DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2023

Title: Interactive Python Code Explainer

Author: Mabilo Mufhulufheli

Project Abbreviation: PYCODEX

Supervisor(s): Gary Stewart

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	15
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	10
System Development and Implementation	*0*	*20*	15
Results, Findings and Conclusions	*10*	*20*	10
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**	

A Multimodal Approach to Introductory Coding

Mufhulufheli Mabilo
University of Cape Town
Cape Town, South Africa
MBLMUF001@myuct.ac.za

Abstract

Introductory programming can pose challenges for numerous first-year Computer Science students, often leading to feelings of difficulty and being overwhelmed. Skills like using for-loops, creating algorithms, and thinking logically are fresh abilities that programmers must cultivate. This growth takes time and requires continuous interaction with Computer Science materials. An increase in engagement levels can be accomplished by integrating strategies that minimize context switching, coupled with the capability to elucidate code through the use of voice audio capabilities. However, despite their frequent effectiveness, designing educational systems with the capacity to reduce context switching and provide code explanations through voice audio can pose intricate challenges. This research paper is centered on the improvement of user-friendliness and the reduction of context switching within the existing Python learning tool, Online Python Tutor. It investigates various methods to enhance the tool's accessibility for Computer Science students by incorporating reduced context switching and the capacity to explain code concepts in a manner tailored for novice users through voice audio presentations of the code. The results indicate that reduced context switching benefits students by consolidating all materials and learning content onto a single page. Furthermore, in terms of the capacity to explain code concepts, it is observed that the approach effectively elucidates all relevant concepts comprehensively.

1 Introduction

In an era characterized by swift technological advancements and pervasive digital transformation, there has been an unprecedented surge in the global demand for computer scientists. As industries spanning diverse sectors integrate technology into their fundamental operations, the role of a computer scientist has transcended from a specialized expertise to becoming a pivotal cornerstone of modern societal and industrial progress[12].

While the enrollment of computer science undergraduate has experienced significant growth, a stark contrast emerges when examining the trajectory of computer science graduates. The path of enrollment growth, evident at the undergraduate level, encounters a formidable obstacle as it progresses to the realm of graduates largely attributed to elevated dropout and failure rates [1].

Upon a thorough examination of these heightened dropout rates, a conspicuous pattern emerges, with the primary point of attrition for computer science students being within their first year of studies. This pivotal juncture functions as a defining moment, essentially dictating whether students will continue their pursuit of a computer science degree. The analysis reveals a concerning statistic, as approximately 67% [12] of students exhibit the ability to successfully navigate their introductory programming courses. The palpable disparity between the number of students who pass these foundational courses and those who encounter difficulties underscores a significant challenge within the computer science curriculum [12].

The core issue underlying this challenge is the inconsistency in engagement demonstrated by a substantial number of computer science students with course content. This lack of consistent engagement hampers the assimilation of fundamental programming skills, ultimately leading to a decline in their enthusiasm for the field of computer science. The initial stages of programming present a formidable challenge, demanding sustained effort for comprehension. A solution to address this issue involves the implementation of a web application that supports reduced context switching and offers the capability to elucidate code through the integration of voice audio capabilities[11].

1.1 Problem statement

Current computing tools provide users with the opportunity to enhance their proficiency in Python by furnishing a platform where they can execute their code incrementally, allowing them to observe the outcomes of each line in a step-by-step manner. While tools with these capabilities do exist, their effectiveness in educating novice programmers in the art of coding may have certain limitations. The learning curve in programming is substantial, especially for individuals who are beginners in the field of Computer Science. While these tools provide an interactive environment for code experimentation, they may not offer a comprehensive coverage of the foundational concepts and intricacies required for a holistic understanding of programming principles.

This paper presents the integration of Online Python Tutor, an open-source platform that enhances code comprehension through the use of visual debugging. Through the provision of a dynamic visualization of code execution, Online Python Tutor enables individuals to acquire insights into program behavior, pinpoint errors, and elevate their overall programming proficiency.

While Online Python Tutor has not been implemented at the University of Cape Town, its effectiveness is evident through its adoption at prestigious institutions like MIT, the University of Washington, and UC Berkeley, where it assumes a crucial role in teaching introductory computer science to first-year students. This statistic serves as a compelling testament to the platform's utility, with over 30,000 individuals utilizing it on a monthly basis as of 2013[6]. Furthermore, the tool's integration into three web-based digital Python textbook projects serves as strong evidence of its recognition and relevance within educational contexts.

However, it's important to acknowledge that the learning tool may not provide optimal user-friendliness for first-year computer science students. Notably, the tool primarily caters to individuals who prefer visual and text-based learning styles. Both in the code visualization tool and the error messages, the use of complex terminology, which might be unfamiliar to beginners, could pose challenges to effective comprehension. Despite the inclusion of a visual debugger within Online Python Tutor, its clarity might be limited, as it mainly highlights the current line in a program, omitting a clear display of altered and referenced variables[6].

In order to improve the usability and advantages of the Online Python Tutor web application for first-year students, the following features have been incorporated:

- **Reduced Context Switching.** - Context switching, defined as the frequent toggling between two interfaces during task execution, can exert a significant impact on the learning process. This phenomenon carries the potential to impose a formidable cognitive load introduced by context switching can disrupt concentration and divert attention away from the code itself. By mitigating this cognitive load, distractions are minimized, providing learners with the capacity to engage in a more profound immersion with the code. This reduction in cognitive burden subsequently contributes to expediting the process of comprehending complex coding structures. In response to this challenge, a noteworthy feature is introduced. An interface window is designed to minimize the need for disruptive context switching. This supplementary window accommodates a comprehensive array of content, inducing elucidations of essential coding concepts and practical tasks that necessitate the application of code. By centralizing these resources within the interface, the feature enhances the fluidity of the learning experience. This strategic integration serves to create a seamless environment wherein learners can engage deeply with coding concepts and tasks without the impediments posed by the constant back-and-forth context switching. Ultimately, the reduced cognitive load resulting from this feature empowers learners to devote more focused time and attention to unraveling the intricacies of coding, intimately accelerating the mastery of code comprehension.
- **Code Audio Explanations** - Recognizing that not all students adhere to visual, reading or writing learning styles, we introduce an audio explanation feature. The tool predominantly caters to the learning modalities, potentially leaving a subset of students underserved. This disparity underscores the need for diversity in pedagogical approaches. The proposed audio feature fills this void by embracing an alternative learning style. This learning approach resonates who grasp content most effectively when presented in an audio format. Consequently, the introduction of audio explanations represents a concerted effort to cater for this learning style. Loops, if-statements, variable assignments, and pivotal coding structures are encompassed within the scope of this feature. With the integration of this feature, learners can expect more comprehensive explanations for intricate code segments that might otherwise prove challenging to comprehend. When a student submits code for compiling, the audio feature generates a succinct yet informative summary of the entered code. This verbal elucidation breaks down the code's intricacies, offering insights into the underlying logic and facilitating the development of a deeper understanding.

The aims of the project are:

- The development of an innovative audio feature aims to enhance the educational experience within the web application. This feature is designed to provide explicit and coherent explanations for Python code input by students, with the overarching goal of ensuring heightened clarity and improved comprehension during code compilation and execution. By implementing this feature, the research seeks to explore how audio-based explanations can facilitate a more effective learning experience, particularly for those who may benefit from auditory learning modalities. Ultimately, this enhancement aims to contribute to more successful learning outcomes among the user base.
- The implementation of reduced context switching in the web application which will serves as a strategic objective. This aim is directed towards enhancing the user experience by minimizing the need for users to switch between multiple pages or interfaces when

working with the application. The research aims to investigate how this reduction in context switching can benefit students by facilitating a more streamlined and focused interaction with the tool. The ultimate goal is to improve the overall usability and effectiveness of the web application as an educational resource for students, contributing to their learning outcomes and comprehension of Python programming concepts.

