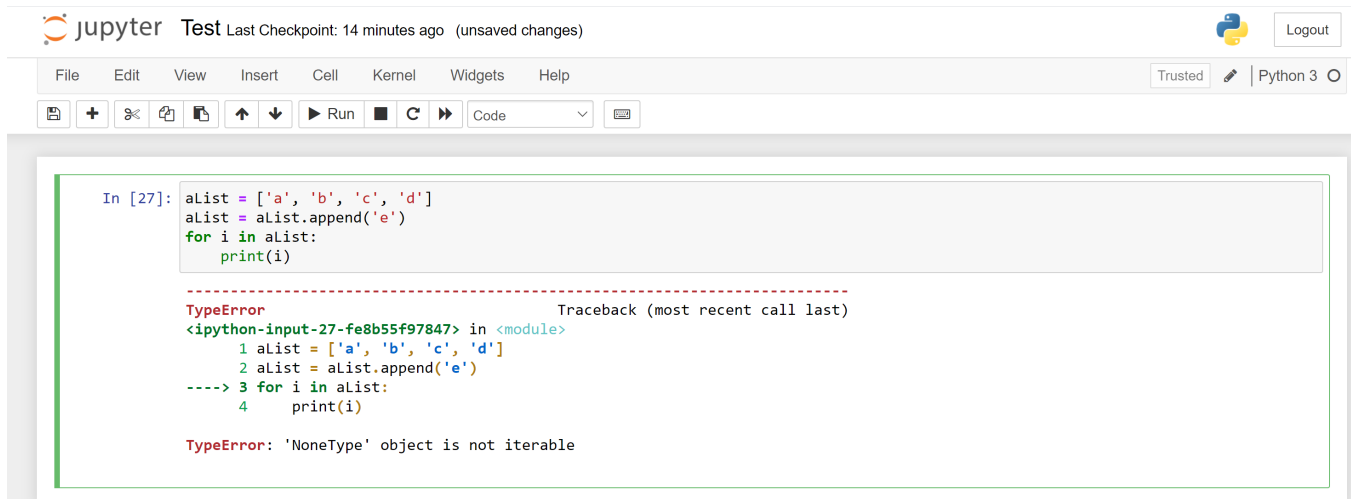
1.1 Problem Definition

We propose to develop Python Compiler Error Enhancer(**PYCEE**) packages for the Jupyter Notebook Platform. The first package called **JUPYCEE** (Jupyter **PYCEE**) which would automatically query the compiler error message on Stack Overflow². The query then returns the answer results, which are then summarised and shown to the programmer, which would help the programmer with an enhanced message to solve the error. The second proposed variant is called **JUPYCEEDOC** (Jupyter **PYCEE** Documentation). **JUPYCEEDOC** would retrieve the data from the official Python documentation for the specific compiler error messages as opposed to **JUPYCEE**, which retrieves and summarises answers from Stack Overflow. Along with **JUPYCEE** and **JUPYCEEDOC**, we also propose the third version of **PYCEE** called Enhanced **JUPYCEE** (**EJUPYCEE**), which would be the combination of both **JUPYCEE** and **JUPYCEEDOC**. Comparison of all three versions of **PYCEE** would then be made by conducting a survey among a group of participants, and an in-depth analysis of the results produced would be done.

The remaining part of the paper is organized as follows. Section 2 discusses about the Related work. Section 3 lists the tools and technologies used for implementation. Section 4 details the proposed methodologies. Section 5 lists the various research questions asked as part of this work. Section 6 discusses about the result analysis part. Section 7 concludes the work with discussion on future scope.

¹<https://insights.stackoverflow.com/survey/2019>

²<https://stackoverflow.com/>



```

In [27]: alist = ['a', 'b', 'c', 'd']
        alist = alist.append('e')
        for i in alist:
            print(i)

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-27-fe8b55f97847> in <module>
      1 alist = ['a', 'b', 'c', 'd']
      2 alist = alist.append('e')
----> 3 for i in alist:
      4     print(i)

TypeError: 'NoneType' object is not iterable

