|  |  |
| --- | --- |
| The Coversheet | |
| Student Name  (unless anonymized) | Gowthami Siddi |
| Student Number  (as shown on student ID card): | 240200846 |
| Word Count / Pages / Duration / Other Limits: | Word count - 4257 |
| Attempt Number: | 1 |
| Date of Submission: | 19/12/2024 |

|  |  |
| --- | --- |
| I have read and understood the [Academic Misconduct statement](https://blog.yorksj.ac.uk/assessment/coversheet-statements/). | Tick to confirm ☒ |
| I have read and understood the [Generative Artificial Intelligence use statement](https://blog.yorksj.ac.uk/assessment/coversheet-statements/). | Tick to confirm ☒ |
| I am satisfied that I have met the Learning Outcomes of this assignment  (please check the Assignment Brief if you are unsure) | ​​​ Met ☒ |

|  |
| --- |
| Self-Assessment – If there are particular aspects of your assignment on which you would like feedback, please indicate below.  Optional for students |
| Suggested prompt questions-  How have you developed or progressed your learning in this work?  What do you feel is the strongest part of this submission?  What feedback would you give yourself?  What part(s) of this assignment are you still unsure about? |
|  |

|  |  |
| --- | --- |
| **Assessor’s Feedback** (may be delivered in line with the submission) | |
| Were the learning outcomes met? | Yes ☐ If not, what was not met: |
| Assessor’s response to the student’s submission, request for feedback and / or self-assessment (feedback): | |
| What specific actions should the student undertake to progress their learning? (feedforward): | |
| Please take this and other feedback to your next academic tutorial to plan your future work. | |

# 

# **Table of Contents**

[1. Introduction 1](#_heading=h.lnxbz9)

[1.1 Research Questions 2](#_heading=h.35nkun2)

[2. Project Plan 2](#_heading=h.1ksv4uv)

[2.1 Objectives 2](#_heading=h.44sinio)

[2.2 Timeline 3](#_heading=h.2jxsxqh)

[3. Literature Review 3](#_heading=h.z337ya)

[3.1 Existing Python Intelligence Tutors 4](#_heading=h.3j2qqm3)

[3.2 Challenges in Existing System 4](#_heading=h.1y810tw)

[3.3 Insights for Development 5](#_heading=h.4i7ojhp)

[4. Development of the Intelligent Tutoring System 6](#_heading=h.2xcytpi)

[4.1 Ontology Design 9](#_heading=h.1ci93xb)

[4.2 Interface Development using Python 1](#_heading=h.2bn6wsx)1

[4.3 Integration and Testing 1](#_heading=h.qsh70q)1

[4.4 Limitations : 1](#_heading=h.3as4poj)2

[5. Conclusion 13](#_heading=h.1pxezwc)

[6. References 1](#_heading=h.49x2ik5)5

# 

# 

# **List of Figures**

Fig-1: Gantt chart ……………………………………………………………………………………...3

Fig-2: Code……………………………………………………………………………………………...7

Fig-3: User Interface…………………………………………………………………………………....8

Fig-4: Ontology………………………………………………………………………………………….9

Fig-5: Ontology OWX Code…………………………………………………………………………...10

# **Intelligent Tutoring System for Python Programming**

## **Introduction**

Programming has become an indispensable skill, especially in the educational curriculum of computer science and also requires the need to demand effective tools for teaching programming languages. The following report focuses on the design and development of an Intelligent Python Tutor, a smart tutoring system specifically designed for fresh beginners into the world of learning Python. (Omankwu et al., 2021)

Python Tutor introduces basic concepts, such as syntax, control structures that use loops and simple data structures, the basic concepts of programming. What makes it different from Python Tutor is its approach: an interactive approach for learners to actively participate in coding exercises as users tackle problems in real-time and get immediate constructive feedback. Learning is reinforced and errors can immediately be corrected while making understanding deepen.

The platform is educational and fun as it integrates practice problem-solving with theory; thus, it is an interesting experience in learning. Python Tutor is more than just a coding tutor; it makes users confident by letting them learn how to be great at their own pace. The system helps people get rid of the intimidation that they face when programming and replaces it with curiosity and enthusiasm. (Yang et al., 2024)

The rapid advancements in educational technology have profoundly impacted programming education, especially with the integration of intelligent systems. These developments have revolutionized conventional teaching methodologies, fostering more personalized and effective learning experiences for students. The use of intelligent systems promotes adaptive learning environments, enabling educators to customize their strategies to address the unique needs of each student, ultimately improving educational outcomes in programming.

How can an Intelligent Tutoring System leveraging knowledge representation, dynamic feedback, and progress tracking improve the learning effectiveness, engagement, and adaptability of Python programming for beginners compared to existing platforms.