2 Background

2.1 Computer Science

Computer Science is a multidimensional field encompassing a wide array of phenomena intrinsic to the realm of computation. It constitutes the systematic exploration of algorithmic processes, encompassing their theory, analysis, design, efficiency, implementation, and practical application. At its core, computer science delves into the intricate world of information structures, investigating how data is organized, stored, retrieved, and processed. Furthermore, the discipline addresses the formidable challenge of complexity, with the aim of comprehending and managing intricate systems. Central to computer science is the mechanization of abstraction, where intricate concepts find translation into computational procedures. This synthesis of knowledge culminates in a discipline that not only facilitates problem-solving and innovation but also forms the foundational framework underpinning the trajectory of modern technological advancement [4].

2.2 Navigating Challenges in Computer Science Education

In the midst of an ever-evolving integration into a technology-driven world, universities are confronted with heightened expectations to consistently produce computer science graduates equipped with a profound understanding of the field. However, the prevailing reality is that universities often grapple with the challenge of keeping pace with the burgeoning demand. This heightened demand has led to a conspicuous scarcity of qualified computer science professionals, contributing to one of the most pressing labor shortages within the South African Computer Science Sector [2]. This critical shortage has a cascading impact, as the number of computer science graduates emerging from higher education institutions is in decline.[9].

The primary cause of the disparity between the demand for computer science graduates and the available supply can be traced back to the early stages of computer science education. The substantial challenge posed by high dropout and failure rates in first-year introductory computer science courses becomes evident. This attribution phenomenon significantly disrupts the pipeline of prospective computer science graduates. Consequently, initiatives aimed at increasing the number of qualified professionals are impeded by the difficulty in retaining students in these foundational courses.[7]

2.3 Cultivating Ongoing Student Engagement in Computer Science

The domain of introductory programming places students in direct contact with the challenge of familiarizing themselves with programming concepts such as 'for-loops,' all the while refining their problem-solving skills by methodically integrating these unfamiliar concepts to craft functional code. For a significant number of students, grasping these principles proves to be a formidable task. When students lack the motivation to consistently engage with these concepts and make an effort to comprehend them, it frequently results in a reluctance to pursue further studies in Computer Science. [9].

Scholarly insights highlight the paramount importance of sustained practice in the development of programming skills. Law et al [8] emphasize that programming proficiency flourishes through persistent practice, underscoring the notion that mastery evolves through ongoing application. Hawi's research underscores the necessity for active engagement in grasping course content, suggesting that students deeply invested in their studies tend to achieve superior learning outcomes [6]. A similar perspective is echoed by Tan and associates, who assert that introductory programming demands the cultivation of high-order thinking skills—a pursuit reliant on unwavering practice and unwavering determination.[6]

2.4 Online Python Tutor

Online Python Tutor traces its origins back to January 2010 when it began as a side project of a graduate student. The project was inspired by the creator's own experiences in teaching Python using complex stack and heap diagrams on a whiteboard [5]. This endeavor was fueled by the advent of progressively advanced web browsers and technologies, opening up possibilities for the creation of an educational program visualization tool that could seamlessly function within a web browser environment. [5].

The primary objective behind the design of Online Python Tutor was to create a tool that Python instructors and students would prefer to use, either independently or in conjunction with traditional whiteboard and PowerPoint diagram methods. Their journey began by distilling common attributes from prevalent program visualization tools and then adapting these features to align with the constraints of a web browser. Over the course of three years, the user base for Online Python Tutor experienced organic growth, largely propelled by word-of-mouth endorsements[7].

During their evolution, they systematically refined their initial design, introducing innovative features such as support for nested functions and lambda expressions. Their development process was profoundly influenced by their vibrant user community, which consisted of diverse computer science instructors, including teaching assistants and professors from institutions of varying sizes. It even extended to instructors overseeing free online computer science courses that collectively attracted over 200,000 enrolled students[7].

3 Design

The role of design in achieving the project's objectives is paramount. To fulfill its aims, the project builds upon the foundation of an existing Python tool, Online Python Tutor. This web application enables user interaction by offering a text editor where Python code can be input, resulting in code outputs and audio explanations. Additionally, learners have the opportunity to access video lessons covering both coding and theoretical aspects. Importantly, the design ethos also focuses on reducing context switching, allowing students to maintain their focus on a single page during their interaction with the tool. The design philosophy revolves around strengthening fundamental programming concepts. Through this thoughtful design approach, the project aims to provide a comprehensive and engaging platform for acquiring a deep understanding of basic programming concepts.

3.1 Educational Aspects

Central to the educational aspect of the project is the selection of Python as the programming language for the web application. This decision is grounded in Python's distinctive attributes that align with effective teaching methodologies. Renowned for its clear and concise syntax, Python is celebrated for its user-friendly nature, making it accessible even to first-year students. Its global prevalence further solidifies its suitability as a teaching tool for introductory programming.

Importantly, this choice is in alignment with the University of Cape Town (UCT) first-year computer science courses, where Python serves as the designated introductory language. Notably, participants for the web application testing phase will be recruited from the University of Cape Town, specifically in the Computer Science module CSC1010H or CSC1011H, ensuring a targeted and contextually relevant evaluation process.

3.2 Error Management

Managing errors constitutes a fundamental aspect of the web application's design. It is presumed that first-year computer science students possess familiarity with elementary Python syntax and the ability to construct code that adheres to the language's conventions. The code they input undergoes rigorous examination by a standard debugger module (bdb). This debugger methodically scans for and identifies the following basic types of errors, pinpointing the line numbers containing syntactic inaccuracies:

- Syntax Errors - These encompass violations of Python's grammar rules.
- Indentation Errors - Occur when code is not correctly aligned.
- Variable Name Errors - Arise from referencing undefined or misspelled variables.
- Function Name Errors - Stem from calling undefined or incorrectly spelled functions.
- Runtime Errors - Occur during code execution due to issues like division by zero or invalid data operations.

With its comprehensive coverage, the debugger adeptly handles this spectrum of errors that students may encounter during their coding endeavors.

3.3 Core Programming Concepts

The web application is primarily designed for users with limited programming exposure, specifically targeting students in their first year of introductory computer science courses. Due to the project's time constraints, a carefully selected subset of fundamental programming concepts has been identified for inclusion in the web application:

- Functions and parameter passing.
- Variables and primitive data types.
- Recursion calls.
- For-loops and While-loops.
- Expression and assignments.
- Code Execution.

3.4 Learning Curve

The incorporation of new features into Online Python Tutor was not devoid of challenges. The tool's open-source nature necessitated a comprehensive understanding of its architecture and functionality before augmentations could be introduced. This preliminary phase of familiarization demanded time and effort to grasp the intricacies of the existing system.

Moreover, in addition to addressing the architectural intricacies, the project embarked on the complex journey of enhancing code explanations. This involved the exploration of various Natural Language Processing (NLP) models to ensure that the code explanations generated by the web application were clear and coherent in a natural language. Additionally, the project aimed to introduce voice libraries that delivered explanations in a human-like manner, steering clear of the typical robotic voice associated with many text-to-speech systems. Achieving this required a meticulous search for voice libraries and technologies that could provide a more natural and understandable voice audio component to the code explanations.Subsequently, the task involved devising strategies to seamlessly integrate the proposed enhancements, both in terms of code explanations and voice audio, into the framework while ensuring compatibility and cohesion with the existing system.

This learning curve was essential to guarantee that the additions harmonized with the tool's overarching objectives and user experience.The dual challenges of grasping the tool's architecture and enhancing code explanations with human-like voice audio were emblematic of the project's commitment to providing users with a rich and user-friendly educational experience.