```

Figure 1: Error example

2 RELATED WORK

Thiselton et al. [11] presents a similar proposal for enhancing the python compiler error messages for Sublime Text IDE. They have also used Stack Overflow and Python Documentation website for querying for the errors obtained. Our work was influenced by their work, where we replaced the IDE with Jupyter Notebook and we also added the proposal for the enhanced version. The paper [3] presents a control/intervention study on the effectiveness of enhancing Java compiler error messages. In the results, the intervention group experienced reductions in the number of overall errors. Nienaltowski et al. [7] explore whether the form of compiler error messages can help students who are learning to program and also find out whether additional information in error messages benefits novices. Their results proved that more detailed messages do not necessarily simplify the understanding of errors but that it matters more where information is placed and how it is structured. Luca Ponzanelli et al. [8] developed an Eclipse plugin called Seahawk, formulates queries automatically from the active context in the IDE, and presents a ranked and interactive list of results to users. Using drag & drop options, users can import code samples in discussions and link Stack Overflow discussions with source code persistently as support for teamwork. Luca Ponzanelli et al. [9] proposed an approach that automatically retrieves related discussions from Stack Overflow, evaluates their relevance. If a given confidence threshold is surpassed, their model notifies the developer about the available help. They implemented this approach using an Eclipse plugin called Prompter.

Campbell et al. [4] developed a content assist called NLP2Code mainly to assist with code snippets, including API usage examples, template code, and algorithms. This was integrated into the Eclipse IDE. NLP2Code helps developers in reducing the context switch occurring between a web browser being used searching for code snippets and an IDE. Their implementation was based on natural language tasks queried from the threads of questions on Stack Overflow. Hongyu Zhang et al. [14] developed a similar tool for code

assisting called Bing Developer Assistant (BDA) that recommends sample code queried from public software repositories like GitHub and web sites like Stack Overflow. BDA answers a query for code snippets that implement an API. A large number of code snippets can be stored within the IDE Visual Studio, which the programmers can reuse.

Christoph Treude et al. [13] conducted a survey that experimented to what extent the Stack Overflow code fragments proved to be self-explanatory from the developers' point of view. Their study was a means to understand and address the information needs of a developer related to code fragments. Alquaimi et al. [1] presented a tool called LAMBADDOC, which detects lambda expressions in Java programming language and can automatically generate natural language documentation. Being a new feature in the Java language, developers find it difficult to comprehend lambda expressions. LAMBADDOC was developed as a goal to help the developers read lambda expressions and understand how they are used.

3 TOOLS AND TECHNOLOGIES USED

Since we are working on enhancing python compiler error messages, we have just used Python Programming language for implementation purpose. We have used basic python libraries like pandas, numpy and re. For summarisation we used nltk packages and beautiful soup for web scraping and so on. The platform or the compiler used for building the packages is Jupyter Notebook. Any IDE that supports Jupyter Notebook can be used as the platform.

4 PROPOSED METHODOLOGY

The proposed methodology can be divided into two subsections. First to develop the python packages and second to conduct an evaluation survey. These two methodologies are discussed in the following two sections.

4.1 Python Packages

We propose to develop the three variants of PYCEEE(Python Compiler Error Enhancer): **JUPYCEE**, **JUPYCEEDOC** and **EJUPYCEE**. The methodology for all three versions of PYCEEE is explained below.

- (1) **JUPYCEE**, when encounters a Python compiler error, parse the corresponding error message. It then finds the affected filename and libraries that have been imported. It also looks if any specific error message has been produced by the compiler and then constructs a Stack Overflow query. Depending upon the type of error, a different approach is carried out. Then Stack Overflow query is carried out in such a way it returns threads tagged with python, which contains at least one answer. The thread is sorted by relevance using the Stack Overflow search algorithm³. Only the first page of the result is taken for further processing. It will contain up to ten Stack Overflow threads, and the first accepted answer is selected. In case if no answer has been selected, the answer with the highest score is selected with the criteria of the score is greater than zero.

The selected answer is then customized and summarised text summarising algorithm with python nltk packages. The final summarised answer is shown to the programmer in Jupyter Notebook.

