Grenze International Journal of Engineering and Technology, June Issue

Reinforcement Learning based Interactive Programming Tutor for Learning Python

G. Srihitha[1], K. M. S. Laasya[2] and Krishnaprasad TR[3]
[1-3]Department of Computer Science and Engineering, Amrita School of Computing Amrita Vishwa Vidyapeetham Amritapuri, Kerala, India
Email: srihithagaddam7@gmail.com, kmslaasya@gmail.com, krishnaprasadtr@am.amrita.edu

***Abstract*— Conventional methodology of a classroom-based learning generates less curiosity amongst students learning programming for the first time. To overcome this challenge we propose an interactive clickable tool that maintains the student interest for a longer duration with respect to learning programming. We use Reinforcement Learning based reward visualization together with sequential logical progress of concepts to learners. This framework combines Q-learning, adaptive rewards and AI-generated exercises to create a more engaging and self-correcting experience for the user compared to traditional teaching methods. Upon simulation, we get highly satisfactory results for criteria such as Sequential order, Educational value and Overall value of 100%, 94% and 97% respectively.**

***Keywords*— Reinforcement Learning, Q-Learning, Python, Interactive Learning, AI.**

I. INTRODUCTION

In recent times Python is the most used programming language with simple and readable syntax. Python became a preferred language for introductory learning for programming[1]. Learning Python in traditional approach makes students to read the theory and memorize syntax. Such an approach generates a disinclination towards programming. Students act as passive recipients of information through lectures, textbooks, or videos. This lack of active engagement makes it harder for students to retain concepts and apply them [2]. So learning Python through a game-based approach is more effective than traditional methods. It makes programming interactive and increases motivation, problem solving skills[3]. It reduces fear of errors and helps beginners develop logical thinking in a fun environment. It makes students learn faster and provides a deeper conceptual and practical use of Python. Studies on Generative AI integration in education indicate that AI-driven systems can improve learner motivation, adaptive instructional design, and self-regulated learning [4]. Educational systems based on Reinforcement Learning have also demonstrated the ability to analyse learner behaviour and provide adaptive learning assistance in cognitive learning environments [5] .

In this work, we are proposing a Reinforcement Learning (RL) [6] based grid and click approach towards learning programming. Our work is primarily focused on students who are learning programming for the first time. The idea behind using Q-learning is that it learns through trial and error, which is similar to the way students learn from mistakes and repeated practice. Research on RL agents in simulation environments highlights the importance of reward-based learning and sequential decision-making for adaptive systems [7]. Q-Learning is a fundamental concept involved in RL. The idea for using Q-learning here is it learns from trials and errors which is very similar to that of how a student learns from mistakes and errors.

Grenze ID: 01.GIJET.12.2.574

This concept makes RL useful in educational purposes. Similarly, game-oriented Reinforcement Learning environments have shown improved interaction and engagement through reward-driven learning mechanisms [8]. Recent studies on sparse reward environments further demonstrate how RL systems can continuously improve performance through feedback and adaptive exploration strategies [9].
The proposed system incorporates gamification elements such as rewards, levels, and progress tracking to enhance user engagement and interactivity. For dynamic content generation we use a grid environment integrated with the AI model Groq (different from Grok). This integration enables the real-time generation of new Python programming problems and exercises for beginners. Additionally, the system generates rewards for each correct option selected by the user, thus providing for an interactive and visually appealing medium for students to learn programming. This unique property makes this platform more interactive and flexible than other methods. The contribution of this paper is in designing a dynamic interactive click-based interface to learn and conceptualize introductory programming. Provide a visualized display of RL based rewards with every step of sequential progress.

