Effective Personalized AI Tutors via LLM-Guided Reinforcement Learning

Angel Tsai-Hsuan Chung,[1,*] Botong Zhang,[2] Ling-Chieh Kung,[3]

Hamsa Bastani,[1,*,†] Osbert Bastani[2,*,†]

[1]Department of Operations, Information, and Decisions, University of Pennsylvania, Philadelphia, USA

[2]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, USA

[3]Department of Information Management, National Taiwan University Taipei, Taiwan

[*]To whom correspondence should be addressed (emails: angelchg@wharton.upenn.edu, hamsab@wharton.upenn.edu, obastani@seas.upenn.edu); †Equal last author

Generative AI (GenAI) is rapidly reshaping education by unlocking the potential for personalized tutoring. Yet, emerging platforms largely focus on GenAI chatbot tutors that reactively answer student questions. We hypothesize that the efficacy of GenAI chatbot tutors can be substantially improved by proactively guiding student learning. To test this, we design a novel tutoring platform that tightly integrates a carefully-designed GenAI chatbot with a reinforcement learning algorithm for sequencing practice problems. Critically, this algorithm leverages rich signals from student-chatbot interactions to adaptively select practice problems of an appropriate difficulty level. In partnership with the Taipei City Government and American Institute in Taiwan, we deployed our tutoring platform in conjunction with a five-month course to teach Python to students across ten high schools. We randomized students between a fixed

practice problem sequence and our adaptive sequencing algorithm. We find that adaptive sequencing increased unassisted final exam performance by 0.15 standard deviations (equivalent to 6-9 months of schooling by some estimates); mediation analysis suggests that gains were driven by increased engagement. Our work provides large-scale field evidence that student-chatbot interactions provide valuable signals for proactively optimizing and personalizing student learning.

1 Introduction

With the rapid recent progress in Generative Artificial Intelligence (GenAI), there has been substantial interest in leveraging it to improve education. A major focus has been on designing personalized tutors based on Large Language Models (LLMs) (*1–6*), which gives students on-demand access to a GenAI chatbot that answers student questions and attempts to resolve their misunderstandings. Developing effective tutoring systems has become increasingly urgent as technological change is accelerating the need for worker reskilling (*7, 8*).

A key lesson from decades of research—from Computer-Assisted Instruction (*9*) to MOOCs (*10*) to initiatives such as One Laptop per Child (*11*)—is that access to educational technology alone is insufficient to improve outcomes; successful deployments require careful design and implementation that aims to integrate with existing pedagogical context rather than replace it. GenAI chatbot tutors are no different—a recent meta-analysis synthesizing 51 experimental studies finds that while ChatGPT has positive impacts on learning on average, there is substantial heterogeneity in outcomes (*3*). Recent work has highlighted the importance of good design on the efficacy of GenAI chatbot tutors, studying the impact of prompt design (*2*), optimizing chatbot features (*12–17*), or improving LLM accuracy on educational tasks (*18, 19*).

However, these approaches all focus on the problem of improving the chatbot, and stay

within the reactive paradigm of responding to student queries (e.g., debugging code upon request) (*20*). Yet, effectively promoting learning requires substantially more than simply conversing with the student. Evidence suggests that learning requires students to engage in productive struggle (*21, 22*), spending time working on challenging problems that help develop critical thinking and problem solving skills. The challenge with GenAI chatbot tutors is that students often lack the skills needed to effectively self-regulate their learning (*23–25*), making it difficult for them to formulate queries that significantly improve their learning outcomes.

In contrast, effective techniques such as mastery learning (*26, 27*), desirable difficulty (*28*), zone of proximal development (*29*), and productive struggle (*21, 30*) focus on engaging the student by providing them with challenging practice problems that are suitable for their current knowledge and skill. While these approaches traditionally require extensive instructor oversight (*31–33*), there has been substantial interest in scaling these techniques by designing algorithms that automate them (*34–37*). A fundamental challenge facing these algorithms is the need to estimate the student's current knowledge state, which captures how well they have learned the underlying concept. Accurate and efficient estimation of a rich knowledge state is key to rapidly adapting difficulty so the student does not waste valuable time solving problems they have already mastered. However, existing algorithms face a critical information bottleneck—they only have access to coarse proxies for the knowledge state, such as whether the student's solutions are correct or how much time they spent solving each problem; as a consequence, they can only effectively estimate a highly simplified approximation of the student's true knowledge state. For instance, the popular Bayesian knowledge tracing (BKT) algorithm (*38, 39*) for personalized learning relies on binary signals (correct or incorrect solutions) to estimate a binary knowledge state (learned or unlearned), which is a very coarse approximation of student understanding.

We argue that these shortcomings can be addressed by tightly integrating adaptive problem sequencing into GenAI chatbot tutors, through two key strategies: (1) using LLMs to extract

richer semantic signals from student solution attempts, and (2) extracting signals from student-chatbot interactions. To test our hypothesis, we propose a novel reinforcement learning algorithm that leverages signals from student-platform interactions—using LLMs both to measure quality of student-chatbot conversations and to analyze student solution attempts—to accurately estimate a continuous measure of the student's current knowledge state. To facilitate this rich feedback and knowledge state representation, our algorithm leverages a novel state estimation procedure based on particle filtering (*40*), as well as a novel planning procedure based on model predictive control (MPC) (*41*). Our algorithm rapidly estimates a rich approximation of the student's knowledge state, and performs granular personalization of practice problem difficulty levels based on this estimate. Furthermore, we developed a tutoring platform that tightly integrates this algorithm with a carefully-designed GenAI chatbot tutor.

In collaboration with the Taipei City Government, the American Institute in Taiwan, and the American Innovation Center (which operates under U.S. Department of State oversight), we deployed our tutoring platform through a five-month-long "AI for Python Learning" course. This course augmented standard lecture-based instruction with practice on our tutoring platform. We conducted a randomized controlled trial (RCT) across 10 participating high schools in Taipei, and we evaluate the impact of our system on semester-long learning outcomes for 770 students. All students were provided with access to the same course material and GenAI chatbot tutor, but we randomized students between our personalized problem sequencing algorithm versus a fixed problem sequence based on standard practice. To the best of our knowledge, our study provides a substantially larger-scale and longer-horizon assessment of the efficacy of GenAI tutoring compared to existing work.[1]

[1]The meta-analysis (*3*) synthesizes 51 (quasi-)experimental studies published between November 2022 and February 2025 (72 coded effect sizes). For the learning-performance outcome specifically ($n = 44$ independent studies), the duration distribution is heavily short-horizon that only $n = 13$ studies lasting $>$8 weeks (i.e., 31/44 $\approx$ 70% are $\leq$8 weeks). Reported samples are typically small: median per-arm sample sizes are $\approx$35–37 students, and $\sim$89% of comparisons involve fewer than 100 students per arm. In addition, we surveyed existing experimental

Our results show that personalized sequencing improves outcomes compared to fixed sequencing by 0.15 standard deviations on an in-person written exam completed without AI assistance, which translates into as much as 6-9 months of additional schooling according to some estimates (*42, 43*). Notably, these gains were achieved without increasing instruction time or teacher workload. Furthermore, our mediation analysis suggests that these gains are mediated by increased student engagement, addressing a long-standing challenge in educational technology of sustaining engagement beyond the "novelty effect," where interest typically rapidly declines following initial excitement (*44–46*). Our work provides rigorous field evidence that leveraging rich behavioral signals from student-platform interactions can drive automated personalization of student practice to substantially improve learning outcomes.

2 GenAI Tutoring System

In this section, we describe our GenAI tutoring system. Both lecture-based instruction and solving practice problems occur on our platform, organized into modules that the students must complete in order. Our GenAI tutoring system is responsible for (i) generating a bank of practice problems (this step happens before the start of the course), (ii) helping students solve a sequence of practice problems for each module, and (iii) selecting the sequence of practice problems to maintain engagement. A key novelty of our system is that it leverages LLM-based features—e.g., student-tutor interactions and code-edit traces—to estimate and track each student's learning progress (see Fig. 1 for illustrative examples); we find that these traces serve as a more precise proxy for learning progress than conventional measures (see Supplement B.4). In our experimental design, (i) and (ii) are provided across all conditions, whereas access to (iii) is randomized, with the goal of evaluating whether it is effective at improving learning

evidence for the efficacy of adaptive problem sequencing, and find that most existing experiments conflate multiple interventions or compare to weak baselines; see Supplement G for a detailed discussion.

outcomes. We describe these three key components of our GenAI tutoring system here, and provide additional details in Supplement A.

2.1 Problem Bank Generation

Personalized problem sequencing requires a large bank of candidate problems, which requires significantly more effort to generate. To scale this process, our system leverages a "self-validating" LLM to help generate high-quality, auto-gradable practice problems that are aligned with the course materials (see details in Supplement A.2). This component takes a human-in-the-loop approach, generating candidate problems that are edited by a human expert before being added to our problem bank. We adopt a human-in-the-loop workflow where teaching assistants review and edit validated candidates for quality and correctness before adding them to the problem bank.

To ensure that the problems are aligned with the course material, our problem generator uses retrieval-augmented generation (RAG) (*47*) from instructional materials, including lecture slides and past practice problems. Given this knowledge base, we prompt an LLM (specifically, GPT-4o) to generate candidate practice problems. Prompts specify the course module, target concept, desired problem type (coding, debugging, or true/false) (see details in Supplement H.1), and difficulty level. The LLM also generates the scaffolding required for autograding, including either the answer for true/false problems or test cases for coding and debugging problems. To ensure quality and correctness, each generated problem is validated through a Python script checking consistency among the question, solution, and test cases. When validation fails, error messages are passed to a separate LLM (specifically, Claude) for iterative revision until the problem passes. Finally, a teaching assistant checks all validated problems for language clarity and alignment with course objectives before adding them to the problem bank. We provide additional details in Supplement H.1.

(a) Answer-Seeking Behavior

Task: *Write code to determine if a number is odd.*

User: Can you tell me how to solve this problem?

AI: *Of course, but can you first tell me what you've tried...*

User: I have literally no idea, can you give me a hint?

AI: *No problem. We can use the "remainder" to determine...*

User: Do I need to use "if" or something like that here?

AI: *Yes, exactly! You can use "if" conditional statements...*

User: **Please tell me the answer.**

AI: *Before we discuss the answer, tell me what you've tried...*

(b) Concept-Understanding Behavior

Task: *Remove first occurrence of an element from a list.*

User: [Shares code attempt with typo: `1st` instead of `lst`]

AI: *There are some errors here, `1st` should be `lst`...*

User: Can you explain `if lst[i] == elem`?

AI: *Of course! This line finds elements equal to `elem`...*

User: What does `elem` represent?

AI: *`elem` is a parameter representing the element to remove...*

User: What's the difference between `remove` and `pop`?

AI: *`remove()` removes by value; `pop()` removes by index...*

(c) Not Meaningful Edits (Format Only)

Task: *Sum even numbers from 1 to n.*

```
--- deleted
+++ added
@@
 def even_sum(n):
     sum = 0
-    for i in range(1,n+1):
-        if(i%2==0):
+    for i in range(1, n+1):
+        if i % 2 == 0:
             sum += i
     return sum
```

(d) Meaningful Edits (Functional Changes)

Task: *Parse/validate IPv4; return `[a,b,c,d]` or `None`.*

```
--- deleted
+++ added
@@
 def solution(string):
     aList = string.split('.')
-    for i in range(len(aList)):
-        aList[i] = int(aList[i])
-    return aList
+    if len(aList) != 4:
+        return None
+    for i in range(len(aList)):
+        if 0 <= int(aList[i]) <=
↪  255:
+            aList[i] = int(aList[i])
+        else:
+            return None
+    return aList
```

Figure 1: **Examples of LLM-based features showing a student's learning progress**. The top row shows AI tutor chat interactions (translated to English): (a) answer-seeking behavior vs. (b) concept-understanding engagement. The bottom row shows student's code editing patterns: (c) format-only changes (in students' code-edit histories, the most common non-meaningful edits include typos, formatting, or stylistic changes) vs. (d) meaningful functional changes.

2.2 GenAI Chatbot Tutor

The core of our GenAI tutor is a web interface that presents students with a sequence of practice problems along with a chatbot designed to help answer questions and resolve any confusion. Our interface shows the current practice problem to the student, and provides them with space to write and execute code and to chat with the GenAI tutor. Students can submit solutions and receive immediate feedback from our autograder; they can submit as many times as desired without penalty. They must correctly solve one problem before continuing to the next. The interface also disables copy/paste functionality to ensure students remain on our platform.

Our platform includes an LLM-based chatbot that is available throughout practice, enabling students to ask questions, request hints, and interpret error messages. The tutor is prompted to avoid providing answers unless students first demonstrate substantial effort (see Fig. 1(a)-(b) for examples, and Supplement A.3 for details). Notably, our platform records time-stamped logs of learner activity, including student and chatbot messages as well as all code edits and submissions (see Fig. 1(c)-(d) for examples). These behavioral traces are critical for us to infer the student's knowledge state; in addition, we also use them in our empirical analysis to help understand the mechanisms underlying our improvements in learning outcomes. We provide additional details on interface, including screenshots, in Supplement A.

2.3 Personalized Problem Sequencing

A key novelty of our system is how it selects the sequence of problems for students to solve. A solution requires two components: (i) a method for estimating the student's current *knowledge state*—which summarizes their mastery of the skills in the current module—based on observations of their performance (e.g., the rate of correct solutions or the time taken to solve problems), and (ii) a policy for selecting the next optimal problem to give the student, based on the estimated knowledge state. Existing algorithms rely on very limited feedback for knowledge

state estimation, which in turn limits the complexity of the knowledge state representation. We build on Bayesian knowledge tracing (*38*), which formulates the optimal problem sequencing problem as a Partially Observed Markov Decision Process (POMDP) (*48*), a general framework for encoding sequential decision-making problems where the true state (in our case, the student's knowledge state) is unobserved and must be estimated based on observations. However, their POMDP is simple—their estimation algorithm only considers binary observations of whether the student successfully solved a problem, so they correspondingly encode the knowledge state as a binary indicator of mastery of the current topic. In their setting, simplicity is necessary due to limited feedback from student interactions. Furthermore, their simple knowledge state also makes selecting the optimal problem straightforward.

In our setting, binary feedback is untenable since students rarely obtain the correct solution on their first try (e.g., due to syntax errors rather than misunderstandings); thus, the feedback would always be negative, and Bayesian knowledge tracing would conclude that the student never achieves mastery. Instead, we design a POMDP that leverages the substantially richer feedback provided by our system. We consider a knowledge state that encodes a continuous measure of mastery; it includes three parameters (i) the student's initial performance, (ii) the student's mastery ceiling (the maximum performance they achieve if they hypothetically solve all of the problems), and (iii) their learning speed (how much they improve after solving a problem). Based on its current estimate of the knowledge state, our algorithm (described below) must select the difficulty level of the student's next practice problem (out of four). The knowledge state evolves according to a deterministic transition function; at a high level, the student's mastery increases depending on the student's learning speed, the number of attempts they needed to solve the problem, and the selected problem difficulty; hyperparameters in the transition function were calibrated based on data collected from a pilot study, which we describe in Section 3 and Supplement C. Finally, our algorithm receives stochastic feedback based on the student's

interaction with our platform—most notably, an LLM's estimate of the number of serious attempts required to solve the problem and the quality of the student-chatbot interactions—and updates its estimate of the knowledge state accordingly.

Our algorithm uses a particle filter for belief state estimation (*40, 49*); it maintains a discrete probability distribution over a beam of plausible knowledge states, and performs posterior updates to this distribution whenever it receives feedback. The belief state estimate is initialized based on the prediction from a random forest model trained on historical data from our pilot study; this model uses features from both a pre-study survey as well as performance in earlier modules when available. To select an optimal decision, our algorithm uses model predictive control (MPC) (*41*); at each step, it samples a large number of possible decision sequences up to a finite number of steps, simulates the outcome for each decision sequence, selects the sequence that achieves the highest performance, and takes the first decision in that sequence.

Finally, to speed up convergence, our algorithm uses model-based reinforcement learning, where it trains a machine learning model on historical data to help predict the initial belief state for a given student-module pair. At a high level, we derive features from historical data on each student's interactions with our platform, including features based on using an LLM to analyze student-chatbot conversations. This strategy enables our belief state estimates to converge more rapidly, enabling us to achieve strong performance earlier in the problem sequence.

We provide additional details on our algorithm in Supplement B, as well as simulation experiments supporting the importance of LLM-derived behavioral information from student-platform interactions.

3 Field Experiment

In partnership with the Taipei City Government, the American Institute in Taiwan, and the American Innovation Center, we developed and ran an "AI for Python Learning" Program based

on our platform. The goal of this program was to teach high school students introductory data science skills in parallel with Python programming skills, in alignment with the government's broader digital literacy and bilingual education goals. The instructional content and materials were based on a ten-module online course developed by professors at National Taiwan University for Coursera, where it has enrolled more than 44,000 learners and received a 4.9/5 average rating, demonstrating its high quality. The program included a formal government-endorsed certification (valid for college applications), providing a strong incentive for students to perform well. To earn the certificate, each student had to (i) complete the required homework for each of the ten modules on our platform (see program schedule in Supplement D.2), and (ii) pass a final, in-person, written certification examination.

In Summer 2024, we ran a pilot study on an early version of the platform, which we detail in Supplement C. Then, in Fall 2024, we worked with our government partners to recruit schools and students for the main certification program; our government partners helped recruit and obtain informed consent from recruited participants (see Supplement D.1 for details). Both the pilot and the main study were IRB-approved at the University of Pennsylvania; the main study was pre-registered (see Supplement D.4 for details).

A total of 1,047 students from 10 high schools in Taipei (8 public, 2 private) participated in the program for the main study. We report results on a sample of 770 students, excluding students that did not meet our pre-registered criteria (e.g., dropped out, flagged for cheating)[2] or that are not directly comparable to the standard program (see Supplement E.1 for details). In January 2025, the government officially launched the program, which ran through the end of June 2025, for a total learning period of approximately five months (see Supplement D.2 for details). In June 2025, we administered the final in-person certification examination and hired external graders to score all the exams. We randomized all students at the individual level into

[2]We do not find significant differences in attrition across both arms; see Supplement E.3.

one of two groups with equal probability:

1. **Treatment (Personalized problem sequence):** Students solve practice problems selected adaptively by our POMDP-based personalized problem sequencing algorithm.

2. **Control (Fixed problem sequence):** Students solve a fixed sequence of problems that progresses from easy to hard within each module; this strategy mirrors common practice.

In particular, the learning platform, including instructional content and tutor chatbot are the same in both groups, and the only difference is in terms of the problem sequence. To ensure comparable exposure by difficulty, all students—regardless of the condition—are required to complete at least two "very hard," two "hard," and two "medium" questions in each module.[3]

We conducted baseline surveys at the beginning of the program to collect details on students' demographics, parents' education, motivation, and pre-existing Python ability (see Supplement D.3). Table 7 in Supplement E.2 shows that these student characteristics are well balanced between experimental conditions.

Our primary outcome is student performance on the final certification exam administered in-person at participating schools with no digital devices allowed; this exam assesses conceptual understanding and problem-solving skills in Python (see Supplement H.2 for details). Finally, in Summer 2025, trained graders scored all exams using a common rubric. We provide additional details on our experimental design in Supplement D.

4 Evaluation

4.1 Main Results

We evaluate the impact of our personalized problem sequencing algorithm on certification exam performance using an Intention-to-Treat (ITT) analysis with ordinary least squares regression

[3]Each module contains 40 auto-graded practice problems (coding, debugging, and true/false), generated via the LLM-based pipeline described in Section 2.1. Homework assignments typically require 15 questions per module.

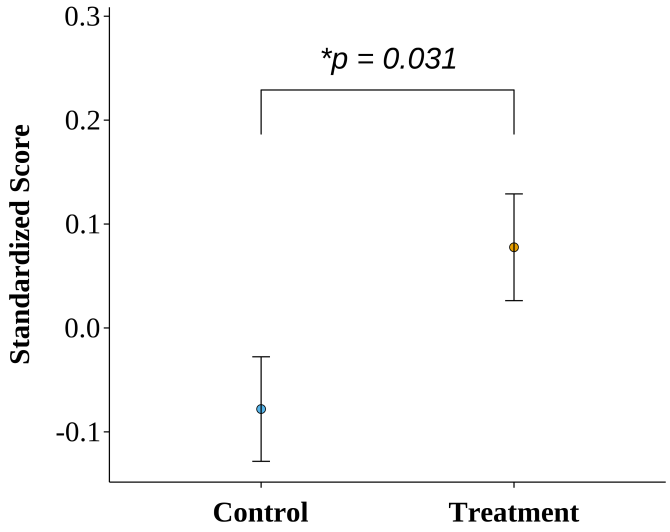
(OLS); we provide details on this regression in Supplement F.1. Our results show that the treatment group outperforms the control group by 0.156 standard deviations ($p < 0.05$, see Fig. 2). This effect remains similar when we control for baseline covariates, school fixed effects, and grade-level fixed effects, showing gains of 0.150 standard deviations ($p < 0.05$, see Fig. 2b, column 2). Based on some estimates from the literature (*42, 43*), the average gain of 0.150 standard deviations is equivalent to approximately six to nine months of additional schooling;[4] while these numbers may differ from one domain to another, they illustrate that the improvements we obtain are substantial. Our results demonstrate that integrating interventions such as a personalized problem sequencing can substantially improve the efficacy of GenAI tutors. We provide additional robustness checks in Supplement F.1, including analyses using alternative standard error specifications (heteroskedasticity-robust and school-clustered) and the raw scores as the outcome; our findings are robust across specifications.

4.2 Closing the Gap: Heterogeneity and Equity

Next, we explore treatment heterogeneity to understand whether the effects of personalized problem sequencing vary by student background. First, we classify students into two types based on their prior Python skill—specifically, beginners (53%, consisting of students who self-identified as "first-time learners" or "beginners" in the baseline survey) and those with prior Python skill (39%, consisting of students who self-identified as having "intermediate" or "proficient" coding skills in the baseline survey).[5] Our results (see Fig. 3) indicate that beginners

[4]Under the World Bank's "Equivalent Years of Schooling" (EYOS) framework (*42*), the pooled Skills Toward Employability and Productivity (STEP) benchmark implies approximately 0.15–0.21 SD of learning per school year. However, the framework highlights substantial cross-system heterogeneity, with higher-performing settings (e.g., the U.S. and Finland) showing approximately 0.20–0.30 SD annual learning gains per year. Using the faster rate as a more comparable benchmark for Taiwan implies that a 0.15 SD improvement in learning (our main finding) translates to approximately 6-9 months of schooling.

[5]We exclude 6 students (1%) who reported national competition-level Python skill, since they had already mastered the curriculum prior to the course and would reflect ceiling effects. 7% of the students did not complete the baseline survey and are excluded from this analysis as well.



(a) Average standardized score by treatment condition.

	Dependent variable:	
	Standardized exam score	
	(1)	(2)
Treatment	0.156*	0.150*
	(0.072)	(0.067)
Controls	No	Yes
School fixed effect	No	Yes
Grade-level fixed effect	No	Yes
Observations (N)	770	716
Adjusted R^2	0.005	0.226

Notes: Standard errors in parentheses. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

(b) OLS result summary.

