UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2023

Title:

Enhancing The Interactivity and Visualization Features of The Online Python Tutor Web Application

Author:

Siviwe Qolohle

Project Abbreviation:

PyCodEx

Supervisor(s):

Mr Gary Stewart

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	15
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	15
System Development and Implementation	*0*	*20*	15
Results, Findings and Conclusions	*10*	*20*	15
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	**60**

Enhancing a The Interactivity and Visualization Features of The Online Python Tutor Web Application

Siviwe Qolohle
Department Of Computer Science
University of Cape Town
Cape Town, Western Cape
qlhsiv001@myuct.ac.za

ABSTRACT

Grasping the basics of programming is challenging for first years who have no prior experience to the skill. Tools which allow users to improve their programming skills do exist, these tools are however not very beginner friendly. They cannot be seen as being beginner friendly because they are mainly practicing tools for individuals who are already skilled at programming, rather than tools that focus on assisting users in grasping the basics of programming. These tools are often not comprehensive enough and it may take a while for the user to fully grasp how they operate. This paper discusses the enhancement of the Online Python Tutor web application, to make it more beginner friendly. Online Python Tutor is web application that allows students to visualize any block of code, to improve their understanding of how each line of code affects the overall code and output. This paper discusses the visual and interactivity enhancements that have been made to make Online Python Tutor more beginner friendly. The effectiveness of the features was tested with first-year Computer Science students. From the results, the usage of colour to link a line of code in the editor, with content in the visualization tool, and explanations in the interactivity tool, proved to be helpful to students. The interactivity feature proved to be more helpful than the visualization features. This is partly because users saw the feature as one that could assist in debugging.

1 INTRODUCTION AND MOTIVATION

Introductory computer science courses tend to have high failure rates at tertiary institutions [1]. Many students struggle with programming upon their arrival at tertiary institutions as many of them arrive with no prior exposure to the skill. Initially the subject can come across as being too abstract for beginners. Grasping the programming terminologies can also be difficult for beginners. With more careers incorporating programming into their day-to-day activities, universities are beginning to incorporate the skill into the curricula of many different degrees across faculties [1, 15]. This increases the urgency of finding a way to smoothen students' experiences when learning how to program. Struggling to initially grasp the basics of the skill can have unexpected effects on a student's experience at university. Struggling with the course can result in forced extension of students' degrees. Some students arrive at university without the intention of enrolling into an extended version of a subject but struggling to keep up with the pace of a single semester computer science course automatically forces them to extend their degree by a year. The high failure rates reflect students' inability to grasp the fundamental topics of programming.

1.1 Visualisation

Programming greatly challenges an individual's problem solving and abstract thinking skills [6]. Visualizations assist in making programming less abstract and allows one to form mental images of complex systems and how their components interact [7, 8, 9]. Visualisation allows students to see how variables are affected by a sequence of instructions [2, 14]. Incorporating colour into code visualisation tools is helpful. Past studies have found that the incorporation of colour has a positive impact on a learner's learning experience. The studies found that incorporating colour into learning material assists in grasping a learners' attention and that colours are processed automatically [19].

When being taught how to program, static tools are often used, these tools include images, written explanations in lecture slides and textbooks. Static tools make it difficult for beginners to grasp the basics of programming, highlighting the urgency for dynamic teaching tools [16]. The need for dynamic teaching tools is not only urgent in introductory programming courses, but it is needed throughout one's programming degree. When learning different programming techniques in CSC3003S such a 'brute force' and 'dynamic programming' dynamic teaching tools are crucial for students to fully grasp the steps that need to be taken for each programming technique. Topics such as searching and sorting algorithms need to be fully understood by all Computer Science students [16]. For students to understand these topics, dynamic teaching methods need to be utilized.

Past studies have found that the incorporation of colour has a positive impact on a learner's learning experience. The studies found that incorporating colour into learning material assists in grasping a learners' attention. Studies have also found that colours are processed automatically [19].

1.2 Interactivity

When learning how to program, interactivity plays a crucial role. There is a vast difference between attending lectures where one is shown pre-constructed solutions to problems, and physically coding one's own solution to a problem. A first-year student may be very consistent when it comes to attending their computer science courses and engaging with their lecturer but may struggle greatly when it comes to the coding assignment. This highlights the importance of interactivity when being taught how to code. Interactive code teaching applications have proven to be very successful when it comes to teaching beginners [5]. In experiments comparing the effectiveness of focusing on visualisation when it comes to programming, with the effectiveness of focusing on interactivity, interactivity has been proven to be more successful [2]. Students have been found to learn and understand programming better when completing practical assessments [4]. This highlights the importance of increasing the time students spend physically coding, rather learning the theory. When it comes to being taught about programming, some concepts only make sense when one is actively programming, as it automatically forces one to self-teach some programming techniques and terminologies.

It has been found that learners who passively make use of visualizations to learn are outperformed by learners actively engaging with the visualizations [10]. One of the ways in which students have been engaging with the visualizations is by making predictions of what changes will occur in the visualization after a change in the code [11]. A common practice that is viewed as being effective when teaching programming is execution history. This is because students often forget what happened in the previous lines of their code [10]. Another common practice which is viewed as being effective when teaching students how to code is supporting flexible execution control. This allows the user to proceed forwards and backwards when analyzing their code [10]. allowing users to enter their own input into an editor is also viewed as being effective as it results in students engaging more with the visualizations [10]. Having questions testing students' understanding is also recommended as it allows students to reflect on the visualization that they have such observed to check whether they truly understand it [10]. It is also important to include explanations within the visualizations. This assists students in better understanding the visualizations [10].
Engagement Taxonomy was introduced by Naps et al [10], which consists of six different levels of interactivity respective to a visualization [6]. The six levels are 'No Viewing', 'Viewing', 'Responding', 'Changing', 'Constructing' and 'presenting' [10]. At each step from the No Viewing level to the Presenting level, the interaction with the visualization increases [10].

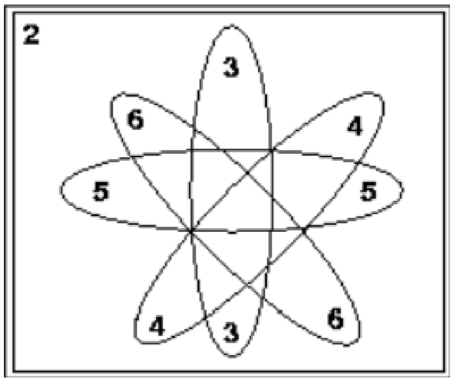
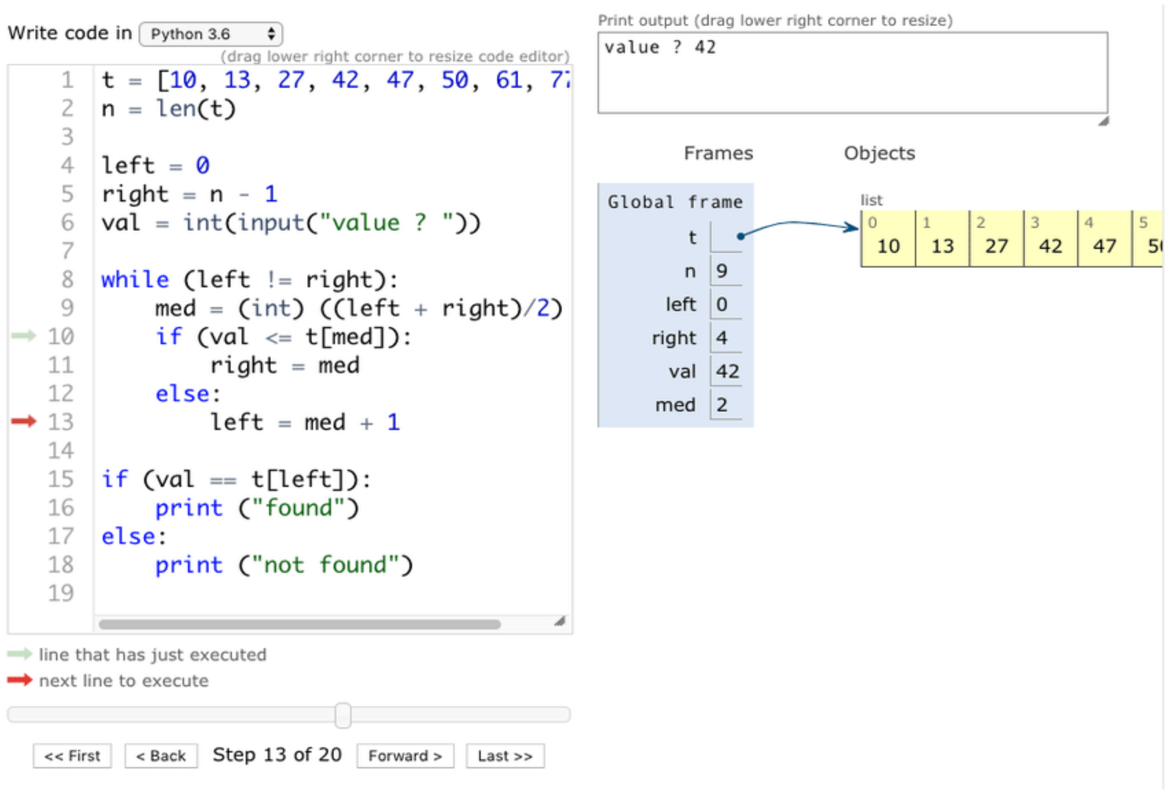