Python was selected as the primary programming language due to its ease of use, widespread popularity, and significant applicability across multiple fields, including data science, web development, and artificial intelligence.

What Python Tutor does is innovative through thoughtful design and hands-on methodology towards empowering even the novice programmer. This platform does not just transfer technical skills but makes the journey fulfilling and rewarding. By bridging the gap between theory and procedure, Python Tutor arms the users with deep foundational knowledge and skills with which a user would feel more confident in continuing his programming journey. (M. Wang, W. Wu and Y. Liang)

The primary aim of the system is to address the challenges faced by beginner programmers while providing a comprehensive and interactive learning environment.

### **1.1 Research Questions**

* How does an ITS influence the confidence and learning trajectory of beginner programmers compared to traditional teaching methods?
* What is the impact of real-time dynamic feedback on novice programmers' learning outcomes and problem-solving skills?
* How does progress tracking in an Intelligent Tutoring System improve engagement and learning effectiveness in programming education?

## **Project Plan**

### **2.1 Objectives**

* Design a knowledge representation system that facilitates the establishment of relationships between major programming concepts such as syntax, loops, conditional statements and context-aware hint and suggestion generation for learners.
* Organize interconnected ideas and entities to clearly demonstrate their relationships. Promote the integration and connection of information from various sources by utilizing standardized terminology.
* An Interactive User Interface with Python Libraries allows learners to test the code that gets real-time feedback for errors with explanations. Bui (2019)
* Provide dynamic error detection and feedback mechanisms that examine syntax and logic errors in the code and offer personalized hints and correction suggestions.
* Present instances and assignments that illustrate authentic programming situations to improve practical comprehension.
* Improve compatibility with widely used coding platforms and tools, including GitHub etc., to ensure smooth transitions between academic and professional settings.
* Integrate extensive accessibility features, including support for screen readers, various input methods, and customizable themes for the user interface.
* It is crucial to ensure that the system can support a growing number of users while preserving high performance levels and an enjoyable user experience.
* To overcome usability and accessibility challenges, design an intuitive, user-friendly interface that is friendly to beginners and compatible with assistive technologies.
* Enhance the debugging system to incorporate support for advanced concepts, including recursion, object-oriented programming, and algorithm optimization.
* Test the system's effectiveness through user testing, focusing on learning outcomes, engagement, and error correction capabilities.

### **2.2 Timeline**

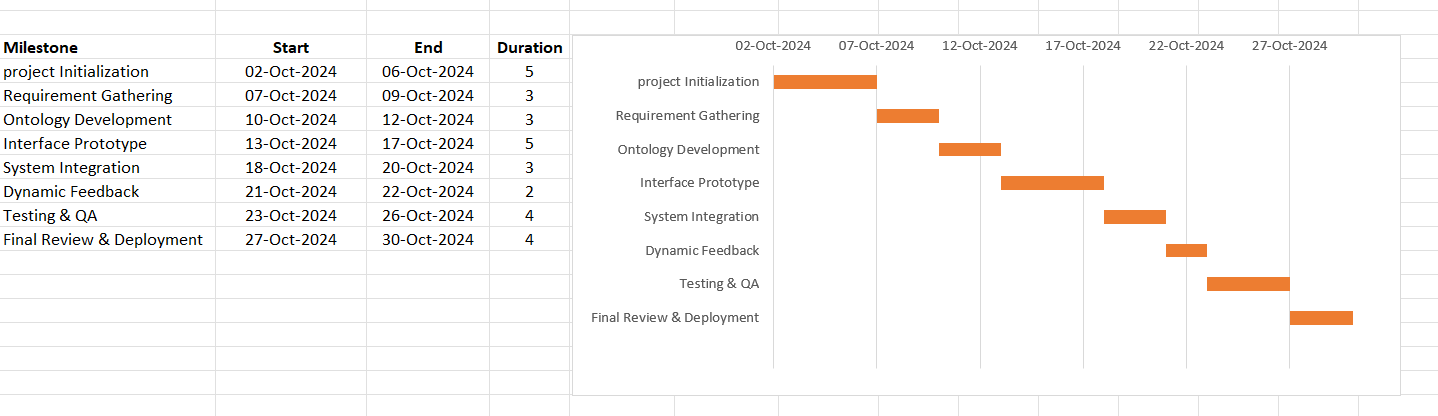


Fig-1 Timeline Gantt chart

## **Literature Review**

(Meng Wang, Wenjun Wu, and Yu Liang,2020) Proposed the concept, structure, and implementation of a novel intelligent tutoring system designed specifically for beginners in C and Python programming. They implemented the system by integrating features such as code classification, program error repair, and personalized knowledge tracing into an existing online programming practice platform. Their approach enhanced the platform's intelligence and significantly improved its ability to support students in learning programming effectively.