Figure 2: **Effect of personalized problem sequencing on exam performance.** (a) Dots show average standardized score by group with standard error. The bracketed annotation reports the two-sided t-test comparing average standardized scores between the treatment and control groups ($p = 0.031$), indicating a statistically significant difference in mean performance. (b) OLS estimates of the effect of personalized problem sequencing on standardized exam scores. Column (1) presents a baseline specification without controls, and Column (2) includes baseline controls and school and grade-level fixed effects. The treatment coefficient is positive and statistically significant across specifications.

with no prior skill benefit more from personalization: among these students, personalization increases performance by 0.215 SD ($p = 0.012$), whereas effects are negligible for students with prior skill (0.008 SD, $p = 0.941$). This pattern is consistent with the literature that adaptive systems often deliver the largest gains for less-experienced learners (*50*). Fixed problem sequences are typically calibrated for the median student, and lack the flexibility to accommodate heterogeneity in learning speeds. Personalized problem sequencing addresses this issue by adapting difficulty based on the student's current performance.

We also examine the impact of our intervention for different school tiers—in Taiwan, high school admission is determined by a national standardized exam, the Comprehensive Assessment

Program (CAP), so schools are clearly ranked by academic selectivity. Specifically, we classify schools as higher- or lower-tier based on entrance exam scores, and examine whether gains vary with school selectivity. As shown in Fig. 3, estimated effects are positive for both school types, with larger gains in lower-tier schools (0.173 SD, $p = 0.045$) and a smaller, statistically insignificant estimate in higher-tier schools (0.039 SD, $p = 0.752$). For both prior Python skill and school tier, we also estimate interaction regressions with terms *RL*$\times$*subgroup*, and find consistent results; see Supplement F.3 for details.

Overall, these results suggest that our personalized problem sequencing intervention can deliver equitable learning gains without widening educational inequality, and may even help bridge the gap between advantaged and disadvantaged students.

4.3 Potential Mechanisms: Productive AI Use and Sustained Engagement

Finally, we examine the mechanisms underlying the performance gains from our personalized problem sequencing algorithm. First, we find that students in the treatment group interact with the tutor chatbot in more productive ways. Second, we find that our performance gains are driven by increased engagement (measured by time-on-task and persistence), but not by completing more problems or being pushed toward uniformly harder material. Together, these results suggest that personalized problem sequencing has the potential to improve student engagement, addressing the retention challenge in existing educational technologies (*44–46*).

More productive use of the tutor chatbot. First, we analyze student interactions with our tutor chatbot. From the student-chatbot conversation history for each practice problem, we construct a chat quality score (on a scale of 1-10) by using LLM-as-a-judge (*51*) to distinguish productive conversations (e.g., explanation-seeking and iterative debugging) from unproductive ones (e.g., answer-seeking); see Figure 1(a)-(b) for examples. Recall that all students have access

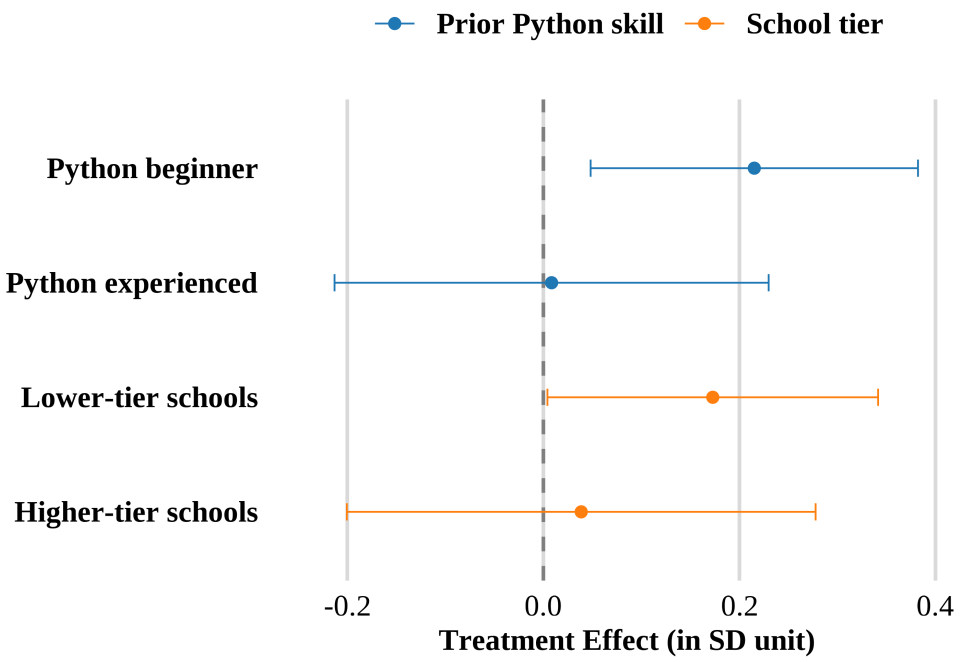
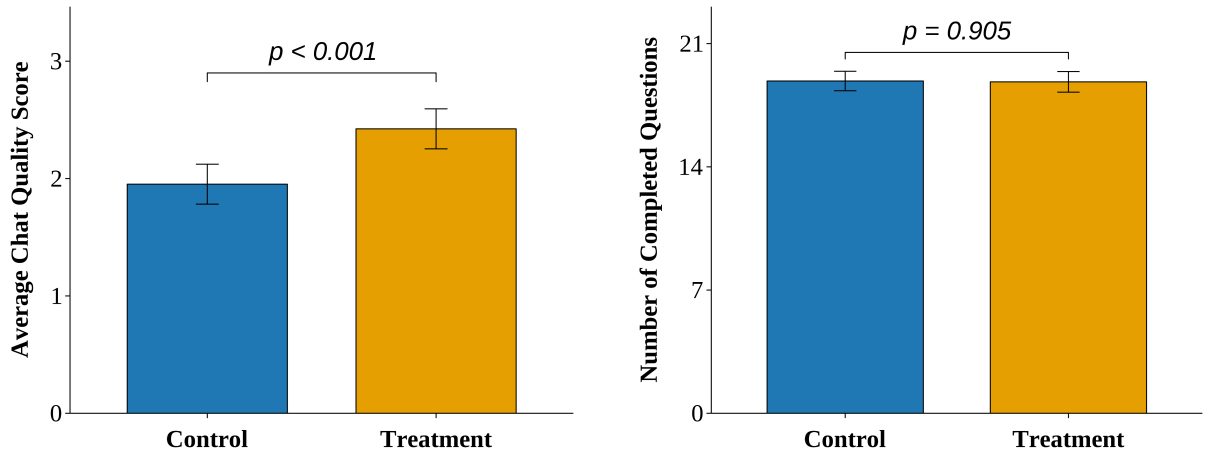


Figure 3: **Treatment effects by prior Python skill and school tier.** The plot reports subgroup-specific treatment effects on the standardized outcome (in units of SD of exam performance across all students). Points denote estimated effects and horizontal bars denote 95% confidence intervals; the dashed vertical line marks zero effect. Blue indicates subgroups defined by prior Python skill, and orange indicates subgroups defined by school tier.

to the same GenAI chatbot tutor with the same guardrails (e.g., prompted to avoid providing answers unless students first demonstrate substantial effort, see Supplement A.3).

As shown in Fig. 4a, students in the treatment group exhibit significantly higher chat quality according to our metric. This pattern suggests that personalized problem sequencing encourages students to engage with the tutor chatbot in ways that support deeper understanding and active problem-solving, supporting our hypothesis that improving other aspects of GenAI tutors can be more productive than focusing on the tutor chatbot in isolation.

Engagement mediates our performance improvement. Next, we examine the mechanism by which personalized problem sequencing improves exam outcomes. Following a standard



(a) Chat quality score

(b) Number of completed questions

Figure 4: **Engagement metrics by treatment assignment.** (a) Chat quality score comparing treatment and control groups, showing higher quality AI interactions in the treatment group. (b) Number of completed practice problems, showing similar completion rates across groups. Bars show group means with 95% confidence intervals. Bracketed annotations report two-sided t-tests: chat quality shows a statistically significant difference ($p < 0.001$), while completion rates do not differ significantly ($p = 0.905$).

mediation analysis (*52*), we study two engagement mediators: total time spent on the platform (*totalTime*) and total number of attempts (*totalAttempt*). We estimate the mediation models conditioning on the same rich set of observed controls as in our main analysis (e.g., baseline characteristics and fixed effects), which helps mitigate concerns that the mediator–outcome relationship is driven by observable differences in student background or program context. Our results show strong positive indirect effects through both measures (ACME: 0.185 SD via *totalTime* and 0.149 SD via *totalAttempt*, both $p < 0.001$), while the direct effects are small and statistically insignificant (ADE: -0.030 SD, $p = 0.690$; and 0.003 SD, $p = 0.984$) (see Table 16 in Supplement F.4). Given a total effect of approximately 0.15 SD, these estimates imply that the treatment effect is almost fully mediated by increased engagement.[6]

[6]The estimated proportion mediated can exceed 1 in finite samples when the estimated direct and indirect components have opposite signs and/or due to sampling variability; we interpret these estimates as evidence that engagement accounts for essentially all of the total effect.

Nevertheless, because engagement is not randomly assigned, mediator–outcome confounding may still arise from unobserved factors, so we do not interpret these mediation estimates as necessarily causal. To assess how strong such unobserved confounding would need to be to overturn our conclusions, we conduct sensitivity analyses that indicate the mediated effect would be eliminated only under moderate levels of unobserved confounding (see Supplement F.4 for details). These results support our hypothesis that personalized problem sequencing improves learning by sustaining student engagement throughout the semester. We provide detailed results and robustness checks, including an alternative path analysis specification using structural equation modeling (SEM), in Supplement F.4.

Ruling out alternative explanations. Finally, we rule out two alternative explanations for our performance improvements. First, we show evidence that the gains are not driven by students simply completing more practice problems. We find that the average number of questions completed is nearly identical in the treatment and control groups (see Fig. 4b).[7] This result suggests that personalization benefits are derived from effort spent solving each problem instead of the number of problems solved—i.e., from increased practice quality instead of quantity.

Second, while the treatment group encounters harder questions on average, we find evidence that this increase is insufficient to explain our performance gains. We cannot perform either a regression or a mediation analysis on data from our main experiment, since practice problem difficulty is not assigned randomly; thus, we instead use data from our pilot study, where difficulty was assigned randomly.[8] In a specification that does not allow for heterogeneity, average problem difficulty is not statistically significantly related to exam performance (see Table 1, Column 1, where $\beta_{\text{Difficulty}} = 0.417$, $p = 0.283$). Furthermore, if we allow the effect to vary by prior

[7]The majority of students complete no more than the required number of practice problems in the homework.

[8]Students used an earlier version of the platform that offered the same functionalities. Question difficulty was categorized into four discrete levels.

Table 1: **Regression summary: Effects of practice-question difficulty**

	Dependent variable:	
	Standardized exam score	
	(1)	(2)
Difficulty	0.417	0.271
	(0.358)	(0.252)
Prior Python	0.909^{*}	-3.739
	(0.375)	(1.598)
Difficulty × Prior Python		2.542^{*}
		(0.863)
Controls	Yes	Yes
School fixed effect	Yes	Yes
Observations	28	28
Adjusted R^2	0.800	0.905

Notes: Standard errors in parentheses. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$. All models include the same set of controls (AI tutor assignment, prior Python background, programming background, parent education, gender, and study majors) and school fixed effect (Supplement C).

coding background, then we find a significant positive interaction between difficulty and coding background (see Table 1, Column 2, where $\beta_{\text{Difficulty}\times\text{PriorPython}} = 2.542$, $p < 0.05$). These results suggest that assigning harder questions disproportionately benefits students with prior Python skill, and does not help students without prior experience. In other words, problem sequencing must be personalized to provide benefits for all students.

We provide details on these analyses in Supplement C. Collectively, our results suggest that increased engagement rather than solving more practice problems or harder practice problems is the main driver behind our effect.

5 Conclusion

Our work demonstrates that the potential of GenAI tutors to improve educational outcomes lies not only in tutors based on chatbots, but by enabling students to engage productively with practice problems. To test this hypothesis, we have developed a GenAI tutoring system that combines a tutor chatbot with a reinforcement learning algorithm for personalized problem sequencing. Whereas prior algorithms are bottlenecked by coarse performance proxies, our algorithm translates complex behavioral traces including code edits and student-chatbot conversations into rich signals of student mastery. Based on this system, we have performed a randomized controlled trial involving 770 high school students, demonstrating that our system improved performance on an end-of-semester exam by 0.15 standard deviations—equivalent to as much as six to nine months of additional schooling by some estimates—without increasing instruction time or teacher workload. Importantly, our results suggest that these gains were driven by increasing student engagement with the practice problems. Maintaining student engagement has been a key challenge in education, especially for long-term educational technology programs. Our findings demonstrate how personalization—not of the tutor chatbot, but rather of the problem sequence—can improve student engagement in a way that benefits longer term outcomes.

More broadly, while our evaluation focused on introductory Python learning among high school students, introductory programming shares similar features with many other educational settings, including heterogeneity in baseline proficiency, steep learning curves, and the importance of productive struggle. Beyond traditional education settings, these features are also prominent in workplace reskilling, which is becoming an increasingly important problem due to the potential impact of GenAI on the labor market (*7, 8*). Our findings suggest that AI tutoring systems that tightly integrate task sequencing into chatbot tutors have the potential to enable effective and personalized learning/reskilling at scale.

References

1. A. Extance, *Nature* **623**, 474 (2023).

2. H. Bastani, *et al.*, *Proceedings of the National Academy of Sciences* **122**, e2422633122 (2025).

3. J. Wang, W. Fan, *Humanities and Social Sciences Communications* **12**, 1 (2025).

4. D. Sun, A. Boudouaia, C. Zhu, Y. Li, *International Journal of Educational Technology in Higher Education* **21**, 14 (2024).

5. E. Kasneci, *et al.*, *Learning and individual differences* **103**, 102274 (2023).

6. D. M. Johnson, W. Doss, C. M. Estepp, *Journal of Research in Technical Careers* **8**, 1 (2024).

7. T. Eloundou, S. Manning, P. Mishkin, D. Rock, *Science* **384**, 1306 (2024).

8. S. Morandini, *et al.*, *Informing Science* **26**, 39 (2023).

9. J. Cristia, P. Ibarrarán, S. Cueto, A. Santiago, E. Severín, *American Economic Journal: Applied Economics* **9**, 295 (2017).

10. S. Papadakis, *Advances in Mobile Learning Educational Research* **3**, 682 (2023).

11. A. J. Lamb, J. M. Weiner, *International Journal of Education in Mathematics, Science and Technology* **6**, 136 (2018).

12. R. Liu, *et al.*, *Proceedings of the 55th ACM technical symposium on computer science education V. 1* (2024), pp. 750–756.

13. R. E. Wang, A. T. Ribeiro, C. D. Robinson, S. Loeb, D. Demszky, *Annenberg Institute for School Reform at Brown University* (2024).

14. S. Shetye, *Studies in Applied Linguistics and TESOL* **24** (2024).

15. H. Kumar, D. M. Rothschild, D. G. Goldstein, J. M. Hofman, *International Conference on Artificial Intelligence in Education* (Springer, 2025), pp. 60–75.

16. G. Kestin, K. Miller, A. Klales, T. Milbourne, G. Ponti, *Scientific Reports* **15**, 17458 (2025).

17. H. Nam, *et al.*, *arXiv preprint arXiv:2507.16252* (2025).

18. T. H. Trinh, Y. Wu, Q. V. Le, H. He, T. Luong, *Nature* **625**, 476 (2024).

19. J. He-Yueya, N. D. Goodman, E. Brunskill, *arXiv preprint arXiv:2403.02795* (2024).

20. A. M. Sidorkin, *Available at SSRN 5230565* (2025).

21. H. K. Warshauer, *Journal of Mathematics Teacher Education* **18**, 375 (2015).

22. H. Bastani, O. Bastani, A. T.-H. Chung, Incentive-compatible human-ai collaboration via adversarial tasks (2026). Working paper, University of Pennsylvania, The Wharton School.

23. R. A. Bjork, J. Dunlosky, N. Kornell, *Annual review of psychology* **64**, 417 (2013).

24. J. Dunlosky, A. R. Lipko, *Current Directions in Psychological Science* **16**, 228 (2007).

25. S. Poulidis, H. Bastani, O. Bastani, *Available at SSRN 5604932* (2025).

26. B. S. Bloom, *Evaluation comment* **1**, n2 (1968).

27. C.-L. C. Kulik, J. A. Kulik, R. L. Bangert-Drowns, *Review of educational research* **60**, 265 (1990).

28. E. L. Bjork, R. A. Bjork, *et al.*, *Psychology and the real world: Essays illustrating fundamental contributions to society* **2**, 56 (2011).

29. L. S. Vygotsky, *Mind in society: The development of higher psychological processes*, vol. 86 (Harvard university press, 1978).

30. M. Kapur, *Cognition and instruction* **26**, 379 (2008).

31. J. A. Kulik, C.-L. C. Kulik, P. A. Cohen, *American psychologist* **34**, 307 (1979).

32. T. R. Guskey (1986).

33. R. E. Slavin, *Review of educational research* **57**, 175 (1987).

34. B. Clement, D. Roy, P. Oudeyer, M. Lopes, *Journal of Educational Data Mining* **7**, 20 (2015).

35. A. N. Rafferty, E. Brunskill, T. L. Griffiths, P. Shafto, *Cognitive Science* **40**, 1290 (2016).

36. S. Shen, M. S. Ausin, B. Mostafavi, M. Chi, *Proceedings of the 26th conference on user modeling, adaptation and personalization* (2018), pp. 43–51.

37. A. Segal, Y. B. David, J. J. Williams, K. Gal, O. Shalom, *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP)* (2018), pp. 111–119.

38. A. T. Corbett, J. R. Anderson, *User modeling and user-adapted interaction* **4**, 253 (1994).

39. I. Šarić-Grgić, A. Grubišić, A. Gašpar, *User modeling and user-adapted interaction* **34**, 1127 (2024).

40. S. Thrun, *Advances in neural information processing systems* **12** (1999).

41. C. E. Garcia, D. M. Prett, M. Morari, *Automatica* **25**, 335 (1989).

42. D. K. Evans, F. Yuan, *World Bank* (2019).

43. S. J. Ritchie, E. M. Tucker-Drob, *Psychological science* **29**, 1358 (2018).

44. C. H.-H. Tsay, A. K. Kofinas, S. K. Trivedi, Y. Yang, *Journal of Computer Assisted Learning* **36**, 128 (2020).

45. J. Reich, *EDUCAUSE Review (Online)* (2014).

46. G. Eysenbach, *Journal of medical Internet research* **7**, e402 (2005).

47. P. Lewis, *et al.*, *Advances in neural information processing systems* **33**, 9459 (2020).

48. L. P. Kaelbling, M. L. Littman, A. R. Cassandra, *Artificial Intelligence* **101**, 99 (1998).

49. M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, *IEEE Transactions on signal processing* **50**, 174 (2002).

50. S. Doroudi, V. Aleven, E. Brunskill, *International Journal of Artificial Intelligence in Education* **29**, 568 (2019).

51. L. Zheng, *et al.*, *Advances in neural information processing systems* **36**, 46595 (2023).

52. K. Imai, L. Keele, D. Tingley, *Psychological Methods* **15**, 309 (2010).

53. J. B. Carroll, *Teachers college record* **64**, 1 (1963).

54. C. H. Papadimitriou, J. N. Tsitsiklis, *Mathematics of operations research* **12**, 441 (1987).

55. G. Tesauro, G. R. Galperin, *Advances in Neural Information Processing Systems* (1996), vol. 9, pp. 1068–1074.

56. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, *arXiv preprint arXiv:1707.06347* (2017).

57. R. H. Lindeman, P. F. Merenda, R. Z. Gold, *et al.*, *Introduction to bivariate and multivariate analysis*, vol. 4 (Scott, Foresman Glenview, IL, 1980).

58. A. Sen, *et al.*, *International Educational Data Mining Society* (2018).

59. Y. B. David, A. Segal, Y. Gal, *Proceedings of the sixth international conference on Learning Analytics & Knowledge* (2016), pp. 354–363.

60. C. Schatten, Sequencing in intelligent tutoring systems based on online learning recommenders, Ph.D. thesis (2017).

61. G. Zhou, J. Wang, C. F. Lynch, M. Chi, *International Educational Data Mining Society* (2017).

62. S. Shen, M. Chi, *Proceedings of the 2016 conference on user modeling adaptation and personalization* (2016), pp. 37–44.

63. S. Shen, B. Mostafavi, C. Lynch, T. Barnes, M. Chi, *International conference on artificial intelligence in education* (Springer, 2018), pp. 327–331.

64. J. P. Rowe, J. C. Lester, *International conference on artificial intelligence in education* (Springer, 2015), pp. 419–428.

65. J. Beck, B. P. Woolf, C. R. Beal, *AAAI/IAAI* **2000**, 1 (2000).

66. K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark, *International journal of artificial intelligence in education* **8**, 30 (1997).

67. J. F. Pane, B. A. Griffin, D. F. McCaffrey, R. Karam, *Educational Evaluation and Policy Analysis* **36**, 127 (2014).

68. R. Belfer, E. Kochmar, I. V. Serban, *International Conference on Artificial Intelligence in Education* (Springer, 2022), pp. 724–730.

69. S. Ritter, J. Kulikowich, P.-W. Lei, C. L. McGuire, P. Morgan, *Frontiers in artificial intelligence and applications* **162**, 13 (2007).

70. A. P. Jaciw, J. V. Cabalo, M.-T. Vu, *Grantee Submission* (2007).

71. J. F. Pane, D. F. McCaffrey, M. E. Slaughter, J. L. Steele, G. S. Ikemoto, *Journal of Research on Educational Effectiveness* **3**, 254 (2010).

72. J. Roschelle, M. Feng, R. F. Murphy, C. A. Mason, *AERA open* **2**, 2332858416673968 (2016).

73. B. Clément, H. Sauzéon, D. Roy, P.-Y. Oudeyer, *arXiv preprint arXiv:2402.01669* (2024).

74. G. Zhou, H. Azizsoltani, M. S. Ausin, T. Barnes, M. Chi, *International journal of artificial intelligence in education* **32**, 454 (2022).

75. J. Bassen, *et al.*, *Proceedings of the 2020 CHI conference on human factors in computing systems* (2020), pp. 1–12.

76. T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, Z. Popovic, *AAMAS* (2014), vol. 1077.

77. R. Schmucker, N. Pachapurkar, S. Bala, M. Shah, T. Mitchell, *European Conference on Technology Enhanced Learning* (Springer, 2023), pp. 383–398.

78. E. Prihar, A. Haim, A. Sales, N. Heffernan, *Proceedings of the Ninth ACM Conference on Learning@ Scale* (2022), pp. 1–11.

79. W. Harrison, S. Higgins, E. Dobson, G. Uwimpuhwe (2024).

80. N. Alam, B. Mostafavi, S. Tithi, M. Chi, T. Barnes (Proceedings of the 17th International Conference on Educational Data Mining, 2024).