Figure 2: Possible Overlaps in the Engagement Taxonomy. Basic regions are 2 (Viewing), 3 (Responding), 4 (Changing), 5 (Constructing), 6 (Presenting)

Figure 1: Engagement Taxonomy [10]

When it comes to the Engagement Taxonomy, each level in the taxonomy (after 'No viewing') does not necessarily exist on its own [10]. There are multiple combinations of how the levels can interact with each other to improve the learner's understanding. For example, a feature can be designed where the Changing level is combined with Responding level [10]. The Viewing level is present in each of the combinations [10]. The third level is Responding. This involves answering questions about the visualization [10]. These questions including Prediction (how the visualization gets affected by a line of code), coding (matching the code to the correct visualization), efficiency analysis of the code, and finally, debugging (checking whether errors are present in the given code) [10]. The responding level requires the user to constantly look back and reanalyze the code, which assists in improving their understanding [10]. The next level is Changing. Changing includes allowing the user to edit the code and provide their own inputs [10]. The next level is Constructing. This level involves constructing your own visualization [10]. The final level is Presenting. This level consists of the learner presenting a visualization to a group of people for discussion and feedback [10]. This could assist in deepening one's understanding of the visualization by getting an idea of other people's interpretation of the visualization. The visualization doesn't necessarily have to belong to the student [10].

1.3 Online Python Tutor


Write code in Python 3.6
(drag lower right corner to resize code editor)
Print output (drag lower right corner to resize)
Frames
Objects
line that has just executed
next line to execute
<< First
< Back
Step 13 of 20
Forward >
Last >>

```
t = [10, 13, 27, 42, 47, 50, 61, 7
n = len(t)

left = 0
right = n - 1
val = int(input("value ? "))

while (left != right):
    med = (int) ((left + right)/2)
    if (val <= t[med]):
        right = med
    else:
        left = med + 1

if (val == t[left]):
    print ("found")
else:
    print ("not found")
```

```
value ? 42
```

Global frame	
t	
n	9
left	0
right	4
val	42
med	2

list

0	1	2	3	4	5
10	13	27	42	47	5

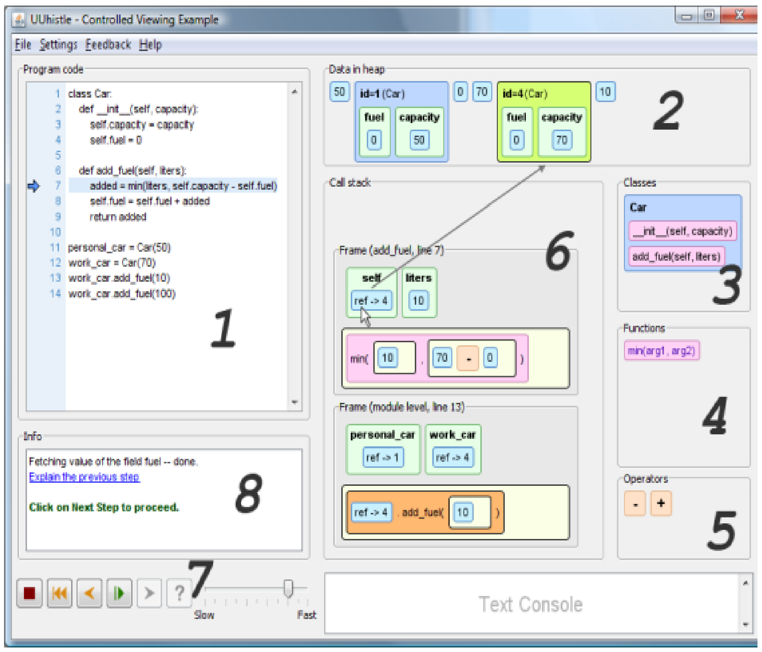
Figure 2: Online Python Tutor

This project builds upon an existing web application known as Online Python Tutor (OPT). OPT (see figure 2) is a platform that allows individuals to code in an editor like other well-known IDEs however OPT allows students to visualize every step of the program [12]. OPT is one of two active code visualisation tools which allow users code in python (the other being UUhistle). However, from the two, OPT is the only web-based application that does not need to be downloaded prior to use [12]. OPT is also easily accessible, one must just enter a URL. OPT has been used in introductory Computer Science courses at multiple universities [12]. These universities include MIT, UC Berkley, University of Washington, and the University of Waterloo [12]. In 2013, it was recorded that over 30,000 individuals used OPT per month. Embedding the web application into other websites is also quite simple. The application was created to move visualisations of code from hand-drawn visualisations on whiteboards to automatically generated digital visualisations. The backend is mainly coded with Python and the frontend is coded using HTML, JavaScript, and CSS. D3 and jsPlumb are the two main JavaScript libraries that are used. jsPlumb is used for the arrows and D3 is used for mapping the trace elements to their corresponding HTML elements. The main ways in which the application is used is firstly as an aid when teaching in a classroom setting; secondly, it is embedded in digital textbooks and thirdly is it used by students for studying or practising [12].

OPT has been recognized to help with topics such as recursion, return values in functions, understanding the flow of a block of code, knowing the lifespan of local variables as well as understanding the referencing of variables. Some students have viewed the diagrams as being very similar to the diagrams they physically draw to understand the structure of their code [12]. The error messages are seen as being easy for beginners to understand compared to the standard python error messages. The web application is easy to augment and has been augmented multiple times. The application has been merged into digital textbooks making it easy for the reader to put the content they are learning into practice. Some of the digital textbooks are structured in such a way that the user can interact with the code that is displayed in the textbook. A textbook that has made use of this feature is *How to Think Like a Computer Scientist: Interactive Edition*. The textbook is widely used, highlighting the importance of interactivity and additional practicing tools in introductory programming courses [12].

1.4 Related Work

UUhistle is a program written in Java, that has both a predominantly visual mode and interactive mode (see figure 3) [13]. The program can be run on its own or on the web. The predominantly visual mode is referred to as Controlled Viewing and it allows the user to observe as the program executes showing



```
class Car:
    def __init__(self, capacity):
        self.capacity = capacity
        self.fuel = 0

    def add_fuel(self, liters):
        added = min(liters, self.capacity - self.fuel)
        self.fuel = self.fuel + added
        return added

personal_car = Car(50)
work_car = Car(70)
work_car.add_fuel(10)
work_car.add_fuel(100)
```

Fetching value of the field fuel -- done.
Explain the previous step

Click on Next Step to proceed.

the different steps the program follows. The code is displayed on the left (with the current line highlighted). On the right side are the components which make up the program, these components include the classes, variables, functions, and operators. As each line is executed, the changes to variables are visualized as well as the steps taken when a method is called by a variable. The changes made by each transition to the next line are displayed on the right. In the Controlled Viewing state, the user can also make use of buttons such as Stop, Rewind, Undo and Next Step as the lines execute.