3.5 Streamlined Development Framework and Architecture

The development of the web application strictly adhered to the guiding principles of simplicity and modularity, primarily driven by the constraints of time. This approach was instrumental in expediting the debugging process and elevating the overall efficiency of the development effort. Embracing a modular structure played a pivotal role in ensuring that the codebase retained its clarity, thus facilitating the rapid identification and resolution of issues as they arose.

Furthermore, the adoption of modularity bestowed a significant advantage in terms of seamless scalability, permitting the integration of new features and components with minimal disruption. This architectural choice not only cultivated ease of maintenance but also facilitated the allocation of responsibilities among the research team during the development phase. The commitment to simplicity and modularity, even within the confines of time constraints, was instrumental in ensuring the project's success, streamlining development processes, and preserving the project's adaptability for accommodating future enhancements.

One noteworthy enhancement within the web application is the audio feature, which elucidates user-inputted code. This feature leverages the open-source Natural Language Processing model called Spacy, the architecture digram in **Figure** 8. This model harnesses the execution trace to translate code results into human-readable text. Subsequently, this interpretive text is seamlessly processed through the Google Text-to-Speech Python library (commonly known as gTTS), the architecture digram in **Figure** 6 of gTTS, culminating in an auditory synergy that underpins the development framework, harnessing diverse tools to create a unified and effective learning experience.

4 Implementation

At the core of the project's realization lies the Online Python Tutor, a web application intricately woven together with a Python-powered backend and a frontend elegantly crafted using standard web technologies: HTML, CSS, and Javascript. The working of the web application is on **Figure** 5.

4.1 Backend

The Online Python Tutor backend serves as a critical component of the system, taking the source code of a Python program as input and producing an execution trace as output. This backend executes the input program under the supervision of the standard Python debugger module (bdb). It halts execution after each executed line, meticulously recording the program's runtime state. The resulting trace is an ordered list of execution points, with each point containing essential information, including:

- The line number about to execute.
- The type of instruction (ordinary single step, exception, function call, or function return).
- A mapping of global variable names to their current values at the execution point.
- An ordered list of stack frames, with each frame containing a map of local variable names to their current values.
- The current state of the heap.
- The program's output up to that execution point.

The backend encodes this trace in JSON format, leveraging Python data types serialized into native JSON types with additional metadata tags [3]. To prevent infinite loops and overly long traces, the backend terminates execution after 300 executed lines. While this limit may appear small for real-world programs, it suffices for pedagogical code examples targeted by students. Notably, the trace can be verbose due to its storage of complete snapshots of the stack and heap at every execution point. Traces from pedagogical code examples typically range from 10KB to 200KB in size.

The backend supports both Python 2 and 3, a critical feature given that courses are taught using both of these major language variants. It can be hosted on any web server capable of running CGI scripts (using Python 2 or 3) or on the Google App Engine platform (Python 2.7). When a user's web browser initiates an HTTP GET request to the backend and submits the Python program for visualization, the backend generates the trace and returns it to the browser in JSON format.

To ensure security, particularly when running untrusted Python code from the web, the backend implements sandboxing. This safeguard prevents the execution of potentially harmful constructs like eval, exec, file I/O, and most module imports (except for a customizable whitelist of modules such as math). Additionally, many web servers implement "defense in depth" measures to provide extra layers of protection.

4.1.1 Audio Explanation Generation

The educational aspect of the web application relies on the processes within spaCy NLP, a proficient library renowned for its linguistic analysis capabilities. SpaCy plays a pivotal role in dissecting user-provided Python code, unraveling its syntactic structure and grammatical intricacies. This process begins with spaCy's exceptional ability to perform linguistic dissection, meticulously breaking down the code lines into tokens, providing a comprehensive understanding of each line's textual essence. This detailed linguistic analysis forms a solid foundation for subsequent enhancements.

The culmination of spaCy's analysis results in insightful explanations, enhancing the linguistic analysis with contextual depth and enabling the elucidation of code intricacies that might elude casual observation. As the process progresses within the spaCy module, the intricate logic woven into the code is unveiled, peeling back the layers to offer an intricate portrayal of how code components interact. Users are not only provided with an understanding of the code's outcome but also with insights into the intricacies of its execution.

Alongside spaCy, the Text-to-Speech (TTS) technology gTTS assumes a vital role in the transformation of textual explanations into auditory elucidations, promoting a multi-modal learning approach. The resulting audio summaries enable learners to grasp the code's nuances through auditory immersion. This collaborative effort between spaCy NLP and gTTS enriches the educational experience of the web application, catering to diverse learning preferences and enhancing code comprehension.

The generated audio explanations, crafted through the collaboration of spaCy NLP and gTTS, are seamlessly integrated into the web application's frontend. This integration ensures that students have convenient access to the audio elucidations, which can be played at their discretion. Furthermore, the web application empowers students with control over the audio experience, providing the capability to adjust the playback speed. This feature allows students to tailor the audio explanations to their preferred pace of learning, whether they prefer a slower, more detailed explanation or a quicker overview. By returning the audio explanations to the frontend with speed control options, the web application enhances the overall educational experience, promoting personalized and effective code comprehension.

4.2 FrontEnd

The frontend of the Online Python Tutor is a user-friendly website located at pythontutor.com. It is designed to be compatible with all modern web browsers without requiring students to install any extensions or plugins. The user interface is straightforward, presenting users with a text box where they can type or paste Python programs. In informal testing, the webpage loads quickly, with examples similar to **Figure** 11 loading in approximately 5 seconds on a fast Wifi and around 10 seconds on a slow Wi-Fi. Once the initial webpage is loaded, all interactions take place in the browser, and there are no additional server calls. Therefore, stepping forwards and backwards through code execution refreshes the display instantaneously.

One of the primary technical challenges in the frontend is rendering objects in a clear and aesthetically pleasing manner. Since the heap can be a complex graph of objects, a naive rendering approach could result in a tangled mess of boxes and pointers. To address this, the frontend employs several layout heuristics inspired by diagrams in textbooks and lecture slides:

- Each frame variable and heap object is stacked vertically in the order of their creation during program execution.
- Objects remain in the same location relative to other objects until there are no more pointers to them.
- Compound objects that contain pointers to other compound objects are nested inside the outer one.
- Linked lists are rendered specially by laying out node objects horizontally.

To achieve these layout heuristics, the frontend uses two main JavaScript libraries: D3 to map execution trace objects to corresponding HTML elements and jsPlumb to draw pointers as arrows. The entire visualizer GUI is encapsulated in a JavaScript ExecutionVisualizer class, making it easy to embed within a webpage.

4.2.1 Video lessons

In the pursuit of simplifying code comprehension, Online Python Tutor harnesses a valuable resource - video tutorials. These videos serve as guiding lights, assisting students in navigating the complexities of programming. By seamlessly integrating video tutorials, the platform offers a unique advantage, enabling students to grasp concepts even before embarking on coding.

The integration of video tutorials entails a meticulously designed process, with these videos finding their home in Firebase Database Storage, forming a harmonious ensemble of educational content. At the core of this educational experience, the "videoPlayer" component takes center stage. This dedicated space serves as the canvas where educational content comes to life, providing the platform upon which the journey of knowledge gracefully unfolds. When a user interacts with the dropdown menu to select a different video, as shown in **Figure** 10 , a seamless transition in content occurs. The chosen video takes center stage, and its content is seamlessly loaded into the video player.

The intricately orchestrated integration of video lessons and code snippets creates an environment that nurtures optimal learning conditions. Theory video lessons and code fragments coexist seamlessly, elevating the learning experience for students. As users explore diverse coding scenarios, video tutorials act as guiding beacons, shedding light on the path to mastery. These videos, elegantly stored within Firebase Database Storage, transcend their role as mere visual aids. They empower students with a profound understanding even before they embark on their coding journey, enriching and enhancing the entire learning process.