81. E. Naslund-Hadley, *et al.*, *Innovations for Poverty Action* (2025).

82. W. Ma, O. O. Adesope, J. C. Nesbit, Q. Liu, *Journal of educational psychology* **106**, 901 (2014).

End Notes

Acknowledgments We are grateful for the close partnership of the Department of Education of the Taipei City Government, the American Institute in Taiwan, American Innovation Center, and the local non-governmental organization Better Together for NextGen Taiwan. We acknowledge invaluable software development support from Bahati Tadjuidje Boniface, Jerry Chih-Yuan Huang, Zihan (Crescent) Xiong, Haosen Ge, and Hingis Chiao-Hsin Chang. We thank our research assistants—Allan Zhang, Cheng-Ying Wu, Harvani Sumawijaya, and Astrid (Pei-Chen) Hsu—as well as TA lead Hsiao-Chiao Lin, Chia-Chien Yang, and the many TAs and graders from the National Taiwan University. We especially thank the teachers at participating schools for their critical role in program implementation. We also thank Lizzy Pecora for administrative support, along with Eddy Yen-Ting Lin and Peter Jia-Wei Cui (Better Together NextGen Taiwan); Yu-Hsin Cho and Chiu-Fang Chiu of the Taipei City Government Department of Education—together with the Taipei City Government's administrative staff, IT team, and government officials—for implementation and logistical support. Finally, we appreciate administrative support from Jerry Weng and the staff and government officials of the American Institute in Taiwan.

Funding: This research was supported by generous funding from the Wharton AI & Analytics Initiative, Wharton Mack Institute for Innovation Management, an Amazon Research Award

(Fall 2023), the Taipei City Government, and the American Institute in Taiwan.

Author contributions: HB, OB, and AC contributed to the study design and methodology. HB, OB, AC, and BZ contributed to manuscript writing. AC and BZ analyzed the results under the supervision of HB and OB. AC and LK contributed to study implementation.

Competing interests: There are no competing interests to declare.

Ethics & Inclusion Statement: This research was approved by the University of Pennsylvania Institutional Review Board (Protocol # 855951) and granted exempt status under Category 1. It was conducted in close partnership with the Taipei City Government and the American Institute in Taiwan, subject to a formal Memorandum of Understanding (MOU) between the University of Pennsylvania and the Taipei City Government. We collaborated closely with policymakers and frontline personnel, conducted training sessions, and aligned our approach with data privacy policy and existing government system and priorities. Roles and responsibilities were clearly defined in advance.

Materials & Correspondence. Correspondence should be addressed to angelchg@wharton.upenn.edu, hamsab@wharton.upenn.edu, and obastani@seas.upenn.edu

Data and materials availability:

To support further research, all data and code will be made available through a Github repository upon publication. Replication code and data is accessible here now: https://tinyurl.com/yx7vsrtc

Supplementary Information

A describes the personalized AI learning system. B details the RL algorithm (POMDP formulation, belief updates, and MPC planning), along with supporting simulation results. Next, C reports details of the pilot study, D reports the experimental protocol (pre-registration and implementation details), E reports data processing and balance checks, and F reports additional analyses supporting the main evaluation, including robustness checks, heterogeneity, mediation, and student feedback. We summarize related work in G and provide course-related materials in H, including example practice problems and the certification exam.

A Personalized AI Learning System

A.1 introduces the user interface and function of our web platform; A.2 provides the RAG problem-generation pipeline; A.3 describes the embedded AI tutor.

A.1 Web Platform

Our web-based learning platform integrates lecture videos, coding practice problems, and AI tutoring into a single interface. It is integrated with the government's e-learning system (Taipei CooC-Cloud), which students access through government-authenticated accounts linked to their official academic records; this authentication mechanism mitigates spillover concerns as students cannot easily access other accounts, and furthermore, easily adheres to the Personal Data Protection Act (PDPA) by limiting researcher access to de-identified student information.

After logging in, each student is taken to a course dashboard that lists all modules in the certificate curriculum (see Fig. 5). Modules are released sequentially; a student can access module $m + 1$ only after completing the required work in module m. This design enforces a common concept ordering while allowing within-module personalization of practice.

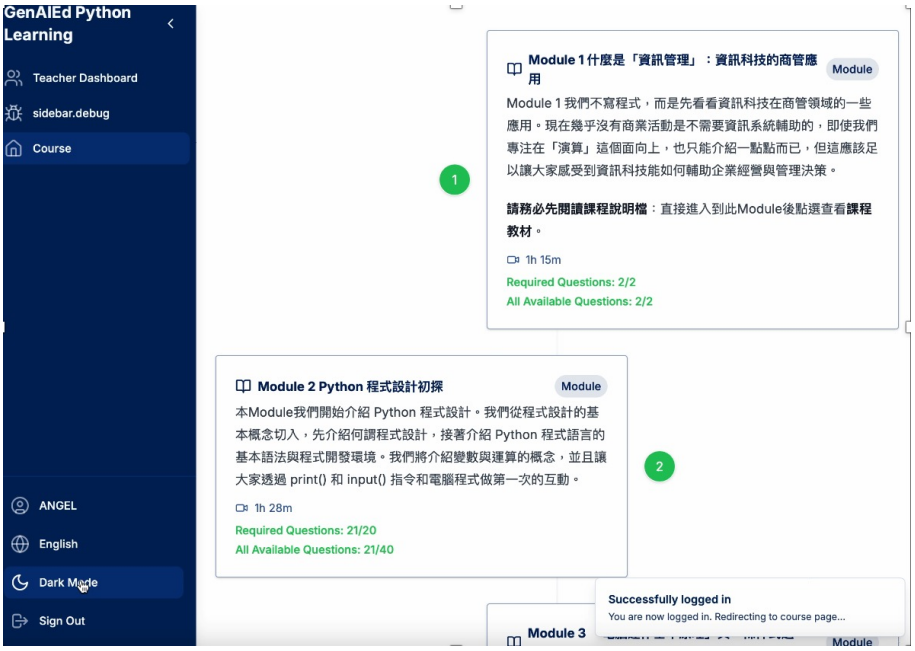


Figure 5: **Menu of Modules on our AI Tutoring Platform**

Within a module, a student starts with a lecture video first. The main interface is divided into three functional areas. As Fig. 6 shows, the upper left panel contains a video player that presents the instructor's lecture for the module. In the upper right, students can work through example problems that mirror those demonstrated in the video, allowing learners to immediately practice the concepts they are seeing in the lecture without leaving the platform.

In the lower-left panel is an LLM-powered AI tutor that is available 24/7 (see Fig. 7). Students can query the tutor with Python-related questions, such as asking for clarification of module concepts, hints on a problem, or explanations of error messages. The tutor is prompted to avoid providing direct answers, and to only provide hints after students have first demonstrated effort to solve the problem.

Once the student feels ready, they can click "Start practicing" to enter the practice mode (see Fig. 8). Practice problems appear one at a time and fall into three types (see examples in H.1): (i) coding questions, where students must write a Python program to satisfy a set of hidden test

Figure 6: **Learning Mode on our AI Tutoring Platform**

cases; (ii) debugging questions, where students must identify and correct errors in provided code; and (iii) true/false questions assessing conceptual understanding. All work is completed directly in the browser-based editor; students do not need to open an external development environment.

For coding and debugging questions, the editor allows students to run code, save intermediate versions, and submit final answers. Submissions are executed on the server, and the platform returns immediate feedback on correctness; when a submission is incorrect, the interface reports failing test cases and/or runtime errors (see Fig. 9a). Students can also report issues encountered during practice, to which the research team responded within 12 hours (see Fig. 9b).

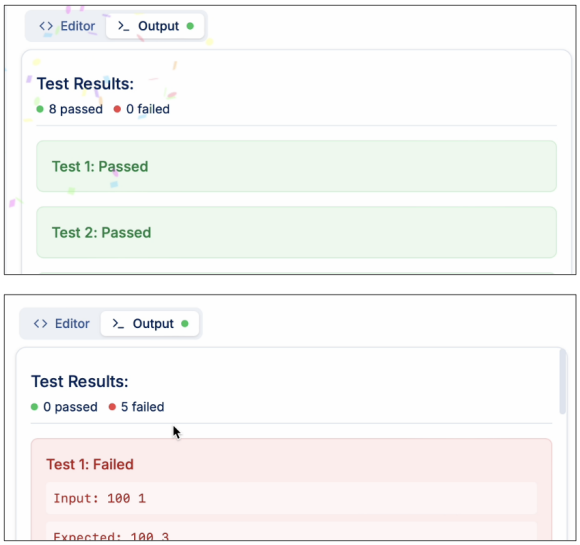
Students can only move to the next question after answering the current question correctly. Each module contains a bank of 40 potential questions; homework assignments typically require each student to complete 15 questions. To ensure comparable exposure across experimental conditions, both treatment and control students were required to complete at least two "very hard," two "hard," and two "medium" questions in each module. Aside from this constraint,

Figure 7: **GenAI Chatbot Tutor on our AI Tutoring Platform.**

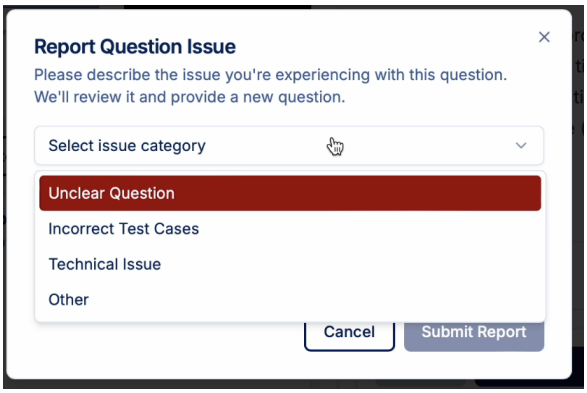
for control students, the sequence follows a fixed ordering from easy to hard, and for treatment students, the RL policy selects the next question adaptively based on the inferred belief state.

The platform records a detailed, time-stamped log of all student activities, including code edits and saved versions, run and submission events, practice outcomes, navigation across modules and questions, and student's conversation history with the AI tutor. These data support both the construction of the learner model used by the RL policy and the subsequent empirical analysis of learning behavior and outcomes.

Figure 8: **Practice Mode**



(a) Immediate practice feedback



(b) Issue reporting

Figure 9: **Student practicing feedback and issue reporting interface.** (a) Students receive immediate feedback after they submit an answer. When tests fail, students can view detailed error information; (b) Students can use the report button to report any issues.

A.2 RAG System for Problem Generation

We generate Python programming questions based on instructional materials using the RAG-based LLM pipeline shown in Fig. 10. For each generation request, we prompt the LLM with (i) the main instructional content, (ii) optional additional content (e.g., prior modules), and (iii) a list of existing questions to discourage duplicates. The model is instructed to output a JSON object containing two coding questions (in English with a traditional Chinese translation), along with a template solution stub, a working sample solution, and a comprehensive assert-based test suite. To encourage variety in the generated questions (e.g., different scenarios and wording), we use sampling settings that allow some randomness (temperature $= 1.0$, top-$p = 0.7$, top-$k = 200$).

The system prompt provides the educational content and high-level role instructions, while the user prompt specifies the required output format and constraints (difficulty levels, bilingual problem statements, and test-suite requirements); we show the system prompt in Figure 11 (we omit the user prompt due to its long length). We generate questions in batches of two.

A.3 GenAI Chatbot Tutor

Tutor integration and shared context. Students interacted with our GenAI tutor (powered by OpenAI's GPT-4o model) embedded in the learning platform during practice. The tutor was available in both the treatment (RL sequencing) and control (fixed sequencing) conditions, and the tutoring model and prompting pipeline were held constant across conditions.

At each practice step, the tutor received a user prompt formed by concatenating the student's query with the current practice problem, relevant metadata (e.g., difficulty), and the sample solution code and/or answer key. The tutor was instructed in the system prompt (see Fig. 12) to keep responses brief and to explain concepts at a middle-school level. This design enabled the tutor to provide accurate, targeted hints while enforcing a strict "no direct answers" policy.

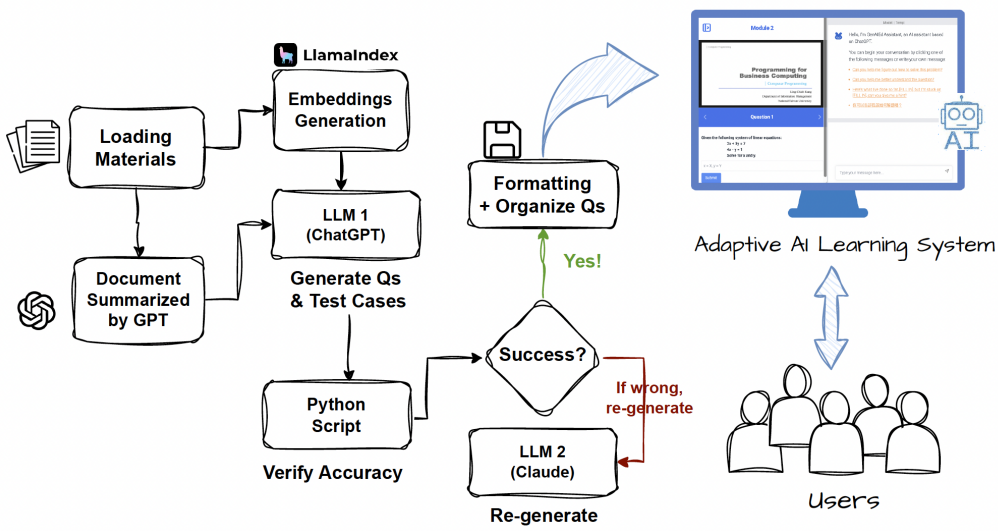


Figure 10: **Human-AI RAG pipeline for practice problem generation.** Course materials and past exams are retrieved as reference contents for an LLM (ChatGPT) to generate programming questions, solutions, and test cases. Each generated output is validated using a Python script that checks functional correctness and consistency. When validation fails, a secondary LLM (Claude) iteratively revises the components until they pass. Teaching assistants then review finalized questions for clarity and quality before deployment.

System prompt

```
You are a knowledgeable AI assistant designed to help generate questions
    based on educational content.
Main content:
{MAIN_DOCUMENT_TEXT}

Additional content:
{ADDITIONAL_DOCUMENT_TEXT}

Your goal is to create well-structured, meaningful, and educational
    coding questions based on the content provided.
You may combine the concepts taught in the additional contents with the
    main content for generating unique and insightful questions.
You should always ensure the correctness of your output and follow the
    question generation format carefully.
When answering, adhere strictly to Python programming conventions and
    ensure the output format is clean and structured.
```

Figure 11: **System prompt for RAG practice problem generation pipeline.**

Base system prompt.

```
You are a Python tutor.  Your goal is to help
students develop a deeper understanding of
Python while keeping them focused on learning.
Keep responses brief, concise, and to the
point---exercise restraint.
Explain to me like I'm a middle school student and
make sure to use the solution code provided in the
context.
```

Figure 12: **Common base system instruction (used in all conditions)**

Pedagogy randomization. The focus of this paper is on RL-based sequencing of practice problem difficulty. However, we also randomized the AI tutor's system prompt in this study (see our pre-registered plan in D.4). All our pedagogy prompts shared the following instructions: (i) do not provide direct answers or full code; (ii) diagnose whether the student's difficulty is problem formulation versus Python syntax/logic and respond accordingly; (iii) keep responses concise; (iv) maintain role consistency; and (v) respond in Traditional Chinese when the student writes in Chinese. They only differed in pedagogical emphasis. We find no meaningful or statistically significant differences across these pedagogy conditions in exam performance or engagement outcomes; accordingly, we do not focus on pedagogy comparisons in this paper. We show that our results are robust to this additional randomization in F.1.

B Reinforcement Learning for Problem Sequencing

In this section, we describe our reinforcement learning algorithm for adaptively sequencing student practice problems. At a high level, we use a model-based reinforcement learning algorithm, where we first predict the parameters θ of a Partially Observed Markov Decision Process (POMDP), and then use optimal control within this POMDP for problem sequencing.

Each POMDP targets a single student-module pair—i.e., the sequence of problems that a specific student works on for a specific module. We train a machine learning model to predict the parameters of this POMDP based on historical data, using features such as student characteristics and their progression in previous modules.

Critically, LLMs perform several key roles in this algorithm: (1) they form the basis of the observation space of our POMDP, analyzing student code submissions to accurately measure progression, (2) they are used to construct several key features for the machine learning model, including analysis of student-tutor interactions. We provide simulation evidence that these features play a critical role in improving the performance of our algorithm, highlighting the benefits of tight integration of AI tutors and POMDP-based problem sequencing.

Due to the limited availability of pilot data, many of our design decisions had to be made on the basis of intuition. Thus, the primary validation of our reinforcement learning algorithm is our randomized controlled trial. Nevertheless, we provide accuracy and simulation-based results to help justify some of these decisions.

B.1 describes the POMDP design; B.2 details our model-based reinforcement learning approach; B.3 outlines the simulation setup and baseline implementations; and B.4 reports the simulation results, including empirical evidence for the value of using LLM-derived signals to estimate learning progression.

B.1 Partially Observable Markov Decision Process (POMDP)

We begin by providing a high-level overview of the POMDP, and then provide details on each component of the POMDP.

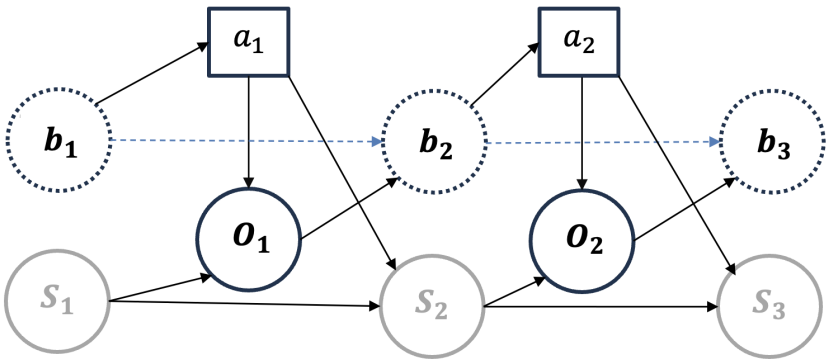


Figure 13: **POMDP-Based Reinforcement Learning System**

B.1.1 Overview

Modeling a student's learning progression is challenging because knowledge is *latent* (i.e., unobserved); it evolves with practice and generates only noisy observations (e.g., how long they take to solve a problem). Partially observed Markov Decision Processes (POMDP) have been applied to solving this challenge. Following this line of work, we formulate the adaptive sequencing of our practice problems as a POMDP. This framework enables the system to select the next difficulty level while uncertain about the student's underlying state, refining its estimate via Bayesian filtering as new interaction data arrive.

Formally, our POMDP consists of: (i) a latent learner state s_t, (ii) an action a_t selecting the difficulty, (iii) an observation o_{t+1} summarizing the student's effort and performance, and (iv) a belief state b_t representing a probability distribution over plausible learner models. At each decision epoch $t \in \{0, 1, \ldots, T_{\max} - 1\}$, the policy selects a *difficulty level* for the next practice problem (in our setting, Easy/Medium/Hard/Very Hard) based on the current belief state b_t. After the student attempts the problem, the platform observes o_{t+1} and computes the belief state b_{t+1} for the next step. Fig. 13 illustrates this system.

Our modeling choices diverge from standard approaches in the literature like Bayesian

Knowledge Tracing (BKT). BKT uses a very simple POMDP that includes binary performance feedback (whether the student successfully solves the problem) to track a binary mastery indicator (whether the student has learned the concept). In our setting, however, binary feedback is uninformative: students iterate until they reach a correct solution, making eventual correctness universal; furthermore, correctness on the very first try is rare (due to syntax or other superficial errors). To obtain a signal that meaningfully tracks progress, we could use the number of attempts required to reach a solution. However, the raw attempt counts (i.e., how many times the student submitted the code to our server to check for correctness) can be inflated by superficial resubmissions, with variation across students on how frequently they make submissions (see our analysis in B.4).

To this end, we need to filter attempts to count *meaningful* submissions. Naïve metrics such as edit distance may not accurately capture whether one code submission has substantially semantic differences from another. Thus, a key feature of our system is to instead employ an LLM to identify meaningful attempts from the student's code edit trajectory. Specifically, we prompt an LLM to take the submission history as input and output whether the latest one is meaningful according to a set of guidelines. Given a sequence of student submissions, we provide this sequence to the LLM and ask it to count the number of meaningful code submissions; we show the prompt we use to do so in Figure 14.

This design naturally leads to a discrete observation space $o_t \in \mathbb{N}$; we considered a number of different distributions, and found that Poisson distributions best matched our pilot data. To handle the increased complexity of the observation space, we also need to increase the complexity of the latent state; otherwise, the latent state would fail to accurately capture variation in observations. Since Poisson distributions have a single real-valued parameter, we correspondingly use a real-valued latent state $s_t \in \mathbb{R}$.

The transition function for our POMDP is hand-designed to match patterns observed in the

Prompt for counting meaningful attempts

```
You are given a list of coding history for a user which contains
various code submissions the question below.

{coding_history}

Question:
{question.content}

Count the number of meaningful code submissions in the user's
history.
For each code submission in the history, compare it with its
previous submission and count it as 1 ONLY if it introduces a
meaningful change in terms of logic, approach, or functionality.
The same code as the previous submission or like small typos,
formatting, or minor variable name changes without changing logic,
or provide empty or nonsense code should be counted as 0.
The final count must be an integer between 0 and
{len(coding_history_list)}.  Return only the integer value with
no additional text.
```

Figure 14: **Prompt used to ask an LLM to count the number of meaningful submissions.**

historical data, including the following salient features: (i) students start from some high initial number of attempts, (ii) students gradually make fewer attempts as they solve problems, and (iii) students converge to some baseline number of attempts. In addition, our transitions assume that students learn more when given harder problems.

Finally, the main parameters of our POMDP are the student's initial state, learning speed, and asymptotic mastery. Given these parameters, the state transitions are deterministic; thus, our belief state is a distribution over these parameters. For state estimation, we use a modified version of a standard Bayesian particle filter for belief state estimation; roughly speaking, the particles come from historical data (specifically, from our pilot), and we maintain a distribution over these particles. Then, belief state estimation amounts to performing posterior updates to this distribution. For control, we use a stochastic model predictive control strategy, where we estimate the value of each action using Monte Carlo rollouts and then select the action with the

best estimated value.

B.1.2 Latent States, Actions, Transitions, and Observations

The design of our POMDP is guided by a key insight from our pilot data—namely, the distinction between "fast" and "slow" learners (here, we use the term "learner" to refer to a student-module pair). Even among students who ultimately performed well, we observed substantial heterogeneity in the speed of progression through difficulty levels. Slower progression did not imply lower capability, but instead reflected a different learning speed that required more practice opportunities before advancing. This pattern aligns with Mastery Learning (*26*) and Carroll's Model of School Learning (*53*), which emphasize that aptitude is closely related to the time required to learn rather than whether learning is possible.

