
Teaching Portfolio

Project Maieutic 學問 · keeping students thinking in the age of AI

Dr. CHAN Chung

Teaching Fellow, Department of Computer Science, City University of Hong Kong · October 2026

1 Teaching Statement

My single teaching aim is to help students keep thinking, and therefore keep learning, so that they remain employable in the age of artificial intelligence. I use the Socratic idea behind the name of my current initiative, *Project Maieutic* (學問): learning is not achieved by receiving direct answers, but by asking the right questions. In computer science this is urgent, because modern generative AI can produce bug-free code at the press of a button, and students who rely on it skip the computational thinking and debugging that make them employable. A first-year student who scored 99.5% on assignments yet 30% on the midterm showed the risk clearly: the AI had removed the thinking and created an illusion of competence.

To keep the thinking, I do not ban AI; I make it a tool that serves one purpose, the student's learning. The digital platforms I have built (DIVE, E-Quiz, the AI-assisted debugger, SocraticAI) are therefore a means, not an end: they let a student run, modify, and debug real code; receive hints rather than answers; and learn to question what an AI produces. This philosophy is stated plainly in the course materials, which open with Socrates, "I cannot teach anybody anything, I can only make them think," and close with the question "Can you teach AI to teach you?"

2 Curriculum Design for the AI Era

These practices serve a discovery-enriched curriculum (DEC), CityU's model in which courseware is **D**iversified, **I**nteractive, **V**ersatile, and **E**ngaging (the DIVE environment). I design courses and courseware for it so that students of very different backgrounds can explore freely, get meaningful feedback, reuse the tools across subjects, and want to go deeper.

In 2025/26 I led a combined first-year programming course of more than 600 students and deployed Team-Based Learning (TBL) at scale. When the course is split from 2026/27, the split itself is a pedagogical decision matched to audience and depth:

- **CS1302A (new, Semester A, about 200 CS students)** gives computer science students a deeper treatment of programming, emphasising agentic and specification-driven coding and raising questions on language design throughout the lectures.
- **CS1302 (Semester B, more than 400 non-CS students)** is the first-year programming course for students from many departments across the University, including CENG, CSCI, CB, Physics and computational mathematics. I also manage the courseware and digital learning materials for its Semester A offering, which I do not personally teach while running CS1302A.
- **CS2310 (about 70 students)** covers C++ and object-oriented programming with Team-Based Learning.

This structure lets the same philosophy serve two goals at once: non-CS students reach a strong level of programming, and CS first-years are pushed toward agentic coding and language design.

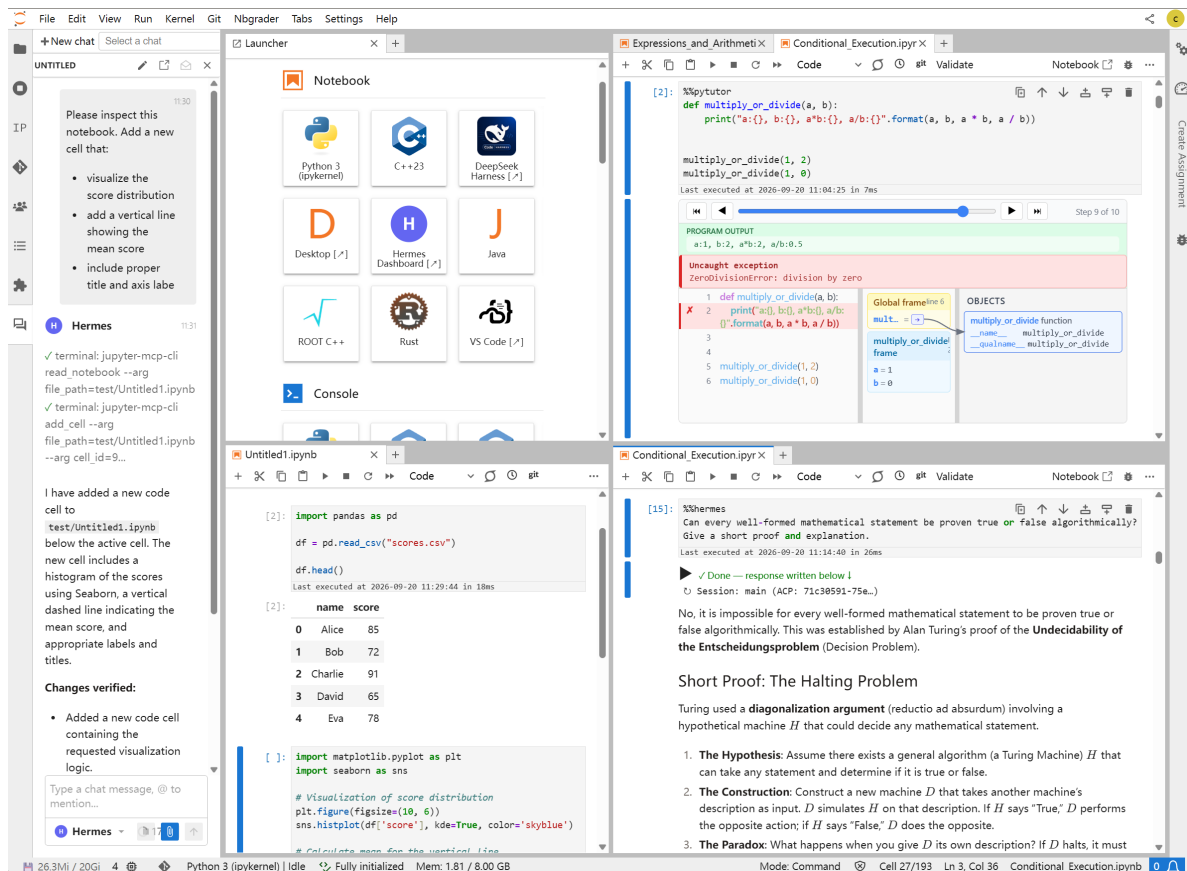
3 Teaching Pedagogies and Practices

3.1 Innovativeness: a Socratic, agent-augmented environment

- **DIVE** (github.com/dive4dec/jupyter) runs interactive, executable teaching materials on an in-house JupyterHub / Kubernetes cluster, so students can modify, execute, and debug code and access shared CPU/GPU resources without any local setup [1].
- **E-Quiz** (github.com/dive4dec/e-quiz) is a self-contained, programmable online exam system integrated with Canvas; it lets students check and re-attempt answers interactively and underpins fair, computer-based assessment [4].
- **SocraticAI** is a family of small language models built on Qwen 3.5 (1.5B parameters, taken as-is) that give hints and guiding questions rather than solutions, inside a visual debugger that runs entirely in the browser [2, 5]. When the teaching team discovered that the model could be jailbroken to obtain direct answers, I turned the finding into a project; a first-year student fixed it by retraining the model so that it can no longer be bypassed. It is now used by hundreds of students in exams without removing the thinking.
- **CE-Quiz** runs collaborative, in-class programming in which the differences among student solutions drive debugging and peer learning, supported by classmates, PALSI leaders, and the AI-assisted debugger [3].
- **Project Maieutic** (dive4dec.github.io/obsidian-vault) brings these together as a specification-driven, agentic approach: students write clear specifications, guide a self-learning agent to implement and test them, and critically audit what the AI produces. The specification-driven agentic-coding courseware is public at ccha23.github.io/cs1302iedb/agentic-coding, and the supporting infrastructure (`jupyter-ai-hermes`, `hermes-acp-sdk`, and `jupyterlab-pwa`) is open source and can be re-deployed by any institution.

3.2 The Human-Agent Loop: students as learners and teachers

The sharpest expression of the philosophy is the human-agent coding loop, worked out in the specification-driven agentic-coding materials and in the summer pilot that produced the IEEE TALE 2026 paper [6]. Rather than minimising human intervention, the design deliberately keeps the student in the loop: students learn by inspecting agent-generated code and explanations, and they teach the agent through clearer specifications, stronger tests, and evidence-based diagnoses. During specification the agent questions the student; during verification the student writes tests and uses failures, logs, and traces to drive revisions; and during code review the student asks the agent to explain its work while judging its correctness. Debugging becomes visible, because every revision requires the student to give the agent actionable feedback.



The DIVE workspace with the DeepSeek Harness available as a launchable app: the student opens DIVE (JupyterLab) and selects the harness to use, so the self-learning Hermes agent and the DeepSeek Harness (dsh) can be invoked for different purposes. Here the agent reads the live notebook and edits it through tool calling (left), while the student specifies the goal and audits the result. The same environment exposes in-notebook magics, `%hermes` and `%pytutor`, so the student can question the agent, run and debug code, and verify its work without leaving the cell.

3.3 A curriculum built, not assembled: what is new in 2026

First-year programming at leading institutions (UC Berkeley CS61A, MIT 6.0001, Stanford CS106A) rests on a recognisable set of pieces: a standard textbook, lecture-driven homework, and a shared cloud tool for visual debugging. CS61A is the closest analogue; it opened a coding-agents module in week 11 of a 15-week course (Fall 2026 syllabus, cs61a.org/fa26). Those courses treat AI as a topical addition introduced part-way through.

The curriculum developed here differs on five points, and each is a design decision made rather than inherited:

- **Specification-driven agentic coding is the spine from the first week, not a late-semester module.** Students learn to prompt, steer, and audit the self-learning Hermes agent inside the DIVE workspace and the DeepSeek Harness, using `%hermes` and `%pytutor` cell magics and the in-browser debuggers. This is what the IEEE TALE 2026 paper formalises [6].
- **The courseware is authored as live notebooks and notebook-based books, not adopted from a standard textbook.** The agentic-coding units ship as Jupyter notebooks that pull directly into the DIVE workspace; AI Agent cells run inside the narrative,

and the materials explicitly teach students to ask the follow-up questions that go deeper (github.com/dive4dec/cs1302_edb, ccha23.github.io/cs1302iedb/agent-coding).

- **The visual debuggers are built, not adopted.** Classic Online Python Tutor requires a public server, needs a live network, and can be prompted to hand out the answer. The OPT_Mentor and OPT_CPP debuggers run entirely in the browser on the newest stable interpreters (Python 3.14 via Pyodide v314.0.2, C++23 via xeus-cpp 0.10.0 and Clang 21.1.8), run offline in an exam hall, and pair the frame state with Socratic hints rather than direct fixes.
- **First-years build and publish open-source packages to PyPI, and run a first-year capstone equivalent.** In the 2025/26 summer pilot, five first-year participants shipped `hermes-acp-sdk`, `traceback-coach`, `slideshow`, `socratic-watchdog`, and `ultrasonic-img-steg` as reusable, dependency-tested packages whose source is public.
- **AI misuse is redesigned around, not prohibited.** CE-Quiz collaborative coding, the TBL restructure at scale, and SocraticAI hints-not-answers are designed so that a student who attempts to prompt their way out of the task produces a wrong result, not a faster answer. Where the common response is a proctored “no AI allowed” rule, the design here makes the rule unnecessary by making the shortcut fail.

To my knowledge, none of the three courses above runs this combination in first-year. The 2025/26 outcomes in the Effectiveness subsection (failure rate from 9.5% to 4.0%, average midterm from 54% to 70%, non-CS students reaching a strong command of programming) are the first signal that the design transfers.

3.4 Effectiveness: students keep thinking, and it shows

- **Objective learning outcomes.** In the first full-scale TBL semester (2025/26, combined course of more than 600 students), the CS1302 failure rate fell from 9.5% to 4.0% and the average midterm score rose from 54% to 70%.
- **Non-CS students perform well.** Students from many departments, including Physics and computational mathematics, complete CS1302 with a strong command of programming and debugging.
- **First-years become co-creators.** Rather than only consuming technology, first-year students now build and ship it. In the summer pilot behind the IEEE TALE 2026 paper [6], most first-year participants moved from an initial idea to a working package prototype in just two weeks, and a group specified, built, and published five open-source packages to PyPI (`hermes-acp-sdk`, `traceback-coach`, `slideshow`, `socratic-watchdog`, `ultrasonic-img-steg`), and their first-year projects were showcased at the 2026 Info Day (3 October). These packages and their demonstrations are part of the work submitted to IEEE TALE 2026.
- **Consistently high Learning Outcome scores,** a College of Engineering Outstanding Teaching Award (2020-21), and strongly positive student feedback.

4 Students as Co-Creators

A distinguishing feature of my practice is that students move from learners to contributors. The reusable packages also create a peer-assessment practice: teams exchange repositories, use an agent to understand another team’s code, and submit tested pull requests; both the contribution and the review decision (accept a sound change, or reject a flawed one with a technical

reason) are assessed, so students learn to critique as well as to produce. First-years who do well return as PALS leaders in later offerings; we hire our graduates to develop and maintain DIVE, E-Quiz, and the AI-enabled infrastructure (for example, automating the download of about 300 student answer scripts for HKIE accreditation); and graduate and postgraduate researchers guide first-years in building projects and writing papers. As a result, several undergraduates, including first-years, are now co-authors on accepted conference papers and maintainers of open-source software, and they take on their first-year projects to publish reusable software, not just pass a course. The platforms and student projects, with demonstration videos, are public at dive4dec.github.io/obsidian-vault.

5 Supporting materials

See the platforms in motion. A video worth a thousand words: demonstration videos of the DIVE workspace, E-Quiz, the visual debuggers, and the DeepSeek Harness, together with the full project page, are on the Project Maieutic site, dive4dec.github.io/obsidian-vault (§The Platforms).



5.1 DIVE / JupyterHub

The DIVE platform is an executable book launchable on JupyterHub and in the browser, with interactive Python/C++, GPU access, and integrated AI.

The screenshot displays the DIVE platform interface. On the left is a navigation menu with sections like '1. Introduction to Computer Programming', '2. Values and Variables', '3. Expressions and Arithmetic', '4. Conditional Execution', '5. Iteration', '6. Using Functions', '7. Writing Function', '8. Objects', '9. More on Functions', '10. Lists and Tuples', '11. Dictionaries and Sets', '12. Monte Carlo Simulation and Linear Algebra', and '13. Review Questions'. Below this is a 'LAB ASSIGNMENTS' section with tasks like '1. Setup', '2. Card guessing game', '3. Calculators', '4. Improved Quadratic Equation Solver', '5. Big Number Conversion', '6. Mastermind', '7. Pandas', '8. Combinatorics', '9. Cybersecurity', '10. Information Theory', and '11. Monte Carlo and Root Finding'. The main content area shows a Jupyter notebook with a code cell containing a function `estimate_chance_of_winning_by_change` and a test cell. Below the code is a section titled '11.2. Solving a 3-by-3 system of linear equations' which explains that `numpy` has a module `linalg` for linear algebra and provides a function called `solve`. It shows the system of equations $2x_0 + 2x_1 = 1$ and $2x_1 = 1$, and then the solution using `np.linalg.solve`. The matrix form is shown as $\begin{bmatrix} 2 & 2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

The same courseware powers both CS1302A and CS1302, and is open to staff and students with a valid CS account (JupyterLab, VS Code, and classic notebooks). The same platform has also carried a graduate data-mining course and a research-level information-theory course, each

with its own tooling and GPU needs, which shows that one environment can serve very different audiences.

For agentic coding, DIVE provisions a separate, isolated environment for each student. Because agents consume many model requests, the stack is built to be fair and sustainable: requests are routed through a proxy that enforces per-student quotas, and the models are self-hosted on institutional GPU nodes, so student records and prompts stay under institutional control.

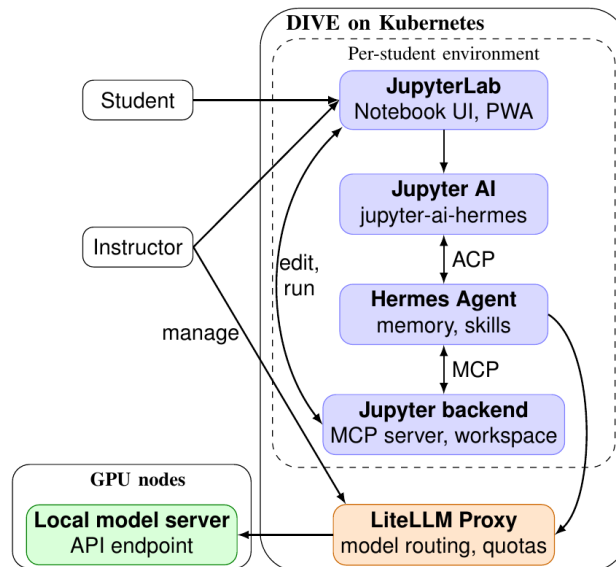


Fig. 1: Architecture of the agentic coding environment in DIVE. ACP connects Jupyter AI to a per-student Hermes Agent, while MCP exposes live notebook context and operations. LiteLLM manages model routing and quotas for the local model server hosted on GPU nodes.

Architecture of the agentic coding environment in DIVE (from the IEEE TALE 2026 paper [6]): a per-student environment on Kubernetes, with the agent bridging the notebook (ACP) and the workspace (MCP), and a proxy routing model calls to a self-hosted local model on the GPU nodes. The instructor retains a management path to each student's workspace.

5.2 E-Quiz

E-Quiz delivers programmable, computer-based exams and practice with immediate, AI-assisted feedback.

Question 1

Correct

Mark 1.00 out of 1.00

Suppose `dic` is a `dict` containing the grades of students in one quiz with `key:value` pairs, where `key` is student number and `value` is the grade.

Please define a function `grade_to_decision` such that `grade_to_decision(dic)` can

- substitute the value/grade in each item in `dic` to `'Fail'` if the grade of the student is lower than 60, to `'Pass'` otherwise;
- return `True` if every student get a `'Pass'` value, and return `False` otherwise.

For example, when `dic = {'1001':99, '1002':60, '1003':55}`, the item `'1001':99` means the grade of the student with student number 1001 is 99. Then `grade_to_decision(dic)` is expected to mutate `dic` to `{'1001': 'Pass', '1002': 'Pass', '1003': 'Fail'}`, and return `False`.

For example:

Test	Result
<pre>dic = {'1001':99, '1002':60, '1003':55} assert grade_to_decision(dic) == False assert dic == {'1001': 'Pass', '1002': 'Pass', '1003': 'Fail'}</pre>	

Answer: (penalty regime: 10, 20, ... %)

```

1 def grade_to_decision(dic):
2     output = True
3     for sid in dic:
4         if dic[sid] >=60:
5             dic[sid] = 'Pass'
6         else:
7             dic[sid] = 'Fail'
8             output = False
9     return output

```

Check

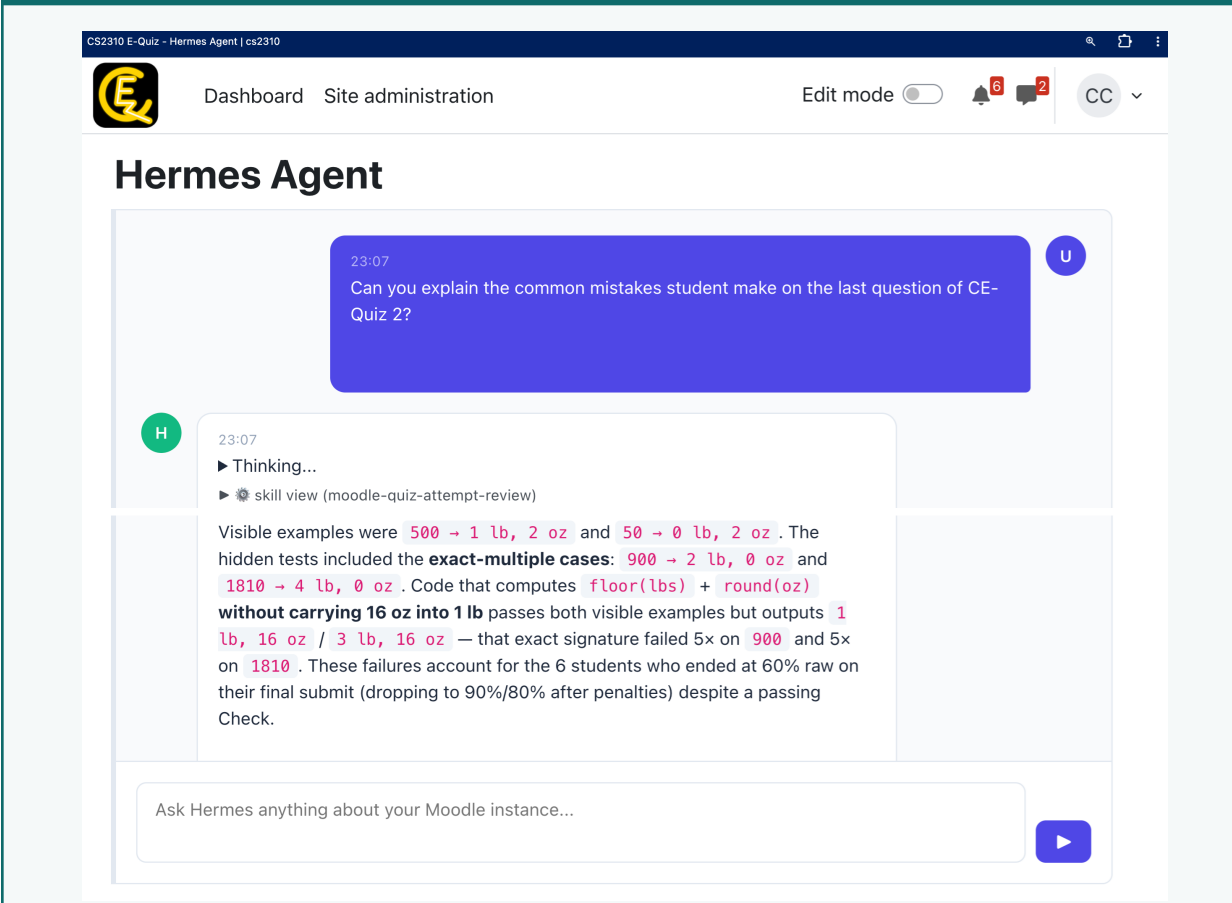
Test	Result
<pre>dic = {'1001':99, '1002':60, '1003':55} assert grade_to_decision(dic) == False assert dic == {'1001': 'Pass', '1002': 'Pass', '1003': 'Fail'}</pre>	✓

Students can visualise the execution of their answers with an in-browser interpreter, and E-Quiz supports sophisticated mathematical questions that accept algebraically equivalent answers and give partial credit, so students discover the right answer rather than being spoon-fed one.

Two design choices matter most for fair, large-scale computer-based assessment. First, Extensive Modular Randomization (EMR) regenerates each question per student and per attempt, which discourages cross-checking answers. In a course where five plagiarism cases were flagged before EMR (four of them confirmed), none were identified after its introduction, while the mean exam score actually rose, from 80.8% to 82.4%, with a tighter spread. Second, the system is built to scale: it was load-tested to 500 concurrent users with no errors, and students rate its features favourably (around 4.2 to 4.6 out of 5), with about four in five preferring it for exams. In one student's words, it makes "exams less stressful, especially for those with math components."

Why a self-contained LMS with an agent. A traditional, centrally managed LMS does not give a teacher enough power to learn the individual progress of their own students, let alone to install or build the learning tools a course needs. And being able to chat with a large language model is not enough either: unlike an agent, a chat model cannot make changes, perform complicated database queries, or learn new skills to complete new tasks. Because E-Quiz is open source and self-contained, it hands that power to the individual teacher, and the self-learning agent carries it out.

Agentic E-Quiz: the self-learning Hermes agent inside a self-contained LMS



The screenshot shows the E-Quiz interface. At the top, there's a header bar with the title "CS2310 E-Quiz - Hermes Agent | cs2310". Below it, a navigation bar includes a logo, "Dashboard", "Site administration", "Edit mode" toggle, notification icons (6 and 2), and a Creative Commons license icon. The main content area is titled "Hermes Agent". It shows a conversation where a user (U) asks: "Can you explain the common mistakes student make on the last question of CE-Quiz 2?". The agent (H) responds with a detailed analysis: "Visible examples were 500 → 1 lb, 2 oz and 50 → 0 lb, 2 oz. The hidden tests included the exact-multiple cases: 900 → 2 lb, 0 oz and 1810 → 4 lb, 0 oz. Code that computes floor(lbs) + round(oz) without carrying 16 oz into 1 lb passes both visible examples but outputs 1 lb, 16 oz / 3 lb, 16 oz — that exact signature failed 5x on 900 and 5x on 1810. These failures account for the 6 students who ended at 60% raw on their final submit (dropping to 90%/80% after penalties) despite a passing Check." At the bottom, there's a text input field "Ask Hermes anything about your Moodle instance..." and a play button icon.

A teacher asks the integrated Hermes agent, in plain language, to explain the mistakes students made on a quiz. The agent reads the quiz data, runs a skill, and reports back with a diagnosis, with the exact examples that were failing. (From the Project Maieutic poster.)

A self-contained LMS with an AI agent inside. E-Quiz, a self-contained learning management system built on Moodle, is published as an open-source tool and is already used by other teachers in the department. It is also where I introduced the self-learning Hermes agent. Because the whole platform runs on the institution's own servers, a teacher can ask the agent, in plain language, to help with their course without writing any code and without their students' data leaving campus. In practice a teacher can ask it to summarise the most common mistakes students made on an assignment, help design a learning activity, or read the system logs and fix an issue; they can also reach the LMS through a messaging app on their phone and teach the agent new skills so it completes the tasks they need. Because E-Quiz is open source and self-contained, a teacher can be the administrator of their own instance and install or develop the plugins their course requires. In the AI era, when applications can be built quickly, that makes E-Quiz a platform any teacher, including teachers outside computer science, can extend for their own teaching.

5.3 The software stack: packages developed for one environment

The platforms are not products bought and configured; they are a stack of more than twenty open-source packages developed over several years and assembled in a single repository, github.com/dive4dec/dive-deploy, that any institution can re-deploy with one command. Their

central aim is to put AI agents within a student's reach, in the place where they work: `jupyter-ai-hermes` brings a self-learning agent into the Jupyter environment, and the DeepSeek Harness is available as a launchable app in DIVE so students can switch between agents for different purposes. Because the agents are this accessible, even first-year undergraduates can deploy non-trivial open-source packages, train new AI models, and publish at prestigious international conferences. The models the agents call on are open-weights and run on CityU's own servers, a choice that is sustainable and scalable in the long term: it supports roughly 800 students using agents every semester, some running the heavy agentic coding needed to complete a final year project, and avoids a large recurring bill for external model subscriptions, on the order of a million dollars per year. The main packages, grouped by function, are listed below with their public repositories, each of which carries its own documentation and examples.

- **Workspace and agent integration.**

- github.com/dive4dec/jupyter: the DIVE Jupyter environments (JupyterHub, VS Code, classic notebooks, GPU group servers).
- github.com/dive4dec/jupyter-ai-hermes: integrates the self-learning Hermes Agent into Jupyter.
- github.com/dive4dec/jupyter-ai-hermes-magics: the `%hermes` and `%pytutor` cell magics (persistent ACP connection, multi-session, streaming).
- github.com/dive4dec/jupyter-hermes-proxy: proxies the Hermes Agent to each student session.
- [jupyter-dsh-proxy](https://pypi.org/project/jupyter-dsh-proxy/) (PyPI): proxies the DeepSeek Harness web UI to each student session.
- github.com/dive4dec/jupyter-python-tutor: step-by-step Python execution in notebooks.
- github.com/dive4dec/jupyter-cpp-tutor: step-by-step C++ execution in notebooks.
- github.com/dive4dec/jupyterlab-pwa: makes JupyterLab a Progressive Web App.

- **Assessment.**

- github.com/ccha23/e-quiz: the self-contained, programmable exam system.
- github.com/dive4dec/local_hermesagent: the Moodle plugin that runs the Hermes Agent inside E-Quiz.
- github.com/dive4dec/local_hermes-synapse: the Hermes agent plugin and skills that let teachers analyse student submissions directly from the course site; plus LTI login, LTI groups, PWA, and profile-lock plugins for the E-Quiz deployment.

- **Debugging and visualisation.**

- github.com/dive4dec/OPT_Mentor: a visual Python debugger with Socratic AI hints, running entirely in the browser (WebLLM, no server).
- github.com/dive4dec/OPT_CPP: a serverless visual C++ debugger.
- github.com/dive4dec/optlite: a serverless Python tutor on Pyodide.

- **Model serving and fair use.**

- github.com/dive4dec/dive_ai: deploys the self-hosted models, including the SocraticAI family.

- github.com/dive4dec/jupyterhub-litellm: per-student API keys with budget enforcement.
- github.com/dive4dec/litellm: the OpenAI-compatible proxy that fronts the shared model endpoints.
- github.com/dive4dec/dsh-openai-shim: the OpenAI-compatibility layer inside the DeepSeek Harness.
- **Capacity management.** pypi.org/project/jupyterhub-storage-usage lets the hub administrator monitor each student's storage usage across the JupyterHub cluster, and pypi.org/project/jupyter-storage-usage lets a student check their own usage inside the workspace; both are published on PyPI. Many JupyterHub storage backends do not enforce a per-user quota, so these add one, making each student's usage visible and bounded.

The two in-browser debuggers are the most visible to students: both step code line by line, show frames, objects, and memory, and pair the error with a Socratic hint rather than an answer. Scan a code, or open the repository directly, to try them.



OPT_Mentor (Python): step 39 of 46, with frames, objects, and an AI hint. Live demo: ccha23.github.io/OPTM/live.html



OPT_CPP (C++): step 44 of 62, with stack, print output, and heap. Live demo: dive4dec.github.io/OPT_CPP

The in-browser visual debuggers. The demos run live in a browser tab, no login; the QR codes open the same pages on a phone.

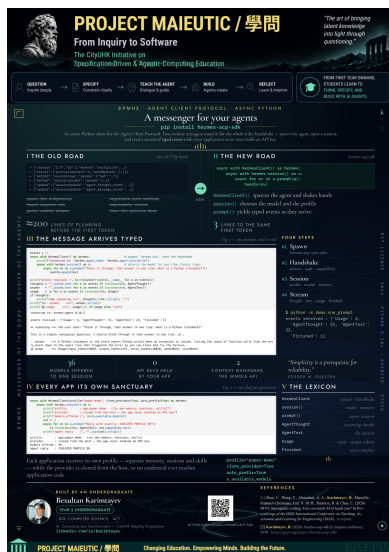
Two design principles run through the stack. First, everything is open source, so students can read, inspect, and extend the very tools that teach them; the agentic-coding courseware builds on this directly. Second, the stack is built to be re-deployed elsewhere: the dive-deploy repository assembles all components with one command, and only the Moodle question-type plugins require institution-specific adjustment. The two exceptions to the open-source principle are deliberate adaptations of existing open source, overleaf-cep and jobeinabox, which are kept for completeness.

5.4 Student-Developed Packages

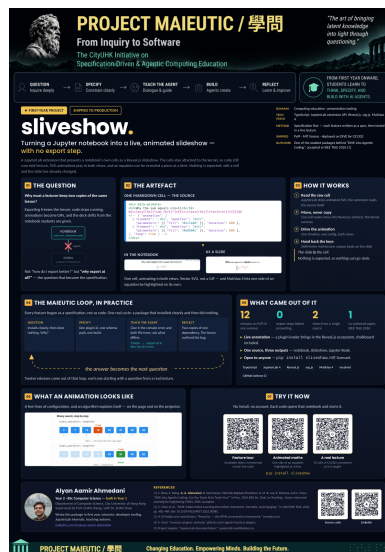
See the five first-year packages in motion. Each has a demonstration video and a full project poster (PDF) on the Project Maieutic site, dive4dec.github.io/obsidian-vault (§Student First-Year Projects); scan the code or open the URL.



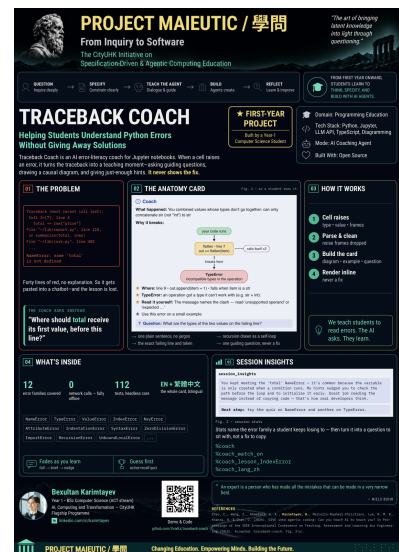
All five first-year packages were specified, built, and shipped to PyPI by first-year undergraduates. Their posters, shown below, cover the full range: an ACP client, interactive presentation, error explanation, signal processing, and an agent watchdog.



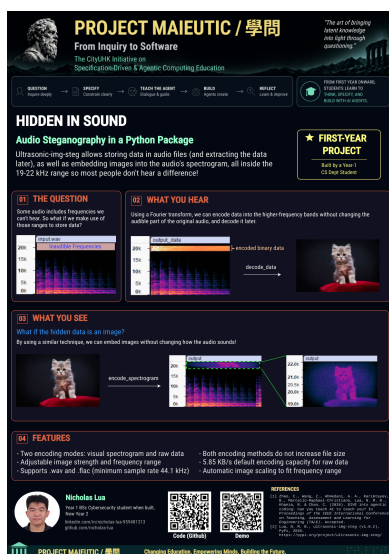
hermes-acp-sdk



slideshow



traceback-coach



ultrasonic-img-steg



socratic-watchdog

Posters of the five first-year packages, each built and published to PyPI and submitted to IEEE TALE 2026. Open the site (or scan the code above) for the full posters and demonstration videos.

6 Dissemination and Outreach

- **Peer-reviewed papers.** IEEE TALE: DIVE [1], E-Quiz [4], SocraticAI [2], and *DIVE into Agentic Coding* [6]; INTED 2026 (CE-Quiz) [3]; EDULEARN 2026 (SocraticAI) [5].
- **Government and schools.** Demonstrated E-Quiz and CE-Quiz to the Education Bureau and school leaders (28 April 2026, video-recorded); spoke on AI in Science Education to 65 secondary school teachers.
- **Gifted secondary students.** Invited as a mentor for the CityUHK EE Gifted Education Fund Programme (Institute of Future Learning) and gave a talk on generative AI in education to about 50 gifted secondary students (F1-F5).
- **Prospective students.** Built the [CityUHK CS Admission FAQ](#), a 400+ note, AI-agent-powered site (built on the DIVE agentic-coding platform, with an optional local model for privacy) that answered prospective students' JUPAS questions at the 2026 Info Day and is proposed for adoption on the department website; because the site itself runs on an AI agent, it shows incoming students the agentic-coding approach that underpins the first-year programme.
- **Open courseware.** Publicly accessible, browser-based courseware for CS1302, CS2310, and CS5483; student packages on PyPI.

Acknowledgments

This environment would not exist without the CSLab, OCIO, CSC, and the many students and staff who have used, improved, and extended it, and without the support of the Talent and Education Development office and the Office of the Provost and Deputy President for Team-Based Learning, PALSI, and the digital learning initiatives on which this work builds.

References

- [1] Chung Chan, Assan Kozhin, Qihang Liang, Glenn Salter, Ruoqin Tang, Chunxiao Ye, and Chao Zhao. DIVE: Make online learning diversified, interactive, versatile, and engaging. In *2022 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 2022.
- [2] Chung Chan, Bogdan Suvernev, Zimeng Wang, Qihang Liang, and Chao Zhao. Making GenAI socratic for computational thinking. In *2025 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 2025.
- [3] Chung Chan, Bogdan Suvernev, Chao Zhao, and Zimeng Wang. CE-Quiz: AI-assisted debugging in exams. In *INTED 2026: 20th International Technology, Education and Development Conference*. IATED, 2026.
- [4] Chung Chan, Chao Zhao, Wai Tong Chau, Yu Zhou, Qihang Liang, and Michael Lim. E-quiz: Empowering educators with a self-contained and programmable online exam system. In *2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 1–8. IEEE, 2023.
- [5] Bogdan Suvernev, Chao Zhao, Zimeng Wang, and Chung Chan. SocraticAI: On training models that make students think. In *Proceedings of EDULEARN 2026: 18th International Conference on Education and New Learning Technologies*. IATED, 2026.

- [6] Chao Zhao, Zimeng Wang, Alyan Aamir Ahmedani, Bexultan Karimtayev, Marcello Raphael Christiano, Nicholas Marcus Bayani Lua, Rauan Khamza, Bogdan Suvernev, and Chung Chan. DIVE into agentic coding: Can you teach ai to teach you? In *2026 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 2026. Accepted.