SECURE LOCAL GENERATIVE AI CODING ASSISTANT: A CDIO APPROACH

Zineb Hidila, Mohamed Tabaa.

Multidisciplinary Research and Innovation Laboratory (LPRI)
Moroccan School of Engineering Sciences (EMSI)

ABSTRACT

Generative Artificial Intelligence, referred to as GEN AI, is a buzzword that has gained immense popularity in recent years. The progress in generative AI has given rise to many applications in the domain of education. We have seen many of those models as coding assistants that enhance students' learning experience in introductory programming courses such as Python. In this paper, we present a new tool as a coding assistant that integrates the strengths of these models in computing education, showing how a CDIO approach can be employed to develop secure and effective AI-powered coding assistants that support novice learners while preserving academic integrity and ethics involved in such tools. This paper discusses the implementation of a containerized solution utilizing generative AI to provide accessible, secure, and efficient offline development tools. We have developed an AI code assistant that works offline, ensuring better privacy for students and learners based on the **qwen2.5-coder:0.5b** model and features a user-friendly Streamlit interface, all within a Docker container for offline execution to facilitate easy deployment and scalability and ensure security. This tool aligns with many CDIO standards to ensure a better educational experience both for learners and educators, which will be discussed in depth. Afterward, we will present the ethical implications of the implementation and then present the different components of the tool, to finally discuss in depth what can be improved for such solutions.

KEYWORDS

GENAI, CDIO, GENERATIVE AI, CODING ASSISTANT, STANDARDS: 2, 3, 5, 8, 11

INTRODUCTION

Ensuring secure tools for education remains one of the priorities in this era of AI. Natural language models have gained a lot of attention lately (the Chatgpt example Tan, 2023) not only for the usage rates worldwide, but also for how it switched the educational framework within the role of educators. Initially, irresponsible handling changed the way students deal with the information given, ignoring how those models can sometimes falsify answers, which is a wrong starting ground when learning a new subject. In addition, personal data are also collected, making educators responsible for structuring and defining how students can access that information.

In the era of artificial intelligence, we observe that many students lack technical depth and critical thinking aspects from the obsessive usage rate of AI tools such as generative AI. For this reason, it is crucial to develop innovative tools that both guide and challenge students, ensuring they remain actively involved as the ”human-in-the-loop” when utilizing AI.(Mollick and Mollick, 2023).

Researchers have suggested various approaches to address this issue, including the design of collaborative AI algorithms that leverage interpretable structures like knowledge graphs and decision trees (Agostinelli et al., 2021), but the simplest yet effective one is to implement AI agents that run locally and offline following ethics guidelines and providing students with a real-time learning experience. In this paper, we propose a CDIO framework-based tool that offers a structured way to construct a secure local AI coding assistant for students. This tool is capable to generate code, quizzes, and providing recommendations to students in a controlled environment, allowing them to maintain their agency and problem-solving skills (Agostinelli et al., 2021).

The CDIO framework **(CDIO Initiative, 2021)** is a well-established approach for both engineering education, which can be adapted to develop a secure local AI coding assistant. The key steps in our proposed Framework are:

• **Conceive**: by identifying the essential objectives and requirements for the secure local AI coding assistant, such as ensuring student activity, encouraging critical thinking, while preserving the ethical AI principles

• **Design**: by developing the architecture and core functionalities of the assistant, including the AI model, algorithms, knowledge representation, and the user interface.

• **Implement**: Creating the coding assistant as an extra learning tool and combining the designed components while respecting security measures.

• **Operate**: Deploy the assistant in the classroom setting, providing training and support to both students and teachers.

The paper is structured as follows, we start by outlining the CDIO standards with the coding assistant, by explaining the impact, the ethical implications and the relevance of the tool. Then we describe in detail the **HelloPy** coding assistant, the components, features and the study that was performed in an engineering school to assess students through the different expertise level while conforming with the syllabus of the python course. Then we display different examples of prompts and responses. Finally, we discuss and conclude the results of the study through students’ assessment.