The second mode of UUhistle is a Visual Program Simulation (VPS) (see figure 4) [13]. In this mode, the student takes on the role of the computer. The objective of this mode is to give the student a deeper understanding of what the computer does when a program is run. In other words, in the VPS mode, the student manually does the changes observed on the right-hand side of the Controlled Viewing (figure 1) mode in each line of code. When a method is created, the user is expected to store it in memory; when the method is called by a variable, it is the user's responsibility to drag the function and enter the required parameter (as with the Controlled Viewing mode, the stored method is displayed on the right side of the screen).

2 SYSTEM DESIGN

The processes that were followed throughout the project to make the enhancements to Online Python Tutor. These processes were both the Waterfall Model as well as the Agile development Model.

2.1 Requirements

The aim of the project is to make Online Python Tutor more beginner friendly by increasing interactivity and improving the visualization features of the application. When it comes to visualization, previous papers have highlighted programming being seen as being too abstract, and the goal of the visualization feature is to reduce this. The existing application does consist of a detailed visualization system, but beginners may struggle to grasp the basics of the system at first. To make change**s** in the visualization system more noticeable, highlighting has been introduced where the variable being referred to in a specific line of code is highlighted. in both the editor and the visualization system. One of papers discussed above mention the importance of having explanations when teaching programming. As a result of this, explanations are given to the user when python keywords are typed into the editor (see Appendix A.3).

For the interactivity section of the project a variable testing system has been introduced. A paper discussed above highlighted the importance of having questions when introducing an Interactivity feature. The questions encourage the user to reflect on the code and determine whether they fully understand each of steps of the code. The way in which the feature works is that the user enters a block code, and the user is tested on whether they understand how a variable is affected in each line. As the user steps through the code, they are asked to predict which variable will be changed or introduced as well as what the new value of the variable is. There is an arrow which points to the line that the user must.

See Appendix A.2 for specific requirements.

3 IMPLEMENTATIONS

The way in which the frontend interacts with the backend is that the code entered in the editor is converted into a string and is sent to the backend with the use of a Get Request. In the backend, an execution trace is produced and is sent to the frontend in a JSON format. To implement the interactivity and visualisation features, the frontend was mainly edited (see Appendix B1). To make the changes, existing JavaScript, HTML and CSS codes were edited. The main JavaScript program that is used for Online Python Tutor is pytutor.js and the main CSS program used is pytutor.css. The HTML file used for the live version of Online Python Tutor is live.html. The main program to which changes were made is the opt-live.js program. This is the JavaScript file that is used to control the frontend of the live version of Online Python Tutor. The way in which this program was edited for the Interactivity feature was

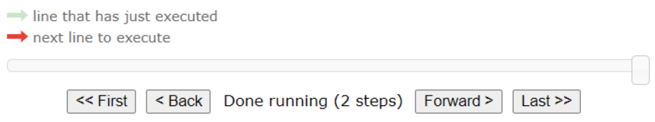


Figure 4: Online Python Tutor Navigation Buttons

through synchronizing the changing variable data with the 'First', 'Forward', 'Back' and 'Last' steps (See figure 4). The way in which the changes at each line of code of are determined is by writing the code typed into the editor, into a JSON file and extracting the global variables called in each line as well as the value of the variable (see Appendix B.2). This data gets written into another JSON file which serves as input into the opt-live.js file. The way in which the input is used in the opt-live.js file as is that the index in the dictionary containing the global variables and their values is monitored. The index in the dictionary corresponds with the current line in the editor. The index corresponds with each of the step buttons stated above. This means that if the user presses 'Forward' the index is incremented by one in the dictionary and if 'First' is pressed the index value becomes zero. To ensure that the predicted answer for line of code is not displayed at the same time at which the user is expected to make the prediction, the predicted solution value is always one index ahead of the index of the current line being executed. Determining the prediction line is made simple by the fact that the existing Online Python Tutor editor points to the line that will be executed next with a red arrow. When the user presses the forward button the solution for the previous line is provided. Although the changes can be seen in the visualisation tool, a comprehensive solution is provided in the interactivity feature (see Appendix A.1 for Interactivity feature with variable highlighting).

The visualization and interactivity feature have a similar foundation. The visualisation feature also needs to be synchronized with the buttons discussed above as well as with the line that is currently being executed in the editor. As the user steps through the code, the variable that has a value that is currently being changed is highlighted in the visualization tool. This is done to assist the student in navigating through the visualisation tool with ease. The variable that gets highlighted depends on the variable in dictionary being pointed to when the given button (stated above) is pressed. The only backend programs that these changes make use of are the bottle_server.py program (see Appendix A.1 for Interactivity feature with variable highlighting).

4 TESTING AND EVALUATION

The root question of the Testing and Evaluation sections is: "Do the visual and interactivity enhancements of Online Python Tutor assist in making the web application beginner friendly?". There were three components to the testing process, namely the Overall Application Testing, Unit Testing and User Testing. When testing a web application, one must mainly focus 'correctness', 'security' and 'quality' [17]. Seeing that Online Python Tutor is open source, testing the security of the web application is not as crucial. When testing the correctness and quality, the testing methods discussed below were used. The testing process followed an agile development model. The Agile Development model ensured that project was on the right path towards meeting the requirements. Integration testing as well as well Unit testing was applied. Frequent meetings with the supervisor also fell into the category of Agile Development. Due to limited time, testing with participants was only completed once.

4.1.1. Overall Application Testing Integration Testing was completed. This tested whether all additional features did not interfere affect the functioning of the original application. When taking the user step by step through each line of code, the current line in the editor needs to correspond with the change in the visualisation tool. This also implies that the highlighted line number in the editor gutter must correspond with the highlighted variable in the code visualisation tool (if the value of a variable has changed). The steps in the in the editor, as well as the corresponding changes in the visualisation tool also need to merge with the Interactivity tool. This is done is by testing whether the highlighted line in the editor corresponds with the prediction solution in the Interactivity tool.

4.1.2 Unit Testing. This is the independent testing of each component of the application [18]. When doing this, the application needs to be broken up into as many separate components as possible. When performing unit testing on the editor, it was tested whether highlighting one variable, results in all of occurrences of variable getting highlighted. It was tested that when stepping through the code, the gutter was highlighted in the position of that line. The following unit test was performed on the keywords within the editor. It was determined whether when typing a keyword (such as 'for' or 'print'), an explanation of word is displayed. Once a new component was added to web application, the entire application was tested to determine whether the overall application continues to run smoothly with the addition of the new component.

4.1 User Testing

This testing process included the participants who we recruited. The testing was conducted in two phases. In the first phase users were initially observed as the navigated throughout the application, thereafter they were asked to complete certain tasks. The second phase consisted of users completing a survey which allowed us to determine the effectiveness and understandability of the enhancements.

4.1.1 Participants. The participants for the project were CSC1010H and CSC1011H students. This course is an introductory programming course that is usually completed by first-year students. Seeing that we are currently in the second semester, participants were familiar with the basics of programming. There was total of 10 participants.

4.1.2 Testing Procedure. Students were recruited at a CSC1010H lecture. They were informed of the purpose and background of the project. They were also informed of the incentive to participate, which was a R250 Takealot voucher. The students who signed chose specific dates and times. the testing process ran for a week. Before the testing process the students were given a Student Affairs approved consent form to complete. The first step of the testing process was observing the students as they navigated through the application. at first, we ask the participants to interact with application without giving them any information. Many students enquired whether there was a manual or introduction section. The students navigated through the interactivity and visualisation features. For the visual feature, the students were asked a enter code and interact with the buttons and slider to see the effects of the affordances on the code. Thereafter, the students we asked to make use of the visualisation feature. This was the only instruction given seeing that the feature had its own instructions. After their interactions with the applications, the students were asked to complete a survey rated the effectiveness, structure, functioning and helpfulness of each feature as well as the overall web application (see Appendix C1).

4.2 Evaluation of Testing Methods

4.2.1 Reliability of Result. Seeing that the number of participants was very low, the results may not be as reliable. The project only had 10 participants. Another issue is that the application has been designed for beginners, but the testing was done with participants who understand the basics of programming. The participants the application was tested on may not find the application as helpful as beginner who does not know the basis of programming.

5 FINDINGS AND DISCUSSION

When testing the web application, both qualitative and quantitative testing techniques we used. The qualitative techniques were observing the participants as they navigated through the application. Throughout this process, the applicants were given tasks to complete, and as they were completing those tasks, they were asked questions to determine their understanding of the feature they were focusing on. The quantitative testing method was survey that each participant was asked to complete (see Appendix C.1 and D.1).

5.1 Findings from Observations

When observing the students, it was found that they were not sure about how the application works and what exactly they were expected to do. A student enquired about instructions or a guide that helped the user navigate through the application. the student suggested we include an introductory video to assist a new user. When the first participant started typing the code, they typed a line of code that accepts input, and they had to be informed that the enhanced version for this project does not take input. The following participants were informed prior to typing their code that the application does not take in input.

When focusing on the visualisation feature the users did not seem to instantly get the affordances of the 'First', 'Forward', 'Back' and 'Last' buttons. The did not interact with the buttons until they were explained to them. The slider on top of the buttons did not seem to be visible enough as the students did not notice it. A participant suggested that I add colour that highlights steps in the slider that have already been executed in one colour and steps that are yet to be executed in another colour. The students recommended enhancing the overall colour for the interface. Students struggled to initially observe the link between the colour in the editor indicating the current, and the variable being highlighted.

When it comes to the interactivity feature, the students did pay as much attention to the 'Variable Testing' and assumed that the instructions were not linked to the web application. They asked a lot of questions regarding how the feature works (regardless of the description given above) and struggled to understand the objective of the feature. A student stated that the feature might be helpful for a beginner but does not seem necessary for them. The students also struggled with understanding the marking system for the interactivity feature.

5.2 Findings from Survey

5.2.1 Visualisation. Initially the participants did not grasp the purpose behind the changing colours and highlighting within the editor and visualisation tool. Upon explaining the purpose behind the feature to the participants, they understood why it had been implemented and found the feature to be helpful. Some participants stated that the feature could assist beginners in grasping the effect of each line of code, on the overall program. This feedback explains the low ratings for the *Understandability of Highlight Feature* and the higher ratings for the *Helpfulness of Highlight Feature*.

5.2.2 Description Pop-ups. When testing the features with the participants, the pop-up descriptions of the python keywords were displayed within the editor. Many participants complained that the descriptions were covering their code, making it difficult for them proceed with their coding. Although the positioning of the pop-ups was initially not ideal, participants did express that they found them to be helpful and convenient. This is mainly because, if someone has not coded in a while, they likely to forget the syntax for the different python keywords. When one can code in multiple languages, they are likely to confuse the syntax for one language, with another. This explains the moderate rating for the *Helpfulness of Pop-ups Explanations*. Seeing that the pop-ups were intended to assist beginners in recalling the syntax keyword, they were written to be as simple as possible. This explains the high ratings for *Understandability of Pop-ups Descriptions*.

5.2.3 Interactivity. The interactivity feature was executed in the form a Variable Test. The description for and instructions of the test were written above the test. After reading the description and instructions, the users were still not aware of the objective of the test, as well as how to complete the test. After explaining the objective and instructions, the students were quickly able to grasp how to use the feature and complete the test. Multiple students suggested the inclusion of a diagram to make understanding the test simpler. This explains the moderate rating for the *Comprehensiveness of Interactivity Feature*. Many participants found the testing feature to be helpful, especially for beginners. Many of them expressed how test would be helpful for debugging code, hence the high rating for *Helpfulness of Interactivity Feature*. The fact that the participants had more than a semester's worth experience in programming in python explains the low rating for the *Difficulty of Interactivity Feature.*

5.2 Discussion

5.2.1 Reflection of Methods. The evaluation methods did seem to be effective as we could see the struggles a user would experience whilst navigating through the application. Further instruction should have been given prior to simply asking the user to the interact with the application. Students struggled with a lot of the terminology and words used in the application. Some words they struggled with were simple, however in the context of the application they were not. An example of a word is 'Step'. The students did not initially grasp that step implies changing lines. The same applies for the 'First', 'Forward', 'Back' and 'Last' buttons. The students did not see the corelation between the navigation buttons and the code. The students frequently hesitated and asked questions throughout the testing process for each feature. This implied that the overall web application was not very comprehensive for someone who has never used it before.

5.2.2 Effectiveness of Visualisation Features. Overall, the students found the inclusion of the highlighting features to be effective. The explanations for the keywords were also seen as been helpful. The main changes that the students desired but could not be implemented was the explanation of each line of their code. When comparing the participants' ratings for the highlighting feature to their ratings for the pop-up explanations, it appears that there is a greater need for more comprehensive explanations of each of the python keywords. This may indicate users' difficulty in remembering when to apply a specific keyword or the syntax associated with the key.

5.2.3 Effectiveness of Interactivity Feature. The main goal of the feature was to be seen as testing feature however, although it was seen as testing feature by some, may students view it as being a debugger. Setting tests where the students are given a block of code and are expected to predict the changes for each line was discussed. It was considered because it was brought up that predicting the changes in each line of your own code may not be as challenging, since you wrote the code. It was suggested that the feature would be more challenging if students were being tested with code they are not familiar with. The new testing feature would be expected to have 'hard', 'medium' and 'easy' levels.

5.2.4 *Target Users.* During the user testing, when interacting with and being asked about the visualisation and interactivity features, many participants stated that the features would be helpful for beginners. A user suggested testing the web application on high school students. More accurate results would be obtained when testing the features with First-year students who have just arrived at university and have no coding experience. High school students are also ideal participants for the testing.

5.2.5 Participants' Challenges. Throughout the user testing, many participants enquired about error messages and about whether the web application would assist them in getting a better explanation for their error messages. When navigating through the visualisation and interactivity features, some users expressed that the step-by-step highlighting and the interactivity feature would assist them in the debugging process. From this it can be concluded that one of the main issues for programmers with the skillset that the participants have, is error-handling.

5.2.6 *Visualisations vs Interactivity*. When comparing the ratings for the *Helpfulness of Highlight Feature, Helpfulness of Pop-up Explanations*, and *Helpfulness of Interactivity Feature*, the Interactivity feature is seen as being more helpful than the two visualisation features.

5.2.7 Limitations of the Tool. Students had to be informed in the advance that the editor does not take in input. Students had to be informed of this because the first student attempted to write a line of code that accepted input. Students also expected the code to allow the importing of modules. The tool also only catered for Python users.

6 FUTURE WORK

Seeing that multiple students enquired about features that focus on error messages, a future project could be to improve the error messages for Online Python Tutor. This feature would also include providing a potential solution for common errors. Seeing that the usage of technology, including mobile devices, is continuing to increase, creating a mobile version of Online Python Tutor could be a good idea. This would provide individuals with a convenient way of learning how to program.

7 CONCLUSIONS

Learning how to program is seen as a daunting task for many individuals, and struggling to grasp the fundamentals results many individuals giving up on acquiring the skill of programming. Online Python Tutor is a web application that assists one in learning how to programme by providing a visualisation tool that maps out all the changes that occur in the program when a line of code is executed. This project focused on improving Online Python Tutor by making it more beginner friendly. In programming, variables play a crucial in giving someone an idea of what is happening when programming. As a result of this, the visualisation and interactivity features of the project focus heavily on variable assignment. The visualisation tools consist of highlighting variable changes the visualisation tool and providing explanations for common Python terminology. The interactivity tool is a testing feature. The incorporation of these features did prove to helpful and useful, however, it was found that they would be more beneficial to individuals who have no experience in programming, unlike the participants of the project, who had at least 6 months' worth of experience in programming. The project found that there is a need for more basic and comprehensive explanations and learning material for fundamental python topics. Multiple studies have proven that when teaching a skill, making use of interactivity is better than focusing solely on visualisation. When it comes to helpfulness, the success of the interactivity feature, compared to the success of the visualisation feature supports the conclusions made by the past studies.

REFERENCES

[1]
Yirsaw Ayalew, Ethel Tshukudu, and Moemedi Lefoane. 2018. Factors Affecting Programming Performance of First Year Students at a University in Botswana. *African Journal of Research in Mathematics, Science and Technology Education* 22, 3 (2018), 363-373. DOI:https://doi.org/10.1080/18117295.2018.1540169
[2]
Imre BENDE. 2022. Data Visualization in Programming Education. Acta Didactica Napocensia 15, 1 (2022), 52-60. DOI: https://doi.org/10.24193/adn.15.1.5
[3]
Anabela Gomes and António José Mendes. 2007. An environment to improve programming education. In Proceedings of the 2007 international conference on Computer systems and technologies (CompSysTech '07). Association for Computing Machinery, New York, NY, USA, Article 88, 1–6. https://doi-org.ezproxy.uct.ac.za/10.1145/1330598.1330691
[4]
Emma Riese and Stefan Stenbom. 2023. Engineering Students' Experiences of Assessment in Introductory Computer Science Courses. *IEEE Transactions on Education* 66, 4 (2023), 350-359. DOI:https://doi.org/10.1109/te.2023.3238895
[5]
Prasad, A. et al. (2022) Programming skills: Visualization, interaction, home language and problem solving. Education and information technologies. [Online] 27 (3), 3197–3223
[6]
Monika Mladenović, Žana Žanko, and Marin Aglić Čuvić. 2020. The impact of using program visualization techniques on learning basic programming concepts at the K–12 level. *Computer Applications in Engineering Education* 29, 1 (2020), 145-159. DOI:https://doi.org/10.1002/cae.22315
[7]
T. Ball, and S. G. Eick, Software visualization in the large, Computer 29 (1996), no. 4, 33–43.
[8]
C. E. Hmelo, and M. Guzdial, Of black and glass boxes: Scaffolding for doing and learning, Proc. 1996 Int. Conf. Learn. Sci., Evanston, IL, 1996, pp. 128–134.
[9]
J. Sorva, V. Karavirta, and L. Malmi, A review of generic program visualization systems for introductory programming education, ACM Trans. Comput. Educ. 13 (2013), no. 4, 1–64
[10]
T. L. Naps et al., Exploring the role of visualization and engagement in computer science education, SIGCSE Bull. 35 (2002), no. 2, 131–152

[11]
Lawrence, A. W. Empirical Studies of the Value of Algorithm Animation in Algorithm Understanding. PhD thesis, Department of Computer Science, Georgia Institute of Technology, 1993
[12]
Philip J. Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). Association for Computing Machinery, New York, NY, USA, 579–584. https://doi-org.ezproxy.uct.ac.za/10.1145/2445196.2445368

[13]
Juha Sorva and Teemu Sirkiä. 2010. UUhistle: a software tool for visual program simulation. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling '10). Association for Computing Machinery, New York, NY, USA, 49–54. https://doi-
[14]
Šimoňák, Slavomír..2014 "Using algorithm visualizations in computer science education" Open Computer Science, vol. 4, no. 3, pp. 183-190. https://doi.org/10.2478/s13537-014-0215-4
[15]
Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bull. 33, 4 (December 2001), 125–180. https://doi-org.ezproxy.uct.ac.za/10.1145/572139.57218
[16]
Eric Fouh, Monika Akbar, and Clifford A. Shaffer. 2012. The Role of Visualization in Computer Science Education. Computers in the Schools 29, 1 (2012), 95-117. DOI:https://doi.org/10.1080/07380569.2012.651422

[17]Yuan-Fang Li, Paramjit K. Das, and David L. Dowe. 2014. Two decades of Web application testing—A survey of recent advances. Information Systems 43, (2014), 20-54. DOI:https://doi.org/10.1016/j.is.2014.02.001

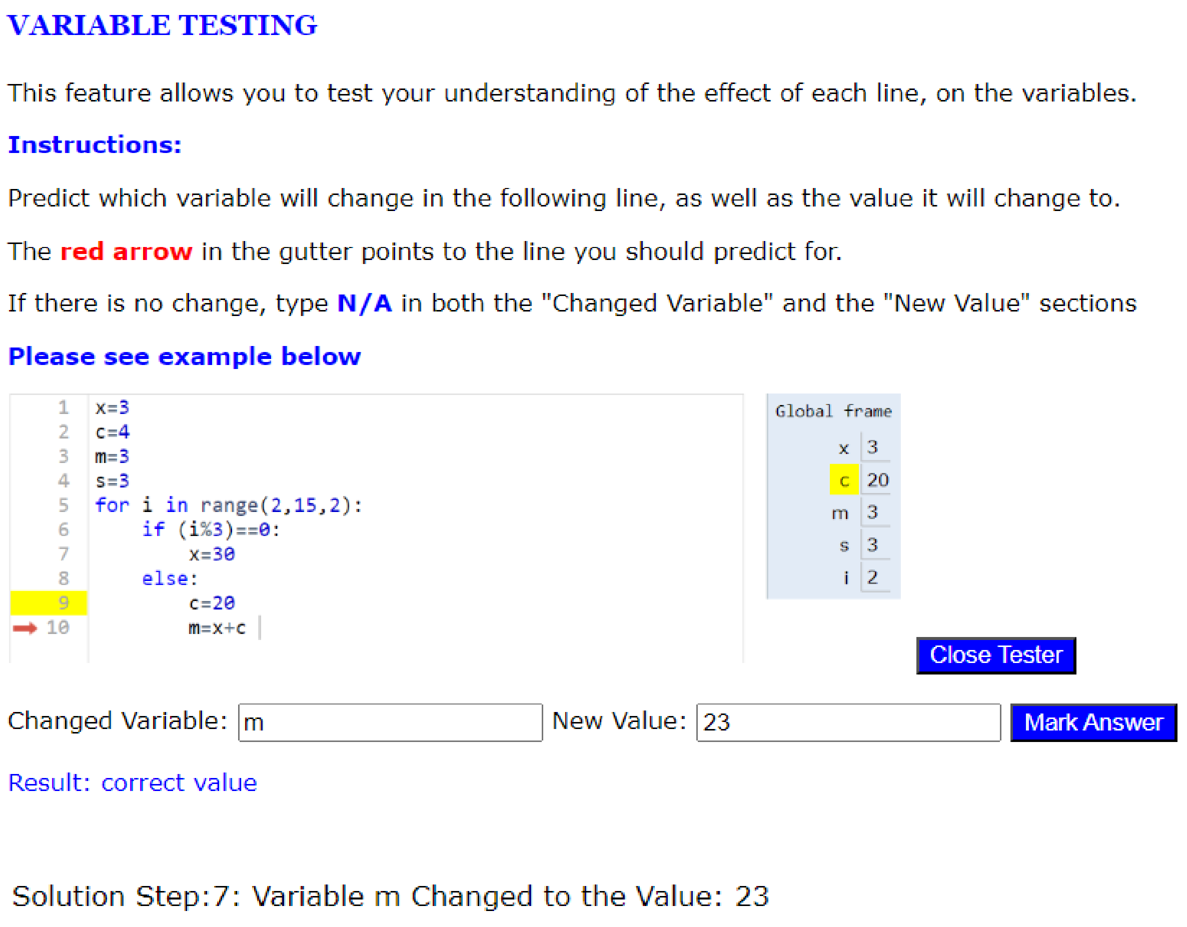
[18]2010. Put unit testing to work in your i programs: testing small programs, or units, within your applications can lead to big improvements in software quality. SQL Server Magazine 12. Retrieved September 7, 2023 from https://go-gale-com.ezproxy.uct.ac.za/ps/i.do?p=AONE&u=unict&id=GALE|A242305314&v=2.1&it=r

[19] Bo Chang, Renmei Xu, and Tiffany Watt. 2018. The Impact of Colors on Learning. Retrieved September 15, 2023 from https://newprairiepress.org/cgi/viewcontent.cgi?article=4001&context=aerc

SUPPLEMENTARY INFORMATION

A SYSTEM DESIGN

A.1 Interactivity Feature with Highlighting Component



VARIABLE TESTING

This feature allows you to test your understanding of the effect of each line, on the variables.

Instructions:

Predict which variable will change in the following line, as well as the value it will change to.

The **red arrow** in the gutter points to the line you should predict for.