Priyanka Sharma and Mayuri Harkishan proposed an Intelligent Tutoring System (ITS) for online programming courses, focusing on user interaction and a subset of computer programming languages. Their design integrates intelligent algorithms, such as optimized ant colony, to enhance formative assessments and provide high-quality feedback. The system aims to offer an immersive learning experience, enabling students, particularly in Pacific regional campuses, to learn programming more effectively and efficiently compared to conventional online methods.

Omankwu Obinnaya Chinecherem, Osodeke Efe Charles, and Onwuzo Chioma Julia proposed the design of an Intelligent Tutoring System (ITS) for teaching Python, aimed at helping students learn the language easily and effectively. The system offers beginner-level lessons and exercises, adapting to individual student needs and progressing gradually from simple to complex concepts. By simulating a one-on-one teaching experience, the ITS saves time, allows students to learn at their own pace, and eliminates the fear of making mistakes. The evaluation of the tutor received excellent feedback from both students and teachers.

Zhiyu Fan and Ashish Dandekar from the National University of Singapore proposed an Intelligent Tutoring System (ITS) designed to provide automated feedback for first-year programming courses. Unlike existing automated grading systems, their ITS focuses on delivering customized error feedback to students. The system was incrementally developed as part of a multi-year Software Engineering course, where third-year students contribute to its improvement each year. This approach not only enhances the tutoring system but also allows its immediate deployment for first-year students, creating a practical and collaborative learning environment for all stakeholders.

### **3.1 Existing Python Intelligence Tutors**

* **Codecademy:** Codecademy provides a comprehensive way to learn Python, teaching things like basic syntax, loops, and Data structures. It has interactive exercises, so you keep practising as you go along. Not ideal is the fact that it doesn't do real-time debugging nor provide explicit ways of troubleshooting particular errors. Still, that's more of a minor annoyance than anything else. Still, it is a great resource for getting familiar with Python.
* **SoloLearn:** SoloLearn is a mobile-friendly learning platform offering gamified lessons for both beginners and experts. While engaging, it only allows limited feedback on predefined examples and doesn't dynamically change based on user mistakes.Chrysafiadi et al. (2022)
* **Python Tutor by Philip Guo:** It is good for exhibiting the execution of a code, visually, showing users how Python programs work line by line. There are no personalized hints; it's not about correcting wrong logical or syntactical errors.

### **3.2 Challenges in Existing System**

The existing systems teaching programming concepts do not possess adaptive features or styles that fit personal or individual learning habits.

The mechanisms for feedback in existing platforms are rather general and do not provide provisions for personalization. Error detection typically gets stuck at the surface level without helping users realize where exactly they made a mistake. Chrysafiadi et al. (2022).

### **3.3 Insights for Development**

To address these gaps, Python Tutor (ITS) aims to combine the strengths of existing platforms while introducing features such as Matellio

* **Dynamic Error Feedback:** Analyze the code for syntax and logic errors in submitted user code and provide targeted hints and solutions.
* **Enhanced Accessibility:** Incorporate functionalities such as compatibility with screen readers, customizable themes, and various input methods.
* **Real – Time Scenarios:** Integrate experiential learning opportunities to bridge the gap between theoretical knowledge and practical application.
* **Knowledge Representation:** Connect concepts such as loops, conditionals, and data structures so that the system can even offer context-aware advice. Develop relationships among programming concepts, such as loops and conditionals, to provide context-sensitive support.
* **Access Without Internet:** Explore options to ensure the system remains functional without depending on continuous internet access.
* **Tracking:** Track performance through exercising and adjust exercises to fit weak areas so that a personalized learning trajectory is ensured.
* **Better Approach:** Create functionalities that address various learning preferences, including visual, auditory, and kinesthetic methods.
* **Multi-language Support:** Programming learners originate from a variety of linguistic backgrounds; therefore, providing multilingual support can significantly improve accessibility.
* **Efficiency and Expandability:** Enhance the system to deliver immediate feedback while supporting a growing number of users.
* **Security and Privacy of Data:** It is essential to manage student information with care, particularly within educational environments.
* **Collaborative Roles:** Create materials for collaborative initiatives and peer evaluations to enhance teamwork.
* **Advanced Topics Support:** As students advance in their educational endeavors, they require resources and support to adeptly navigate the growing complexity of subjects such as algorithms, data structures, machine learning, and web development.

The incorporation of these insights can enhance the effectiveness, engagement, and scalability of the learning experience in Python programming within the ITS framework.