ALIGNMENT OF AI CODING ASSISTANT DEVELOPMENT WITH CDIO STANDARDS:

Creating an AI coding assistant serves as an example for appropriate learning practices and an effective teaching tool, which are aligned with the CDIO framework. The necessity of defined learning outcomes that cover many skills described in Standard 2 aligns with the code assistant by creating adaptable activities that encourage disciplinary, interpersonal, and personal skills while boosting critical thinking, problem-solving, and cooperation

Moreover, the creation of the tool projects the interdisciplinary aspect which is highlighted in the third standard by integrating computer sciences, machine learning, deep learning and product building for an easy safe to use tool. On the other hand, the fifth standard is a projection of the life cycle of the creation of the coding assistant by following the cycle of **conceive-design-implement-operate** (CDIO) model. The educational challenges involved in this process come from real needs in education, such as reducing instructor's workload, generating engaging exercises or projecting real-world scenarios that are refined through the last phase of operating through performance metrics and feedback from students. Moving to Integrated learning experiences, which is standard number seven, is mirrored from the assistant as a tool that generates exercises that are adapted to any technical skill level without diverging from real-world scenarios. According to Standard 8, it supports active learning via interactive tasks that engage students.

Lastly, we have the eleventh Standard by the coding assistant's capability to generate real time feedback as the student starts engaging with many levels of exercises by providing corrected solutions that track the learning progress, which essentially helps foster continuous improvement in educational outcomes. **Table 1** displays the different standards associated with the coding assistant and its relevance.

Sustainable and ethical implications

The fast advancements in generative AI, such as large language models and diffusion models, have opened a broad range of applications in various domains, including education (Denny et al., 2023). However, using such AI-powered tools in educational settings raises many different ethical concerns, from plagiarism, to over-reliance on Artificial Intelligence, and the potential for biased or misleading outputs sometimes referred to as hallucinations (Kazemitabaar et al., 2023) (Denny et al., 2023. To address these concerns, ethically using AI must be one of the major considerations when designing and implementing those tools (Sharples, 2023). Furthermore, it is essential to design these tools to enhance the learning experience of students, while maintaining the fundamental principles of academic integrity.

The sustainable development concepts as part of the CDIO framework reflects a commitment to prepare students to this rapid evolution of technological adaptation of tools in their curriculum to have responsible engineering practices. By challenging them to consider the ethical implications in the problem-solving process, we ensure that not only are we training those students to be technically proficient but also to make them evaluate the impacts in the learning process by fostering a mindset of sustainable and ethical responsibilities. As industries and organizations globally adopt socially responsible practices, engineers with this knowledge will be better prepared to lead and contribute effectively.

Table 1. Relevance of the coding assistant within the CDIO standards

CDIO Standard	Description	Relevance to AI Coding Assistant
2	Covers personal, interpersonal, product, process, and service-building skills	The assistant creates exercises that are aligned with outcomes in programming, problem-solving, and critical thinking.
3	Including technical and non-technical skills in the learning activity	The assistant combines concepts from computer science, machine learning and educational principles.
5	Encourage students to conceive, design, implement, and operate systems by displaying the tool and explaining the concepts involved	Reducing instructor workload or creating adaptive programming exercises for students. Selecting appropriate algorithms, themes, and user interfaces to ensure effectiveness. Fine-tuning its outputs, ensuring they meet educational goals, and addressing any technical limitations, such as hallucination or bias. Continuous monitoring and refinement ensure that it remains effective and aligns with evolving educational needs
7	Encourage learning through real-world applications	The assistant offers exercises based on real-world scenarios, promoting applied learning and an interdisciplinary approach.
8	Promotes student engagement through interactive methods	Students engage in tasks generated by the assistant and receive immediate feedback to actively improve their skills.
11	Evaluates student learning through various assessments methods	The assistant monitors progress, delivers real-time feedback, and dynamically evaluates learners' needs.