Designing a POMDP with a real-valued latent state can lead to a complicated belief state estimation procedure. Thus, to simplify the design of our POMDP, we encapsulate the uncertainty over the latent state in a set of time-invariant parameters θ, which are specific to the current learner (i.e., student-module pair). For a fixed θ, the POMDP transitions are deterministic, depending only on the selected actions. With these two choices, the belief state can be expressed purely as a distribution over θ, with the corresponding distribution s_t over the latent state being derived by simulating the deterministic POMDP transitions for the chosen actions. The parameter vector $\theta = (c_1, c_2, \tau)$ includes the following information:

- c_1 controls the learner's *initial mastery*
- c_2 is the learner's long-run *mastery ceiling* (the asymptotic attempt rate under mastery)
- τ controls the speed at which mastery improves (note that lower latent state corresponds to higher mastery).

These parameters are estimated from normalized historical data and lie within the compact set

$$c_1, c_2 \in [1.0, 5.0], \qquad \tau \in [0.1, 0.5].$$

Learning proceeds over a finite horizon of at maximum $T_{\max} = 40$ steps.[9] The latent effort variable SE_t denotes the learner's underlying expected effort required at time t to solve a practice problem correctly (e.g., expected number of meaningful attempts on a reference task); importantly, lower values indicate higher proficiency. The full latent state is

$$s_t = (\theta, t, \mathrm{SE}_t) = (c_1, c_2, \tau, t, \mathrm{SE}_t).$$

The initial effort level is specified as

$$\mathrm{SE}_0 = c_1 + c_2.$$

At each step t, the platform selects a difficulty level

$$a_t \in \mathcal{A} = \{0, 1, 2, 3\},$$

with larger values indicating harder questions. Conditional on (s_t, a_t), the latent effort evolves according to the transition function

$$\mathrm{SE}_{t+1} = \big(\mathrm{SE}_t - c_2\big) \exp(-\Delta_t) + c_2, \tag{1}$$

where

$$\Delta_t = \tau(\eta + a_t \delta_{\text{knowledge}}), \tag{2}$$

where $\eta = 1.2$, $\delta_{\text{knowledge}} = 1$. Because $\mathrm{SE}_0 > c_2$ and $\Delta_t > 0$, Eq. (1) says the expected effort SE_t (e.g., expected number of attempts on a reference task) required to achieve a correct solution decreases monotonically toward c_2, with faster convergence for larger τ and higher assigned difficulty a_t.

[9] 40 is the maximum number of questions available for each module.

After the learner interacts with the practice problem chosen at time t, the platform observes a noisy effort signal $o_{t+1} \in \mathbb{N}_0$. In our implementation, o_{t+1} is the LLM-estimated number of *meaningful* attempts required to reach a correct solution (i.e., substantively distinct solution attempts), which filters out superficial resubmissions and better reflects the student's cognitive effort and learning progression. We model o_{t+1} as a Poisson random variable, which is well-suited for nonnegative count data and captures the empirical pattern that effort becomes both larger on average and more variable for harder tasks:

$$o_{t+1} \mid s_{t+1}, a_t \sim \text{Poisson}(\lambda_{t+1}),$$

with rate

$$\lambda_{t+1} = \max\{\text{SE}_{t+1}(1 + a_t)(1 - \text{slip}), \epsilon\} \tag{3}$$

where $\text{slip} = 0.1$ denotes the probability of an incorrect response despite having the required knowledge, and $\epsilon = 10^{-6}$ is a small constant for numerical stability.

Thus, harder tasks (i.e., larger a_t) tend to induce larger observed effort, all else being equal. Under the Poisson model, larger λ_{t+1} implies both a higher mean and a higher variance, which parsimoniously reflects increased dispersion in attempt counts on more difficult items.

B.1.3 Belief State and Belief State Estimation

The true state is unobserved because θ and SE_t are latent. Thus, we maintain a belief distribution over the learner's latent parameters θ and propagate an associated derived effort trajectory. For computational feasibility, we use a standard particle filtering algorithm for belief state estimation. Concretely, we approximate the belief state over the hyperparameter θ based on a finite set of plausible hyperparameter choices (called *particles*):

$$\theta^{(j)} = \left(c_1^{(j)}, c_2^{(j)}, \tau^{(j)}\right), \qquad j \in [n] = \{1, \ldots, n\}.$$

These particles are constructed from historical data, as we describe in B.2. The underlying effort $\mathrm{SE}_t^{(j)}$ is updated online for each particle via the transition function; that is, the belief space is approximated by propagating particles over θ according to the transition function. In more detail, we represent the belief state b_t at time t as a distribution over the time-invariant parameters θ, with

$$b_t(\theta) \approx \hat{b}_t = \sum_{j=1}^{n} p_t^{(j)} \delta_{\theta^{(j)}}(\theta), \qquad p_t^{(j)} \geq 0, \qquad \sum_{j=1}^{n} p_t^{(j)} = 1, \tag{4}$$

where $\delta_{\theta^{(j)}}$ is the Dirac measure at $\theta^{(j)}$. The corresponding distribution over the latent state is

$$s_t^{(j)} = \left(\theta^{(j)}, t, \mathrm{SE}_t^{(j)}\right).$$

It remains to describe how our belief state is initialized and updated. First, the initial belief state $\hat{b}_0$ is based on predictions from historical data, as described in B.2. Next, given belief $\hat{b}_t$, action a_t, and observation o_{t+1}, we update particle weights via three steps:

- **Step 1: Predictive propagation of derived effort.** For each particle j with state

$$s_t^{(j)} = \left(\theta^{(j)}, t, \mathrm{SE}_t^{(j)}\right), \tag{5}$$

  we propagate its derived effort using the transition function:

$$\mathrm{SE}_{t+1}^{(j)} = \left(\mathrm{SE}_t^{(j)} - c_2^{(j)}\right) \exp\left(-\Delta_t^{(j)}\right) + c_2^{(j)}, \tag{6}$$

  where

$$\Delta_t^{(j)} = \tau^{(j)} \left(\eta + a_t \delta_{\text{knowledge}}\right). \tag{7}$$

  The particle $\theta^{(j)}$ itself remains unchanged, while $\mathrm{SE}_t^{(j)}$ evolves deterministically given a_t.

- **Step 2: Likelihood weighting using the Poisson observation model.** For each particle, the corresponding Poisson rate is

$$\lambda_{t+1}^{(j)} = \max\{\mathrm{SE}_{t+1}^{(j)}(1 + a_t)(1 - \text{slip}), \epsilon\}, \tag{8}$$

so the likelihood of observing o_{t+1} is

$$\ell^{(j)} = P(o_{t+1} \mid \theta^{(j)}, \mathrm{SE}_{t+1}^{(j)}, a_t) = \frac{\left(\lambda_{t+1}^{(j)}\right)^{o_{t+1}} \exp\left(-\lambda_{t+1}^{(j)}\right)}{o_{t+1}!}. \tag{9}$$

We compute unnormalized posterior weights

$$\tilde{p}_{t+1}^{(j)} = p_t^{(j)} \ell_j, \qquad j = 1, \ldots, n, \tag{10}$$

and normalize:

$$p_{t+1}^{(j)} = \frac{\tilde{p}_{t+1}^{(j)}}{\sum_{j'=1}^{n} \tilde{p}_{t+1}^{(j')}}, \qquad j = 1, \ldots, n. \tag{11}$$

- **Step 3: Posterior belief update.** The updated belief over θ is again represented as

$$b_{t+1}(\theta) \approx \hat{b}_{t+1}(\theta) = \sum_{j=1}^{n} p_{t+1}^{(j)} \delta_{\theta^{(j)}}(\theta), \tag{12}$$

and posterior expectations of any function $h(\theta)$ are approximated by

$$\mathbb{E}_{b_{t+1}}[h(\theta)] \approx \sum_{j=1}^{n} p_{t+1}^{(j)} h\big(\theta^{(j)}\big). \tag{13}$$

B.1.4 Reward Function

Our ultimate objective is to help learners achieve mastery of the material. While this goal can be captured by assessing whether the latent state converges to its asymptotic value, such an objective misses many nuances about whether the learner was engaged and whether they found the practice problems to be challenging yet rewarding to solve. These nuances are all critical for sustaining learning.

Thus, we instead design a reward function that captures the idea that students should solve problems at a difficulty level appropriate for their current skill level, which has been extensively studied as the *zone of proximal development (ZPD)* (*29*). There is substantial evidence that keeping students in the ZPD maintains engagement while improving learning. Concretely, our

reward function specifies the best difficulty level for matching the learner's current skill level, thereby increasing difficulty at a pace that reflects their personal learning speed τ. An additional advantage of this approach is that it provides a shaped reward function that guides our control algorithm described in B.1.5.

A challenge with this approach is that it requires substantial human expertise to specify the rate of progression. The pacing encoded by our reward function is based on expert pedagogical knowledge and was calibrated using pilot study trajectories, where we observed the variation in how quickly students progressed even among those who performed well. Our pacing is calibrated to accommodate both fast and slow learners, and to avoid penalizing students who require more attempts in the short run but are on track to achieve mastery.

Specifically, let SE_t be a measure of required effort to succeed where *lower is better*. We define normalized skill SE_t as:

$$\mathrm{NSE}_t = \frac{\mathrm{SE}_t - c_2}{c_1} \tag{14}$$

so lower NSE_t corresponds to fewer attempts/time and thus higher proficiency. Then, we compare NSE_t to calibrated thresholds

$$(\mathrm{threshold}_1, \mathrm{threshold}_2, \mathrm{threshold}_3) = (0.2, 0.4, 0.6),$$

where the thresholds are hyperparameters that we manually selected based on the pilot data. Then, we define the desired (appropriate) difficulty level as

$$a_{\mathrm{exp}} = \sum_{i=1}^{3} \mathbb{1}(\mathrm{NSE}_t \leq \mathrm{threshold}_i)\,, \tag{15}$$

so lower NSE corresponds to higher desired difficulty (from 0 to 3). Finally, the reward for choosing action a_t at time t is

$$r_t = \begin{cases} 1 & \text{if } a_t = a_{\mathrm{exp}}, \\ \max\{0, 1 - 0.25 \cdot |a_t - a_{\mathrm{exp}}|\} & \text{otherwise.} \end{cases} \tag{16}$$

This function assigns a full reward of 1 when the chosen difficulty matches the desired level, and penalizes deviations linearly in $|a_t - a_{\text{exp}}|$. The corresponding objective for a control policy π is to maximize the expected cumulative reward:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T_{\max}-1} r_t \right] \tag{17}$$

where the expectation is taken with respect to the Markov chain induced by taking actions according to π.

B.1.5 Model Predictive Control

Next, we describe our control algorithm for selecting actions (i.e., practice problem difficulty levels) to maximize our POMDP objective. Computing the exact optimal solution is computationally intractable due to the large state space (*54*). Thus, we select actions online using model predictive control (also called receding-horizon control) in conjunction with Monte-Carlo rollouts (*55*). Recall that at step t, the belief state b_t is represented by a distribution over particles, which we denote $\{(s_t^{(j)}, p_t^{(j)})\}_{j=1}^n$. To select the next action, we evaluate each candidate action $a \in \mathcal{A}$ by estimating its expected cumulative shaped reward over a lookahead horizon H. Specifically, for each action a, we sample M Monte-Carlo rollouts. Specifically, for each rollout $m \in [M]$ (we use $M = 50$), we first sample an initial latent state index $j \sim \text{Categorical}(p_t^{(1)}, \ldots, p_t^{(n)})$ and set the simulation state $s_{t,0}^{(m)} = s_t^{(j)}$. Then, we simulate the POMDP over $H = 10$ steps by (i) taking action a at the first step and following a handcrafted policy $\tilde{\pi}$ thereafter; (ii) propagating the latent dynamics via (1)–(2); (iii) sampling observations from the Poisson model (3); and (iv) accumulating the shaped rewards defined in (16). For $\tilde{\pi}$, we randomly sample a state from the current belief state and then act optimally according to that state. This strategy yields the following Monte-Carlo value estimate:

$$\widehat{Q}_t(b_t, a) = \frac{1}{M} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \gamma^h r_{t+h}^{(m)}, \tag{18}$$

where γ is the discount factor (we use $\gamma = 1$ since our horizon is short). We select the action maximizing this value:

$$a_t^* \in \arg\max_{a \in \mathcal{A}} \widehat{Q}_t(b_t, a), \tag{19}$$

execute a_t^* on the platform, and update the belief given the realized observation o_{t+1} (B.1.2). We re-plan at every time step using the current belief state.

B.2 Model-Based Reinforcement Learning

A key remaining question is what to use as the initial belief state over the POMDP parameters (c_1, c_2, τ). One strategy is to use a fixed choice across all learners, relying on the POMDP to update them correctly over time. However, the time horizon of our POMDP is relatively short, leaving little time to obtain accurate estimates for choosing high-reward actions. Thus, we use a model-based reinforcement learning approach, where we predict the parameters from historical data, and form an initial belief state based on these predictions.

Specifically, we train a machine learning model to predict the POMDP parameters based on features derived from behavioral logs from earlier modules, including time-stamped code edits, coding history, and AI-tutor interactions. For our machine learning model, we use a gradient-boosted model (GBM), which achieved the best predictive performance among the models we evaluated. Specifically, we evaluated the performance of different models on data from our pilot; results in B.4 show that GBMs substantially outperform other model families. In our deployment we used our data from our pilot to train the GBM in Modules 1–3 since data from our deployment was not yet available. Starting in Module 4, we instead train the GBM using data from earlier modules.

The outcome predicted by the GBM is whether the student is a slow or fast learner—i.e., the value of $\tau \in \{\tau_{\text{high}}, \tau_{\text{low}}\}$. We found that this was the most important parameter to achieving good performance; furthermore, it was much more difficult to get good accuracy at predicting c_1

and c_2. We estimate the prediction targets for historical data by curve fitting. Specifically, for each given sequence of observations, we compute the *maximum a posteriori* (MAP) estimate of the underlying sequence of states $\text{SE}_0, ..., \text{SE}_T$ by maximizing the Poisson log-likelihood over c_1, c_2, τ using the `curve_fit` routine in `scipy`.

Our features include (i) difficulty-level summaries across attempted items (mean, standard deviation, and slope of a least-squares linear fit over time), (ii) time-to-solve summaries in minutes (mean, standard deviation, and the corresponding linear-fit slope), and (iii) difficulty-adjusted time-to-solve (mean and linear-fit slope), where time is adjusted by assigned difficulty to separate effort from practice problem's difficulty level. We also include LLM-based measures constructed from submission trajectories and AI-tutor conversations, including (iv) meaningful attempts (mean, standard deviation, and linear-fit slope), (v) the average growth rate of attempts required to solve a question correctly, and (vi) a chat-quality score capturing the quality of student–chatbot interactions (using the first observed value to reflect initial help-seeking behavior).

Finally, we use the prediction probability p for $\tau = \tau_{\text{high}}$ from the GBM, along with historical data, to construct the initial belief state b_0. Specifically, we aggregate our MAP estimates from the historical data (our pilot data for Modules 1–3 , and data from earlier modules for Module 4 and onwards) to form $\mathcal{D} = \{(c_{1,i}, c_{2,i}, \tau_i)\}_{i=1}^{n}$; these estimates form the particles in our particle filter. Naïvely, we could use the uniform distribution over these particles as b_0; however, this strategy is not personalized to the current learner. Instead, we use the predicted probability p of $\tau = \tau_{\text{high}}$ from the GBM to form the prior over τ (i.e., $P(\tau = \tau_{\text{high}}) = p$ and $P(\tau = \tau_{\text{low}} = 1 - p)$. For c_1, c_2, we construct the datasets

$$\mathcal{D}_{\text{high}} = \{(c_{1,i}, c_{2,i}, \tau_i) \in \mathcal{D} \mid \tau_i = \tau_{\text{high}}\}$$

$$\mathcal{D}_{\text{low}} = \{(c_{1,i}, c_{2,i}, \tau_i) \in \mathcal{D} \mid \tau_i = \tau_{\text{low}}\}.$$

Then, we form the mixture distribution

$$\begin{aligned}\mathcal{D}_p &= p \cdot \text{Uniform}(\mathcal{D}_{\text{high}}) + (1-p) \cdot \text{Uniform}(\mathcal{D}_{\text{low}}) \\ &= p \sum_{\theta \in \mathcal{D}_{\text{high}}} \frac{\delta_\theta}{n_{\text{high}}} + (1-p) \sum_{\theta \in \mathcal{D}_{\text{low}}} \frac{\delta_\theta}{n_{\text{low}}}\end{aligned}$$

where $n_{\text{high}} = |\mathcal{D}_{\text{high}}|$ and $n_{\text{low}} = |\mathcal{D}_{\text{low}}|$. In other words, we use a uniform distribution over $\mathcal{D}_{\text{high}}$ with probability p, and a uniform distribution over $\mathcal{D}_{\text{low}}$ with probability $1-p$. We use $b_0 = \mathcal{D}_p$ as our initial belief state.

B.3 Simulation Experiment

Next, we describe the setup of our simulation experiment used to evaluate our approach. This experiment is performed entirely on our pilot data (described in C), where problem sequences are randomized.[10] Our results (described in B.4) demonstrate the value of both our reinforcement learning algorithm as well as our use of LLM-derived features and observations.

B.3.1 Experimental Setup

Dataset. Our pilot dataset consists of 43 students with module records from 3 to 15, to simulate the algorithms. We restrict to 37 students who had completed at least 7 of the 15 modules to ensure reliability of the simulation. We randomly split them into a training set of 27 students for fitting the prediction model and a test set of 10 students for evaluating both prediction accuracy and POMDP decision-making performance. For each module, we simulate 15 steps, as homework assignments typically consist of 15 questions per module (see Section 3).

[10]We cannot use data from our main study since problem sequences are either fixed (for the control group) or adaptive (for the treatment group), so there is no random variation that can be leveraged for evaluation purposes.

Algorithm 1 Simulation Evaluation of our POMDP-Based Decision-Making Policy

Input: Student ID, module IDs, ground truth (c_1, c_2, τ), initial belief probabilities $\{p_0^{(j)}\}_{j=1}^n$
Output: Sequences $\{a_t\}, \{o_t\}, \{r_t\}, \{b_t\}$
Initialize:
$\{(c_1^{(j)}, c_2^{(j)}, \tau^{(j)})\}_{j=1}^n \leftarrow$ particles from historical data
$b_0 \leftarrow$ belief state with probabilities $\{p_0^{(j)}\}_{j=1}^n$ over n states
Initialize each state $s_0^{(j)}$ with $SE_0^{(j)} = c_1^{(j)} + c_2^{(j)}$
True State: $SE_0^{\text{true}} \leftarrow c_1 + c_2$

for $t = 1, 2, \ldots, T$ **do**
  Action Selection:
    for *each hypothesis state* $s_{t-1}^{(j)}$ *in belief* b_{t-1} **do**
      $a^{(j)*} \leftarrow$ optimal action selected by our policy for state $s_{t-1}^{(j)}$
    end
    $a_t \leftarrow \arg\max_{a \in \{0,1,2,3\}} \sum_{j:a^{(j)*}=a} p_{t-1}^{(j)}$
    Environment Step:
    Execute a_t; observe o_t, reward r_t
    True State Update:
    $\Delta^{\text{true}} \leftarrow \tau \cdot (1.2 + a_t \delta_{knowledge})$
    $SE_t^{\text{true}} \leftarrow (SE_{t-1}^{\text{true}} - c_2) \cdot \exp(-\Delta^{\text{true}}) + c_2$
    Belief Update:
    for *each hypothesis state* $s_{t-1}^{(j)}$ *in belief* $b^{(t-1)}$ **do**
      State Transition:
      $\Delta^{(j)} \leftarrow \tau^{(j)} \cdot (1.2 + a_t \delta_{knowledge})$
      $SE_t^{(j)} \leftarrow (SE_{t-1}^{(j)} - c_2^{(j)}) \cdot \exp(-\Delta^{(j)}) + c_2^{(j)}$
      $s_{t-1}^{(j)} \leftarrow$ update state with new $SE_t^{(j)}$
      Observation Likelihood:
      $\lambda^{(j)} \leftarrow \max(SE_t^{(j)} \cdot (1 + a_t)(1 - \text{slip}), \epsilon)$
      $\ell^{(j)} \leftarrow P(o_t \mid s_t^{(j)}) = \frac{(\lambda^j)^{o_t} \exp(-\lambda^{(j)})}{o_t!}$
    end
    Belief Normalization:
    $p_t^{(j)} \leftarrow \frac{p_{t-1}^{(j)} \cdot \ell^{(j)}}{\sum_{i=1}^n p_{t-1}^{(i)} \cdot \ell^{(i)}}$ for all j
    $b_t \leftarrow$ updated belief with states $\{s_t^{(j)}\}$ and probabilities $\{p_t^{(j)}\}$
    if *maximum steps reached or all modules are completed* **then**
      break
    end
end
return All sequences

Methodology. First, we describe our simulation environment. The main information required to construct our simulation environment is an initial state distribution. To this end, we use the method described in B.2 to estimate the initial state of each student in our dataset. Then, we use the uniform distribution over the training (resp., test) students as our initial state distribution for our training (resp., test) environment. Then, for each approach (our algorithm or one of our baselines, described below), we perform training on the training environment, and evaluate performance on the test environment. To do so, each algorithm is given simulation access to our training and test environments, which we perform as follows (see Algorithm 1 for a summary). First, we randomly sample an initial state, and initialize our POMDP with that initial state as the true state. Then, at each step, (1) the algorithm selects an action to take based on the observations so far, (2) we transition the true state based on that action, and (3) based on the updated state, we produce an observation (which is always given to the algorithm) and a reward (which is given to the algorithm during training; it is only used for evaluation during testing). At the end of the horizon, we assess performance by summing the rewards accrued across steps. This simulation provides a rigorous held-out evaluation of our methodology under the assumption that our POMDP transitions, observations, and rewards accurately reflect the real task.

Baselines. We compare our adaptive POMDP sequencing algorithm against several baselines, including a binary-state HMM sequencing rule (i.e., the existing Bayesian knowledge tracing algorithm), a LSTM-based PPO policy, random sequencing, and fixed sequencing, as well as ablations of our algorithm that differ in how they initialize and update learner beliefs (no historical data and uniform over historical data). We describe these approaches in detail below.

Algorithm 2 Simulation Evaluation of the HMM-Based Decision-Making Policy

Input: Student ID, module IDs, ground truth (c_1, c_2, τ)
Output: Sequences $\{a_t\}, \{o_t\}, \{r_t\}, \{p_t^{\text{learned}}\}$
Initialize:
$p_0^{\text{learned}} \leftarrow 0$
$a_0 \leftarrow 0$
True State: $SE_0^{\text{true}} \leftarrow c_1 + c_2$
for $t = 1, 2, \ldots, T$ **do**
  Action Selection:
  if $p_{t-1}^{\textit{learned}} \geq 0.95$ ***and*** $a_{t-1} < 3$ **then**
    $a_t \leftarrow a_{t-1} + 1$
  else
    $a_t \leftarrow a_{t-1}$
  end
  Environment Step:
  Execute a_t; observe o_t, reward r_t
  $o_t^{\text{bin}} \leftarrow \mathbf{1}[o_t > 5]$
  True State Update:
  $\Delta^{\text{true}} \leftarrow \tau \cdot (1.2 + a_t \delta_{knowledge})$
  $SE_t^{\text{true}} \leftarrow (SE_{t-1}^{\text{true}} - c_2) \cdot \exp(-\Delta^{\text{true}}) + c_2$
  Belief Prediction:
  $P_{01} \leftarrow 1 - \exp(-\tau(1 + a_t))$
  $p_{t|t-1}^{\text{learned}} \leftarrow p_{t-1}^{\text{learned}} + (1 - p_{t-1}^{\text{learned}}) P_{01}$
  Belief Update:
  $\ell_L \leftarrow P(o_t^{\text{bin}} \mid \text{Learned})$
  $\ell_N \leftarrow P(o_t^{\text{bin}} \mid \text{Not Learned})$
  $p_t^{\text{learned}} \leftarrow \dfrac{p_{t|t-1}^{\text{learned}} \ell_L}{p_{t|t-1}^{\text{learned}} \ell_L + (1 - p_{t|t-1}^{\text{learned}}) \ell_N}$
  if *maximum steps reached or all modules are completed* **then**
    break
  end
end
return All sequences

- **Hidden Markov Model (HMM)**: We first implement a Hidden Markov Model (HMM) baseline method for adaptive difficulty sequencing in the simulation. The method models student learning as a two-state Markov process and employs a threshold-based policy to dynamically adjust question difficulty based on probabilistic estimates of learning state.

The HMM models student learning using a two-state space: (i) State N (Not Learned): The student has not yet mastered the material; performance is characterized by an initial latent effort variable SE_0 that evolves over time, and (ii) State L (Learned): The student has mastered the material; performance is characterized by c_2. The system maintains a belief state $p_{\text{learned}} \in [0, 1]$, representing the probability that the student is in learned state.

The transition from "Not Learned" to "Learned" follows an exponential decay model

$$P(N \to L \mid \tau, a) = 1 - \exp(-\tau(1 + a)).$$

Given the current belief p_t^{learned} and action a_t, the predicted belief after the transition is

$$p_{t+1|t}^{\text{learned}} = p_t^{\text{learned}} + (1 - p_t^{\text{learned}}) \cdot P(N \to L \mid \tau, a_t).$$

For the observations, we convert the Poisson observation o from B.1.2 into a binary variable using a cutoff threshold:

$$o^{\text{bin}} = \begin{cases} 1 & \text{if } o > 5 \\ 0 & \text{otherwise} \end{cases}$$

The observation likelihoods are:

$$P(o^{\text{bin}} = 1 \mid \text{Learned}) = c_2 \qquad P(o^{\text{bin}} = 1 \mid \text{Not Learned}) = SE_t$$

The belief state is updated based on o_{bin} using Bayes' rule:

$$p_{t+1}^{\text{learned}} = \frac{p_{t+1|t}^{\text{learned}} \cdot P(o^{\text{bin}} \mid \text{Learned})}{p_{t+1|t}^{\text{learned}} \cdot P(o^{\text{bin}} \mid \text{Learned}) + (1 - p_{t+1|t}^{\text{learned}}) \cdot P(o^{\text{bin}} \mid \text{Not Learned})}$$

Finally, the action selection policy uses a simple threshold rule:

$$a_{t+1} = \begin{cases} a_t + 1 & \text{if } p_t^{\text{learned}} \geq 0.95 \\ a_t & \text{otherwise.} \end{cases}$$

We show the evaluation of the HMM algorithm in Algorithm 2.

- **Proximal Policy Optimization (PPO)**: We consider using model-free reinforcement learning to train a policy in our simulator. Specifically, we trained a neural network policy using proximal policy optimization (PPO) (*56*); since PPO does not maintain an explicit belief state, we use a long short-term memory (LSTM) policy to learn latent state. Furthermore, we provide the policy with a richer observation vector that summarizes the interaction history—the agent receives a 6-dimensional observation vector at each step t:

$$\mathbf{x}_t = \left(\frac{t}{T},\ \frac{a_{t-1}}{3},\ \frac{o_{t-1}}{25},\ \frac{\bar{o}_{1:t-1}}{25},\ \frac{\hat{SE}_{t-1}}{20},\ \frac{\hat{SE}_{t-1} - \hat{SE}_1}{20} + 0.5 \right),$$

where T is the horizon length, a_{t-1} is the previous action (0 at the start), o_{t-1} is the most recent Poisson observation, $\bar{o}_{1:t-1} = \frac{1}{t-1}\sum_{i=1}^{t-1} o_i$ is the cumulative mean of all past observations, $\widehat{\text{SE}}_{t-1} = o_{t-1}/((1 + a_{t-1})(1 - \text{slip}))$ estimates the underlying success rate, and the trend in estimated success rate (values below 0.5 indicate declining performance, above 0.5 indicate improvement). All components are clipped to $[0, 1]$.

The policy network uses a single-layer LSTM with 128 hidden units, followed by separate 64-unit policy and value heads. The agent is trained using generalized advantage estimation (GAE) with $\gamma = 0.99$ and $\lambda = 0.95$, a linearly decaying learning rate from 3×10^{-4} to 3×10^{-5}, and an entropy coefficient of 0.1 to encourage exploration. The policy is trained on one million timesteps using the training split of student parameters, and evaluated on the held-out test split. While PPO uses the same difficulty-matching reward and monotonicity penalty as the POMDP, we found that training with the identical reward function led the agent to learn a degenerate policy that always selects the hardest difficulty level. To address this issue, we replace the single first-step penalty with an extended early-step penalty over the first three steps:

$$r_t^{\text{early}} = -\max(a_t - t, 0) \cdot \left(1 - \frac{t}{3}\right) \qquad (\forall t \in \{0, 1, 2\}).$$

- **No historical:** We assume no prior knowledge of θ, and maintain a uniform distribution over 10 particles sampled uniformly at random from the parameter domain.
- **Uniform historical:** At each module, we maintain a uniform distribution over the estimated parameters of all students from the previous modules as our belief state. For the first module, we use the estimated parameters of the first 10 students (excluding the current student) from the current module.
- **Greedy:** We replace our policy with a greedy one-step lookahead policy. At each step, we select the action that maximizes the immediate expected reward.
- **Random sequencing:** We uniformly randomly sample $a_t \sim \text{Uniform}(\{0, 1, 2, 3\})$.
- **Fixed sequencing:** We use a fixed sequence of actions across the 15 steps.

$$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3].$$

- **Oracle:** We initialize the belief state with the student's ground-truth parameters (c_1, c_2, τ) for each module, concentrating all probability mass on the true parameter configuration. Since this approach uses the ground truth state, it serves as an upper bound on the performance achievable by any policy in our POMDP framework.

Metrics. Our main outcome is the cumulative reward; average over the horizon to normalize it. While this is a shaped reward, it is intended to capture the desired behavior of our POMDP policy. In addition, we report the average difficulty level of problems (represented as an integer in $\{0, 1, 2, 3\}$) across the horizon. This metric allows us to assess whether each algorithm is correctly assigning harder problems to high-type students than to low-type students.

Method	**Avg Reward** (High τ)	**Avg Reward** (Low τ)	**Avg Reward** (All)	**Avg Diff.** (High τ)	**Avg Diff.** (Low τ)	**Avg Diff.** (All)
ML (Ours)	**0.975**	**0.903**	**0.942**	2.760	2.684	2.725
ML (no LLM)	0.963	0.901	0.935	2.714	2.687	2.702
HMM	0.793	0.527	0.673	2.000	0.000	1.098
PPO	0.904	0.890	0.898	2.512	2.574	2.540
No hist	0.821	0.871	0.843	2.145	2.029	2.093
Uni hist	0.951	0.896	0.926	2.671	2.575	2.628
Greedy	0.950	0.907	0.931	2.667	2.607	2.640
Random seq	0.492	0.508	0.499	1.544	1.578	1.559
Fixed seq	0.492	0.682	0.578	0.800	0.800	0.800
Oracle	0.988	0.975	0.982	2.800	2.265	2.559

Table 2: Comparison of methods by average reward and difficulty across τ groups (difficulty ranges from 0 to 3).

B.4 Simulation Results

Quantitative results. Table 2 shows a comprehensive comparison of all methods; we also break down performance based on the subset of students with $\tau = \tau_{\text{high}}$ vs. $\tau = \tau_{\text{low}}$. The "Oracle" method serves as a performance upper bound, since it knows the true state. Our POMDP-based policy with ML-based initialization ("ML") achieves the highest overall average reward (0.942) and performs consistently well across for both high τ (0.975) and low τ (0.903) students. The ML method without LLM features ("ML no LLM") shows slightly lower performance (0.935 overall), suggesting that LLM-derived features contribute to improved adaptation. The "Greedy" method performs comparably (0.931 overall), demonstrating that while one-step lookahead can be effective, there is value to accounting for long-term rewards. The "PPO" method achieves competitive performance (0.898 overall) and, notably, exhibits very balanced rewards across high- and low-τ groups (0.904 vs. 0.890). These results suggest that PPO learns a robust global strategy, but does not distinguish as effectively between fast and slow learners as our method.

The "Uniform" historical baseline (0.926) performs slightly worse, highlighting that adapta-

tion based on the initial state has some value, though online adaptation can mitigate shortcomings of prediction. However, the "No historical" baseline (0.843) shows substantially lower performance, particularly for high τ students (0.821); thus, using historical data to initialize the belief state is critical. The "HMM" method shows significantly worse performance, suggesting limited ability to adapt to individual students. Both "Random" and "Fixed" sequencing baselines perform poorly (0.499 and 0.578 respectively), which demonstrates the importance of adapting to individual students.

Qualitative examples. To illustrate how different methods adapt to varying student types, we present qualitative examples across four representative cases that capture the key dimensions of student variability: learning speed (τ) and initial topic familiarity (quantified by the initial gap between the true required effort SE_0 and the mastery ceiling c_2) in Fig. 15 (low τ, large gap), Fig. 16 (low τ, small gap), Fig. 17 (high τ, large gap), and Fig. 18 (high τ, small gap).

Prediction model evaluation. Finally, we also analyze the accuracy of our predictive model. First, Fig. 19 summarizes which behavioral signals are most predictive of the learning speed τ. Notably, several of our LLM-derived features achieve high importance. These features are also important according to several other metrics. Specifically, we found that including LLM features increases AUC from 63.66% to 72.44%; see Table 3. These gains are consistent with the fact that meaningful edit attempts and chat-quality features capture engagement and help-seeking behavior that is difficult to measure in traditional tutoring systems in real time.

Efficacy of LLM-based observations. A key feature of our approach is that our observations are based on an LLM judging the number of meaningful attempted solutions, rather than a conventional observation such as the number of raw attempted solutions. Intuitively, raw attempts are very noisy since students often make sequences of submissions with minor edits.

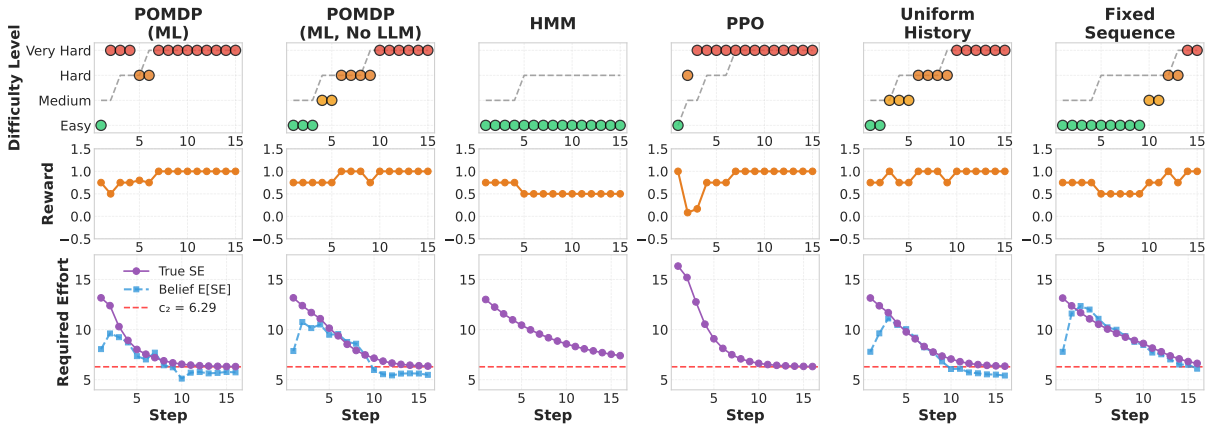


Figure 15: Comparison of adaptive difficulty selection strategies for a student with τ_{low} and large initial gap between the true latent effort variable SE and c_2, indicating limited initial familiarity with the topic. The figure presents three metrics across multiple methods: (top row) **Difficulty Level** showing selected difficulty levels (color-coded: Easy, Medium, Hard, Very Hard) with expected actions as a dashed reference line; (middle row) **Rewards** accumulated at each step; (bottom row) **Required Effort** displaying the true required effort (purple solid line), belief-expected required effort for POMDP methods (blue dashed line), and ground truth c_2 (red dashed line). Each column represents a different method.

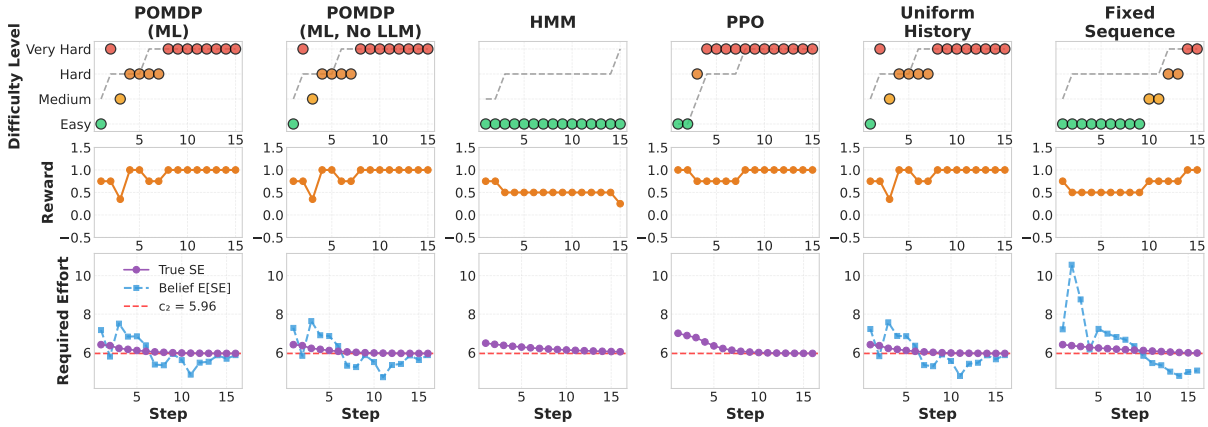


Figure 16: Comparison of adaptive difficulty selection strategies for a student with τ_{low} and small initial gap between the true required effort SE and c_2, indicating high initial familiarity with the topic. Same setup as Figure 15.

To evaluate whether meaningful attempts (*MeaningfulAttempt*) better reflects students' learning progression than the raw number of attempts (*TotalAttempt*), we compare their explanatory power for student learning outcomes. Specifically, we first construct standardized versions of each

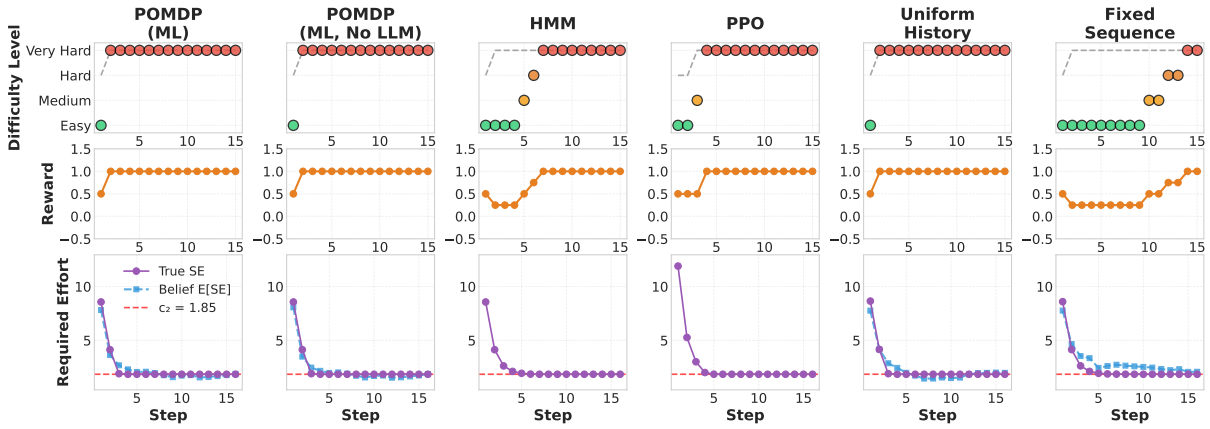


Figure 17: Comparison of adaptive difficulty selection strategies for a student with τ_{high} and large initial gap between the true required effort SE and c_2, indicating limited initial familiarity with the topic. Same setup as Figure 15.

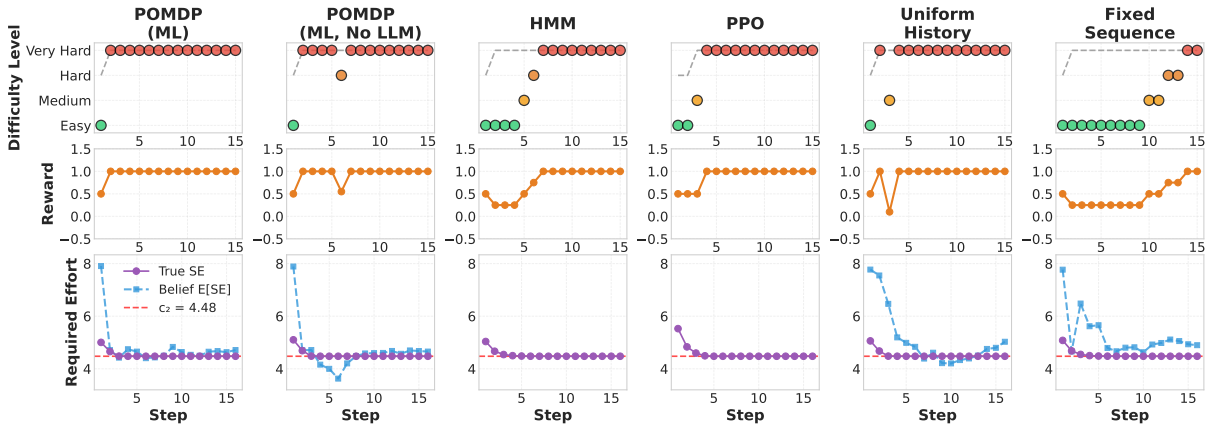


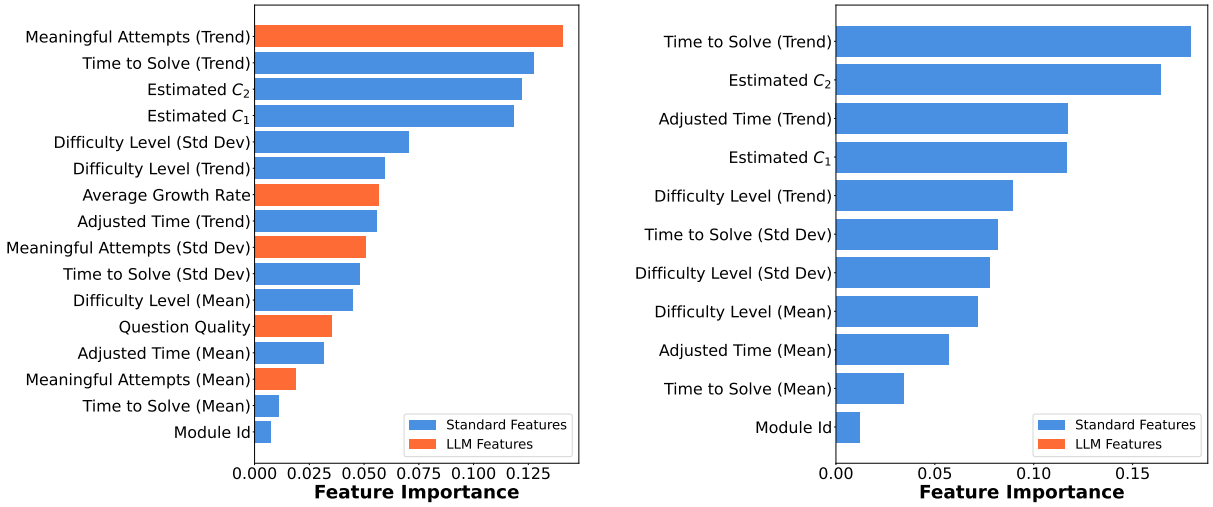
Figure 18: Comparison of adaptive difficulty selection strategies for a student with τ_{high} and small initial gap between the true required effort SE and c_2, indicating high initial familiarity with the topic. Same setup as Figure 15.

predictor (namely, $z^{\text{MeaningfulAttempt}}$ and $z^{\text{TotalAttempt}}$), and then use a sequence of ordinary least squares (OLS) regression models to assess their explanatory power.

First, we consider two univariate specifications:

$$Y_i = \alpha + \beta_1 z_i^{\text{TotalAttempt}} + \varepsilon_i, \tag{20}$$

$$Y_i = \alpha + \beta_2 z_i^{\text{MeaningfulAttempt}} + \varepsilon_i, \tag{21}$$



(a) Feature importance with LLM features

(b) Feature importance for without LLM features

Figure 19: **Feature importance comparison for predicting τ with and without LLM features.**

Table 3: **Classification performance of different models with and without LLM features.** Metrics are computed using a threshold of 0.5. AUC and Accuracy are reported as percentages.

Model Type	LLM Feats	AUC	Acc	TP	FP	TN	FN
Gradient Boosting	Yes	**72.44**	**71.57**	46	19	27	10
	No	63.66	69.61	47	22	24	9
Logistic Regression	Yes	61.88	57.84	31	18	28	25
	No	54.66	53.92	28	19	27	28
Random Forest	Yes	70.30	69.61	41	16	30	15
	No	63.78	59.80	33	18	28	23
XGBoost	Yes	69.57	64.71	42	22	24	14
	No	61.34	65.69	43	22	24	13

as well as a joint specification:

$$Y_i = \alpha + \beta_1 z_i^{\text{TotalAttempt}} + \beta_2 z_i^{\text{MeaningfulAttempt}} + \varepsilon_i. \tag{22}$$

The raw-attempt model in Eq. (20) explains $R^2 = 0.08$ of the variation in student's learning outcome, whereas the meaningful-attempt model in Eq. (21) explains $R^2 = 0.14$. Thus, meaningful attempts account for substantially more outcome variation than raw attempts.

Next, we test whether each observation contributes incremental explanatory power beyond the other using nested-model comparisons. Adding $z^{\text{MeaningfulAttempt}}$ to Eq. (20) produces a large improvement in fit ($F = 67.36$, $p < 0.001$). In contrast, adding $z^{\text{TotalAttempt}}$ to Eq. (21) yields a smaller (though still statistically significant) improvement ($F = 15.66$, $p < 0.001$). These results indicate that meaningful attempts contain substantial information about learning outcomes not captured by raw attempts, but less so vice versa.

Fianlly, because $z^{\text{TotalAttempt}}$ and $z^{\text{MeaningfulAttempt}}$ are correlated, coefficient magnitudes in the joint model can be difficult to interpret due to shared variance. Thus, we complement the nested tests with a variance decomposition based on the Lindeman-Merenda-Gold (LMG) metric (*57*), which tells us the attribution of the model R^2 to regressors averaged over all possible orderings. Applying the LMG decomposition to Eq. (22) shows that $z^{\text{MeaningfulAttempt}}$ accounts for 0.69 of explained variance (69.06%), whereas $z^{\text{TotalAttempt}}$ accounts for 0.31 (30.94%). This result confirms that meaningful attempts contribute most of the explanatory power in the joint model.

Discussion on LLM-derived features and observations. Both our POMDP simulation results and our prediction results demonstrate the value of LLMs in extracting useful signals of student learning progression from student-platform interactions. First, our simulation results as well as the performance of our GBM prediction model on a held-out test set both demonstrate that LLM-derived signals from student-chatbot interactions benefit the prediction of the initial belief state. Second, our analysis of meaningful vs. total attempts demonstrate the value of LLM-derived observations from student code submissions on our platform. Together, these results provide strong support for our hypothesis that student-platform interactions are a valuable source of information that can be used to obtain more accurate estimates of student knowledge state.

C Pilot Study

Before launching our large-scale RCT, we conducted an IRB-approved pilot study to (i) validate the feasibility of the end-to-end AI learning system (question generation, platform logging, and tutoring workflow), (ii) collect early learner-trajectory data for training and calibrating our reinforcement learning (RL) algorithm, and (iii) incorporate user feedback to redesign and re-implement the platform and question bank.

Study context and sample. The pilot was run in the two month accelerated summer term at the National Taiwan University (NTU) in 2024 using the same course content as the main study here. A total of 52 students participated: 21 were high school students or first-year college students, 6 were graduate students, and participants represented multiple universities, with 22 students studying outside of Taipei. Throughout the course, students completed an in-person midterm and final exam covering the course concepts; all exams were graded by human grader to measure learning outcomes. All eligible university students and advanced high school students nationwide could register this class in the summer.

Pilot experimental design. The pilot used a 2×2 randomized controlled design that crossed (i) adaptive sequencing (RL vs. Random) with (ii) support modality (human tutor vs. GPT-4-based tutor). Participating students were uniformly randomly assigned to one of four arms:

1. **Control (No RL+Human Tutor).** Non-adaptive practice with a randomized difficulty sequence; access to standard human tutors.

2. **No RL+GPT-4.** Non-adaptive practice with a randomized difficulty sequence; access to a GPT-4-based tutor.

3. **RL+Human Tutor.** Adaptive practice intended to be personalized via RL; access to

standard human tutors.

4. **RL+GPT-4.** Adaptive practice intended to be personalized via RL; access to a GPT-4-based tutor.

To ensure perceived fairness, the pilot implemented a post-midterm switch (control $\leftrightarrow$ RL+GPT-4; no-RL+GPT-4 $\leftrightarrow$ RL+human), so that each student experienced multiple learning environments over the course.

Lessons for platform and algorithm. During the pilot, we found there were data infrastructure and integration issues that prevented the RL policy from being reliably updated online as intended. As a result, we treat the pilot primarily as a validation of our system and a data collection opportunity rather than a formal test of the fully functional adaptive algorithm. We used the pilot interaction logs—especially the control-arm trajectories with randomized difficulty—to (i) train and calibrate the RL components offline, and (ii) guide a comprehensive redesign, re-implementing the learning platform (logging pipeline, interface usability, and tutoring workflow) and question bank (generation, validation, and quality control). These changes were informed by both quantitative usage traces and structured feedback from students and teaching staff. Note that because the pilot included a mid-course arm switch, we restrict all pilot analyses used for scientific interpretation to data *before* the switch, to avoid carryover effects across conditions.

D Experimental Protocol Details

D.1 outlines teacher and student briefings; D.2 describes the Python certification program schedule and rules; D.3 lists the survey items; and D.4 details our pre-registered analyses.

D.1 Teacher and Student Briefing

Before the start of the program, our team held an in-person briefing with the high school teachers responsible for program implementation at participating schools, focused on introducing the study protocol and demonstrating the learning platform.[11] The briefing covered the overall course structure, the platform interface (lecture videos, practice mode, and the embedded AI tutor), and the rules governing student participation and academic integrity.

Following the briefing, teachers recruited students from their respective schools. All participants provided informed consent, but they were blinded to specific details regarding the random treatment assignment. Students who signed up consisted primarily of 10th-graders (59.5%), followed by 11th graders (23.8%) and 12th graders (16.7%). Regarding prior Python learning experience, 28.4% were first-time learners and 27.0% identified as beginners. While 39.9% reported intermediate experience, only a small fraction identified as proficient (3.8%) or at a competition level (1.0%). We provide more information about the student sample in our balance table (Table 7 in E).

Before the first module, teachers outlined the study's general structure, explaining that the program involved a multi-week sequence of online learning modules culminating in a certification exam. Students were issued unique login credentials and invited to a brief synchronous online orientation. To assist them throughout the program, a platform demonstration video was made accessible for reference at any time. Furthermore, the course materials included a concise set of usage rules and tips for effective AI-tutor interaction (see Fig. 20).

[11] Note that instruction was delivered through pre-recorded videos on our learning platform, so these teachers did not need to prepare or teach the curriculum; their role was limited to supervising program implementation.

AI Tutor Usage Tip and Rules for Students

AI Tutor: Tips for Effective Use

- Think carefully about what you are stuck on; the more specific your question, the more useful the response.
- State what you want explained: a concept, code logic, or the problem statement itself.
- The AI can be wrong. Always evaluate its answers critically—verify by asking follow-up questions, challenging assumptions, and checking the logic. Use this as practice for independent thinking.
- If the explanation is unclear, ask the AI to rephrase in simpler terms or to provide a concrete example.
- Use the AI to clarify any confusion. Do not worry if your question feels basic, and it is okay to ask similar questions again.

AI Tutor: Usage Rules

1. *Rate limit.* To ensure fair access, usage is rate-limited: after every five questions, students must wait 10 seconds before submitting additional AI-tutor queries.
2. *Compliance with usage standards.* Students were required to follow the platform's usage policy:
   - **No abuse.** Ask questions thoughtfully and read the response carefully before following up.
   - **Respect privacy and rights.** Do not share or misuse personal information. Do not use AI-generated content to harass, discriminate, scam, or harm others.
   - **Ethics and social responsibility.** Do not use the AI tutor for illegal activities or to manipulate or mislead others in harmful ways.
   - **Safety and fairness.** Do not attempt to bypass safety mechanisms or conduct unauthorized testing. Report abnormal behavior or risks to the teaching assistants or instructor.
   - **Law and academic integrity.** Do not use the AI tutor for cheating, plagiarism, or other dishonest academic conduct.

Students were instructed to use the AI tutor as a learning aid and to ensure all usage is ethical and compliant. Any violation resulted in revocation of platform access and removal from the program.

Figure 20: **AI tutor usage tips and rules provided to student participants.**

D.2 Certification Program Schedule and Rules

Table 4 presents the course schedule, assignment deadlines, and module outlines for the certification program, along with the lengths of the instructional videos. The program concluded with an on-site paper-based assessment administered by participating schools (see H.2).

Table 4: **Program Schedule and Outline**

Date	**Course Module & Assignment**	**Learning Hours / Course Outline**
2/17 – 2/23 2/24 – 3/2	**Module 1-3: Basics** Assignment 1 **Deadline: 3/12 11:59pm*	This week introduces Python applications in modern business, as well as Python syntax and the programming development environment. Students will interact using basic commands and programs. **Video Length:** M1: 1:14:54 M2: 1:27:49 M3: 1:08:53
3/3 – 3/9 3/10 – 3/16	**Module 4-5: Conditionals** Assignment 2 **Deadline: 3/23 11:59pm*	This week continues the introduction to different variable types and moves into conditional selection teaching, enabling programs to respond based on given conditions. **Video Length:** M4: 0:53:27 M5: 1:00:49
3/17 – 3/23 3/24 – 3/30	**Module 6: Loops (Part 1)** Assignment 3 **Deadline: 3/30 11:59pm*	This week continues with logical operators and introduces loop concepts to build more complex conditional selections, enabling the program to repeat automated tasks. **Video Length:** M6: 0:49:11
3/31 – 4/6 4/7 – 4/13	**Module 7: Loops (Part 2)** Assignment 4 **Deadline: 4/13 11:59pm*	Continuing with loops; learning the application of Lists in data management. **Video Length:** M7: 0:41:18
4/14 – 4/20 4/21 – 4/27	**Module 8: Operations Applications** Assignment 5 **Deadline: 4/27 11:59pm*	This module explores program applications in business topics such as production scheduling and inventory control. **Video Length:** M8: 1:32:42
Continued on next page		

Table 4 – continued from previous page

Date	Course Module & Assignment	Learning Hours / Course Outline
4/28 – 5/4 5/5 – 5/11	**Module 9: Functions (Part 1)** Assignment 6 **Deadline: 5/11 11:59pm*	This week focuses on Functions. **Video Length:** M9: 1:06:57
5/12 – 5/18 5/19 – 5/25	**Module 10: Functions (Part 2)** Assignment 7 **Deadline: 5/25 11:59pm*	Continuing with Functions. This week introduces the application of functions in program development. **Video Length:** M10: 0:56:19
5/26 – 6/6	**On-site Paper-based Assessment**	Schools will select a date to administer the paper-based test to assess learning outcomes and determine eligibility for certificates.

D.3 Survey Instrument

We administered a baseline survey after the introductory module (when students had become familiar with the platform) to obtain statistics on student demographics, motivation, parental education, familiarity with the course material, etc. Table 5 lists the survey questions.

D.4 Pre-Registration Details

Our pre-registration is available here: https://aspredicted.org/6az9cd.pdf. The primary outcome was student performance on the certification exam. We pre-registered two primary analyses: (i) linear regression evaluating treatment effects on test performance with controls for baseline covariates and fixed effects, and (ii) evaluation of different pedagogical strategies in the system prompt. We report the first analysis in Fig. 2b and F.1; for the second analysis, as noted in A.3, we find no meaningful variation across prompt conditions. We supplement our primary regression with unadjusted two-sided t-tests (Fig. 2a) and report effects on students in an accelerated track where the implementation of RL was severely delayed (Table 12; F.1).

Table 5: **Baseline survey questions.**

Item	Survey question (response options)
Q1	**What is your gender?** *(Female; Male; Prefer not to say; Other)*
Q2	**What is the *primary* motivation for you to take this course?** *(Parents required it; Classmates/friends invited me; School/graduation requirement; Enrich my resume; Learn relevant skills in the AI era)*
Q3	**What is the highest education level of your parents?** *(High school or below; Bachelor's degree; Master's degree; Doctoral degree)*
Q4	**Do you attend cram school (including private tutoring), or have you ever attended? If yes, which subjects have you attended for?** *(None; Chinese; English; Math; Science; Programming; Other subjects; Multiple selections allowed)*
Q5	**During the period you attended high school, how was your average academic performance?** *(Top 20; Top 50; Top 100; Top 150; Top 200; Top 300; Top 500 or >500 in class/school rank)*
Q6	**Before taking this course, have you taken any courses related to programming? (Including self-study and online courses; any related learning of more than 20 hours counts. Multiple selections allowed.)** *(No, none or fewer than 20 hours; Yes, Python; Yes, C/C++; Yes, other programming languages; Multiple selections allowed)*
Q7	**What do you think your Python ability or level is?** *(First time learning; Little to no foundation; Average; Proficient; Very proficient and have participated in competitions)*
Q8	**What are your views on learning using generative AI?** *(AI can help me learn; AI may create dependence and hinder my learning)*

We pre-registered several secondary analyses examining heterogeneous treatment effects and student engagement patterns. We report heterogeneity by prior Python experience and school tier in Section 4.2 (with interaction-regression specifications in F.3). Following our pre-registered plan to analyze engagement measures, we examine platform logs and AI chat conversations as potential mechanisms in Section 4.3 and F.4.

Note that our pre-registration included two additional components that we do not emphasize in this paper. First, the pre-registration planned analyses for both university (NTU) and high school samples (conducted as separate deployments on different student populations). However, because the NTU term began earlier, unforeseen delays in integrating our platform with the government e-learning system prevented reliable delivery of our intervention during the NTU deployment. Thus, we only report on the high school student deployment. Second, we randomized the AI tutor prompt across three pedagogical frameworks (Constructivism, Behaviorism, and Social

Learning). We find no meaningful or statistically significant differences across pedagogical conditions in our primary outcomes. Accordingly, we do not discuss these contrasts in this paper; however, we show that our main results remain robust when controlling for these features (see F.1).

E Data Processing, Attrition, and Covariate Balance

E.1 details sample exclusion criteria; E.2 shows covariate balance across arms; and E.3 analyzes differential attrition.

E.1 Exclusion Processing

A total of 1,047 students participated in the program. We construct the main analysis sample by excluding observations that do not follow our pre-specified rule in our pre-registered plan or are not comparable to the standard program (see Table 6).

Following our pre-registration, we exclude 94 students who dropped out of the program or were unable to attend the exam due to schedule conflicts (and therefore do not have an outcome measure) and 29 students who were flagged for cheating during the certification exam. The cheating incident only happened in a single classroom and arose from a misunderstanding of proctoring instructions. We additionally exclude 154 senior high school students due to complications and delays with the implementation of RL. Specifically, (1) the course for seniors was accelerated due to their graduation timeline, (2) the program launch for senior students was moved substantially earlier than initially planned, and (3) key platform integrations were delayed due to governmental constraints. As a consequence, our RL algorithm was only active one month after senior students started the course, which is about half of their course period due to their accelerated timeline. Thus, we would not expect RL to be effective among this student population. For transparency, we report our main analysis on the sample including the

Table 6: **Sample construction and attrition.**

Step	**Excluded**	**% of total**	**Remaining N**
Total enrolled in program	–	–	1,047
Exclude: Certification score = 0 (did not complete exam)	94	8.98%	953
Exclude: Flagged for cheating during certification exam	29	2.77%	924
Exclude: Accelerated senior-student track (non-comparable schedule/exam)	154	14.71%	770
Total excluded	277	26.46%	-
Final analysis sample	-	-	**770**

Notes: Percentages are relative to the total enrolled sample ($N = 1{,}047$).

accelerated senior-student track (see F.1).

E.2 Balance Table

After these exclusions, our final analysis sample contains 770 students. We assess baseline comparability between the treatment and control groups within this final analysis sample. Table 7 reports covariate balance and shows no evidence of systematic differences in observed pre-treatment characteristics across treatment conditions.

E.3 Differential attrition

To assess differential attrition by treatment status, we define an indicator $A_i = 1$ if student i is excluded from the main analysis sample (i.e., not present in the post-exclusion dataset) and $A_i = 0$ otherwise. As shown in Table 8, exclusion rates are nearly identical across arms: $140/524 = 26.7\%$ in control and $137/523 = 26.2\%$ in treatment. We fail to reject equality of attrition rates using both a two-sample proportion test ($p = 0.85$) and a chi-squared test of equal rates ($p = 0.90$).

Furthermore, we test for differential attrition within schools by estimating a linear probability and a logit model of A_i on treatment assignment with school fixed effects:

$$A_i = \alpha + \beta \text{Treatment}_i + \sum_s \gamma_s \mathbb{1}\{\text{School}_i = s\} + \varepsilon_i, \tag{23}$$

Table 7: **Baseline Covariate Balance Between Treatment and Control Groups (N=770)**

Variable	Control	Treated	p-value	p-adj (BH)
Continuous Variables, Mean (SD)				
Parent education (ordinal)	2.37 (0.88)	2.31 (0.78)	0.727	0.844
Has tutoring	0.93 (0.26)	0.91 (0.28)	0.399	0.776
Academic performance in school (ordinal)	4.84 (1.68)	4.79 (1.62)	0.726	0.844
Has programming experience	0.71 (0.46)	0.72 (0.45)	0.959	0.959
Python ability (ordinal)	1.17 (0.96)	1.16 (0.93)	0.938	0.959
AI positive view	0.97 (0.16)	0.98 (0.15)	0.632	0.830
Categorical Variables, n (%)				
Gender			0.271	0.715
Female	121 (31.5%)	131 (33.9%)		
Male	212 (55.2%)	206 (53.4%)		
Prefer not to say	19 (4.9%)	23 (6.0%)		
Other	5 (1.3%)	1 (0.3%)		
Missing	27 (7.0%)	25 (6.5%)		
Motivation			0.013	0.130
Parents	7 (1.8%)	5 (1.3%)		
Classmates or friends	33 (8.6%)	18 (4.7%)		
School or graduation	125 (32.6%)	110 (28.5%)		
Enrich CV	68 (17.7%)	60 (15.5%)		
Learn skill in AI era	123 (32.0%)	168 (43.5%)		
Missing	28 (7.3%)	26 (6.7%)		
Parent Education			0.012	0.130
High school & below	59 (15.4%)	56 (14.5%)		
University	142 (37.0%)	152 (39.4%)		
Master	119 (31.0%)	136 (35.2%)		
PhD	36 (9.4%)	16 (4.1%)		
Missing	28 (7.3%)	26 (6.7%)		
Python Ability			0.658	0.830
First time learner	111 (28.9%)	113 (29.3%)		
Beginner	97 (25.3%)	90 (23.3%)		
Intermediate	129 (33.6%)	143 (37.0%)		
Proficient	14 (3.6%)	13 (3.4%)		
Competition level	5 (1.3%)	1 (0.3%)		
Missing	28 (7.3%)	26 (6.7%)		

Notes: p-values computed from t-test (continuous) or chi-square test (categorical). BH-adjusted p-values control the false discovery rate across all covariates.

Table 8: **Attrition balance by treatment condition.**

	Control	**Treatment**	**Diff. (T–C)**	p**-value**
N (full)	524	523	–	–
Excluded N	140	137	–	–
Attrition rate	0.267	0.262	-0.005	–
Two-sample proportion test		–		0.848
Chi-squared test of equal rates		–		0.903

Notes: "Attrition" indicates exclusion from the main analysis sample. Diff. is Treatment minus Control.

Table 9: **Regression tests for differential attrition.**

	Logit	**LPM**
Treatment indicator	-0.057	-0.0078
	(0.169)	(0.0232)
p-value	0.733	0.736
School fixed effects	Yes	Yes
N	1,047	1,047

Notes: The dependent variable equals 1 if a student is excluded from the main analysis sample and 0 otherwise. The Logit specification estimates Equation (24) and reports coefficients on the log-odds scale. The LPM (linear probability model) estimates Equation (23), where coefficients are interpreted as percentage-point changes in the probability of exclusion. Both specifications include school fixed effects. Standard errors are shown in parentheses.

where Treatment_i indicates assignment to the RL-personalized sequence and the school indicators absorb school-level differences in participation.

$$\Pr(A_i = 1) = \Lambda\left(\alpha + \beta \text{Treatment}_i + \sum_s \gamma_s \mathbb{1}\{\text{School}_i = s\}\right), \tag{24}$$

where $\Lambda(\cdot)$ denotes the logistic CDF.

Table 9 shows that the estimated treatment effect on attrition is small and statistically indistinguishable from zero in both specifications (Logit: $\hat{\beta} = -0.057$, $p = 0.73$; LPM: $\hat{\beta} = -0.78$ percentage points, $p = 0.74$). Overall, these results suggest there is negligible differential attrition between treatment and control, alleviating concerns that sample exclusions bias our estimated treatment effects.

F Evaluation

F.1 details the regression specifications for our main analysis and several robustness checks; F.2 provides evidence ruling out an alternative explanation that increased problem difficulty drives our findings; F.3 presents heterogeneity analyses; F.4 reports mediation analyses; and F.5 summarizes student feedback on the AI tutoring program.

F.1 Main Analysis and Robustness Checks

We analyze the impact of RL-based personalization on exam performance using ordinary least squares (OLS) regressions. For student i, our baseline specification is

$$Y_i = \alpha + \beta RL_i + \mathbf{X}_i'\boldsymbol{\gamma} + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + \varepsilon_i, \tag{25}$$

where Y_i is either the raw exam score or the standardized exam score (computed as a z-score using the mean and standard deviation of the full sample); RL_i is an indicator for treatment assignment to the RL personalization condition, $\mathbf{X}_i$ is a vector of baseline covariates (including prior programming experience, parental education, self-reported prior Python ability, gender, baseline academic performance, and self-reported motivation), and $\delta_{\text{school}(i)}$ and $\delta_{\text{grade}(i)}$ are school and grade-level fixed effects, respectively.

Table 10 reports the effect of the RL-based problem sequencing on students' raw and standardized performance on the final exam. Columns (1) and (2) present OLS estimates without controls; columns (3) and (4) add baseline covariates and school and grade-level fixed effects.

As a robustness check, we also re-estimate the main specification using alternative standard error calculations to account for potential heteroskedasticity and within-school correlation of residuals. Results in Table 11 show that the positive effect of RL personalization remains statistically significant across all three specifications.

Recall that our main analysis omitted senior students due to complications with the imple-

Table 10: **Effect of RL Personalization on Exam Performance**

	Dependent variable			
	(1) Standardized score	(2) Raw score	(3) Standardized score	(4) Raw score
(Intercept)	-0.078	31.820^{***}	-1.273^{***}	3.957
	(0.051)	(1.189)	(0.229)	(5.339)
RL	0.156^{*}	3.636^{*}	0.150^{*}	3.503^{*}
	(0.072)	(1.679)	(0.067)	(1.572)
Has programming experience			0.246^{**}	5.745^{**}
			(0.085)	(1.982)
Parent education (ordinal)			0.018	0.425
			(0.041)	(0.959)
Python ability (ordinal)			0.125^{**}	2.919^{**}
			(0.042)	(0.985)
Male			0.224^{**}	5.231^{**}
			(0.078)	(1.813)
Academic performance (ordinal)			0.079^{***}	1.843^{***}
			(0.021)	(0.499)
School fixed effects	NO	NO	YES	YES
Grade level fixed effects	NO	NO	YES	YES
Motivation controls	NO	NO	YES	YES
R^2	0.006	0.006	0.253	0.253
Adjusted R^2	0.005	0.005	0.226	0.226
Observations	770	770	716	716
F-statistic	4.69 (1, 770)	4.69 (1, 770)	9.34 (25, 716)	9.34 (25, 716)

Notes: Estimates use listwise deletion for missing baseline survey covariates in columns (3) and (4). $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

Table 11: **Effect of RL Personalization on Exam Performance with Alternative Standard Errors**

	Dependent variable: Standardized score		
	(1) OLS	(2) HC1 robust	(3) School-clustered
(Intercept)	-1.273^{**}	-1.273^{***}	-1.273^{*}
	(0.316)	(0.271)	(0.455)
RL	0.150^{*}	0.150^{*}	0.150^{*}
	(0.067)	(0.067)	(0.069)
Python ability (ordinal)	0.125^{**}	0.125^{**}	0.125
	(0.042)	(0.043)	(0.088)
Male	0.224^{**}	0.224^{**}	0.224
	(0.078)	(0.075)	(0.208)
Has programming experience	0.246^{**}	0.246^{**}	0.246
	(0.085)	(0.081)	(0.164)
Parent education (ordinal)	0.018	0.018	0.018
	(0.041)	(0.040)	(0.029)
Academic performance (ordinal)	0.079^{***}	0.079^{***}	0.079^{***}
	(0.021)	(0.022)	(0.020)
School fixed effects	YES	YES	YES
Grade level fixed effects	YES	YES	YES
Motivation controls	YES	YES	YES
R^2	0.253	0.253	0.253
Adjusted R^2	0.226	0.226	0.226
Observations	716	716	716

Notes: Column (1) reports OLS standard errors. Column (2) reports heteroskedasticity-robust (HC1) standard errors. Column (3) reports school-clustered standard errors with CR2 small-sample correction. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

mentation: (1) the course for seniors was accelerated due to their graduation timeline, (2) the program launch for senior students was moved substantially earlier than initially planned, and (3) key platform integrations were delayed due to governmental constraints. As a consequence, our RL algorithm was only active one month after senior students started the course, which is about half of their course period due to their accelerated timeline. Thus, we would not expect to observe significant benefits for these students. Nevertheless, for transparency, we report results for an expanded sample that includes senior students (Table 12). First, column (1) presents the OLS estimate without controls, and column (2) adds baseline covariates as well as school and grade-level fixed effects; these results are directionally the same as our main analysis, but as expected, they are not statistically significant. To disentangle the effects on senior students from those on non-seniors, columns (3)–(4) include a senior indicator and an RL$\times$Senior interaction term (with and without controls and fixed effects, respectively). In both specifications, the main RL coefficient remains positive, whereas the interaction term is negative and statistically insignificant, which is consistent with the hypothesis that our system was effective for non-seniors but not for seniors.

Finally, we examine our primary treatment effect estimates controlling for the randomly-assigned pedagogical approach used in the prompt (these were also randomized at the student-level, see discussion in D.4). Specifically, the AI tutor prompt was randomized across three pedagogical frameworks (Constructivism, Behaviorism, and Social Learning) so that the tutor followed a consistent instructional style throughout a student's interactions.[12] Table 13 reports OLS estimates of Equation (25) but additionally including indicators for the type of randomized AI-tutor pedagogical prompts. Consistent with our main result, assignment to RL-based personalization improves standardized exam scores by 0.145 standard deviations ($p = 0.026$). The

[12]Constructivism uses a student-centered, discovery-based approach that prompts students to articulate the problem in their own words and explain their reasoning; Behaviorism emphasizes structured, repetitive practice with immediate feedback; Social Learning positions the tutor in a peer-like "learning buddy" role rather than a traditional instructor.

estimated coefficients on the pedagogical prompt indicators are noisy and statistically insignificant, and including them leaves the estimated RL effect essentially unchanged, suggesting that the pedagogical style of the prompt does not materially affect our primary treatment estimate.

F.2 Ruling Out "Harder is Better"

Recall that we found that the treatment group received somewhat higher difficulty questions on average than the control group (see Table 14). Thus, one alternative explanation for our results is that the performance gains were driven simply by exposure to more difficult questions. We are unable to rule out this possibility using our RCT data because difficulty assignment was endogenous in both conditions of our study: in the treatment group, difficulty was dynamically determined by the RL policy, whereas in the control group, it followed a fixed, increasing sequence, which means that exposure to harder problems was confounded with student persistence. However, our pilot data—performed on the same course material and on a comparable student population (see C)—does not have this limitation, since in the pilot setting, difficulty was assigned exogenously: in the control arm, difficulty is randomized rather than adaptively chosen. Therefore, we estimate regressions within the pilot control group relating assigned difficulty to learning outcomes (e.g., exam or quiz performance), controlling for baseline covariates and AI tutor assignment. Because difficulty is randomized in this arm, this analysis isolates the causal effect of receiving harder questions holding other factors constant. Specifically, letting student i's average practice-question difficulty be denoted by *AvgDif*, we estimate:

$$Y_i = \alpha + \beta \mathit{AvgDif}_i + \eta \mathit{AI}_i + \theta \mathit{PriorPython}_i + \mathbf{X}_i'\boldsymbol{\gamma} + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + \varepsilon_i, \qquad (26)$$

where Y_i denotes standardized exam score, $\mathit{PriorPython}_i$ is a dummy indicator of whether students have prior Python coding background, $\mathbf{X}_i$ includes baseline demographic controls, and $\delta_{\text{school}(i)}$, $\delta_{\text{grade}(i)}$ are school and grade-level fixed effects.

Table 12: **Results Including Delayed Implementation with Seniors**

	Dependent variable: Standardized score			
	(1)	(2)	(3)	(4)
(Intercept)	0.088^{*}	-1.016^{***}	-0.078	-1.238^{***}
	(0.044)	(0.288)	(0.051)	(0.207)
RL	$0.116^{\dagger}$	$0.114^{\dagger}$	0.156^{*}	0.144^{*}
	(0.063)	(0.058)	(0.072)	(0.067)
Senior			0.110	$0.237^{\dagger}$
			(0.125)	(0.140)
RL $\times$ Senior			-0.219	-0.185
			(0.176)	(0.174)
Has programming experience		0.221^{**}		0.229^{**}
		(0.074)		(0.078)
Parent education (ordinal)		0.015		0.019
		(0.035)		(0.038)
Python ability (ordinal)		0.115^{**}		0.124^{**}
		(0.037)		(0.039)
Male		0.234^{***}		0.258^{***}
		(0.067)		(0.072)
Academic performance (ordinal)		0.070^{***}		0.075^{***}
		(0.018)		(0.019)
School fixed effects	NO	YES	NO	YES
Grade level fixed effects	NO	YES	NO	YES
Motivation controls	NO	YES	NO	YES
R^2	0.004	0.279	0.005	0.235
Adjusted R^2	0.003	0.255	0.002	0.209
Observations	924	839	924	839
F-statistic	3.41 (1, 922)	12.05 (26, 812)	1.61 (3, 920)	9.21 (27, 811)

Notes: Columns (1)–(2) uses the main regression specification, while Columns (3)–(4) additionally includes the senior interaction term (RL $\times$ Senior). $^{\dagger}p < 0.1$, $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

Table 13: **Result of Main Analysis with Pedagogy Controls**

	Dependent variable: Standardized score
	(1)
RL	0.151*
	(0.067)
Prior Python ability (ordinal)	0.125**
	(0.042)
Prior programming experience	0.246**
	(0.085)
Parent education (ordinal)	0.022
	(0.041)
Baseline academic performance (ordinal)	0.078***
	(0.021)
Male	0.232**
	(0.078)
AI Pedagogy: Constructivism	-0.101
	(0.083)
AI Pedagogy: Behaviorism	-0.022
	(0.081)
Grade-level fixed effects	YES
School fixed effects	YES
Motivation controls	YES
R^2	0.255
Adjusted R^2	0.225
Observations	716
F-statistic	8.703 (27, 688)

Notes: OLS estimates with standard errors in parentheses. $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

We also look at heterogeneous treatment effects with respect to students' prior Python coding background (*PriorPython*). To this end, we estimate

$$\begin{aligned} Y_i =& \alpha + \beta_1 \textit{AvgDif}_i + \beta_2 \textit{PriorPython}_i + \beta_3 \big(\textit{AvgDif}_i \times \textit{PriorPython}_i\big) \\ &+ \eta \textit{AI}_i + \mathbf{X}_i \boldsymbol{\gamma} + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + + \varepsilon_i. \end{aligned} \tag{27}$$

Results reported in Table 14 show that increased difficulty alone does not account for improved performance and could potentially harm beginners, supporting the interpretation that the gains in

Table 14: **Pilot study: Practice-question difficulty exposure and assessment performance**

	Dependent variable: Standardized score	
	(1)	(2)
AvgDif	0.417	0.271
	(0.358)	(0.252)
PriorPython	0.909*	-3.739
	(0.375)	(1.598)
AvgDif × PriorPython		2.542*
		(0.863)
AI	0.685*	0.357
	(0.285)	(0.226)
Parent Education	0.542*	0.710**
	(0.213)	(0.158)
ProgrammingBackground	0.296	0.676
	(0.370)	(0.286)
Gender	-0.110	-0.028
	(0.306)	(0.213)
(Intercept)	-1.414	-0.751
	(0.697)	(0.531)
School fixed effects	YES	YES
Study Major control	YES	YES
Observations	28	28
Adjusted R^2	0.800	0.905

Notes: OLS estimates with standard errors in parentheses. AvgDif denotes average assigned practice-question difficulty. 3 observations were deleted due to missingness in survey response (listwise deletion). $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

our main study are driven by adaptive sequencing rather than by mechanical difficulty inflation.

F.3 Heterogeneity Analysis

In addition to using subgroup regression to assess the heterogeneous effect of RL-based personalization, we also extend the main specification in Equation (25) by interacting the treatment indicator with subgroup indicators. First, we define $PythonFamiliar_i$ as an indicator equal to 1

for students who reported *some* prior Python exposure (e.g., basic familiarity but not competition-level proficiency) before entering the program, and 0 otherwise (Python beginners).[13] Then, we estimate:

$$Y_i = \alpha + \beta_1 \textit{RL}_i + \beta_2 \textit{PythonFamiliar}_i + \beta_3 \big(\textit{RL}_i \times \textit{PythonFamiliar}_i\big) + \mathbf{X}_i'\boldsymbol{\gamma} + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + \varepsilon_i, \tag{28}$$

where Y_i is the standardized outcome and $\mathbf{X}_i$ contains the same baseline controls as in the main analysis, along with school and grade-level fixed effects. In this specification, β_1 is the treatment effect for Python beginners, and β_3 captures the *differential* effect for Python-familiar students relative to beginners; thus, the treatment effect for Python-familiar students is $\beta_1 + \beta_3$. Next, we examine heterogeneity with respect to school selectivity by replacing $\textit{PythonFamiliar}_i$ with an indicator $\textit{SchoolTier}_i$, defined using entrance exam scores as described in the main paper. It is equal to 1 for higher-tier school otherwise 0:

$$Y_i = \alpha + \beta_1 \textit{RL}_i + \beta_2 \textit{SchoolTier}_i + \beta_3 \big(\textit{RL}_i \times \textit{SchoolTier}_i\big) + \mathbf{X}_i'\boldsymbol{\gamma} + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + \varepsilon_i. \tag{29}$$

Table 15 reports the estimates from both types of heterogeneous treatment effects. Consistent with Fig. 3 in the main paper, the point estimates indicate that the gains from RL personalization are largest among Python beginners. In the Python-skill interaction specification (Table 15, cols. (1)–(2)), the RL effect is positive and statistically significant for Python beginners (the reference group). In contrast, the incremental effect for Python-familiar students is negative (RL$\times$PythonFamiliar $= -0.249$ SD, $p = 0.066$), implying that the treatment effect for Python-familiar students ($\beta_1 + \beta_3$) is much smaller and not statistically distinguishable from zero. In the school-tier interaction specification (Table 15, cols. (3)–(4)), we find no evidence that RL disproportionately benefits students in higher-tier schools: the interaction term is statistically

[13] We exclude 6 students (1%) who reported national competition-level Python skill, since they had already mastered the curriculum prior to the course and would reflect ceiling effects.

Table 15: **Heterogeneous effects of RL personalization by prior Python skill and school tier.** This table reports interaction regressions that extend Eq. (25) by interacting the RL assignment indicator with (i) prior Python familiarity and (ii) school tier. Columns (1)–(2) use the Python-skill interaction; columns (3)–(4) use the school-tier interaction.

	Prior Python skill interaction		School tier interaction	
	(1) Standardized score	(2) Raw score	(3) Standardized score	(4) Raw score
RL	0.235**	5.498**	0.191*	4.459*
	(0.088)	(2.045)	(0.079)	(1.850)
RL × Subgroup	−0.249†	−5.816†	−0.145	−3.393
	(0.135)	(3.155)	(0.148)	(3.464)
Subgroup indicator	0.414***	9.674***	0.809**	18.882**
	(0.103)	(2.400)	(0.263)	(6.133)
Prior Python ability (ordinal)			0.126**	2.936**
			(0.042)	(0.985)
Has programming experience	0.242**	5.659**	0.243**	5.672**
	(0.083)	(1.940)	(0.085)	(1.984)
Male	0.218**	5.093**	0.225**	5.251**
	(0.077)	(1.803)	(0.078)	(1.813)
Parent education (ordinal)	0.022	0.511	0.017	0.398
	(0.041)	(0.963)	(0.041)	(0.959)
Academic performance (ordinal)	0.078***	1.811***	0.079***	1.849***
	(0.021)	(0.500)	(0.021)	(0.499)
School fixed effects	YES	YES	YES	YES
Grade-level fixed effects	YES	YES	YES	YES
Motivation controls	YES	YES	YES	YES
Observations	710	710	716	716
R^2	0.261	0.261	0.254	0.254
Adj. R^2	0.233	0.233	0.226	0.226

Notes: Standard errors in parentheses. $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$, $^{\dagger}p < 0.10$.

insignificant (RL×SchoolTier $= -0.145$ SD, $p = 0.328$), indicating that performance gains do not systematically favor already advantaged students.

F.4 Mediation Analysis

We conduct a mediation analysis in the potential-outcomes framework following (*52*). We examine two measures of student engagement as mediators: (i) total time spent on the platform (*totalTime*), and (ii) total number of attempts (*totalAttempt*), showing that the RL policy increased performance by driving students to spend more time and effort.

For each mediator $M_i \in \{\textit{totalTime}, \textit{totalAttempt}\}$, we estimate a mediator model and an outcome model, using the same baseline covariates and fixed effects as in the main analysis:

$$M_i = \alpha_M + \tau \textit{RL}_i + \mathbf{X}_i'\boldsymbol{\gamma}_M + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + u_i, \tag{30}$$

$$Y_i = \alpha_Y + \beta \textit{RL}_i + {}_i + \mathbf{X}_i'\boldsymbol{\gamma}_Y + \delta_{\text{school}(i)} + \delta_{\text{grade}(i)} + \varepsilon_i, \tag{31}$$

where Y_i is the standardized score outcome and $\mathbf{X}_i$ includes the same controls as in the main specification. We then use the `mediation` package in R to estimate the average mediation effect (ACME; indirect effect) and the average direct effect (ADE), with nonparametric bootstrap percentile confidence intervals (1,000 simulations), as recommended by (*52*).

Table 16 summarizes the results. Across both mediators, the estimated ACME is positive and precisely estimated, while the ADE is statistically indistinguishable from zero. For *totalTime*, the ACME is 0.193 SD ($p < 0.001$) and the ADE is -0.031 SD ($p = 0.682$); for *totalAttempt*, the ACME is 0.152 SD ($p < 0.001$) and the ADE is 0.009 SD ($p = 0.854$). These estimates imply that the overall treatment effect (≈ 0.15 SD) operates primarily through increased engagement rather than through a residual direct channel.[14]

Finally, we assess robustness to violations of sequential ignorability using the sensitivity analysis in (*52*), parameterized by ρ, the correlation between the unobserved components of the mediator and outcome models. The ACME would be attenuated to zero only if ρ were

[14]The estimated proportion mediated can exceed 1 when the estimated direct and indirect components have opposite signs and/or due to sampling variability, which is common in finite samples. We interpret these estimates as evidence that mediation through engagement accounts for almost all of the total effect.

Table 16: **Mediation analysis: engagement as a mechanism.** ACME denotes the average causal mediation effect (indirect effect) and ADE denotes the average direct effect. Confidence intervals are nonparametric bootstrap percentile intervals with 1,000 simulations.

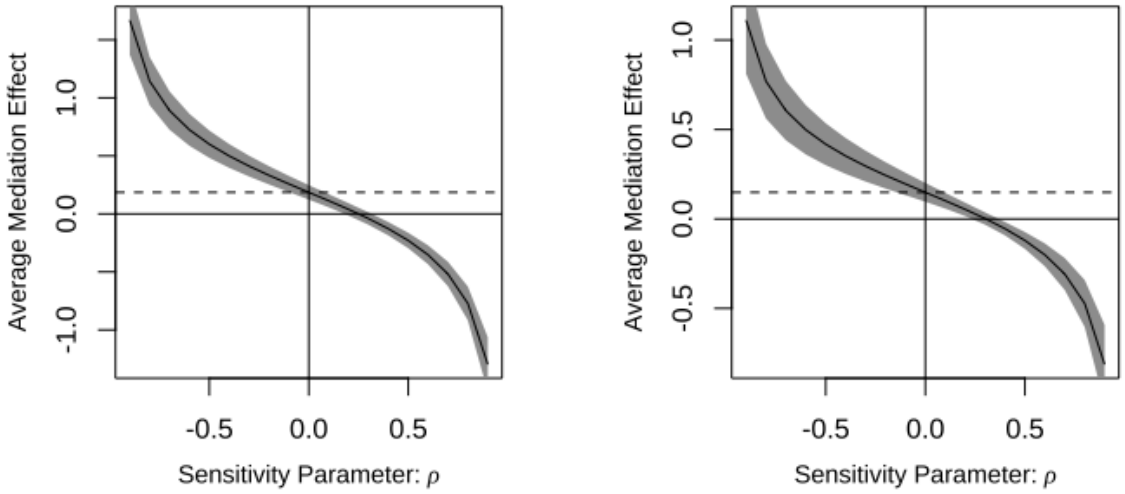
	Mediators	
	Total Time Spent	**Total Attempts**
ACME (indirect effect)	0.193 [0.137,0.252]***	0.153 [0.108,0.200]***
ADE (direct effect)	-0.031 [-0.170,0.109]	0.009 [-0.109,0.139]
Total effect	0.162 [0.034,0.298]*	0.162 [0.037,0.297]*
Proportion mediated	1.189 [0.571,5.113]*	0.943 [0.499,3.148]*
ρ at which ACME $= 0$ (sensitivity)	0.3	0.3
Sample size	712	712
Bootstrap simulations	1,000	1,000

Notes: Entries report point estimates with 95% confidence intervals in brackets. $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$. Sensitivity parameter ρ captures the correlation between unobserved determinants of the mediator and the outcome. The sample size (N=712) is reduced from the main analysis with controls (N=716) due to missing total attempt and total time data for four participants. These participants showed minimal engagement (no homework completion) despite remaining enrolled through the final certification exam.

approximately 0.3 for *totalTime* and 0.3 for *totalAttempt* (see Fig. 21a–21b), indicating that the mediated effect is robust to moderate levels of unobserved confounding.

As a robustness analysis, we conducted path analyses using the *lavaan* package in R to examine whether the effect of the RL policy on exam performance is mediated by increased engagement, measured by two metrics: *total time spent to solve the question* and *total number of attempts*. We use structural equation modeling (SEM) with 1,000 bootstrap resamples to ensure robust inference. The results, summarized in Fig. 22, show a consistent mechanism across both measures. First, the total effect of the RL intervention on exam scores is positive and significant ($\beta_{\text{total}} = 0.158, p = 0.023$); furthermore, this gain is driven entirely by the indirect path through engagement.

For time taken, RL significantly increased total time taken ($a = 173.46, p < 0.001$), which in turn strongly predicted higher exam performance ($b = 0.001, p < 0.001$), yielding a significant



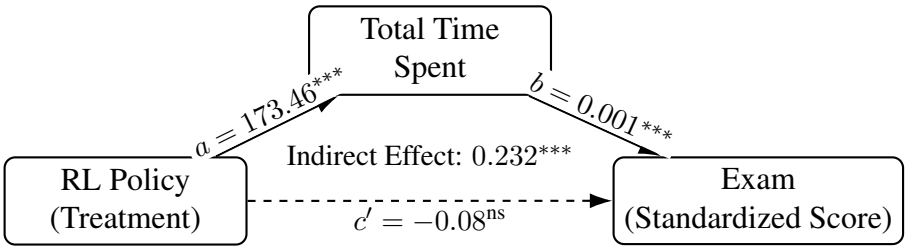
(a) Mediator: total time spent.

(b) Mediator: total number of attempts.

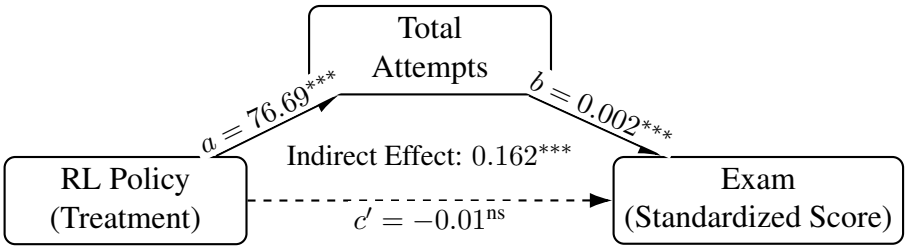
Figure 21: **Sensitivity analysis for mediation through engagement.** Sensitivity plots for the estimated average causal mediation effect (ACME) under potential violations of sequential ignorability. Each panel shows ACME as a function of the sensitivity parameter ρ, the correlation between unobserved determinants of the mediator and the outcome. The solid curve gives the point estimate and the shaded band indicates the 95% confidence interval. The vertical line marks $\rho = 0$ (no unobserved confounding), and the horizontal line marks zero mediated effect.

positive indirect effect ($\beta_{\text{indirect}} = 0.232, p < 0.001$). Similarly, for practice intensity, the RL policy significantly increased the number of attempts ($a = 76.69, p < 0.001$), which also predicted higher scores ($b = 0.002, p < 0.001$), resulting in a significant indirect effect ($\beta_{\text{indirect}} = 0.162, p < 0.001$).

In both models, the direct effect of the RL policy on performance—after accounting for the mediator—was small and statistically indistinguishable from zero ($\beta_{\text{direct (time)}} = -0.081, p = 0.235$; $\beta_{\text{direct (attempts)}} = -0.011, p = 0.869$). This pattern of full mediation across both time spent and practice intensity suggests that the primary driver of the RL system's efficacy is its ability to sustain student effort and extend productive practice.



(a) Mediator: Total Time Spent per Practice Problem (sec)



(b) Mediator: Total Number of Attempts

Figure 22: **Path Analysis Results.** Structural equation models testing the mediation of the treatment effect via (a) Total Time Spent and (b) Total Attempts. Path a represents the effect of RL on the mediator; Path b represents the effect of the mediator on the exam score; Path c' represents the direct effect of RL on the exam score after controlling for the mediator. *Note: Values represent coefficients.* $^{***}p < 0.001$*; ns = not significant.*

F.5 Student Feedback

Halfway through the study (before starting Module 7), we asked students one open-ended question "What are your thoughts and suggestions about using this learning platform?" Overall, the students' responses reflect the successful implementation of our learning platform. We give a summary and example quotes below:

Overall Educational Experience. Despite the complexities of introducing a new system, students responded positively to the core learning experience. Students found the platform effective, specifically noting the clarity of instruction and the diversity of practice materials. The consensus was that the system successfully facilitated a meaningful learning process.

"It is very easy to use, and the explanations are very clear."

"The practice questions are very diverse; I like them a lot."

"The instructor's explanations are very clear...this time I truly felt that I learned."

Addressing System Stability. During the first two weeks, we encountered stability issues—specifically lag and session timeouts—stemming from traffic congestion on the government's digital learning system (SSO) linkage. However, our team moved quickly to mitigate the impact by fixing the technical glitches and extending the deadline for the first required homework. These early issues were resolved quickly and did not persist through the remainder of the curriculum.

"The work session often expires; there are webpage errors that require refreshing."

"Sometimes it crashes and I have to reload repeatedly."

AI Tutor. Students found it to be a convenient companion for identifying mistakes, though we acknowledge there is still room for improvement regarding the precision of its answers.

"Having an AI on the side to ask questions is great—it is very convenient!"

"The AI is useful, but its effectiveness is limited—there is still a lot of room for improvement."

Overall, our work is a proof of the concept that tightly integrating a carefully-designed GenAI chatbot with a reinforcement learning algorithm for sequencing practice problems can be an effective way of improving student learning outcomes. With further improvements in the design of the tutoring platform, chatbot, and delivery, we may anticipate even larger gains.

G Summary of Related Work on Adaptive Learning Policies

To identify real-world evaluations of adaptive learning algorithms related to our work, we began with the empirical literature reviewed by Doroudi's work (*50*), including only those conducted in

Table 17: **Summary of related work on adaptive learning**

Paper	Domain	Population	N	Adaptation Policy	Baseline	Outcome	Significance
Rafferty et al. (2016) (*35*)	Alpha. Arith.	Online (self-recruit)	40	POMDP	Random	Time taken	Sig. (Conditional)
Sen et al. (2018) (*58*)	Chemistry	Online (AMT)	325	ML-generated seq.	Random/Human	Test score	Sig.
Clement et al. (2015) (*34*)	Arithmetic	Ages 7-8	400	MAB Intel. System	Human Expert	Test score	Mixed
David et al. (2016) (*59*)	Mathematics	K-12	73	BKT	Ascending Algo.	Test score	Not Sig.
Schatten (2017) (*60*)	Mathematics	K-12	98	Vygotsky Policy (VPS)	Rule-based	Test score	Not Sig.
Doroudi et al. (2017) (*50*)	Mathematics	Grades 4-5	69	BKT-based adaptive	Spiral difficulty	Test score	Not Sig.
Segal et al. (2018) (*37*)	Mathematics	Grade 7	92	Multi-Armed Bandit	BKT & Ascending	Test score	Not Sig.
Zhou et al. (2017) (*61*)	Probability	Undergrad	153	RL	Random	Test score	Sig.
Shen & Chi (2016) (*62*)	Logic	Undergrad	106	RL	Random	Test score	Not Sig.
Shen et al. (2018) (*36*)	Logic	Undergrad	77	Constrained POMDP	Random	Test score	Sig. (Low knowledge only)
Shen et al. (2018) S1 (*63*)	Logic	Undergrad	105	POMDP	Random	Test score	Not Sig.
Shen et al. (2018) S2 (*63*)	Logic	Undergrad	181	POMDP (w/ features)	Random	Test score	Sig.
Rowe & Lester (2015) (*64*)	Microbiology	Grade 8	75	Modular RL	Random	Test score	Not Sig.
Beck et al. (2000) (*65*)	Arithmetic	Grade 6	97	RL	Human Expert	Time taken	Sig.
Koedinger et al. (1997) (*66*)	Algebra	Grade 9	470	PUPM + Algebra Tutor	Trad. Curriculum	Test score	Sig.
Pane et al. (2014) (*67*)	Algebra	Grades 9-12	18,700	CTAI	Existing Curr.	Test score	Mixed
Pane et al. (2014) (*67*)	Algebra	Grades 6-8	6,800	CTAI	Existing Curr.	Test score	Not Sig.
Belfer et al. (2022) (*68*)	Data Science	Online (self-recruit)	44	Contextual Bandit	Rule-based	Completion rate	Sig.
Ritter et al. (2007) (*69*)	Algebra	Grade 9	426	Cognitive Tutor (+Train)	Existing Curr.	Test score	Sig.
Jaciw et al. (2007) (*70*)	Algebra	Grades 8-13	541	Cognitive Tutor (+Train)	Existing Curr.	Test score	Not Sig.
Pane et al. (2010) (*71*)	Geometry	Grades 9-12	1,640	Cognitive Tutor (+Train)	Existing Curr.	Test score	Not Sig.
Roschelle et al. (2016) (*72*)	Mathematics	Grade 7	2,850	ASSISTments (+PD)	Existing Practice	Test score	Sig.
Clément et al. (2024) (*73*)	Mathematics	Grade 2	265	Multi-Armed Bandit	Human Expert	Test score	Sig.
Zhou et al. (2022) (*74*)	Discrete Math	Undergrad	180	HRL	Random	Test score	Sig.
Bassen et al. (2020) (*75*)	Lin. Algebra	Amazon Employees	1,987	RL (activities)	Self-directed	Test score	Sig.
Mandel et al. (2014) (*76*)	Fractions	Online (Game recruit)	2,000	POMDP/HMM	Random/Expert	Concept count	Sig.
Schmucker et al. (2025) (*77*)	Biology	Online Class	27,268	Bandit (select hint)	Random	Accuracy	Sig.
Prihar et al. (2023) (*78*)	Mathematics	Middle School	3,602	Contextual MAB	Random	Accuracy	Not Sig.
Harrison et al. (2024) (*79*)	Mathematics	UK Secondary School	2,901	PVAE + Bayesian Optimization	Existing Curr.	Test score	Sig.
Alam et al. (2024) (*80*)	Mathematics	Undergrad	212	Deep RL	Ascending difficulty	Test score	Not Sig.
Naslund-Hadleyetal et al. (2025) (*81*)	Mathematics and reading	Age 10-12	1900	rule-based personalization	Existing Curr.	Test score	Not Sig. (Math); Sig (English)

Notes: AMT denotes Amazon Mechanical Turk; BKT denotes Bayesian Knowledge Tracing; MAB denotes Multi-Armed Bandit; PVAE denotes Partial Variational Autoencoder; HRL denotes Hierarchical Reinforcement Learning; CTAI denotes Cognitive Tutor Algebra I.

real-world educational settings (e.g., classrooms or deployed online platforms) and focused on STEM domains such as mathematics, science, or computer science. We expanded the search using the deep research capabilities of frontier large language models (GPT-4.5, Claude Opus 4.5, and Gemini Pro), each prompted with a structured query designed to return studies meeting

strict inclusion criteria. Specifically, studies had to involve adaptive learning algorithms such as Bayesian Knowledge Tracing (BKT), multi-armed bandits, or reinforcement learning, and apply them to adaptively sequence or select learning content. In addition, we only included experimental studies that reported learning outcomes that were comparable between treatment and control groups; we excluded studies focused solely on personalized pedagogy, such as those adapting instructional style rather than content or difficulty order, as well as those limited to simulated settings or offline-only evaluations. This process yielded a preliminary list of candidate papers, which we then reviewed manually to verify that they met all criteria and to extract and summarize key information from each study (see Table 17).

While many studies on BKT and Intelligent Tutoring Systems (ITS) have shown gains over traditional instruction (*39, 82*), evidence is mixed when it comes to adaptive learning systems that aim to improve existing instructional sequencing in STEM education. Studies showing significant effects often include or are confounded with additional supports (e.g., teacher training with Cognitive Tutor or ASSISTments), compare against weak baselines (e.g., random sequencing), or measure outcomes not directly tied to learning (e.g., time or completion rate). Many are short-term or conducted entirely online with crowd-recruited participants, lacking institutional context and the academic stakes that are inherent to a formal education system. The two large-scale field RCTs closely aligned with our work are the evaluation of the Eedi platform (*79*) and Doodle personalized learning platform (*81*). However, Eedi focuses on demonstrating the efficacy of their overall platform rather than studying personalized problem sequencing in isolation; Doodle Learning in Belize reported no significant improvement in math scores.

To the best of our knowledge, our study is the first to show a statistically significant educational impact of mastery-based adaptive learning over expert-informed fixed-sequence practice problems in a large-scale, long-term, institutionally grounded deployment.

H Materials

H.1 provides example practice problems; H.2 provides the in-person certification exam.

H.1 Practice Problem Examples

Each module contains 40 AI-generated questions (Section 2.1) for homework practice, with 10 at each difficulty level (easy, medium, hard, very hard). We prepared 10 additional backup questions per module to address potential issues, totaling approximately 500 questions in our question bank. For generated questions that passed our automated RAG validation procedure, six independent reviewers (three teaching assistants from National Taiwan University and three from our research team) manually verified each question for accuracy and language clarity. Example questions for each difficulty level are shown below:

Easy (True/False)

Statement. The `len()` function can be used to calculate the length of both lists and strings.
Task. Indicate whether the statement is `true` or `false`.

Medium (Debugging)

Task. Write a program that reads a single line of space-separated integers representing students' ages. The program should swap the first and last elements of the list and then output the modified list. The provided code contains bugs that cause the swap to be performed incorrectly. Fix and provide corrected program. (The error will not be in any `print()` statement.)

Buggy code:

```
ages_str = input()
ages = ages_str.split()
for i in range(len(ages)):
   ages[i] = int(ages[i])
ages[0] = ages[-1]
ages[-1] = ages[0]
print(ages)
```

Example test cases:

```
Input:
18 20 22 24

Output:
[24, 20, 22, 18]
5
```

Hard (Coding)

Task. Write a function that calculates tax based on the given income (integer) and returns the tax (float). Rules: if income is less than 10,000 dollars, the tax rate is 3%; if income is between 10,000 and 20,000 dollars (inclusive), the tax rate is 4%; if income is above 20,000 dollars, the tax rate is 5%.

Example test cases.

```
assert calculate_tax(25000) == 1250.0
assert calculate_tax(10000) == 400.0
assert calculate_tax(20000) == 800.0
assert calculate_tax(1000000) == 50000.0
```

Very Hard (Longer coding)

Task. Assume you are a manufacturing manager responsible for scheduling jobs to minimize the maximum completion time (makespan). Given job processing times and the number of machines, compute the minimized makespan under the following strategy: (1) sort job times in descending order, then (2) assign each job (in that order) to the machine with the smallest current load.
Input. Two lines: (1) space-separated job processing times (integers), (2) number of machines (integer).
Output. Print the maximum machine load after assignment (the makespan).

Example test cases:

```
Input:
3 3 3 4 4 5 5 6 7 8
3

Output:
18
```

H.2 Certification Exam

This section provides the bilingual (English/Chinese) certification exam used in our study. We manually designed the exam (i.e., it was not AI-generated) based on the instructor's course materials and evaluation data from previous terms of the same course. The exam was administered in person as a handwritten, paper-based assessment with no digital devices allowed, and lasted 120 minutes.

適性化 AI 輔助程式學習實施計畫
AI for Python Learning

證書測驗
Certification Exam

本次考試爲**手寫**考試，只能用筆在官方提供的答案紙上作答。考試 closed book，考試期間不可以查閱任何資料，也不可以跟其他人直接或間接討論題目和答案。作答時可以不依順序，但要清楚標明題號；可以用任何筆，包含鉛筆，只要清楚就行；筆跡要清晰，如果助教看不懂紙上的文字，就只能扣分。
考試時間共 120 分鐘。總分 100 分，及格分數界定與證書分級標準會在試後公佈。

This is a **handwritten** exam. You can only use a pen to write on the answer sheet provided by the official. The exam is closed book. During the exam, you may not look up any information or discuss the questions and answers with others, directly or indirectly. If you have any questions during the exam, you can raise your hand to ask the teaching assistant. You can answer the questions in any order, but you need to clearly mark the question number. You can use any pen, including pencil, as long as it is clear. Your handwriting must be clear; if the teaching assistant cannot understand the text on the paper, points will be deducted.
The exam time is 120 minutes. Total score: 100 points. The passing grade and certificate classification standards will be announced after the exam.

第一題 Question 1

(12 分；每小題 4 分) 針對以下題目，如果你認爲該敘述正確無誤，請在答案卷寫下「 o 」，反之請寫下「 × 」。你不需要寫你的原因。在題目卷上的作答不會被計分。

(12 points; 4 points each) For the following questions, if you think the statement is correct, write "o" on the answer sheet; otherwise, write "×". You do not need to write your reasoning. Answers written on the question sheet will not be scored.

(a) 同一個清單裡面裝的元素的資料型態必須都相同。
All elements stored in the same list must be of the same data type.

(b) 在 Python 程式裡，break 會跳出包住這個 break 的最內層的 if 區塊。
In a Python program, a break statement exits the innermost if block that encloses it.

(c) 如果有一個清單名爲 A，則 len(A) 會回傳 A 中的元素個數減一的值。
If there is a list named A, then len(A) will return the number of elements in A minus one.

第二題 Question 2

(35 分；每小題 5 分) 在以下的 Python 程式中，輸出結果會是什麼？如果你認爲程式有 syntax error/TypeError/runtime error，就請在答案卷上寫下「error」即可不用解釋；如果你認爲會輸出複數行，就依序一行一行寫下你認爲的輸出。
請在答案紙上寫下你認爲正確的答案。在題目卷上的作答不會被計分。

(35 points; 5 points each) In the following Python programs, what will be the output? If you think the program has a syntax error/TypeError/runtime error, just write "error" on the answer sheet; if you think it will output multiple lines, write down your expected output line by line in order.
Write down the option or answer you think is correct on the answer sheet. Answers written on the question sheet will not be scored.

(a)

```
total = 0
for i in range (5):
    for j in range (3):
        if i + j == 5:
            total += i
print(total)
```

(b)

```
count = 0
values = [1, 2, 3, 4, 5, 6]
for i in range(values):
    if i % 2 == 1:
        count += values[i]
    else:
        continue
print(count)
```

(c)

```
target = 8
values = [9, 8, 7, 7, 6, 4, 7, 3, 8, 5]
for x in values:
    if x <= target + 1 and x >= target - 1:
        print(x)
    else:
        print (0)
```

(d)

```
count = int()
for x in [1, 2, 3, 4, 5, 6]:
    count += 1
print(count)
```

(e)

```
total = 0
```

```
begin = [2, 8, 4, 9]
end = [3, 7, 2, 2, 1]
for f in begin:
    if f >= 5:
        for t in end:
            if f + t == 10:
                total += f
                break
print(total)
```

(f)

```
a = 0.1
b = 0.2
if a + b == 0.3:
    print('Yes')
else:
    print('No')
```

(g)

```
a = 3
b = 2
c = int(a+b)
for i in range(1, c):
    if i == 4:
        continue
    if c % i == 0:
        print(i)
```

第三題 Question 3

(25 分) 針對以下題目，請寫一個完整的程式，去完成指定任務。請使用正確的 Python 語法 (包含排版)，不過如果這份程式概念上是正確的，只是語法、排版、邏輯上有一些瑕疵，那這份程式還是會得到部分分數。同學們得要確保助教們看得懂你的答案才行。如果助教實在看不懂，那份答案就一定會被扣分或甚至零分。

For the following question, please write a complete program for each to accomplish the specified task. Please use correct Python syntax (including formatting). However, if the program is conceptually correct but has some flaws in syntax, formatting, or logic, it will still receive partial credit. You must ensure that teaching assistants can understand your answers. If the teaching assistant really cannot understand, that answer will definitely be deducted points or even receive zero points.

題目敘述：你經營一間連鎖飲料品牌，共有 `store_cnt` 家分店，在每間店都賣同樣的 `drink_cnt` 種飲料。昨天第 i 家店的第 j 種飲料的銷售杯數是 `sales[i-1][j-1]`，換言之 `sales` 是一個二維清單。你想印出一共有幾家店有至少兩種飲料的銷售杯數達到 `target` 杯以上。

請寫一個完整、正確 (包含排版) 的 Python 計算模組，使用已經存好正確數值的 `store_cnt` (整數)、`drink_cnt` (整數)、`sales` (二維整數清單) 和 `target` (整數) 變

數中的資料，進行計算後完成上述任務。[1]。提示：使用迴圈來看每一家店。

Question Statement: You operate a chain beverage brand with `store_cnt` branches. Each store sells exactly `drink_cnt` kinds of drinks. Yesterday, the number of cups sold of the j-th drink in the i-th store was `sales[i-1][j-1]`; in other words, `sales` is a two-dimensional list.

Print how many stores sold **at least two** kinds of drinks whose cup sales reached `target` or more.

Write a complete, correctly formatted Python module that uses the already populated variables `store_cnt` (int), `drink_cnt` (int), `sales` (2-D list of int), and `target` (int) to perform this calculation.[2] Hint: use loop to check each store.

第四題 Question 4

(28 分) 小傑正準備與心儀的女生的第一場約會，不過小傑平日的工作相當繁重，他必須盡力在期限前做完手邊的所有工作才能夠去赴約。具體來說，小傑手邊總共有 n 件工作，每一個工作都有一個固定的處理時間 p_i，以及工作必須完成的的期限 d_i。

對於這些工作，小傑會安排好一個「工作順序」(簡稱「順序」) s 去逐一完成每一項工作，例如 $s = (4, 2, 1, 3)$ 表示小傑要依序做工作 4、工作 2、工作 1，最後做工作 3。當然，任何一項順序都無法保證所有工作都可以在期限前完成。每項工作的「延遲時間」的計算方式如下。首先，把該工作的完成時間 x_i 減去該工作的期限 d_i，如果算出來是正的，該數字就是這項工作的延遲時間; 反之如果是負的，表示該工作在期限前就已經完成了，則該工作的延遲時間爲 0。

如果所有工作的總延遲時間太多，那位女生就會認定小傑工作太繁忙了，進而取消約會，因此小傑必須有辦法評估一個給定順序會導致的總延遲時間。

舉例來說，假設小傑手邊有 4 個工作，依序分別是 1、2、3、4 號工作，這些工作的處理時間依序是 5 小時、4 小時、3 小時、2 小時，完成期限分別是開始工作後的第 6 小時、第 7 小時、第 8 小時、第 10 小時。若小傑安排的工作順序爲 $(1, 4, 2, 3)$，則總延遲時間的計算過程如下:

1. 第一步做 1 號工作並開始計時，5 小時後工作完成，完成的時間爲開始後第 5 小時，不超過給定的期限第 6 小時，總延遲時間目前爲 0。

2. 第二步做 4 號工作並開始計時，2 小時後工作完成，完成的時間爲開始後第 $5 + 2 = 7$ 小時，不超過給定的期限第 10 小時，總延遲時間目前爲 0。

3. 第三步做 2 號工作並開始計時，4 小時後工作完成，完成的時間爲開始後第 $7 + 4 = 11$ 小時，超過給定的期限 $11-7 = 4$ 小時，總延遲時間目前爲 4 小時。

[1]如果這份程式概念上是正確的，只是語法、排版、邏輯上有一些瑕疵，那這份程式還是會得到部份分數。同學們得要確保助教們看得懂這個程式；如果助教實在看不懂，那就一定會被扣分或甚至零分。在此前提下，同學們可以自行決定要不要寫註解；如果覺得自己的程式很容易被理解，就可以不用寫註解。

[2]If the program is conceptually correct but contains minor syntactic, formatting, or logical flaws, partial credit will still be awarded. Students must ensure that the teaching assistants can understand the program; if the TAs truly cannot understand it, points will be deducted or even a zero grade. Under this premise, students may decide whether to include comments; if the program is easy to understand, comments are optional.

4. 第四步做 3 號工作並開始計時，3 小時後工作完成，完成的時間爲開始後第 $11+3=14$ 小時，超過給定的期限 $14-8=6$ 小時，總延遲時間最終爲 10 小時。因此，小傑在這個工作順序下總延遲時間爲 10 小時。

請參考以上例子與依照題目指示，根據給定的工作資訊，輸出給定工作順序下的總延遲時間。

輸入格式: 輸入共有四列：

1. 第一列：單一正整數 n
2. 第二列：n 個正整數 $p_1, p_2, \ldots, p_n$
3. 第三列：n 個正整數 $d_1, d_2, \ldots, d_n$
4. 第四列：n 個正整數 $s_1, s_2, \ldots, s_n$ (工作順序)

同一列內數字以逗號分隔。已知 $1 \leq n \leq 20$，$1 \leq p_i \leq 50$，$1 \leq d_i \leq 1000$，$1 \leq s_i \leq n$ 且 s_i 互不重複。

輸出格式: 輸出一個整數，爲所給順序的總延遲時間。提示 (完成下列函數):

```
def total_delay(n, p, d, s):
    return
```

(28 points) Xiao-Jie is preparing his first date with the girl he likes. However, Xiao-Jie's daily workload is quite heavy, and he must finish all his tasks before their deadlines in order to go on the date. Specifically, Xiao-Jie has a total of n tasks. Each task has a fixed processing time p_i and a deadline d_i by which it must be completed. For these tasks, Xiao-Jie arranges a "task sequence" (simply "sequence") s to complete the tasks one by one; for example, $s = (4, 2, 1, 3)$ means that Xiao-Jie will do task 4, task 2, task 1, and finally task 3 in that order. Of course, no sequence can guarantee that all tasks are finished before their deadlines. The "tardiness" of each task is calculated as follows. First, subtract the task's deadline d_i from its completion time x_i. If the result is positive, that number is the task's tardiness; otherwise, if the result is negative, the task was completed before the deadline, so its tardiness is 0. If the total tardiness of all tasks is too large, the girl will conclude that Xiao-Jie is too busy and cancel the date. Therefore, Xiao-Jie must be able to evaluate the total tardiness produced by a given sequence.

For example, suppose Xiao-Jie has four tasks, numbered 1 through 4. Their processing times are 5 h, 4 h, 3 h, and 2 h, and their deadlines are at hours 6, 7, 8, and 10 after starting work, respectively. If the sequence is $(1, 4, 2, 3)$, the total tardiness is calculated as follows:

1. Do task 1 first and start timing. It finishes 5 h later, at hour 5, which does not exceed its deadline of hour 6; total tardiness is 0.
2. Do task 4 next. It finishes 2 h later, at hour $5 + 2 = 7$, which does not exceed its deadline of hour 10; total tardiness is still 0.
3. Do task 2 next. It finishes 4 h later, at hour $7 + 4 = 11$, which exceeds its deadline by $11 - 7 = 4$ h; total tardiness is now 4 h.

4. Do task 3 last. It finishes 3 h later, at hour $11 + 3 = 14$, which exceeds its deadline by $14 - 8 = 6$ h; total tardiness finally becomes 10 h. Thus, the total tardiness for this sequence is 10 h.

Please refer to the above example and, following the problem instructions, output the total tardiness for the given task sequence based on the provided task information.

Input Four lines:

1. Line 1: a single integer n
2. Line 2: n integers $p_1, \ldots, p_n$ (processing times)
3. Line 3: n integers $d_1, \ldots, d_n$ (deadlines)
4. Line 4: n integers $s_1, \ldots, s_n$ (job order)

Numbers in the same line are comma-separated. Constraints: $1 \leq n \leq 20$, $1 \leq p_i \leq 50$, $1 \leq d_i \leq 1000$, $1 \leq s_i \leq n$ with all s_i distinct.

Output: Output a single integer—the total tardiness for the specified job sequence. Hint (complete the function):

```
def total_delay(n, p, d, s):
    return
```