- (2) **JUPYCEEDOC**, the baseline package for Jupyter Notebook IDE, will query the compiler error when encountered, Exceptions page of the Python API⁴. It will then show the corresponding result in the Jupyter Notebook IDE. It will make use of the type of error produced to show the corresponding message from official PYTHON documentation regarding the error message rather than using the description of the error. The JUPYCEEDOC does not make any summarization as opposed to its other version, JUPYCEE, which summarises the answer from the Stack Overflow.
- (3) **EJUPYCEE**, will be the combination of both JUPYCEEDOC and JUPYCEE. It will show both the result from official PYTHON documentation as well as the summarised answer from Stack Overflow. We hope that this enhanced version of JUPYCEE produce a better result than both version. Figure 2 depicts the over all workflow of the three variants of plugins that will be used.

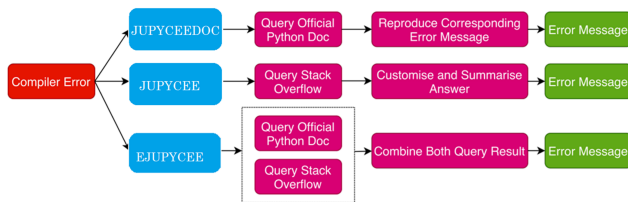


Figure 2: Workflow of different versions of PYCEEE

³<https://meta.stackoverflow.com/questions/355532/how-does-sort-by-relevance-work>

⁴<https://docs.python.org/3/library/exceptions.html>

4.1.1 Text Summarization. Because of the competitive business tendencies many organisations now asks their customers or users for feed backs. These feed backs can be enormous in number from the users at a single day. It becomes quite tedious for a person to sit and analyze the huge number of fed backs. But today's machine learning has went to an extent that it can do the tasks of human. The machine can understand the language of humans using Natural Language Processing. Feedback Summarizer is one of the application that we have used to summarize our feedbacks on our project.

In our work we have used Corpus means a collection of text. It could be data sets of anything containing texts be it poems by a certain poet, bodies of work by a certain author, etc. In this case, we used a data set of pre-determined stop words. We also used Tokenizers, it divides a text into a series of tokens. There are three main tokenizers – word, sentence, and regex tokenizer. We have only used the word and sentence tokenizer.

A python dictionary is used that kept a record of how many times each word appeared in the feedback after removing the stop words. We uses the dictionary over every sentence to know which sentences have the most relevant content in the overall text. Then we assigned score to each sentence depending on the words it contains and the frequency table. Secondly, we created a dictionary to keep the score of each sentence, we went through the dictionary to generate the summary. A simple approach to compare our scores was to find the average score of a sentence. The average itself is a good threshold. After applying the threshold value and stored sentences in order into the summary.

The following figure 3 shows the feedback result before summarization.

```

In: summarize_answer()

lista = lista.sort(lista)

This should be
lista.sort()

The .sort() method is in-place, and returns None. If you want something not in-place, which returns a value, you could use
sorted_list = sorted(lista)

Aside #1: please don't call your lists 'list'. That clobbers the builtin list type.

Aside #2: I'm not sure what this line is meant to do:
print str("value 1a")+str(" ") +str("value 2")+str(" ") +str("value 3a ") +str("value 4")+str("\n")

is it simply
print "value 1a + value 2 = value 3a value 4"

? In other words, I don't know why you're calling str on things which are already str.

Aside #3: sometimes you use print("something") (Python 3 syntax) and sometimes you use print "something" (Python 2). The latter would give you
a SyntaxError in py3, so you must be running 2.1, in which case you probably don't want to get in the habit or you'll wind up printing tuples, with extra
parentheses. I admit that it'll work well enough here, because if there's only one element in the parentheses it's not interpreted as a tuple, but it looks strange
to the pythonic eye...

```

Figure 3: Result before summarization

After applying our feedback summarizer the the result is the following 4, that is better than the actual feedback that carries many words or lines that are of no use while debugging a project.

4.2 Participant Survey Evaluation

The packages were then distributed among different participants with different levels of expertise in PYTHON. Their objective was to program two tasks in python. To ensure that participants did not find the tasks too difficult, they were distributed with four python programming tasks and asked them to choose any two from them. They will be able to work with all three versions of JUPYCEEE

```

lista = list.sort(lista)
This should be

lista.sort()
The .sort() method is in-place, and returns None. If you want something not in-place, which returns a value, you could use

sorted_list = sorted(lista)
Aside #1: please don't call your lists list. Aside #2: I'm not sure what this line is meant to do:

print str("value 1a")+str(" ")+str("value 2")+str(" - ")+str("value 3a ")+str("value 4")+str("
")
is it simply

print "value 1a + value 2 = value 3a value 4"
? Aside #3: sometimes you use print("something") (Python 3 syntax) and sometimes you use print "something" (Python 2).