In conclusion, this section highlights the importance of following relevant standards (**Table 1**) to ensure reliability, effectiveness and ethical responsibilities, making both actors (students and instructors) aware of the impact of their activities. Moreover, the standards conform to the coding assistant shows the establishment of a tool respecting data privacy, fairness and prioritization transparency, not only fulfill technical requirements but also contribute to develop trustworthy, accountable and socially responsible.

HELLOPY CODING ASSISTANT

Overview of the coding assistant

HelloPy is a coding assistant designed to support python learners through personalized experience and interactive support by using the Qween generative coding model (Qwen Team, 2024). The assistant walks through students from different levels to improve, and up-skill their

problem skills by being integrated in a user-friendly environment (Streamlit Inc., 2024) to make it a tool easily intractable with which run in real-time.

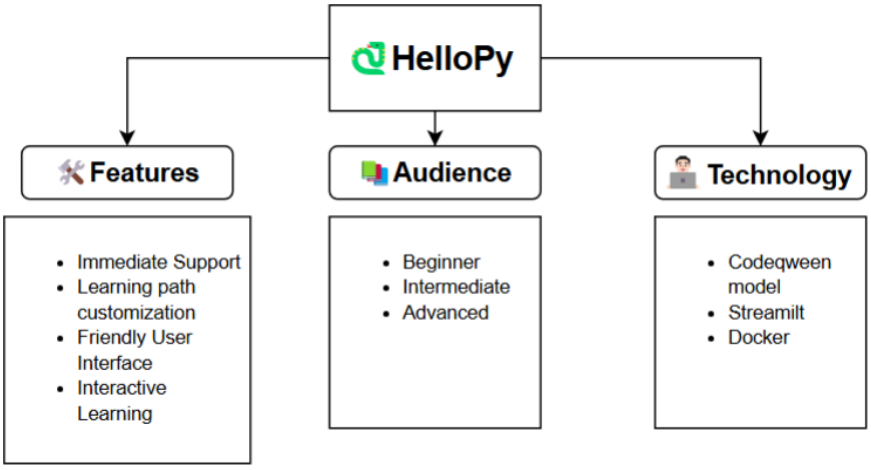

Figure 1. the HelloPy components

Study setting

In our approach, we experimented with the computer science department, in a Moroccan engineering school named **EMSI (École Marocaine des Sciences de l'Ingénieur École Marocaine des Sciences de l'Ingénieur, n.d.)**, where we gradually walk through students to reach an advanced level based on a set of predefined objectives, features, and assessment methods. All while ensuring that learners get adequate and personalized feedback and guidance from both the instructor and the coding assistant. The syllabus of the python programming language has strong foundational knowledge and progresses toward using the framework Django for web development to ensure a smooth transition from beginner to advanced equipping students with job-ready skills.

Level 1: Beginner

<u>Objectives:</u>
• Getting familiarized with the python syntax and basic programming exercises.
• Encouraging students to test different foundations to gain confidence through step-by step explanations and real-time feedback.
<u>Features:</u>
• Guided tutorials from both the instructor and the tool with easy-to-understand explanations.
• Real-time Error Handling.
• Suggesting code.
<u>Assessments:</u>
• Build-in-quizzes after completion of each lesson to test fully the comprehension of students
• Guided coding tutorial and initial challenges.

Level 2: Intermediate

<u>Objectives:</u>
• Develop problem-solving skills and an understanding of modular programming .
• Introduce intermediate concepts like file handling, error handling, and working with libraries (e.g., NumPy, pandas).

• Improve debugging and code optimization skills.
Features:
• Code debugging.
• Contextual explanation.
• Project-based learning.
• Code optimization suggestions.
• Library support.
Assessments:
• Completion of project based on milestones.
• Debugging code.
• Quizzes for intermediate concepts.