An efficient Intelligent Tutoring System designed for Python programming must be flexible, offer tailored feedback, promote collaborative learning, and utilize a range of educational strategies. Emphasizing error identification, gamification, sophisticated debugging tools, and compatibility with real-world coding environments will enable the system to assist learners across all levels of programming expertise.

Furthermore, incorporating real-time data analytics, scalability, and robust security measures will guarantee that the system remains resilient and capable of accommodating an expanding and varied user demographic. By leveraging these insights, developers can foster a more engaging, effective, and efficient educational experience for students of Python programming. Python Tutor will bridge the void between stagnant knowledge resources and the demand for real-time, interactive feedback, creating a complete tool for learning Python programming effectively. Chrysafiadi et al. (2022)

## **Development of the Intelligent Tutoring System**

An Intelligent Tutoring System ITS focused-on teaching basic programming skills using Python to new programmers emphasizes the following core topics: syntax, loops, conditionals, and data structures. It's focused on the aim of providing a learning experience by giving learners the ability to work through exercises and use dynamic feedback in receiving answers.

The document delineates the development methodology, objectives, and characteristics of an Intelligent Tutoring System (ITS) designed to instruct novice learners in Python programming. Essential components encompass research inquiries, objectives, system architecture, ontology, interface design, and evaluation. Additionally, it discusses particular limitations, including an emphasis on basic concepts and the difficulties associated with providing real-time feedback.