4.2.2 Context Switching

The development of the web application was strategically designed with a focus on reducing context switching for users. Context switching refers to the cognitive effort required when transitioning between different tasks or environments. In the context of programming and learning, excessive context switching can be disruptive and hinder the learning process.

To mitigate context switching, the web application offers a seamless and integrated learning experience. It provides users with a unified platform where they can not only write and visualize Python code but also access explanatory audio and video content without the need to switch between multiple tools or websites. This consolidation of resources ensures that learners can maintain their focus on the task at hand, which is understanding and writing Python code.

Additionally, the integration of audio explanations directly within the application reduces the need for users to switch to external resources for clarifications. They can listen to detailed explanations while viewing the code, creating a synchronized and uninterrupted learning experience. Overall, the deliberate design choice to minimize context switching enhances the user's ability to concentrate on learning programming concepts effectively within the web application, ultimately improving the learning outcomes for students, which is shown in **Figure** 10.

5 USER TESTING

During the user testing phase, a total of 10 participants were thoughtfully recruited from the University of Cape Town's computer science modules, specifically CSC1010H and CSC1011H, 5 participants from CSC1010H and 5 participants from CSC1011H. These participants were selected due to their foundational knowledge in Python programming, aligning with the web application's target audience. The testing process was meticulously organized, with participants booking 30-minute testing slots in advance, and reminders were thoughtfully sent to ensure their timely participation. Prior to interacting with the web application, participants digitally signed informed consent forms to ensure they were fully aware of the testing procedure.

An instructional video served as a guide, explaining the effective usage of the web application. Participants were given the freedom to copy and paste Python code from W3Schools into the application's code segment, with the caveat of a 30-minute time constraint to manage the testing sessions efficiently. Feedback collection played a pivotal role, with participants sharing their experiences and insights through surveys. To incentivize their valuable participation, participants were entered into a raffle for a chance to win a R250 Takealot voucher. The insights and feedback gathered during this user testing phase were instrumental in shaping the final form of the educational web application, ensuring its competitiveness and effectiveness within the targeted educational landscape.

6 Survey

The post-user testing survey was comprehensive in its approach, encompassing both quantitative and qualitative aspects to gain valuable insights into participants' engagement and experiences with the web application. In the quantitative section of the survey, the refined User Engagement Scale (UES) was utilized. This abbreviated version of the original scale introduced by O'Brien [10] was chosen to ensure that participants feedback was collected efficiently, considering the potential for participant fatigue due to the concise nature of the refined UES.

By employing this mixed-methods approach, the survey aimed to provide a holistic understanding of how users engaged with the web application and their overall satisfaction with its features and functionalities. The combination of quantitative data from the UES and qualitative insights from open-ended questions that are in **Figure** 12, allowed for a comprehensive evaluation of the web application's effectiveness and user-friendliness. This feedback is invaluable for further refining and enhancing the educational web application, ensuring it meets the needs and expectations of its users.

7 Ethical Considerations and Legal Compliance

Ethical considerations and legal compliance were paramount throughout the development of the educational web application and the user testing process. These aspects were meticulously addressed to safeguard the rights and well-being of participants while adhering to legal regulations.

During the user testing phase, participants were recruited from the CSC1010H and CSC1011H courses at the University of Cape Town, and their participation was entirely voluntary, without any monetary compensation. Participants were provided with comprehensive information regarding the research purpose, their roles, the expected testing duration (maximum 45 minutes), and their right to withdraw consent at any point. Informed consent forms were presented to the students, and their signatures were required before participation. These consent forms outlined data collection and storage procedures, as well as the intended use of the collected data. To protect confidentiality, no personal details were recorded.

The development of the web application adhered to open-source software practices, with appropriate credit given to the original authors for any source code used. Additionally, the videos integrated into the web application were carefully selected to be royalty-free, ensuring full compliance with copyright regulations. These ethical and legal measures were diligently implemented to uphold transparency, safeguard participants' rights, and ensure adherence to applicable laws and regulations throughout the development and testing phases of the educational web application.

8 Results and Analysis

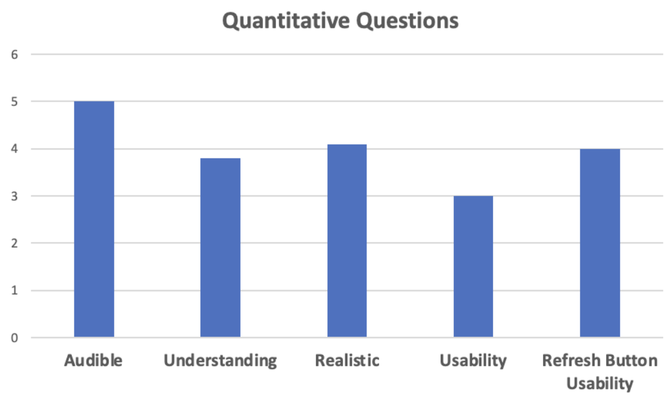


Figure 1. Graph for Quantitative questions

The quantitative feedback received from the first-year students who experienced the audio feature within the web application provides valuable insights into its effectiveness. In terms of audibility, the feature received a high rating of 5, indicating that the audio explanations were clear and easily understandable. However, when it comes to understanding, the rating was 3.3, suggesting that while the audio was audible, there may be room for improvement in terms of comprehensibility. The rating of 4 for the realism of the audio indicates that students found the audio explanations to be authentic and in line with their expectations. In assessing usability, the feature received a rating of 3, indicating that there may be aspects of the user interface that could be further optimized for a more user-friendly experience. Notably, the refresh button's usability received a high rating of 4, suggesting that students found it effective for controlling and replaying the audio explanations.

The qualitative feedback provided by the students regarding the audio feature in the web application sheds light on several important aspects of its usability and effectiveness.

8.1 Realism of the Audio Voice

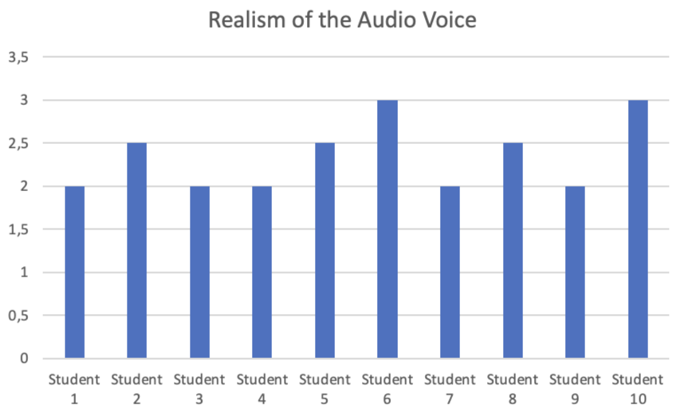


Figure 2. Realism of the Audio Voice(X-Axis(Students) and Y-Axis(Scale 1-5))

Many students expressed that they found the audio voice to be less realistic and more robotic in nature. This feedback highlights the importance of improving the naturalness of the audio voice to create a more immersive and engaging learning experience.

8.2 Ease of Navigation

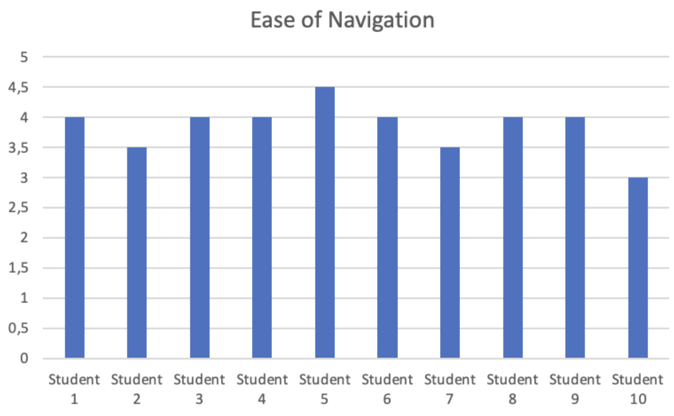


Figure 3. Ease of Navigation(X-Axis(Students) and Y-Axis(Scale 1-5))

Students generally found the audio feature to be user-friendly in terms of navigation. They appreciated the ability to rewind and replay the audio explanations multiple times. Additionally, the control over playback speed was considered a valuable feature, allowing students to adapt the audio to their preferred pace of learning, as shown in **Figure** 7. This feedback indicates that the navigation aspects of the audio feature are well-received.