If there is no change, type **N/A** in both the "Changed Variable" and the "New Value" sections

Please see example below

```
x=3
c=4
m=3
s=3
for i in range(2,15,2):
    if (i%3)==0:
        x=30
    else:
        c=20
        m=x+c
```

Global frame

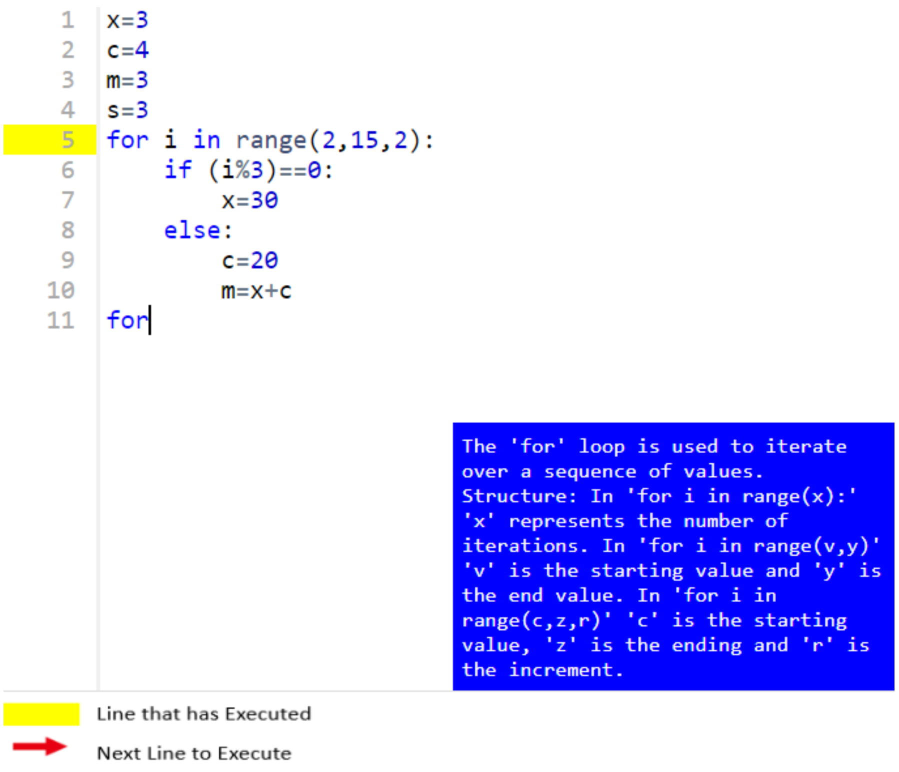
x	3
c	20
m	3
s	3
i	2

Changed Variable: m New Value: 23

Result: correct value

Solution Step:7: Variable m Changed to the Value: 23

A.3 Keyword Explanations

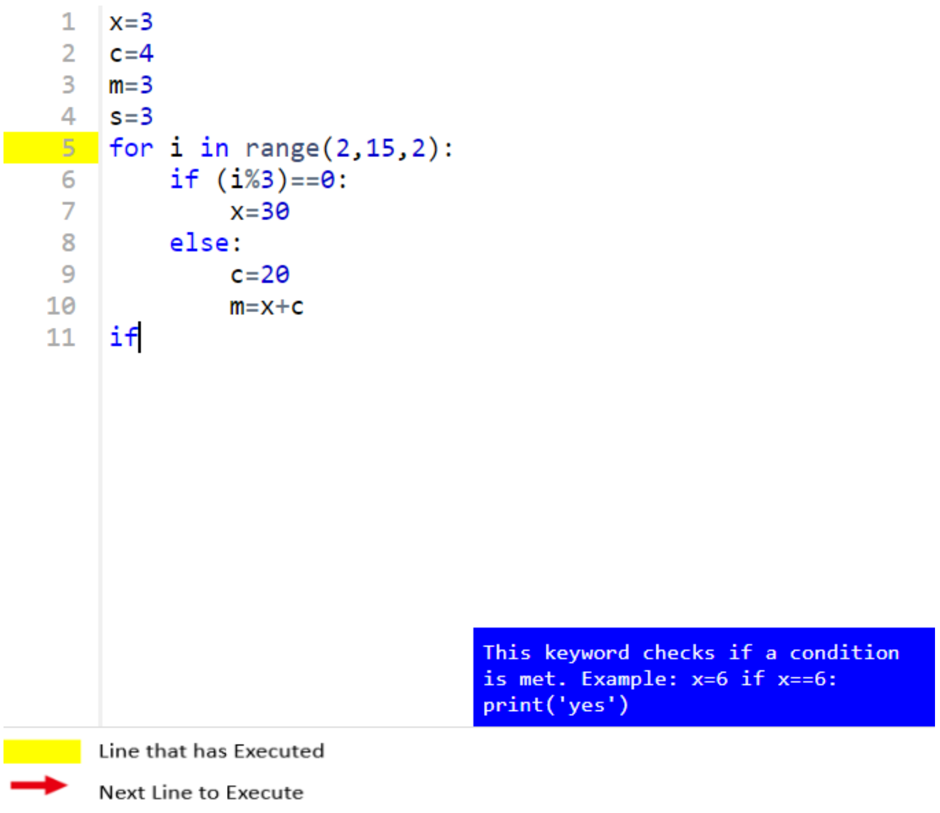


```
x=3
c=4
m=3
s=3
for i in range(2,15,2):
    if (i%3)==0:
        x=30
    else:
        c=20
        m=x+c
for
```

The 'for' loop is used to iterate over a sequence of values. Structure: In 'for i in range(x):' 'x' represents the number of iterations. In 'for i in range(v,y)' 'v' is the starting value and 'y' is the end value. In 'for i in range(c,z,r)' 'c' is the starting value, 'z' is the ending and 'r' is the increment.

🟨 Line that has Executed

→ Next Line to Execute



```
x=3
c=4
m=3
s=3
for i in range(2,15,2):
    if (i%3)==0:
        x=30
    else:
        c=20
        m=x+c
if
```

This keyword checks if a condition is met. Example: x=6 if x==6: print('yes')

🟨 Line that has Executed

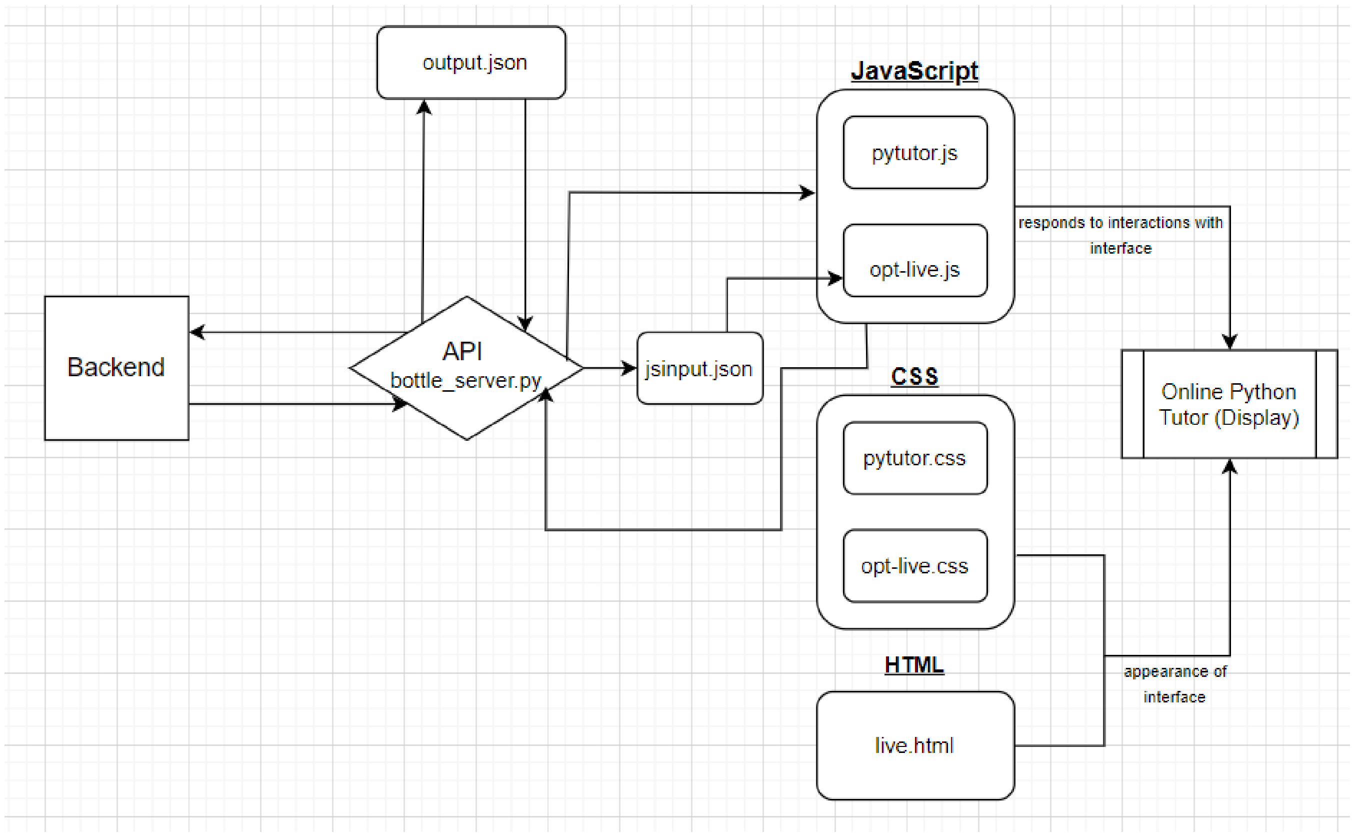
→ Next Line to Execute

A.2 Requirements

Requirement	Details	Justification
Highlight Feature	As the user steps through the code in the editor, the currently executing line number should be highlighted in the gutter of the editor. If the there is a variable in the current line, that is being declared or changed, that variable should be highlighted in the code visualisation tool.	Highlighting is a tool that automatically draws a learner to the variable being highlighted. This tool will also allow the learner to easily detect the variable that has been changed in the editor as well as the value it has changed to.
Descriptions of Keywords	Python basic keywords will have descriptions which will be displayed in the editor when the specific keyword is entered. The keywords that are focused on are ‘print’, ‘if’, ‘elif’, ‘else’, ‘for’ and ‘while’.	Beginners may struggle when determining which keyword to apply at a specific point in their code. This feature reminds them of how each variable is used, as well as what the correct syntax for the variable is.
Variable Testing	As the user steps through their code, they will be given the option to predict the variable that will change in the next line, as well as what the variable will change to.	This allows users to test whether they understand the effect each line of code has on variables and the overall program.

B IMPLEMENTATION

B.1 Frontend Architecture



B.2 Extracting Variables and Values

For the following code:

```
x=10
v=4
```

The following trace file is created (output.json):

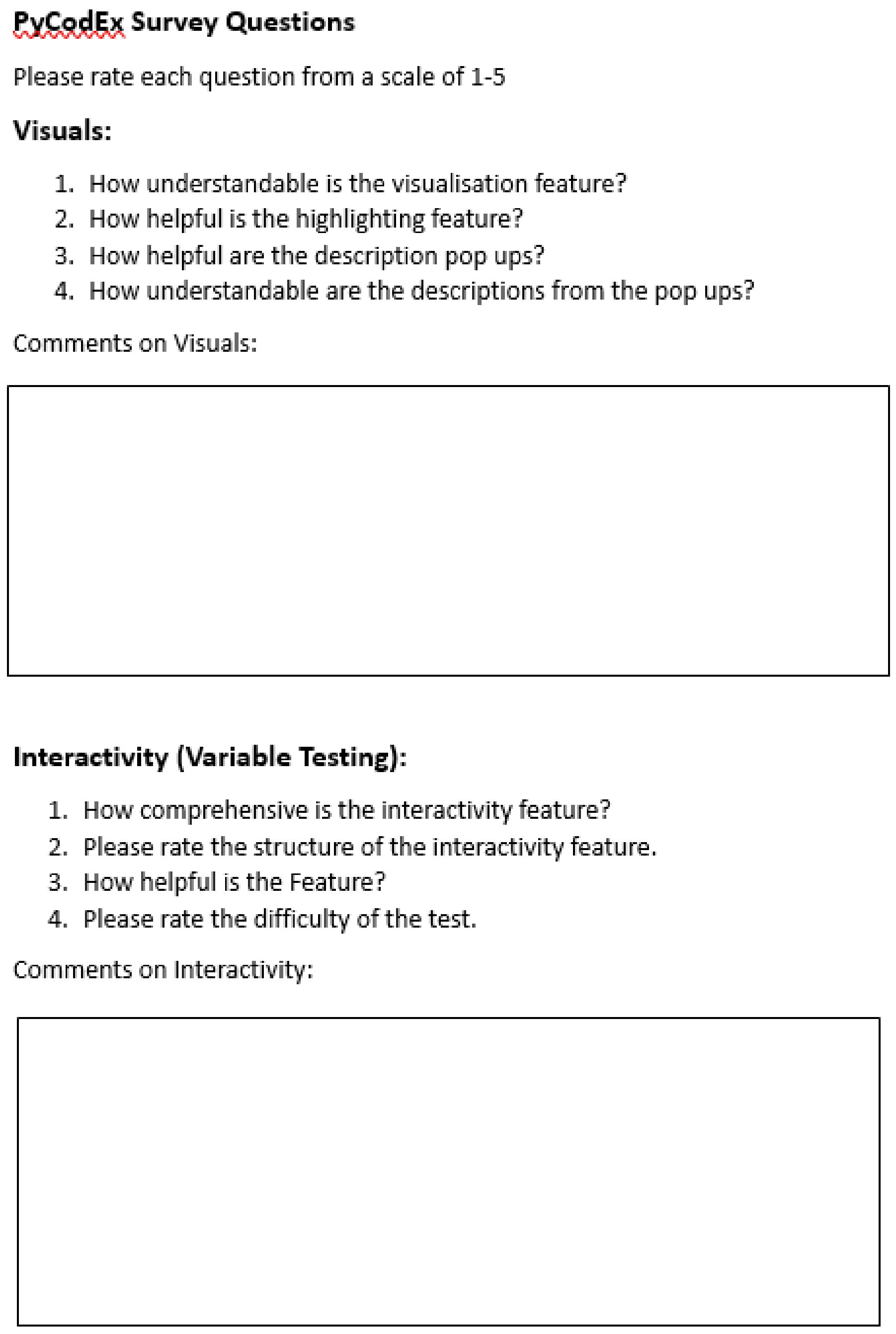
```
{"code": "x=10 \nv=4 \n", "trace": [{"line": 1, "event": "step_line", "func_name":
"<module>", "globals": {}, "ordered_globals": [], "stack_to_render": [], "heap":
{}, "stdout": ""}, {"line": 2, "event": "step_line", "func_name": "<module>",
"globals": {"x": 10}, "ordered_globals": ["x"], "stack_to_render": [], "heap": {},
"stdout": ""}, {"line": 2, "event": "return", "func_name": "<module>", "globals":
{"x": 10, "v": 4}, "ordered_globals": ["x", "v"], "stack_to_render": [], "heap":
{}, "stdout": ""}]}
```

The following values are extracted into the jsinput.json file:

```
{
    "1": "N/A , N/A",
    "2": "x , 10",
    "3": "v , 4"
}
```

C TESTING AND EVALUATION

C.1 User Survey



PyCodEx Survey Questions

Please rate each question from a scale of 1-5

Visuals:

1. How understandable is the visualisation feature?
2. How helpful is the highlighting feature?
3. How helpful are the description pop ups?
4. How understandable are the descriptions from the pop ups?

Comments on Visuals:

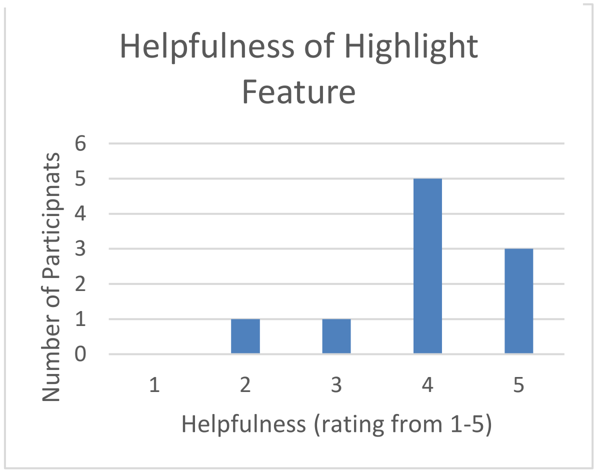
Interactivity (Variable Testing):

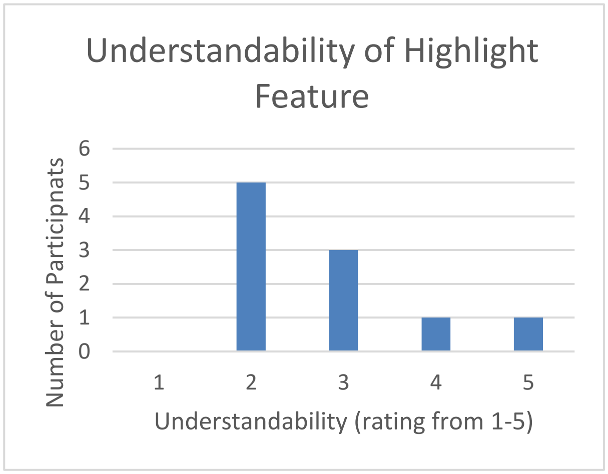
1. How comprehensive is the interactivity feature?
2. Please rate the structure of the interactivity feature.
3. How helpful is the Feature?
4. Please rate the difficulty of the test.

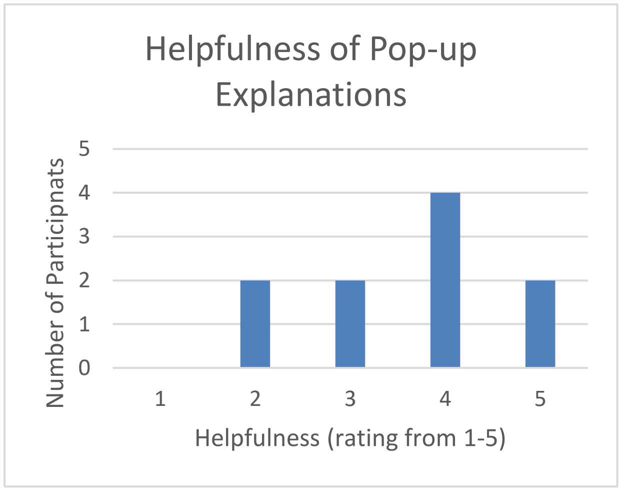
Comments on Interactivity:

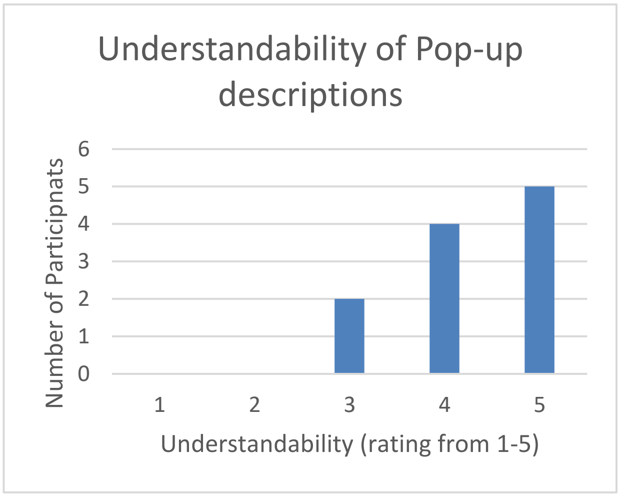
D FINDINGS AND DISCUSSION

D.1 Visualisation Feedback Graphs

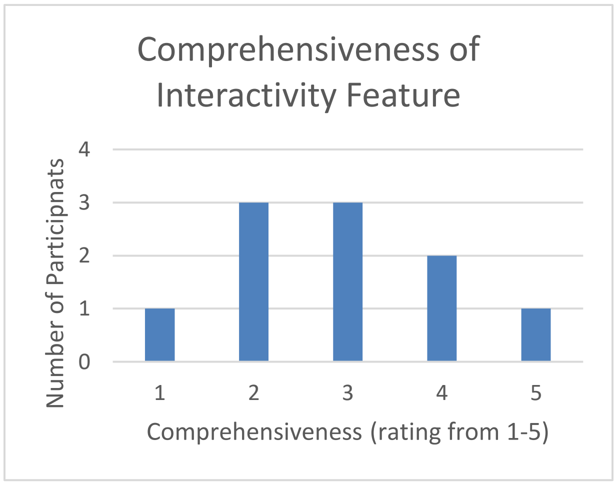


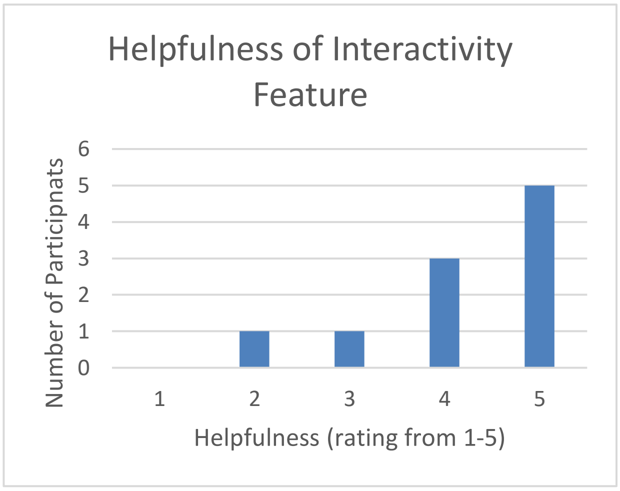


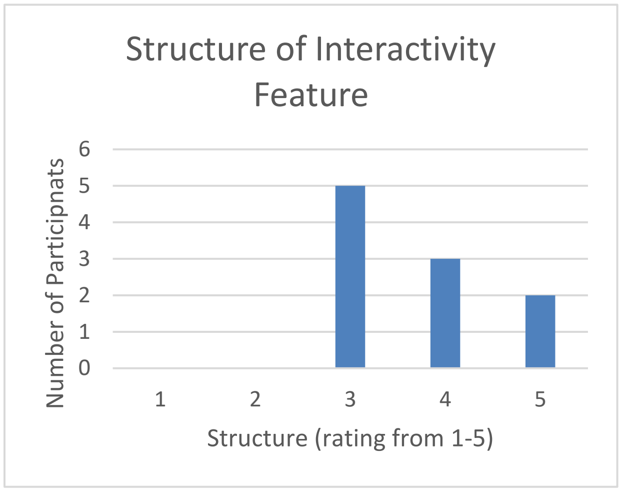


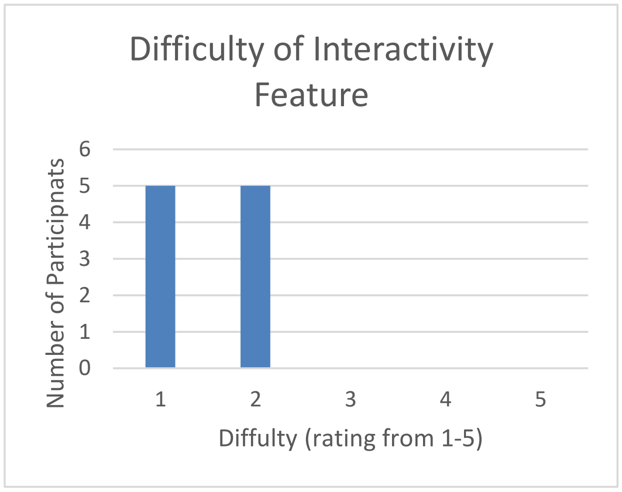


Interactivity Feedback Graphs






Structure of Interactivity Feature
Number of Participnats
0
1
2
3
4
5
6
1
2
3
4
5
Structure (rating from 1-5)


Difficulty of Interactivity Feature
Number of Participnats
0
1
2
3
4
5
6
1
2
3
4
5
Diffulty (rating from 1-5)