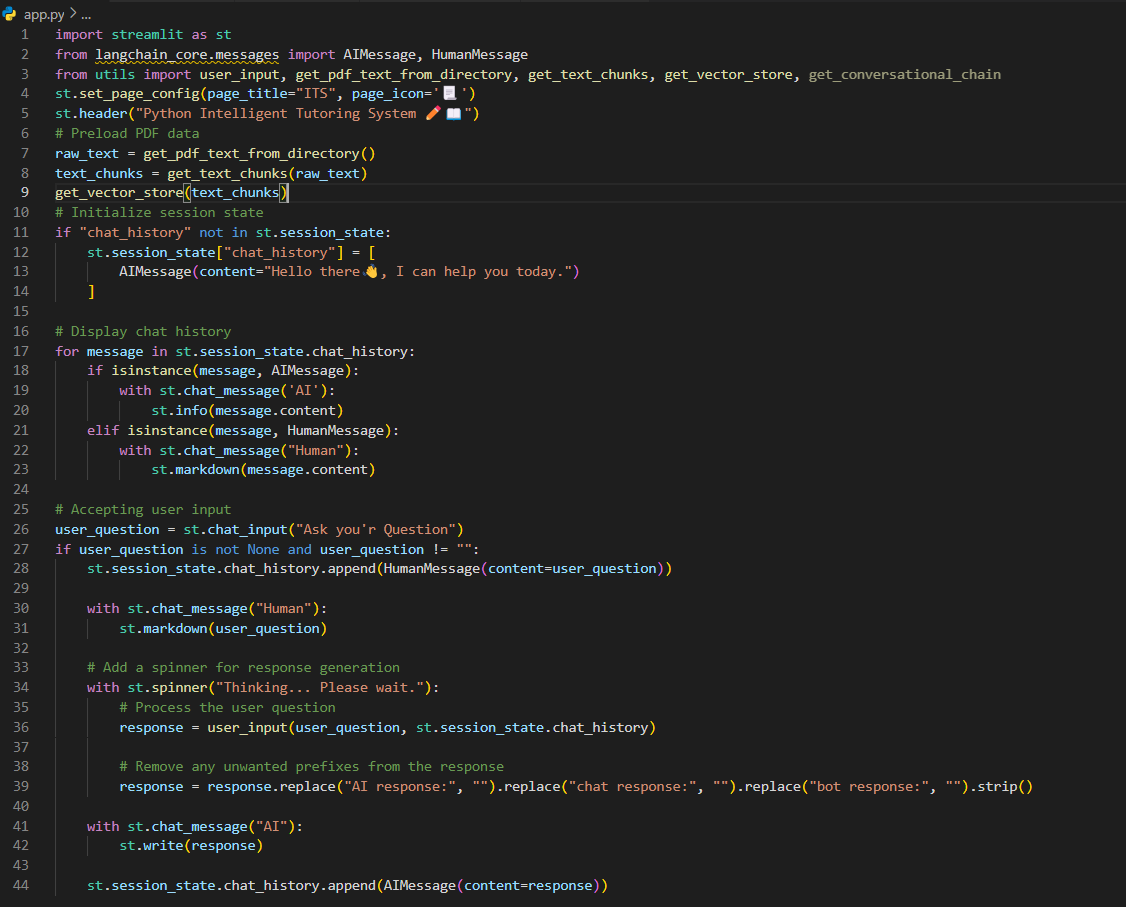


Fig-2 Code of the web UI, Which is developed using Streamlit



Fig-3 Web Application UI

### **4.1 Ontology Design**

**Objective:** Define relationships between programming concepts such as loops, syntax rules

**Process**:

* To create the ontology with ITS as the root class and visualize it using OntoGraf, I followed a step-by-step process in Protege. First, I opened Protege and navigated to the Classes tab. I right-clicked on the default root class, Thing, and created a new subclass named ITS, which became the root of my ontology. I then added child classes under ITS, such as System Features, User and Configuration. Each of these classes had additional subclasses; for example, under System Features, I added User\_Interaction, Text\_Processing\_for\_model\_Training ,Embedding\_Management and Conversational\_System, and under User I added Feedback and Query\_processing. Under Configuration, I added Environment\_setup and API\_Key\_Management.
* Next, I defined relationships between these classes by creating object properties in the Object Properties tab. I linked ITS to System Features using a property called hasFeature, and similarly, I linked ITS to User Configuration with a property called hasConfiguration. This step ensured that all classes were connected and properly organized within the ontology.
* After structuring the classes and properties, I used OntoGraf for visualization. I accessed OntoGraf from the Window → Views → Ontology Views → OntoGraf menu and focused on ITS. By right-clicking on ITS in OntoGraf, I selected Show Neighbors to display its child classes. To make the visualization clear and hierarchical, I applied the Tree Layout option.
* Finally, once the graph was set up as desired, I exported it as a PNG image using OntoGraf's Export feature. This provided a clear, visual representation of my ontology with ITS at the root, making it easy to understand and share.

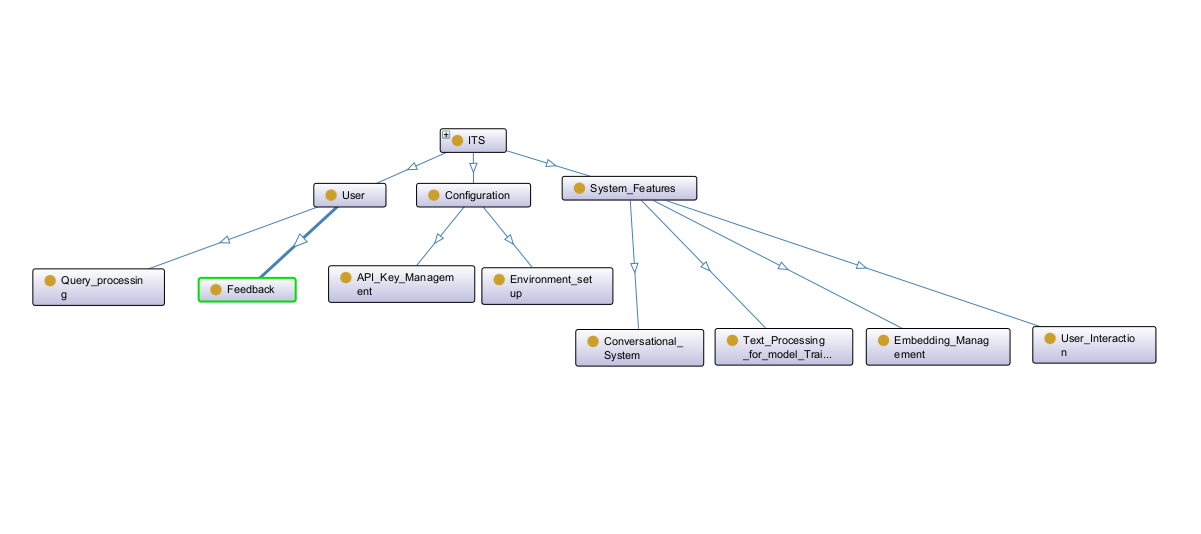
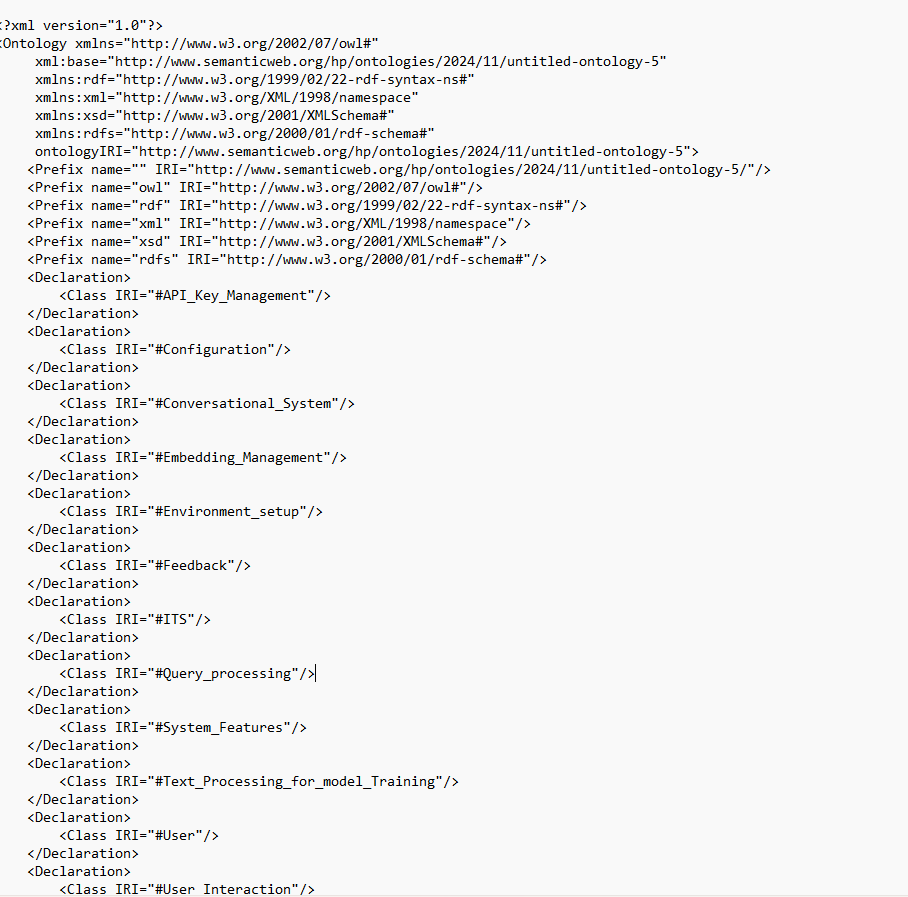