```

Figure 4: Result after summarization

along with the compiler message produced by the IDE itself. These tasks were chosen from ⁵ and ⁶ and are as follows:

- (1) **Removing duplicates from a list** : Write a function that takes a list (with duplicate elements) and returns a new list that contains all the elements of the first list removing duplicates.
- (2) **Password Generator**: Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your code in a main method.
- (3) **Birthday Dictionary**: For this exercise, we will keep track of when our friend's birthdays are, and be able to find that information based on their name. Create a dictionary (in your file) of names and birthdays. When you run your program it should ask the user to enter a name, and return the birthday of that person back to them.
- (4) **Extracting URL from a website**: Write a Python program to extract all the URLs from the webpage python.org that are nested within tags

We prepared an Instructions Manual which contained guidance on how to run the packages for performing the tasks. We then shared all the packages, test notebook files to have a basic understanding of the functionality of the three packages and the Instructions Manual as a zipped file across social media platforms. To best replicate a real programming setting, participants were explicitly given permission to execute the code (that's why we shared with them the full working source code) and to refer to the Internet while completing the study tasks.

If a participant did not encounter any compiler error while completing a task, they were asked to execute the test programs which contained working examples for all the three JUPYCEE variants. Once they were done with the tasks, we requested them to submit a survey, the link for which was given in that Manual. Through this survey they will address the different pros and cons of JUPYCEE, JUPYCEEDOC, EJUPYCEE and default compiler. They would address the helpfulness, satisfactory levels and programmer preference of all the three versions and default compiler produced message. Basically the survey was to help us analyze the results for our Research questions.

⁵<https://www.practicepython.org/>

⁶<https://www.w3resource.com/python-exercises>

5 RESEARCH QUESTIONS

Not much prior work and focus have been done for the compiler error messages related to python prior to the research done in the paper [10]. Since it is not a widely researched area with less significant findings, we propose to establish the following research questions, some of which are the same as the paper and some are our own proposed research questions.

- **RQ1** *How do programmers perceive Python compiler error messages?*

This research question will address the different participant's views on the compiler error messages produced by the IDE itself.

- **RQ2** *How do programmers perceive working with JUPYCEE and JUPYCEEDOC?*

This research question will address the different participant's views on the results produced by the two versions of PYCEE, which are JUPYCEE and JUPYCEEDOC. We will investigate the impact of the results produced by these two plugins and compare their advantages over the compiler error messages produced by the IDE itself in terms of helpfulness, satisfaction level and programmer preferences.

- **RQ3** *What are programmers opinion regarding EJUPYCEE? Was the enhancement made on PYCEE really helpful?*

This research question will address if the EJUPYCEE did really make a positive impact and enhancement over JUPYCEE and JUPYCEEDOC. We will analyze the results produced by this package and compare it over the other two variants, in terms of helpfulness, satisfaction level and programmer preferences.

- **RQ4** *What are programmers opinion regarding different pros and cons of the three different versions of PYCEE?*

This research question will make a comparison between the three versions of PYCEE with the help of participants' opinions and suggestions on improvements.

Table 1: Background of participants

Sl. No	Participant	Count
1	Student	17
2	Programmer	7
3	Teacher	4
4	Software Developer	1
5	Quality Analyst	1
Total		30

6 RESULTS AND DISCUSSION

In this section, we describe the findings for each of our four Research Questions. First we will start with a discussion on the tasks assigned to the participants.

6.1 Participants and Assignment of Tasks

To answer our research questions, we conducted a user study for which thirty people participated. The participants were from different background. Out of which seventeen were students, seven

professional programmers and people from few other categories as well as shown in the Table 1.

The thirty participants were given the freedom to choose any two out of the four tasks as described in Section 4.2. The assignment of tasks to various participants are shown in Table 2. It can be seen that, majority of them opted for the two tasks viz. Removing Duplicates from list and Generating Passwords. Even though all tasks were of equal difficulty level, maybe the participants were not comfortable with using the data structure 'Dictionary' for doing the third task and may not be familiar working with 'Beautiful Soup' for extracting URLs. Participant Number 17, 19, 21 and 22 opted for only one task.

Table 2: Background of participants

Participant	Question 1	Question 2
1	Removing Duplicates	Password Generator
2	Removing Duplicates	Password Generator
3	Removing Duplicates	Password Generator
4	Removing Duplicates	Extract URL
5	Removing Duplicates	Password Generator
6	Removing Duplicates	Birthday Dictionary
7	Removing Duplicates	Password Generator
8	Removing Duplicates	Password Generator
9	Removing Duplicates	Password Generator
10	Removing Duplicates	Password Generator
11	Removing Duplicates	Password Generator
12	Removing Duplicates	Birthday Dictionary
13	Removing Duplicates	Birthday Dictionary
14	Removing Duplicates	Password Generator
15	Password Generator	Birthday Dictionary
16	Password Generator	Extract URL
17	Password Generator	
18	Removing Duplicates	Birthday Dictionary
19	Birthday Dictionary	
20	Removing Duplicates	Birthday Dictionary
21	Password Generator	
22	Removing Duplicates	
23	Removing Duplicates	Password Generator
24	Removing Duplicates	Password Generator
25	Removing Duplicates	Password Generator
26	Removing Duplicates	Password Generator
27	Removing Duplicates	Extract URL
28	Removing Duplicates	Password Generator
29	Removing Duplicates	Password Generator
30	Removing Duplicates	Birthday Dictionary

6.2 RQ1: How do programmers perceive Python compiler error messages?

In this section, we will discuss the various questions asked as part of the Research Question 1, which is their perception regarding the error message provided by a compiler. In order to analyze this data, we collected two information.

6.2.1 How many years of experience do the participants have in Python Programming? The first question asked as part of RQ1 was how many years of experience the participants had in working with Python Programming language. The data collected as part of this question is depicted in Figure 5.

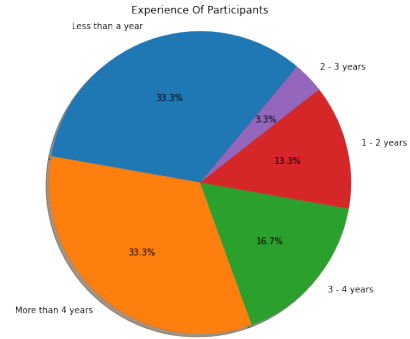


Figure 5: Python programming experience

We can see that 33.3% of people had less than one year of experience in Python programming. People with one to three years constituted 16% in total. Around 50% of people had more than three years of experience. Since majority people had some background, we anticipate that they would not have had much difficulty in performing the task.

6.2.2 How do you rate the helpfulness of the error message provided by default python compiler? The second question asked as part of RQ1 was regarding the helpfulness of the default error message provided by the compiler. The data collected as part of this question is depicted in Figure 6. This figure shows, 50% of people gave less than or equal to 3 ratings and remaining 50% gave more than 4. It is evident that people are generally not satisfied with the default error message provided by the compiler.

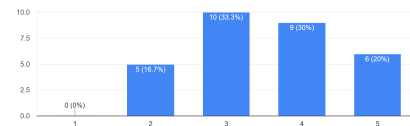


Figure 6: Helpfulness of default python compiler

6.3 RQ2: How do programmers perceive working with JUPYCEE and JUPYCEEDOC?

In this section, we will discuss about the different questions asked as part of the Research Question 2 where in the participant's reaction regarding the two proposed PYCEE variants were analyzed. To address this question, we collected the following data from the participants.

6.3.1 How do you rate the helpfulness of JUPYCEE and JUPYCEEDOC? The first question asked as part of RQ2 was how helpful the participants found with our two variants of PYCEE, viz. JUPYCEE

and JUPYCEEDOC. Our main intention was to determine if the participants found the automatic query response from StackOverflow and Python Documentation helped them in performing the two tasks. The data collected for this is displayed in Figure 7.

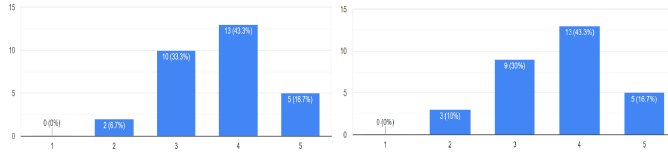


Figure 7: Helpfulness of (a)JUPYCEE and (b)JUPYCEEDOC

From Figure 7, we can conclude that for both JUPYCEE and JUPYCEEDOC, the participants had more or less the same level of perception regarding the helpfulness of both the variants. Around 60% of people gave more than four ratings and around 40% gave less than or equal to 3 ratings. One more important observation from this data is that out of 60% of people who gave more than 4 ratings, only 16% contributed to 5 star rating. Which shows that they still did not find these two variants very much helpful. However, compared to the equivalent response given for the default error message, we can see an increase of 10% in both, which shows they found the automatic response from both JUPYCEE as well as JUPYCEEDOC more helpful in a comparative basis.

6.3.2 How do you rate your satisfaction level with JUPYCEE and JUPYCEEDOC? The second question asked as part of RQ2 was regarding the satisfaction level of working with JUPYCEE and JUPYCEEDOC, apart from knowing how helpful they found these two variants. Their level of satisfaction for both the variants are shown in Figure 8.

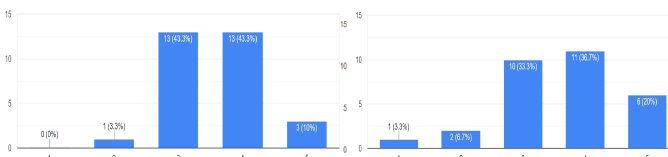


Figure 8: Satisfaction level of (a)JUPYCEE and (b)JUPYCEEDOC

From Figure 8, it can be seen that that around 53% were satisfied with more than 4 stars for JUPYCEE and remaining with less than or equal to three. Similarly, for JUPYCEEDOC, 57% people gave 4 stars and more rating for their level of satisfaction and remaining with less than or equal to three. From these graphs also, we can conclude that people who gave 5 star ratings were less which shows they were not fully satisfied with the packages.

6.4 RQ3: What are programmers opinion regarding EJUPYCEE? Was the enhancement made on PYCEE really helpful?

In this section, we will discuss about the different questions asked as part of the Research Question 3, where the participant's opinion

regarding the enhanced PYCEE variant, i.e. EJUPYCEE were analyzed. We wanted to analyze whether the enhancement done was truly helpful or not. To collect data for this question, we collected the following information from the participants.

6.4.1 How do you rate the helpfulness of EJUPYCEE and how satisfied are you with the enhanced version? The first question asked as part of RQ3 was regarding the helpfulness and satisfaction level of EJUPYCEE. Figure 9 depicts how the participants found EJUPYCEE helpful and their level of satisfaction.

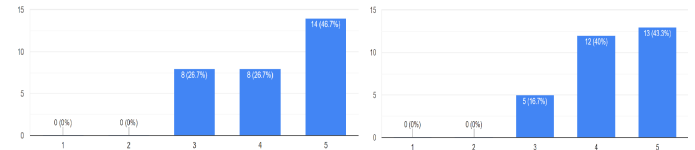


Figure 9: EJUPYCEE (a)Helpfulness and (b)Satisfaction Level

Comparing Figure 9 with Figure 7 (a) and (b), it can be seen that people who gave rating 5 for the older versions were less with only 16.7%. Whereas for EJUPYCEE, we can see a huge increase of 46.7%. Our main objective was to analyze if the enhancement made on EJUPYCEE was truly helpful and how satisfied the participants are. The data obtained as part of this question clearly proves that the enhancement was definitely an improvement. This can be because of the fact that the participants get the combined queried results from both StackOverflow as well as Python documentation using EJUPYCEE. Even though we got an improved result compared to JUPYCEE and JUPYCEEDOC, we lacked better percentage of people of more than 70% to give a 5 star rating. The reasons for this is analyzed as part of RQ4 which is explained in the Section 6.5.

6.4.2 Which of the three PYCEE variants would you prefer for future reference? The second question asked as part of RQ3 was which of the three versions of PYCEE that we have developed would the participants consider for referring in future. The response we got for this data is shown in Figure 10.

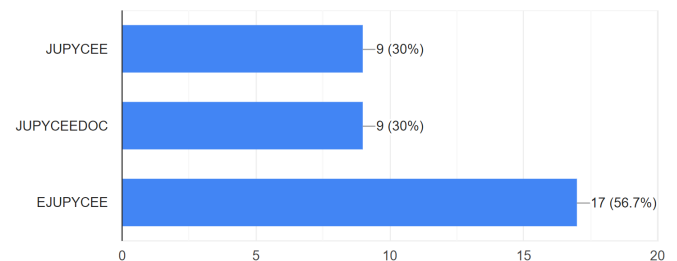


Figure 10: Preferred PYCEE variant

Figure 10 shows that around 30% of participants voted for JUPYCEE and JUPYCEEDOC equally. Whereas for EJUPYCEE, more participants have given votes with 56.7% rating. This proves the results obtained regarding the helpfulness and satisfaction level where EJUPYCEE had better ratings compared to other three.

6.4.3 Resources used for debugging the code. The third question analyzed as part of RQ3 was to get an idea of what all resources the participants had used as reference for resolving the errors obtained as part of the tasks. Apart from using the versions, the participants were given the liberty to refer Google as well. We did not expect that our packages would help them 100 percent. Figure 11 shows the various resources the participants used for debugging the code.

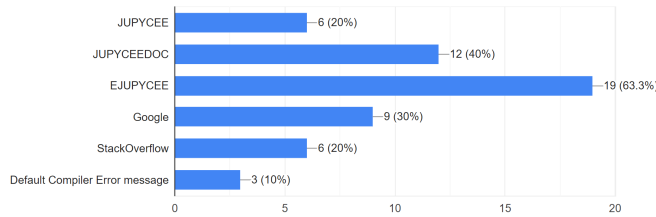


Figure 11: Resources used for debugging the code in completing the tasks

From the results obtained, the majority voting was for EJUPYCEE and JUPYCEEDOC with 63% and 40% respectively. This clearly shows most of the errors the users were able to solve with the help of these two packages. They also depended on Google and Stackoverflow, but their reference was less with 20-30%. The default compiler error message was found to be least helpful with just 10% voting. Since EJUPYCEE obtained the highest rating with 63%, it can be concluded that this is definitely an improvement over the default compiler error message, which was our main proposed problem.

6.5 RQ4: What are programmers opinion regarding different pros and cons of the three different versions of PYCEE?

This section describes the pros and cons of the three variants of pycee according to the individuals who took the survey. We also surveyed the reasons behind choosing one of our variant over other two.

6.5.1 Reasons behind choosing EJUPYCEE over two variants(JUPYCEE & JUPYCEEDOC): . From the feedback received by the survey conducted among 30 participants, they have reported that for multiple error that they encountered for the JUPYCEE and JUPYCEEDOC variants they had to use python documentation and other online resources as well to complete the tasks. The participants when they had to solve they same error using EJUPYCEE faced no hassle of going through other online resources and python documentation to solve the error. It provided result from both Python official documentation and stack-overflow. This helped to save participants time, thus being one of the reason for being popular among participants.

6.5.2 A Discussion on the pros: After analysing the feedback from the individuals we found that EJUPYCEE was the most helpful among the three variants. The participants stated that it helped them save their time for searching the online resources for the error. The participants also stated that the hint provided were useful enough for them to debug their code. The answer displayed from

python official documentation was found to be very specific and helpful. This helped the participants solve their task easily and fast. Many reported it being even helpful for beginner programmer to learn how to code.

6.5.3 A Discussion on the cons: After analysing the feedback from the individual we found out that our variants had multiple flaws as well. The solutions provided by the variants was showing multiple unorganised data that were confusing to understand. Sometimes there were no solutions provided. other than the compilers default error messages. Multiple times the users reported the solution being irrelevant to the encountered error. Some participants also reported the need of multiple answers to be provided. The variants need online access and cannot work without it. The participants also reported lack of answers from other online resources.

7 CONCLUSION AND FUTURE WORK

Programmers come across different compilers error messages that are sometimes uninformative. They seek help from different sources from the internet to solve those exceptions. Keeping these issues in focus, we have created three libraries integrated on Jupyter Notebook to provide enhanced compiler error messages for python. The three variants are JUPYCEE, JUPYCEEDOC and EJUPYCEE. The three variants have different functionalities. JUPYCEE performs queries automatically on stack overflow website regarding the exception and compiler error message thrown. The answer with highest number of vote and that have been accepted is chosen and summarised and then shown to the user. JUPYCEEDOC queries the python official documentation regarding the compiler error and provides the solution within console. Some custom tailored hints are also provided to the user for some specific error that has been found. The third variant EJUPYCEE provides solution by combining the result of both JUPYCEE and JUPYCEEDOC. To find the usefulness of the libraries, we conducted a survey among 30 individuals of different background. The individuals were assigned with 4 different python task out of which they had to complete any 2 of them. They were asked to import the 3 different libraries one at a time to complete the task and then complete the survey. Analysing the response and feedback of the 30 participants we acknowledged that EJUPYCEE was the most convenient among the 3 variants of libraries. This was because it provided custom tailored hint messages, summarised stack-overflow answers along with help from python official documentation. Also from the analysis from the result of survey we can conclude that EJUPYCEE was really helpful, satisfying and time saving. The graphs obtained also shows that online sources in terms of enhancing the compilers default error messages can be used successfully. Thus for the future work we plan to add not only stack-overflow answer but answer from other online resources.

We received multiple feedback from the participants. After analysing those feedback we have acknowledged some functionalities that we plan to integrate in the future. We plan to take into consideration more than one solution from stack-overflow. We would provide multiple answers from other online resources. Multiple participants reported less accurate summarisation, thus we plan to implement a better summariser. Participants had problem viewing the results displayed to them. The visualisation and display of results would

be improved in future version. Further more we plan to use the stack-overflow dataset by implementing a lucen search on it so that our library can be implemented offline.

REFERENCES

- [1] Anwar Alqaimi, Patanamon Thongtanunam, and Christoph Treude. 2019. Automatically generating documentation for lambda expressions in java. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 310–320.
- [2] Brett A. Becker. 2016. An Effective Approach to Enhancing Compiler Error Messages. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) (*SIGCSE '16*). Association for Computing Machinery, New York, NY, USA, 126–131. <https://doi.org/10.1145/2839509.2844584>
- [3] Brett A Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, and Catherine Mooney. 2016. Effective compiler error message enhancement for novice programming students. *Computer Science Education* 26, 2-3 (2016), 148–175.
- [4] Brock Angus Campbell and Christoph Treude. 2017. NLP2Code: Code snippet content assist via natural language tasks. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 628–632.
- [5] Paul Denny, Andrew Luxton-Reilly, and Dave Carpenter. 2014. Enhancing Syntax Error Messages Appears Ineffectual. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education* (Uppsala, Sweden) (*ITiCSE '14*). Association for Computing Machinery, New York, NY, USA, 273–278. <https://doi.org/10.1145/2591708.2591748>
- [6] Guillaume Marceau, Kathi Fisler, and Shriram Krishnamurthi. 2011. Mind Your Language: On Novices' Interactions with Error Messages. In *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (Portland, Oregon, USA) (*Onward! 2011*). Association for Computing Machinery, New York, NY, USA, 3–18. <https://doi.org/10.1145/2048237.2048241>
- [7] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. 2008. Compiler error messages: What can help novices?. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 168–172.
- [8] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack overflow in the ide. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 1295–1298.
- [9] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining stackoverflow to turn the ide into a self-confident programming prompter. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 102–111.
- [10] E. Thiselton and C. Treude. 2019. Enhancing Python Compiler Error Messages via Stack. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12. <https://doi.org/10.1109/ESEM.2019.8870155>
- [11] Emillie Thiselton and Christoph Treude. 2019. Enhancing python compiler error messages via stack. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–12.
- [12] V. Javier Traver. 2010. On Compiler Error Messages: What They <i>Say</i> and What They <i>Mean</i>. *Adv. in Hum.-Comp. Int.* 2010, Article 3 (Jan. 2010), 26 pages. <https://doi.org/10.1155/2010/602570>
- [13] Christoph Treude and Martin P Robillard. 2017. Understanding stack overflow code fragments. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 509–513.
- [14] Hongyu Zhang, Anuj Jain, Gaurav Khandelwal, Chandrashekhar Kaushik, Scott Ge, and Wenxiang Hu. 2016. Bing developer assistant: improving developer productivity by recommending sample code. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 956–961.