8.3 Clarity of the Audio Refresh Button

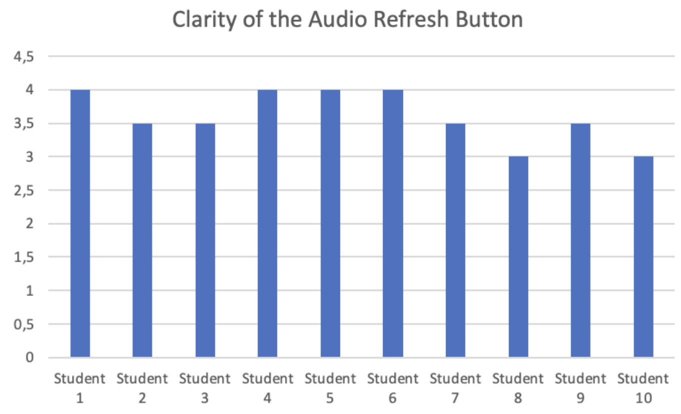


Figure 4. Clarity of the Audio Refresh Button(X-Axis(Students) and Y-Axis(Scale 1-5))

Students mentioned that the tutorial video effectively helped them understand the purpose and functionality of the audio refresh button. This suggests that providing clear instructional resources or tooltips can enhance users' comprehension of various features within the web application.

Overall, the qualitative feedback provides valuable insights into areas for improvement, particularly in enhancing the realism of the audio voice. Additionally, it highlights the success of user-friendly navigation features such as rewind, playback control, and the tutorial video in clarifying the functionality of the refresh button. These insights can guide further refinements to the audio feature, aiming to create a more immersive and intuitive learning experience for students.

9 Conclusion

In conclusion, this research has made significant advancements in addressing the challenge of student engagement in introductory programming courses. By exploring the integration of audio and video lesson features, coupled with reduced context switching, into a web application, we sought to understand their impact on user engagement. While the findings provide valuable insights into the potential benefits of such features, it's important to acknowledge that the results may not be entirely conclusive due to the limited number of participants involved in the study.

Nonetheless, the feedback from students was largely positive, with many expressing that the video lessons were valuable for following along while coding. Additionally, the audio feature received appreciation, particularly when students encountered difficulties in understanding the code they were inputting. These findings indicate that incorporating multimedia elements and reducing context switching within the learning environment can enhance student engagement and comprehension.

Moving forward, further research with a larger and more diverse participant pool could provide deeper insights into the effectiveness of these features. Additionally, continuous refinements to the audio and video components, including addressing the realism of the audio voice, can contribute to an even more enriching educational experience. Ultimately, this research serves as a promising step toward improving the learning outcomes of introductory programming students.

References

[1] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin* 37, 2 (2005), 103–106.

[2] Andres P Calitz, Jean H Greyling, and Margaret DM Cullen. 2014. South African industry ICT graduate skills requirements. *Southern African Computer Lecturers' Association (SACLA)* 1 (2014), 25–26.

[3] Douglas Crockford. 2006. Json (javascript object notation). *URL http://www. json. org* (2006).

[4] José Figueiredo and Francisco García-Peñalvo. 2021. Teaching and Learning Tools for Introductory Programming in University Courses. In *2021 International Symposium on Computers in Education (SIIE)*. 1–6. https://doi.org/10.1109/SIIE53363.2021.9583623

[5] 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*. 579–584.

[6] Nazir Hawi. 2010. Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education* 54, 4 (2010), 1127–1136.

[7] Bassey Isong. 2014. A Methodology for Teaching Computer Programming: first year students' perspective. *International journal of modern education and computer science* 6, 9 (2014), 15.

[8] Kris MY Law, Victor CS Lee, and Yuen-Tak Yu. 2010. Learning motivation in e-learning facilitated computer programming courses. *Computers & Education* 55, 1 (2010), 218–228.

[9] Sasha Nikolic, Montserrat Ros, and David B Hastie. 2018. Teaching programming in common first year engineering: discipline insights applying a flipped learning problem-solving approach. *Australasian Journal of Engineering Education* 23, 1 (2018), 3–14.

[10] Heather L O'Brien, Paul Cairns, and Mark Hall. 2018. A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form. *International Journal of Human-Computer Studies* 112 (2018), 28–39.

[11] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. 39–44.

[12] Stuart Zweben, Jodi Tims, and Yan Timanovsky. 2020. ACM-NDC study 2019—2020: eighth annual study of non-doctoral-granting departments in computing. *ACM Inroads* 11, 3 (2020), 26–37.