Fig-4 Ontology



### 

Fig-5 Ontology OWX Code

### **4.2 Interface Development using Python**

**Objective:** Develop an interactive learning interface where the student inputs their code and receives feedback. Instant Evaluation: Users are able to enter Python code, and the system delivers immediate feedback regarding any errors or successful execution.

User-Centric Design: Created using Streamlit to ensure an intuitive and accessible experience for beginners.

Timely Feedback: Distinguishes among syntax errors, runtime errors, and logical warnings.

**Process:**

* A graphical interface was designed by making use of the streamlit library of Python.
* Features like real-time code analysis, error detection, and suggestions were designed. For instance, a wrong for loop has been designed as an example. A user is faced with hints pertaining to the Loop class of the ontology. Ajie (n.d.)
* Employ ontology-driven error categorization to deliver tailored feedback. Identify specific error regions and offer recommendations in straightforward and comprehensible language.
* Incorporate tools designed for code visualization, provide strategies for debugging, and offer practical exercises. Additionally, introduce gamification components to enhance the learning experience.
* Develop intuitive and accessible interfaces tailored for beginners, employing libraries like Streamlit or PyQt. Incorporate features such as customizable themes, adjustable font sizes, and navigation support.
* The interface should deliver immediate feedback regarding user input, including the detection of both syntax and logical errors as they occur.
* Enhance the interface to support an expanding user base while ensuring peak performance standards are upheld.
* Incorporate functionalities that track user advancement, tailor exercises to address particular difficulties, and keep a comprehensive record of educational accomplishments.

## **4.3 Integration and Testing**

* **Integration**: The interface and the ontology are both allied so that the learning content and feedback mechanisms can work there with no hassle. Utilize Python libraries to manage queries that are based on ontologies. User engagement within the interface, encompassing code submissions, must align with the relevant ontology concepts to facilitate the efficient and dynamic gathering of feedback or recommendations. Conduct API-level testing to verify the precise management of inputs and outputs between the interface and the ontology. It is essential for all educational resources, including interactive error detection and feedback systems, to integrate smoothly across the various components.
* **Testing**: Tested with sample exercises to check functionality and error-free working. So, for testing we can follow the procedure like:
  + **Unit Testing**: Test the components individually by using frameworks like unittest etc.
  + **Integration Testing**: Evaluate the integrated system to determine the harmonious operation of the interface and ontology elements. This evaluation should encompass the following aspects:

- Accuracy of code analysis.

- Retrieval of feedback in alignment with the ontology.

- Responsiveness and usability of the user interface.

* + **System Testing**: Implement comprehensive testing throughout the entire process by utilizing authentic user scenarios. Assess the effectiveness of the learning experience by integrating practice exercises and ensuring that the feedback provided is both precise and constructive.
  + **Usability Testing**: Collect user feedback to enhance the interface design and refine error messages. Utilize tools such as Selenium to replicate user interactions for automated usability testing.
  + **Performance Testing**: Evaluate the system's performance under different load conditions to confirm its capacity to accommodate multiple users simultaneously without considerable delays. Emphasize the significance of delivering real-time feedback and the response times associated with ontology queries.

## **4.4 Limitations:**

* It only focuses on fundamental concepts of syntax, loops, and conditionals while excluding further advanced subjects like object-oriented programming, recursion, or even the data science library. Basic syntax and simple logic errors will be handled with less success with advanced debugging challenges, for instance, multi-threading problems, and algorithm efficiency issues. as well as providing a near-IDE environment would complete its usability.
* The material presented herein is limited to basic programming principles, encompassing syntax, loops, and conditionals. It does not address more complex subjects such as object-oriented programming, recursion, or data science libraries.
* While basic syntax and straightforward logical errors are addressed adequately, there is a lack of sufficient assistance for more complex debugging issues, including multi-threading complications or concerns related to algorithm efficiency.
* As the user base expands or the complexity of code submissions increases, there may be a lag in delivering real-time feedback.
* Acknowledge the existing limitations while suggesting practical solutions to maintain credibility.
* The system fails to sufficiently address the needs of auditory and kinesthetic learners and does not provide support for specific learning preferences.
* It may take time to process real-time feedback due to the increasing user base or large and complex code submissions.
* Not very adaptable to auditory or kinesthetic learners or special learning preferences. - Highly dependent on accuracy and completeness, which may result in gaps or incorrect feedback if the ontology contains errors.
* Doesn't accommodate collaborative learning activities such as peer code reviews, group projects, or shared exercises.
* Not Python-only: the site restricts learners to Python as a language; there isn't any provision for allowing learners to learn other languages.
* It lacks complete accessibility features for disabled users, such as screen readers or alternative input methods. It is inflexible and not adaptable to unconventional coding styles or non-standard inputs. The structure of foundational lessons is fixed and sometimes becomes repetitive or unchallenging for learners who already have some experience. Exercises are mostly theoretical with limited real-world programming scenarios and are, therefore, less practical.
* It requires an active internet connection, which limits its usability for learners in areas with poor connectivity.
* Dynamic feedback processing requires a lot of computational resources, which slows performance in large-scale or resource-constrained environments.

## **Conclusion**

ITSs are considered transformative educational tools as they can personalize feedback and adaptive learning environments that can improve the learning process. Such systems use technologies like artificial intelligence, natural language processing, and ontology-based knowledge representation to facilitate adaptation of the learning environment to meet individual needs, thereby being effective for all types of learners.

The document underscores the profound influence that Intelligent Tutoring Systems (ITS) can exert on the educational landscape, particularly within programming disciplines. These systems:

Provide tailored feedback and individualized learning pathways that address the distinct requirements of each student. Boost academic achievement by offering contextualized error correction and systematically guiding learners through problem-solving strategies. Integrate features such as immediate feedback, progress tracking, and gamification to inspire students and enhance their self-assurance. Enhance the overall educational experience by delivering prompt, constructive feedback, which fosters a deeper understanding of the subject matter and mitigates frustration. Demonstrate adaptability, with potential uses across various educational domains, thereby cultivating skills essential for lifelong learning. ITS platforms are acknowledged as innovative tools for creating effective, engaging, and accessible educational experiences, particularly in the field of programming education.

Highlight the opportunities for practical applications, which involve equipping students to tackle coding challenges that align with industry standards or collaborating with professional development platforms such as GitHub. Investigate the possibilities presented by emerging technologies, including machine learning, natural language processing, and adaptive learning algorithms, to enhance the functionalities of Intelligent Tutoring Systems.

ITSs, therefore, play a major role in guiding users through a step-by-step problem-solving process within programming education. With its emphasis on syntax error, logic error, or errors in the design of a program, it not only corrects the mistakes but explains and gives context-aware solutions, which ensures that not only do learners correct their mistakes but also understand what the concept is.Ijcsis and Koroveshi (2021)

Moreover, ITSs learn dynamically to the pace of each learner's ability and learning style. They can, for example, determine weaknesses, such as issues with loops or conditionals, and provide specific exercises on these topics to improve knowledge of these concepts. Such an adaptive characteristic allows for a constant, continuous progression without being based on the "one-size-fits-all" method applied by traditional teaching approaches.Ijcsis and Koroveshi (2021)

The other significant advantage is that ITSs provide real-time feedback. It is frustrating for a beginner when errors are marked without any explanation. ITSs reduce this by providing immediate, constructive feedback, often accompanied by visualizations or hints, which will help learners correct their mistakes on the spot and improve their confidence in their own abilities. (*AASCIT - Journal - HTML*, n.d.)

ITSs also promote engagement by involving interactive and gamified elements. The existence of features such as tracking, achievements, and rewards facilitates sustained participation, thereby making learning an enjoyable and motivating process. When integrated with user-friendly interfaces and visualization tools, it makes complex topics accessible even to a complete novice.

ITSs also have applications in mathematics, science, and language learning, thus showing their versatility and potential for revolutionizing education. Through the integration of cutting-edge AI technologies with pedagogical principles, ITSs not only enhance knowledge acquisition but also foster critical thinking, problem-solving, and lifelong learning skills. (*AASCIT - Journal - HTML*, n.d.)

**Future Scope:** Highlight potential enhancements, such as the incorporation of advanced programming principles, the ability to accommodate multiple programming languages, or the integration of collaborative learning features like peer review systems.

## **References**

Omankwu, O. C., Osodeke, E. C., Onwuzo, C. J., Okoro, U. B., Department of Computer Science, College of Physical and Applied Sciences, & Michael Okpara University of Agriculture. (2021). An intelligent tutoring system for learning Python. In *International Journal of Scientific & Engineering Research* (Vol. 12, Issue 2, pp. 254–257). <https://www.ijser.org/researchpaper/An-Intelligent-Tutoring-System-for-Learning-Python.pdf>

Yang, A. C., Lin, J., Lin, C., & Ogata, H. (2024). Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education. *Computers and Education: Artificial Intelligence*, *7*, 100309. <https://doi.org/10.1016/j.caeai.2024.100309>

M. Wang, W. Wu and Y. Liang, "A Novel Intelligent Tutoring System For Learning Programming," 2020 International Conference on Development and Application Systems (DAS), Suceava, Romania, 2020, pp. 162-168, doi: 10.1109/DAS49615.2020.9108925.

Huang, Y., Brusilovsky, P., Guerra, J., Koedinger, K., & Schunn, C. (2023). Supporting skill integration in an intelligent tutoring system for code tracing. *Journal of Computer Assisted Learning*, *39*(2), 477-500. <https://doi.org/10.1111/jcal.12757>

Bui, T. H. (2019). Analysis Of The Interoperability Of Intelligent Tutoring Systems For Programming: An Aspect Of Programming Exercises. In *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH* (Vol. 8, Issue 10, pp. 655–656) [Journal-article]. <https://www.ijstr.org/final-print/oct2019/Analysis-Of-The-Interoperability-Of-Intelligent-Tutoring-Systems-For-Programming-An-Aspect-Of-Programming-Exercises.pdf>

Menor, J. V. (2023). Design, development and effectiveness of an intelligent tutoring system using neural network. *AIP Conference Proceedings*, *2683*, 030011. <https://doi.org/10.1063/5.0125246>

Stathaki, A., Kondylakis, H., Marakakis, E., & Kalogerakis, M. (2018). i-Prolog: A Web-Based Intelligent Tutoring System for Learning Prolog. In *Springer eBooks* (pp. 337–346). <https://doi.org/10.1007/978-3-319-76908-0_32>

Chrysafiadi, K., Virvou, M., Tsihrintzis, G. A., & Hatzilygeroudis, I. (2022). Evaluating the user’s experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in Greece. *Education and Information Technologies*, *28*(6), 6453–6483. <https://doi.org/10.1007/s10639-022-11444-3>

Ajie, G. S. (n.d.). *Artificial intelligence in intelligent tutoring systems toward sustainable education\_ a systematic review*. Scribd. <https://www.scribd.com/document/697365685/Artificial-intelligence-in-intelligent-tutoring-systems-toward-sustainable-education-a-systematic-review>

Ijcsis, J. O. C. S., & Koroveshi, J. (2021). TRAINING AN INTELLIGENT TUTORING SYSTEM USING REINFORCEMENT LEARNING. *www.academia.edu*. <https://www.academia.edu/45675697/TRAINING_AN_INTELLIGENT_TUTORING_SYSTEM_USING_REINFORCEMENT_LEARNING>

*AASCIT - Journal - HTML*. (n.d.). <http://article.aascit.org/file/html/9401214.html>

M. Wang, W. Wu and Y. Liang, "A Novel Intelligent Tutoring System For Learning

Programming," 2020 International Conference on Development and Application

Systems (DAS), Suceava, Romania, 2020, pp. 162-168, doi:

10.1109/DAS49615.2020.9108925.

**GitHub Repository link:** <https://github.com/GowthamiSiddi/AI-ITS.git>