Level 3: Advanced

Objectives:
• Mastered advanced Python concepts such as object-oriented programming.
• Expose students to design patterns such as MVC (model-view controller) and MVT (model view template).
• Build projects based on web frameworks and APIs.
Features:
• Algorithmic challenges.
• Templates for real-world applications such as creating RESTful APIs with Flask or Django or training a machine learning model with scikit-learn.
• Performance metrics for the code.
Assessments:
• Completion of capstone projects, such as designing a REST API.
• Performing code tests with real-time feedback.
• Peer code reviews (simulated within the instructor).

RESULTS AND ANALYSIS

Evaluation of Student Learning Outcomes

The analysis was performed on students to assess and track the level of engagement, evaluating both the soft and hard skills aspects. As shown in both table 2 and figure 2, high scores in python coding proficiency (9/10) and assisted problem solving (8/10) is a true example of how including modern tools improve technical competencies.

For self-learning and adaptability with a score of (8/10) is a true example that supports the CDIO emphasis on extended learning and reflective practice. On the other hand, we have moderate to low scores, for debugging and optimization within the Django framework (7/10) and collaboration in projects (6/10) indicates a need to improve and develop extended tools that support collaboration and coordination.

To summarize, the results demonstrate the technical proficiency and direct learning that benefits both the students and instructors by exploring the full spectrum of the CDIO Framework.

Table 2. Evaluation of Student Learning Outcomes

Learning Outcome	Score (0-10)
Python Coding Proficiency – Writing clean, efficient Python code	9
AI-Assisted Problem Solving – Leveraging the generative offline coding assistant	8
Self-Learning & Adaptability – Learning without direct supervision	8
Debugging & Optimization – Identifying and fixing errors efficiently	7
Django Framework Understanding – Concepts, models, and views	7
Collaboration in Projects – Working in teams with Git, delegation, etc.	6

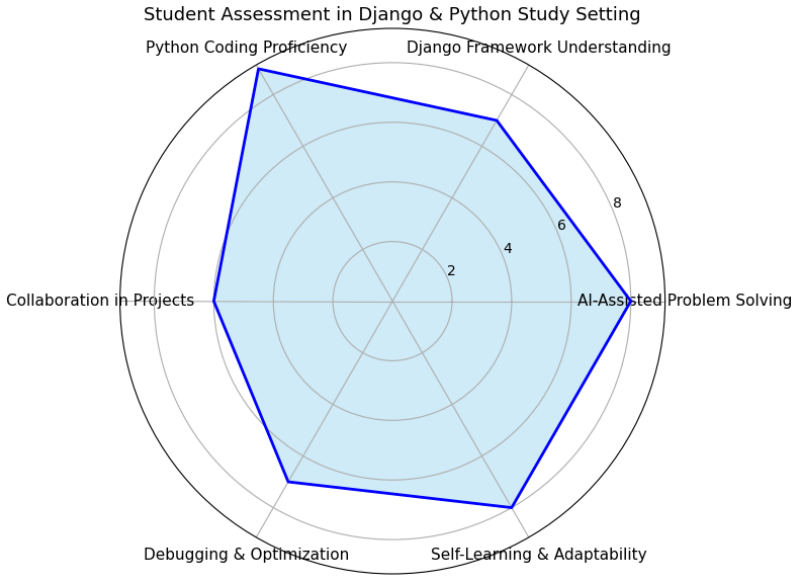

Figure 2. Study Setting for Django/Python with a Generative Offline Assistant

CONCLUSION

The result of the study shows the efficiency of using artificial intelligence, especially generative models to enhance the programming education spectrum across many skill levels. This efficiency is translated through a distinct set of prompts, adaptive learning tracks, and real-time feedback. It has also shown notable potential to engage students and boost their confidence by improving the learning process. The integration of the CDIO approach ensured a seamless transition from theory to real-world applications. Another key finding is that promoting local offline solutions while explaining to students how they can engage in responsible AI is a big step to ensure that the next students generation will develop a mindset that is not only framed in consuming tools but rather thinking if the tool is sustainable ,respecting the ethics of usage and protect their privacy. Furthermore, the solution is a win-win for both instructors and students since it reduces the workload of instructors and exposes students to a safe environment learning since AI is starting to impact teaching, learning and

thinking. To conclude, this paper is an extensive description of a generative AI coding assistant that conforms with the CDIO framework for the python programming language. The set of CDIO standards which applied to the assistant shows how we can leverage education optimally and safely for all the actors in that ecosystem and up-skilling classical teaching methods to merge with the continuous trends of technology with biasing some of the human aspects like critical thinking.

ACKNOWLEDGEMENTS

The author(s) received no financial support for this work.

REFERENCES

Agostinelli, F., Mavalankar, M., Khandelwal, V., Tang, H., Wu, D., Berry, B., Srivastava, B., et al. (2021). Designing children's new learning partner: Collaborative artificial intelligence for learning to solve the rubik's cube [Retrieved January 2025]. *Interaction Design and Children (IDC '21)*. https://doi.org/10.1145/3459990.3465175

CDIO Initiative. (2021). Cdio standards 3.0: Context and rationale. http://www.cdio.org

Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., et al. (2023). Computing education in the era of generative ai. *arXiv (Cornell University)*. https://arxiv.org/abs/2306.02608

École Marocaine des Sciences de l'Ingénieur. (n.d.). Emsi: Leading the way in engineering education. https://www.emsi.ma

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of ai code generators on supporting novice learners in introductory programming. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–23.

Mollick, E., & Mollick, L. (2023). Assigning ai: Seven approaches for students, with prompts. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4475995

Qwen Team. (2024, April). Code with codeqwen1.5. https://qwenlm.github.io/blog/codeqwen1.5

Sharples, M. (2023). Towards social generative ai for education: Theory, practices and ethics. *Learning: Research and Practice*, *9*(2), 159–167.

Streamlit Inc. (2024). Streamlit documentation.

Tan, X. (2023). The impact of chatgpt on education and future prospects. *Highlights in Science Engineering and Technology*. https://doi.org/10.54097/hset.v61i.10285

BIOGRAPHICAL INFORMATION

Zineb Hidila is a professor researcher in the Moroccan school of engineering science and a active member in the Multidisciplinary Research and Innovation Laboratory (LPRI). She holds an MSc degree in transport and logistics optimization and a PhD degree in Artificial Intelligence. Her research and development interests are wide range and include the artificial intelligence applied to cybersecurity, IoT and health. She's also a certified university instructor in Nvidia Deep Learning Institute.

Mohammed Tabaa received the degree in engineering from the Moroccan School of Engineering Science (EMSI), Casablanca, Morocco, in 2011, the master's degree in radiocommunication, and embedded electronic systems from the University of Paul Verlaine, Metz, France, and the Ph.D. and H.D.R. Diploma degrees in electronics systems from the University of Lorraine, Metz, France, in 2014 and 2020, respectively. Since 2015, he has been the Director of the LPRI Private Laboratory attached to the EMSI. His research interests include array of digital signal processing for wireless communications, the IoT, digitalization, renewable energy, and embedded systems. He served on the Organizing Committee and Technical Program Committee of several international conferences, including IEEE ICM, IEEE REPS&GIE, IEEE Systol, INTIS, ASD, JDSI, and FIoE

Corresponding author

Zineb Hidila
Multidisciplinary Research
and Innovation Laboratory (LPRI)
Moroccan School of Engineering
Sciences (EMSI)
Morocco, Casablanca
z.hidila@emsi.ma

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

APPENDIX:

CODING ASSISTANT USER INTERFACE

The tool using the interface consists of three main components:

1. **The home interface**: that explains the role and scope of the assistant.
2. **Coding assistant**: where students ask the assistant about the different learning aspects of the python language (The list is still exhaustive)
1. **FAQs**: that explains the technology involved in HelloPy, from the generative model to the user interface and the docker container for the offline execution.

1.Home Interface:

Figure 3. First page when opening the code assistant.

2.Coding assistant:

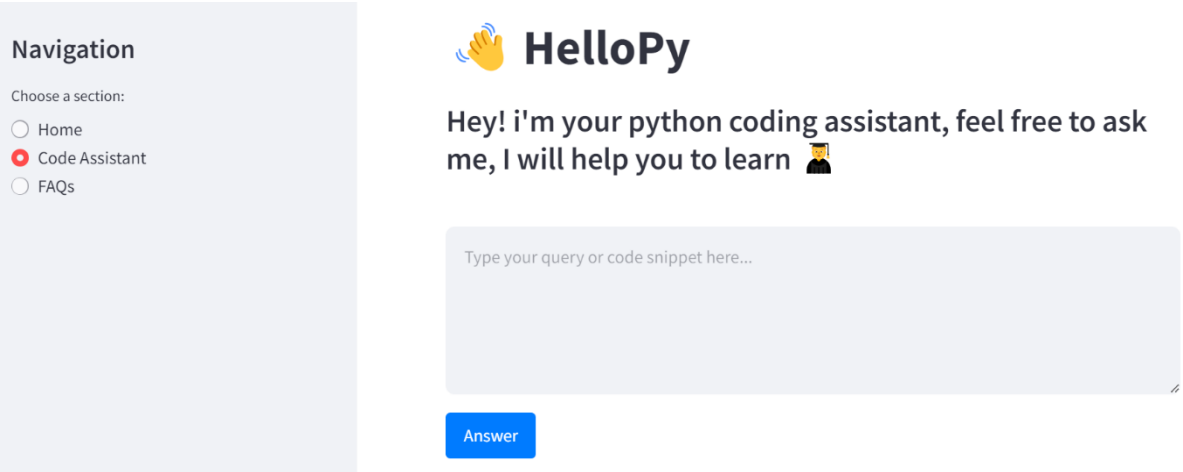

Figure 4. Assistant

3.FAQs:

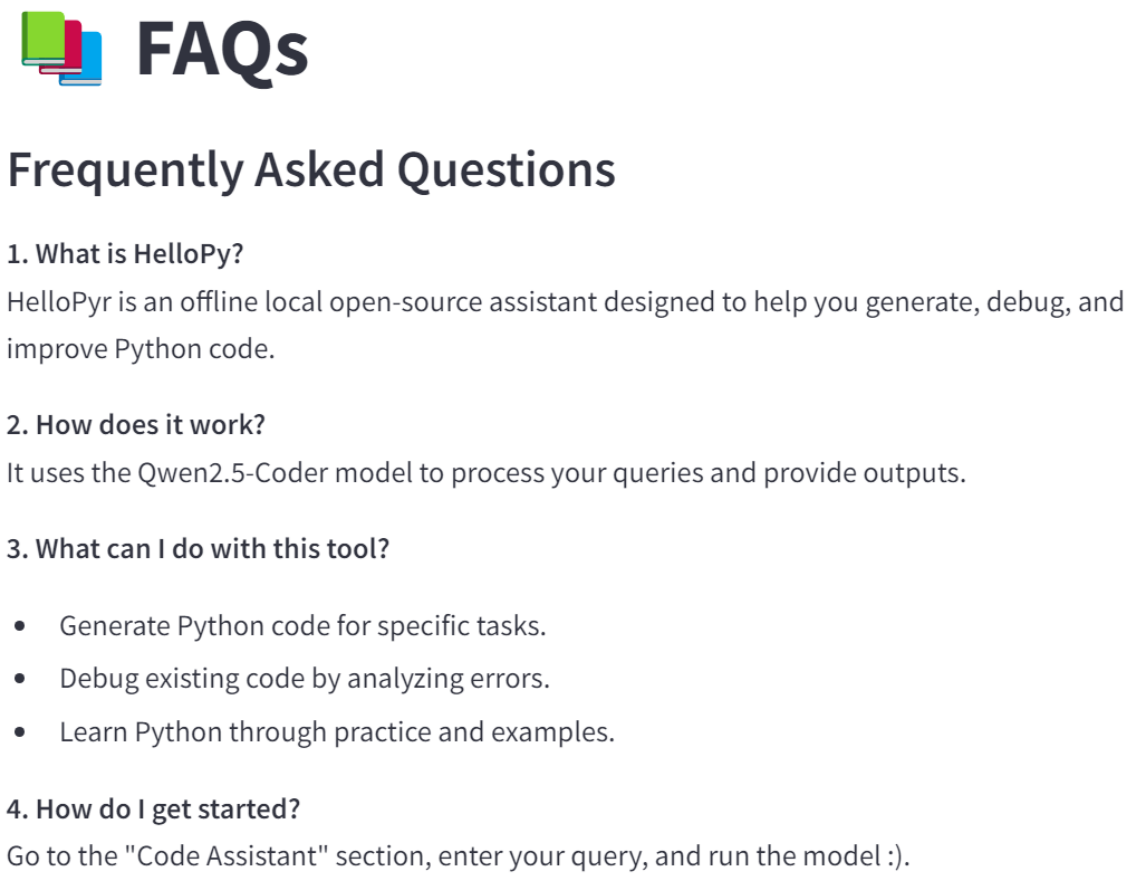

Figure 5. FAQs section

EXAMPLES OF PROMPTS WITH ANSWERS:

The different figures shown below is a small example of the capabilities of the model to generate adequate responses for all the students' level, detailed and well-structured to engage students and encourage them to surpass the difficulties such as communication and confidence.

Figure 6. Student first time asking the assistant

Model Output:

Of course! Python is an excellent language for learning and scripting. Here are some steps you can follow to get started:

1. **Install Python**: Make sure you have Python installed on your machine. You can download it from the official Python website (https://www.python.org/downloads/).

2. **Create a Virtual Environment**: It's a good practice to create a virtual environment to manage dependencies for different projects. This helps avoid conflicts between packages from different projects.

```bash
# Create a virtual environment
python -m venv myenv

# Activate the virtual environment
source myenv/bin/activate # On Windows, use `myenv\Scripts\activate`
```

Figure 7. Output of first time asking the assistant

Figure 8. Beginner knowledge test

Model Output:

Of course! Python is a versatile programming language with a rich ecosystem for various applications. Here's a quick introduction to the basics of Python:

1. **Syntax**:
- Python uses indentation to define blocks of code.
- The `if`, `elif`, and `else` statements are used to control the flow of the program.
- Strings in Python can be single quotes, double quotes, or triple quotes.

2. **Data Types**:
- Numbers: Integers (`int`), floats (`float`), complex numbers (`complex`).
- Lists: Ordered collection of items that can contain elements of any data type.
- Dictionaries: Key-value pairs where keys are unique and values can be of any data type.

3. **Control Structures**:
- For loops: Used to iterate over a sequence (like lists, strings) or for processing data in a loop.
- Conditionals (`if`, `elif`, `else`).
- Functions: Define blocks of code that perform specific tasks and can be reused.

Figure 9. Output of the knowledge test

Figure 10. An advanced student request

Model Output:

Certainly! Here's a simple project idea that you might consider if you're looking for some guidance in Python:

1. **Flask Web Application**: This will serve as your web server and handle incoming requests, such as fetching data from a database or API.

2. **Database Interaction**: Use SQLAlchemy (or other database libraries) to interact with the database where your application needs to store and retrieve data.

3. **API Integration**: Use Flask-RESTful to create RESTful APIs for your application. You can define endpoints that allow users to perform CRUD operations on data.

4. **Data Storage**: Use Amazon S3 or Google Cloud Storage for storing your data locally, if you prefer.

Figure 11. Advanced output request