10 Appendix

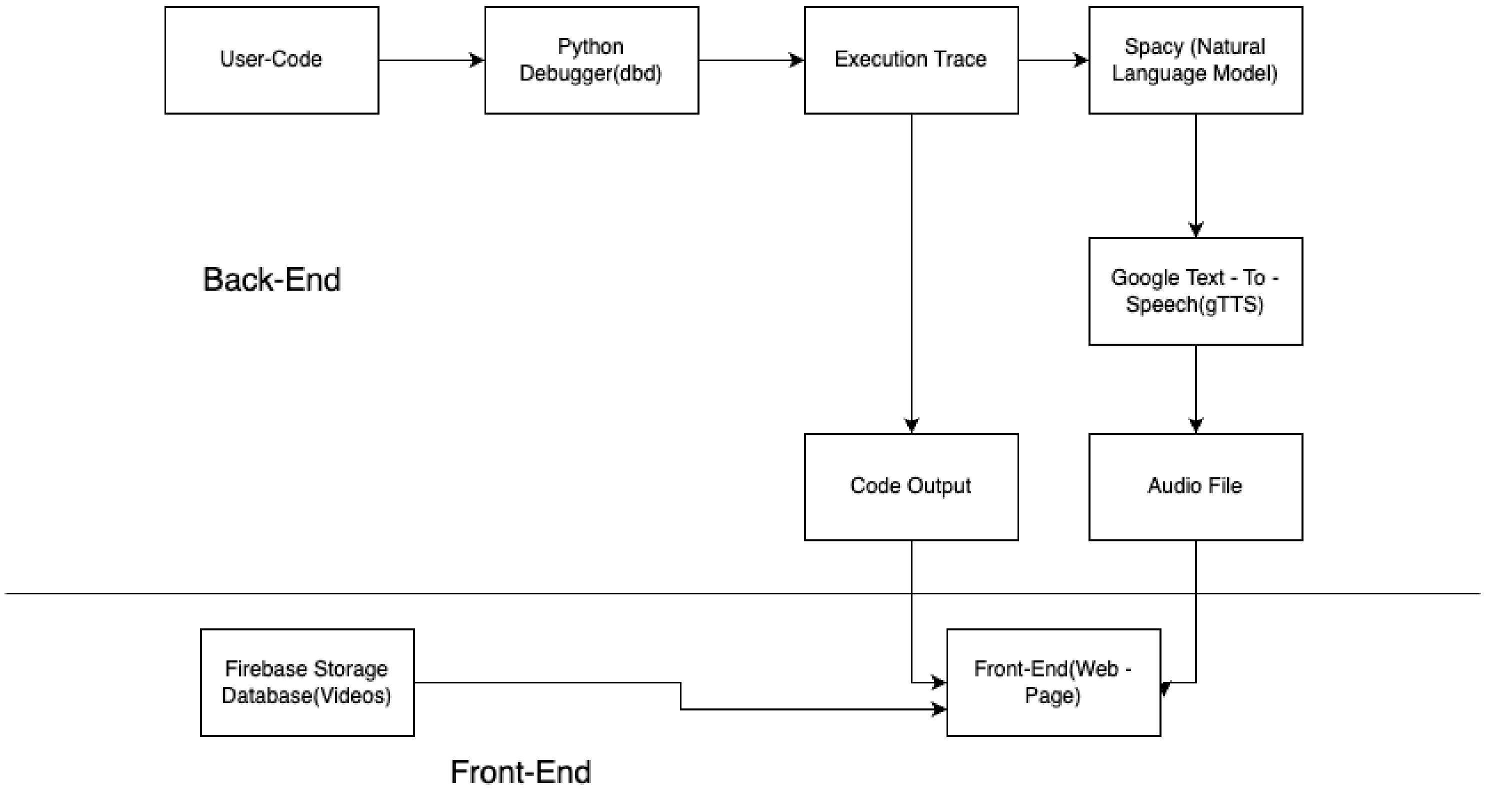


Figure 5. Working of the Web-Application

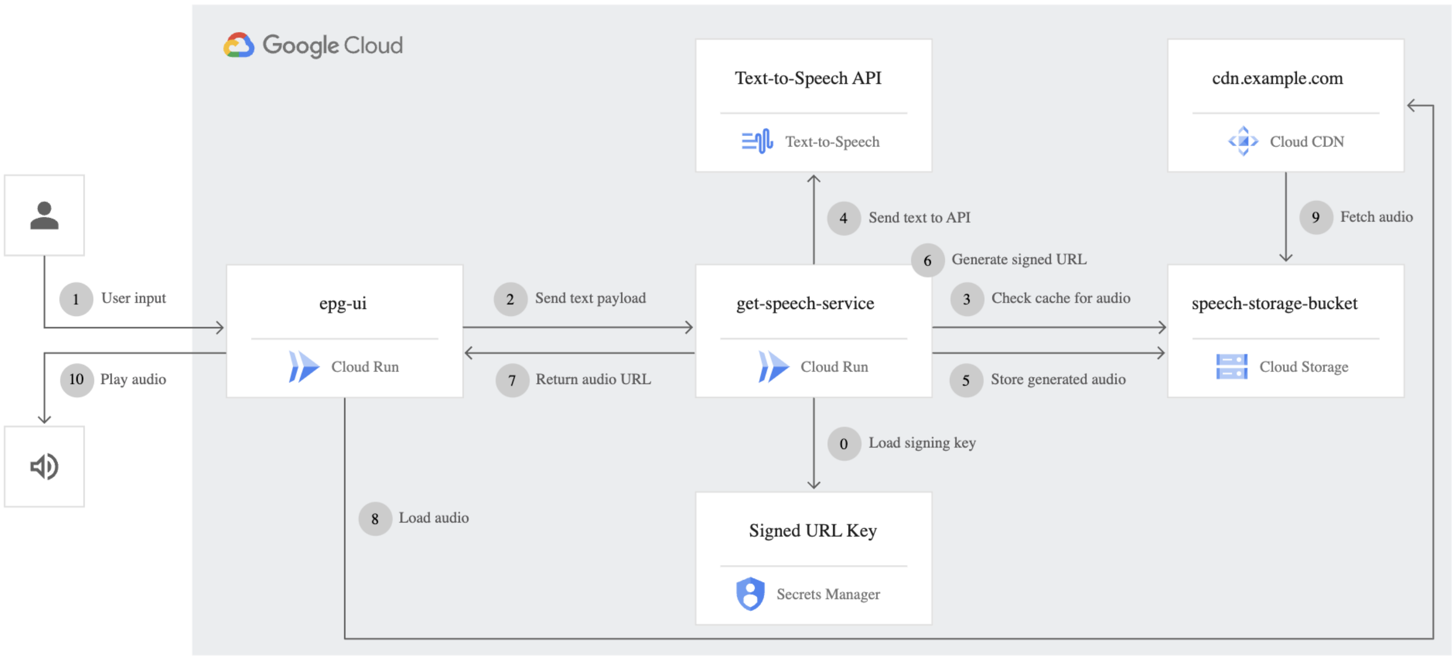


Figure 6. Google Text-To-Speech Architecture Diagram

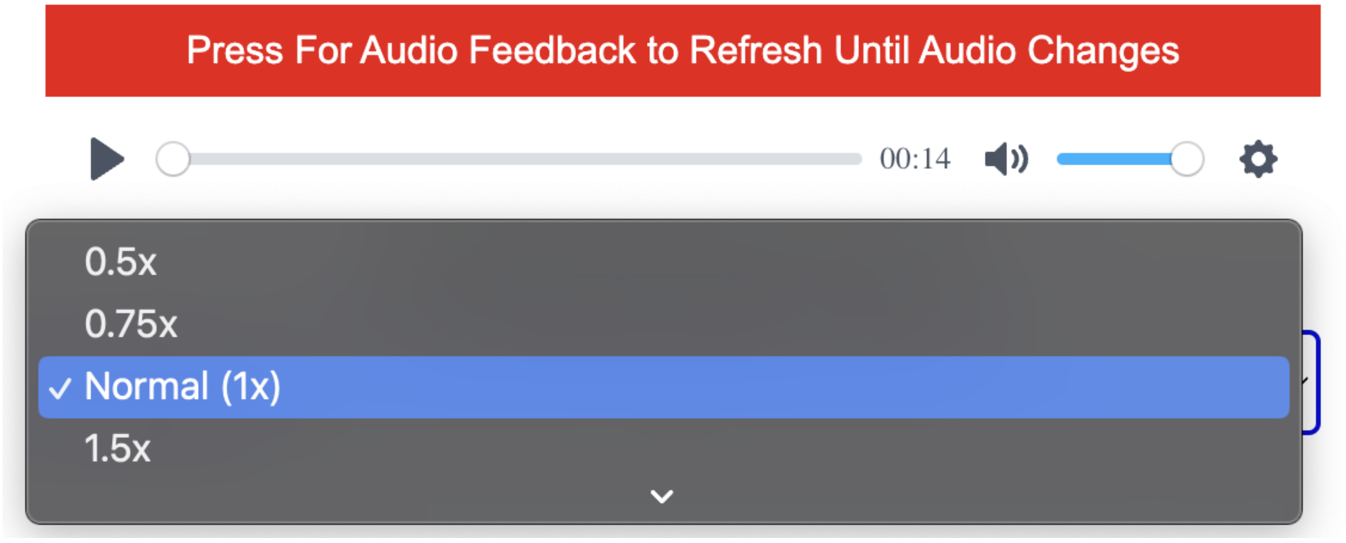


Figure 7. Audio Speed Selection

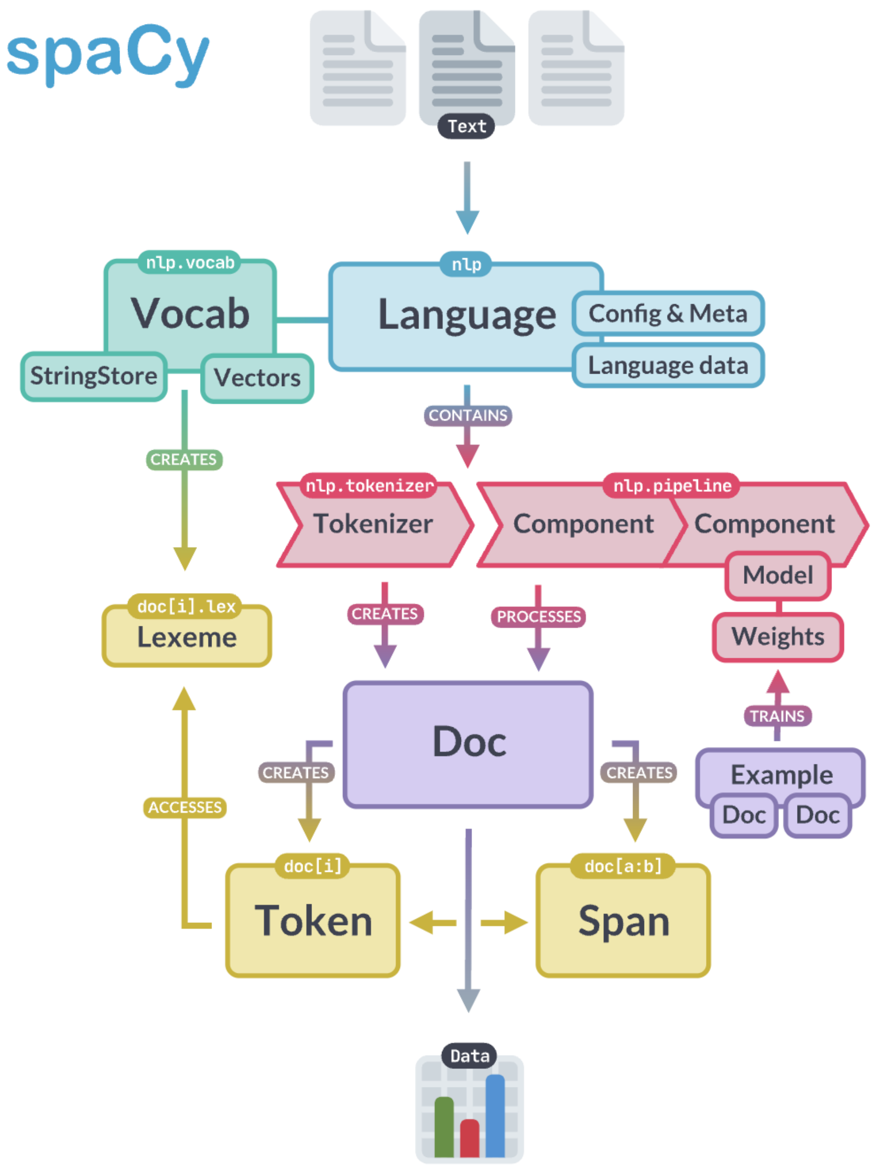


Figure 8. Spacy Architecture Diagram

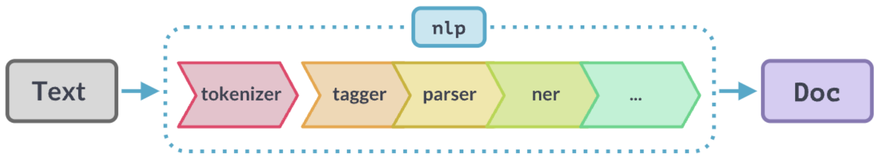
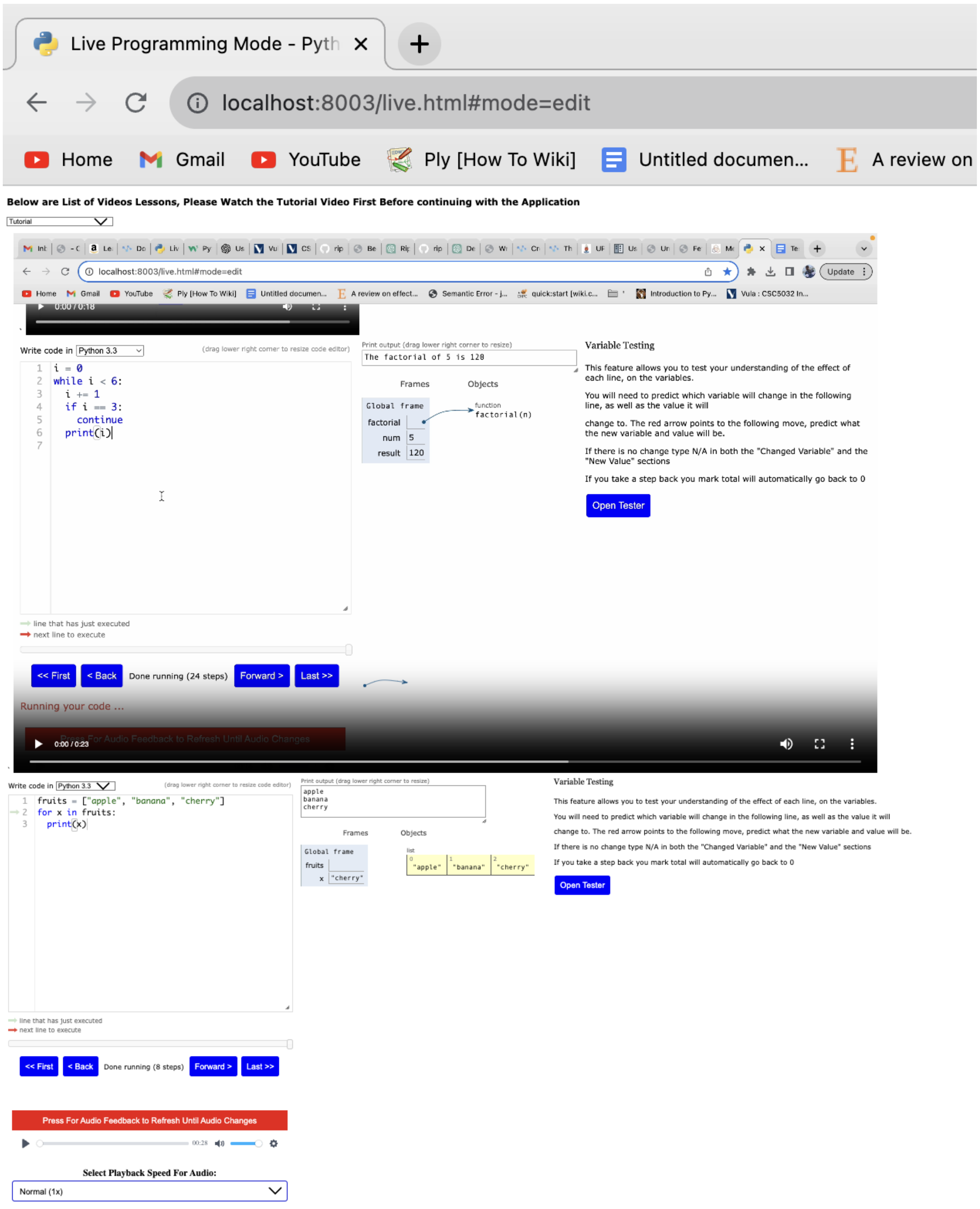


Figure 9. pipeline components



Below are List of Videos Lessons, Please Watch the Tutorial Video First Before continuing with the Application

```python
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
The factorial of 5 is 120
```

Global frame	
factorial	factorial(n)
num	5
result	120

Done running (24 steps)

Running your code ...

Press For Audio Feedback to Refresh Until Audio Changes

Variable Testing

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

You will need to predict which variable will change in the following line, as well as the value it will change to. The red arrow points to the following move, predict what the new variable and value will be.