II. Literature Review

Learning Programming has always been a challenging area, especially for beginners. Researchers have explored multiple approaches over the years to make this process easier and interesting.
One of the earliest studies found that students who received one-on-one tutoring performed better than those in a traditional classroom environment [10]. Since it is not practical for every student to get a personalised tutoring from a human instructor our system provides AI-driven guidance to each learner using their performance and behavioural data. This type of system helps student by guiding them in programming by giving step-by-step feedback and breaking complex problems into smaller tasks, which helps reduce learning time and improve retention [11]. Reinforcement Learning introduced an improved approach to learning systems, where Q-learning allows an agent to learn optimal actions in different states through rewards and penalties based on the Bellman equation [12]. While traditional Q-learning focused on optimising actions through rewards and penalties, our work applies it specifically to programming education by dynamically adapting learning strategies according to student progress. Researchers also applied Q-learning to grid-based and sequential decision-making tasks [13] , demonstrating its ability to learn optimal actions through repeated interactions. This supports the present work, which uses a grid-world structure to guide students through Python learning tasks. While previous studies showed that gamification elements like rewards and levels improved engagement and motivation, our work combines these features with reinforcement learning to create adaptive learning paths for student's progress. This combination of machine learning and game-based learning environments can improve both the system performance and the overall learning experience [14].
Recently, the rise of LLMs has created new possibilities for automated content generation in education. Rule based approaches struggled to produce diverse questions, while neural language models were able to generate more contextually relevant questions across different subjects and different levels [15] . Q-learning is an effective approach for sequential decision making, but its applications to programming education remain limited. Reinforcement learning in educational environments showed good results, though most systems lacked structured grid world-based interaction connected to programming education. AI-based content generation showed potential for creating coding problems, but it was used separately from reinforcement learning environments [16]. As a result, there is only limited research on combining reinforcement learning, grid world interaction, programming education, and AI-generated learning content. The proposed RL Python Tutor aims to explore the integration of these elements within an interactive educational environment.

III. Framework and Methodology

The proposed system employs a interactive grid-based learning environment designed to teach Python programming concepts through structured problem-solving. The environment consists of a 4×4 grid comprising 16 cells, where each problem is decomposed into 4–6 sequential subgoals. Learners must identify and select these subgoals in the correct order to successfully complete a task.

A. Grid Environment and Interaction

Each cell in the grid represents a potential code statement or operation. When a learner clicks on a cell:

- If it is the correct subgoal (selected in the proper sequence) turns green, providing immediate positive visual feedback.

- If it is incorrect cell it turns red, applies a penalty, and displays explanatory feedback highlighting why the choice was inefficient or incorrect.

The grid and the system is designed to include 'trap', i.e deliberate incorrect statements which may appear to be the correct statements. This is to dissuade the students from guessing in a random way and to make each student think in a logical way. Such a 'trap' also leads to the user in improving the concepts of a particular programming approach. The system also deliberately makes the user to think in a sequential way by giving a lesser reward if the user selects a cell out of order. The user is displayed a 'Goal' status upon completing all the sequential (sub goals).

B. Learner Support Features

If the user (who is a beginner in programming) is unable to correctly choose the logical sequence of cells, the system provides the user with a 'Guide me' option. This option provides the user with a subgoal which would make the user think about the correct answer rather than completely making the answer directly visible to the user.

C. Reward and State Systems

A Q-Learning based approach is used to generate rewards. Correct cells are initialized with higher Q-values (0.7–1.0), while incorrect cells start lower (0.3–0.5). These values are dynamically updated based on user interactions. The reward function is defined as:

$R(s, a)=0.5\times I[\text{correct}]-2.5\times I[\text{incorrect}]-0.1$

Where s is the state, a is the action taken (selecting a cell). The rewards are allotted as follows:

- +0.5 is awarded for selecting the correct cell in the proper sequence,
- -2.5 is deducted for incorrect selections,
- -0.1 is a small per-step penalty applied to discourage random clicking.

The state (s) tracks the number of correctly completed subgoals in sequence, determining the current target subgoal.

The Q-value for each cell is updated using the standard Q-learning formula

$Q(s, a)\leftarrow Q(s, a)+\alpha[R+\gamma \max_{a'} Q(s', a')-Q(s, a)]$

Where:

α is the learning rate - controls the weight of new information versus past knowledge, γ (discount factor) determines the importance of future rewards, (s') and (a') represent the next state and possible action

D. AI-powered dynamic problem generation:

To overcome the limitations of static problem sets, the system integrates with a large language model namely 'Groq'. We chose 'Groq' over other LLM models due to it's free availability of API keys, unlike ChatGPT and Grok where uses are expected to pay a certain amount. This enables on-demand generation of new Python programming problems. Users can request a problem by selecting the "Generate" button and providing a topic or problem name. The AI generates appropriate subgoals, places them correctly within the grid, ensures no overlapping content, and randomizes cell positions on each reset. Generated problems are automatically added to the problem menu, ensuring freshness and personalization. A pseudo code of the proposed solution is given below:

Algorithm:
InteractiveGridLearning(problem_topic)
Input: Problem_topic (user-selected or AI-generated)
Output: Learning session with rewards and feedback

GenerateProblem(problem_topic) using Groq
Decompose into 4–6 ordered subgoals
Populate 4×4 grid with subgoals + traps (random positions)
Initialize Q-values for each cell
Set current_state = 0, total_reward = 0

while current_state < number_of_subgoals: do
Display grid and current target subgoal description
User selects cell (action a)
if cell is correct AND in sequential order then
Turn cell GREEN and total_reward += 0.5

Update Q(cell) using Q-learning formula
current_state ← current_state + 1
else
Flash cell RED and total_reward -= 2.5
Show explanatory feedback / trap rationale
Apply step penalty -0.1 and Update Q(cell)
Update all Q-values based on new reward and next state
if user requests "Guide Me" then
Provide hint for current subgoal only

if current state == number_of_subgoals then
Display final reward
Congratulate learner on completing the problem in correct order

IV. RESULT

In order to simulate the RL tool, we use Python with libraries such as sklearn, tkinter, json, numpy, ast, requests etc. Given below is an example of a program using the RL tool. Here we showcase the grid having the program: 'Change to uppercase and calculate the length of the string'. This cells in the grid has code snippets belonging to the Python program of 'Change to uppercase and calculate the length of the string' as shown in Fig. 2. The challenge for a beginner to programming is to click the cells in a sequential manner so as to complete the program in a logical way. If the user clicks an incorrect cell, even if it is not in the correct sequence, the corresponding cell will be highlighted in red colour as shown in the Fig. 3. If the user selects the cell in the correct sequential order, each cell gets highlighted in Green as shown in the Fig. 4 below.

Figure. 1– String program

Figure. 2– Incorrect selection

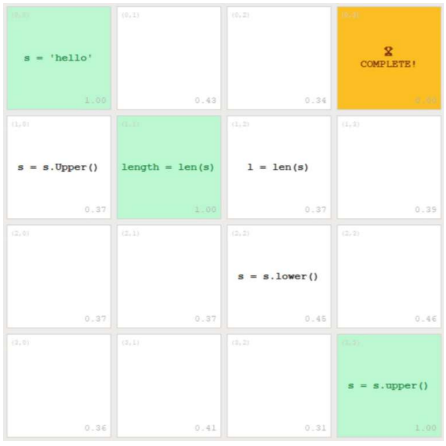

Figure. 3 – Sequentially correct selection

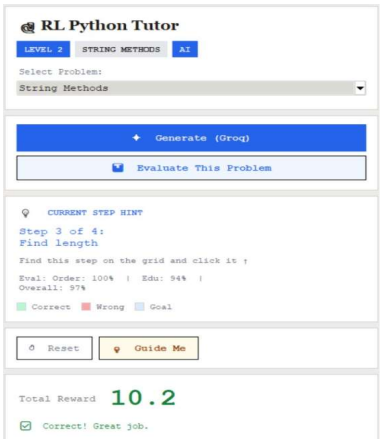

Figure. 4 – Working of reward system

With each correct cell clicked by the user, a cumulative reward is displayed in the user interface. The reward increases if the user clicks the correct cell else the reward decreases as shown in the Fig. 5 above. The 'Guide Me' button, helps the user in providing 'hints' (one hint at a time). The performance of this RL tool is evaluated on certain metrics to understand it's effectiveness. We use the following three evaluation criteria: Sequential order, Educational value, Overall effectiveness score. To check whether the generated problems make logical sense, we created a sequential order validator. This tool takes the subgoals (small code steps) one by one and joins them together. It then tests if the code still works correctly using Python's built-in syntax checker. If upon the addition of any code snippet breaks the code, then the score is reduced by 0.3. Such a process makes sure that subgoals are correct by themselves and also they appear in the correct order. The educational value of each generated problem was evaluated using a heuristic-based scoring function. The function analyses both individual subgoals and the complete code by searching for key string-related programming concepts and methods (e.g., input(), .split(), .strip(), .upper(), len(), and f-strings). Each concept is assigned a predefined weight reflecting its pedagogical importance for beginners.

The Overall effectiveness score is computed by a weighted average of two important factors:

Overall value =0.5×Sequential Order+0.5×Educational Value

As this work is focused on beginner centric interactive learning, the scalar value 0.5 are the same for both the factors to reflect equal importance given to logical correctness and pedagogical usefulness. The resulting score ranges from 0.0 to 1.0, providing a single composite measure of educational effectiveness.

For a basic program of 'Change to uppercase and calculate the length of the string' the values of the metrics are:

TABLE I

Sequential order	100%
Educational value	91%
Overall score	94%

V. CONCLUSION AND FUTURE WORKS

This work provides a grid based interactive approach in understanding the logical flow of writing a program. The interactive environment makes the user more interested in solving a problem or writing a program. We use RL to showcase rewards achieved by the user after each selection of a cell. RL is very important as it helps to track the sequential progress of the logical understanding of the user. The rewards increase if the selected cell is the correct cell and the sequential order is maintained. The rewards decrease otherwise. We use Sequential order, Educational value and Overall effectiveness metric to evaluate this approach. Future work will include further refining of the program by collecting the user feedback per problem and customizing the work for each learner.

REFERENCES

[1] Shein E (2015) Python for beginners. Commun ACM 58:19–21. https://doi.org/10.1145/2716560

[2] Eranki KLN, Moudgalya KM (2016) Program Slicing Technique. In: Proceedings of the 17th Annual Conference on Information Technology Education. ACM, New York, NY, USA, pp 160–165

[3] Sreelakshmi R, McLain M, Rajeshwaran A, et al (2015) Gamification to enhance Learning using Gagne's learning model. In: 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, pp 1–6

[4] Raman R, Achuthan K, Nedungadi P (2026) Generative AI Integration in Education: Theoretical Review and Future Directions Informed by the ADO Framework. Information 17:241. https://doi.org/10.3390/info17030241

[5] Nagarajan H, Inakollu VS, Vancha P, Amudha J (2023) Detection of Reading Impairment from Eye-Gaze Behaviour using Reinforcement Learning. Procedia Comput Sci 218:2734–2743. https://doi.org/10.1016/j.procs.2023.01.245

[6] Sutton RS., Barto AG. (2020) Reinforcement learning: an introduction. The MIT Press

[7] Nair KG, Aravind MJ, Adithyan AN, et al (2026) Evaluation Strategies for Reinforcement Learning Agents in Diverse Simulation Environments. pp 83–103

[8] Niveaditha VR, Sheba Sulthana PR, Bharathi Mohan G, Doss S (2024) Word Based Tic-Tac-Toe Using Reinforcement Learning. In: 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, pp 1–7

[9] Kunchapu A, Kumar RP (2023) Enhancing Generalization in Sparse Reward Environments: A Fusion of Reinforcement Learning and Genetic Algorithms. In: 2023 Global Conference on Information Technologies and Communications (GCITC). IEEE, pp 1–7

[10] BLOOM BS (1984) The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher 13:4–16. https://doi.org/10.3102/0013189X013006004

[11] Anderson JR, Boyle CF, Reiser BJ (1985) Intelligent Tutoring Systems. Science (1979) 228:456–462. https://doi.org/10.1126/science.228.4698.456
[12] Surendran S, Montresor A, Vinodini Ramesh M (2025) A Reinforcement Learning Approach for Routing in Marine Communication Network of Fishing Vessels. SN Comput Sci 6:62. https://doi.org/10.1007/s42979-024-03606-6
[13] T M, Y L, S L, et al (2014) Offline policy evaluation across representations with applications to educational games. AAMAS '14: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems
[14] Hamari J, Koivisto J, Sarsa H (2014) Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification. In: 2014 47th Hawaii International Conference on System Sciences. IEEE, pp 3025–3034
[15] Pedreira O, García F, Brisaboa N, Piattini M (2015) Gamification in software engineering – A systematic mapping. Inf Softw Technol 57:157–168. https://doi.org/10.1016/j.infsof.2014.08.007
[16] Kurdi G, Leo J, Parsia B, et al (2020) A Systematic Review of Automatic Question Generation for Educational Purposes. Int J Artif Intell Educ 30:121–204. https://doi.org/10.1007/s40593-019-00186-y