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

If you take a step back you mark total will automatically go back to 0

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

Global frame	
fruits	list: 0 "apple", 1 "banana", 2 "cherry"
x	"cherry"

Done running (8 steps)

Variable Testing

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

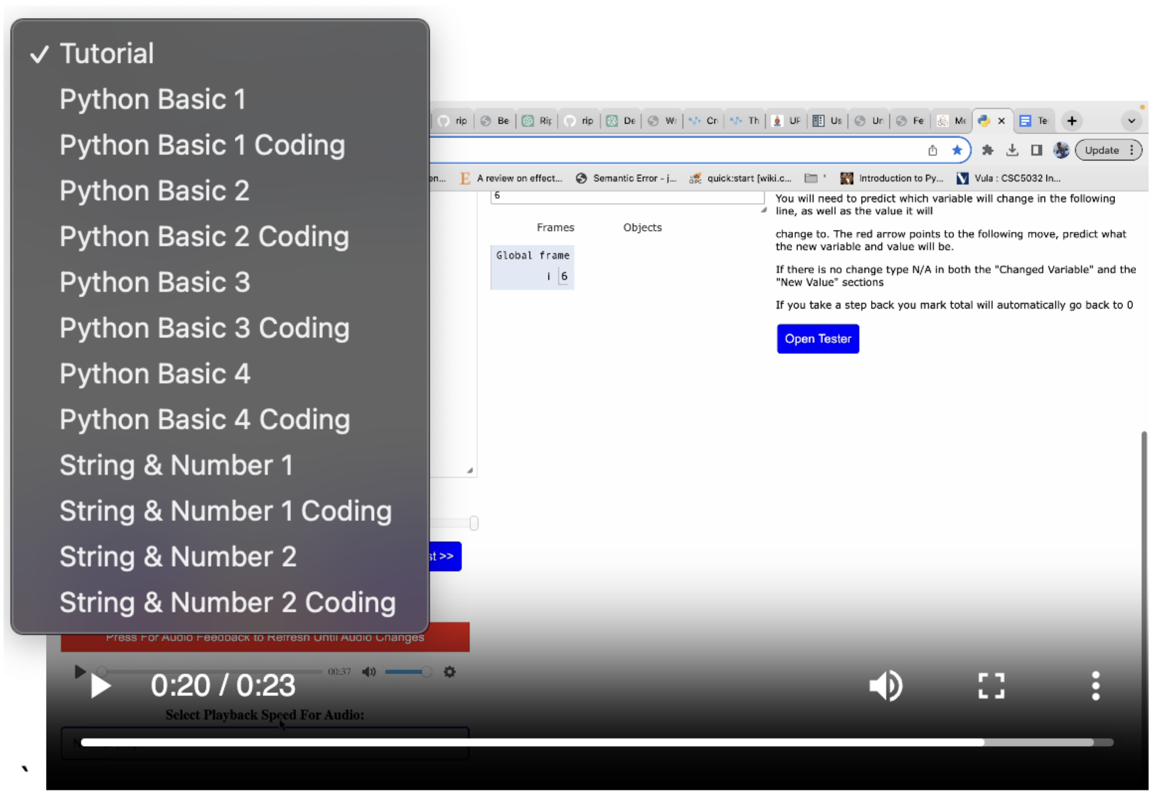
You will need to predict which variable will change in the following line, as well as the value it will change to. The red arrow points to the following move, predict what the new variable and value will be.

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

If you take a step back you mark total will automatically go back to 0

Press For Audio Feedback to Refresh Until Audio Changes

Figure 10. Context Switch



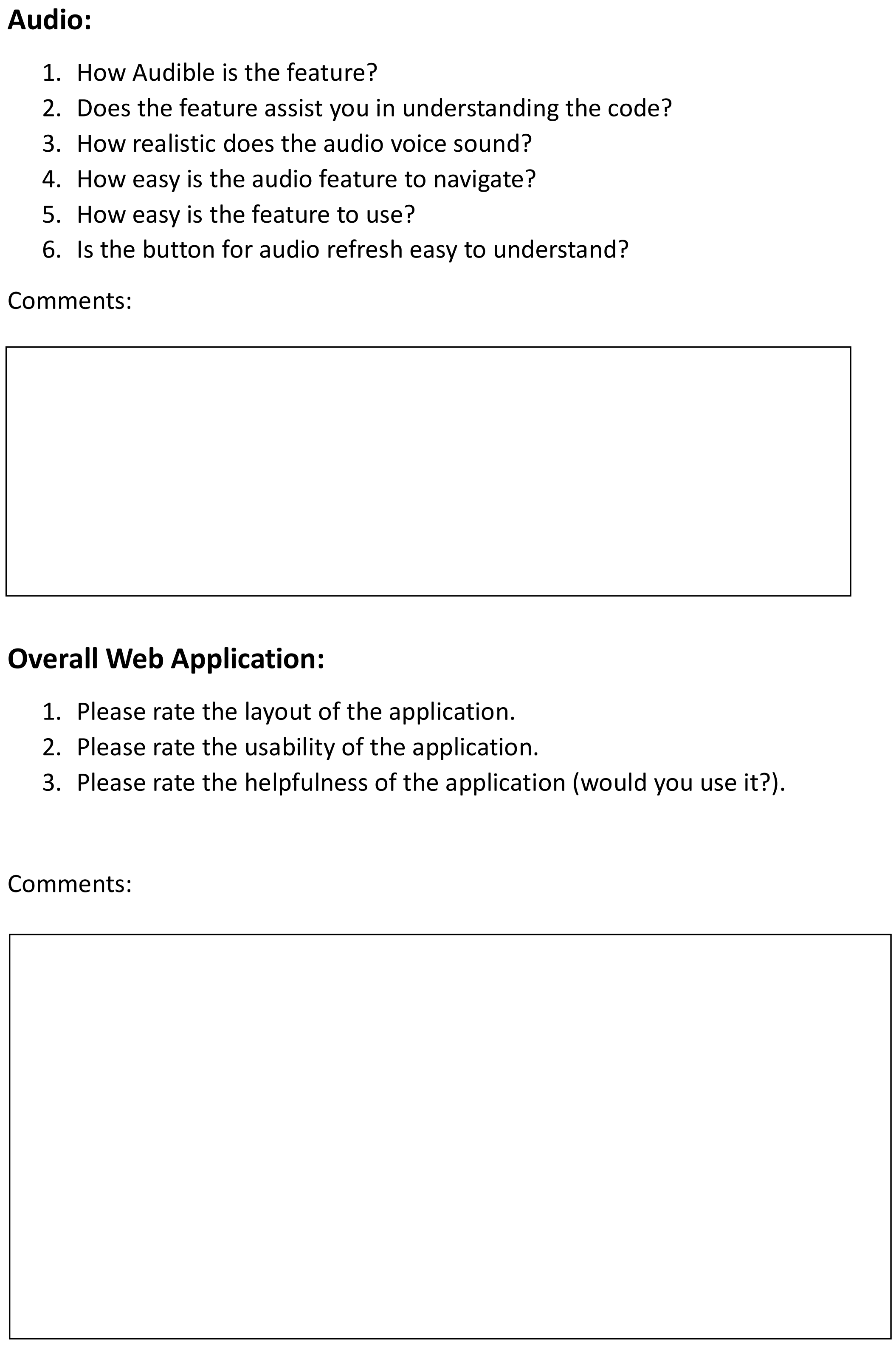
You will need to predict which variable will change in the following line, as well as the value it will change to. The red arrow points to the following move, predict what the new variable and value will be.

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

If you take a step back you mark total will automatically go back to 0

Figure 11. Videos Scroll Bar

Figure 12. Survey Questions



Audio:

1. How Audible is the feature?
2. Does the feature assist you in understanding the code?
3. How realistic does the audio voice sound?
4. How easy is the audio feature to navigate?
5. How easy is the feature to use?
6. Is the button for audio refresh easy to understand?

Comments:

Overall Web Application:

1. Please rate the layout of the application.
2. Please rate the usability of the application.
3. Please rate the helpfulness of the application (would you use it?).

Comments: