

Designing Constrained AI Assistance for Novice Programmers

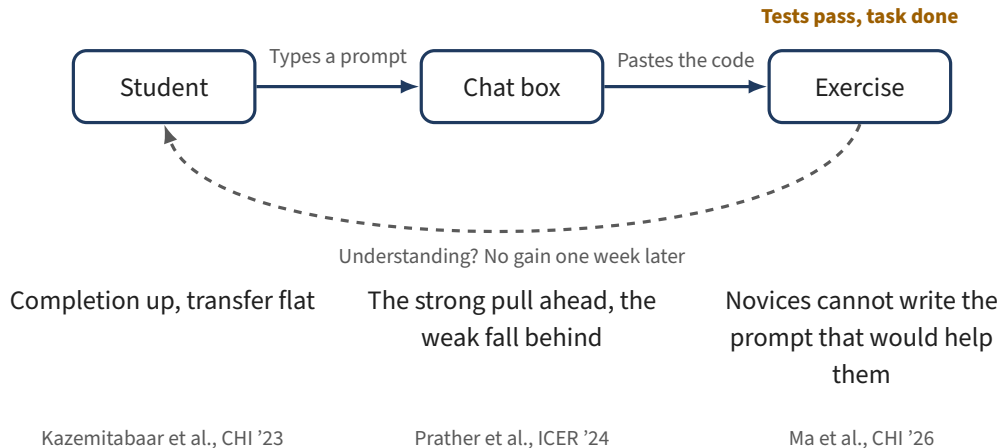
Tensions and Hypotheses from a JupyterLab Hint System

Alexandre De Masi • Chen Wang • Laurent Moccozet

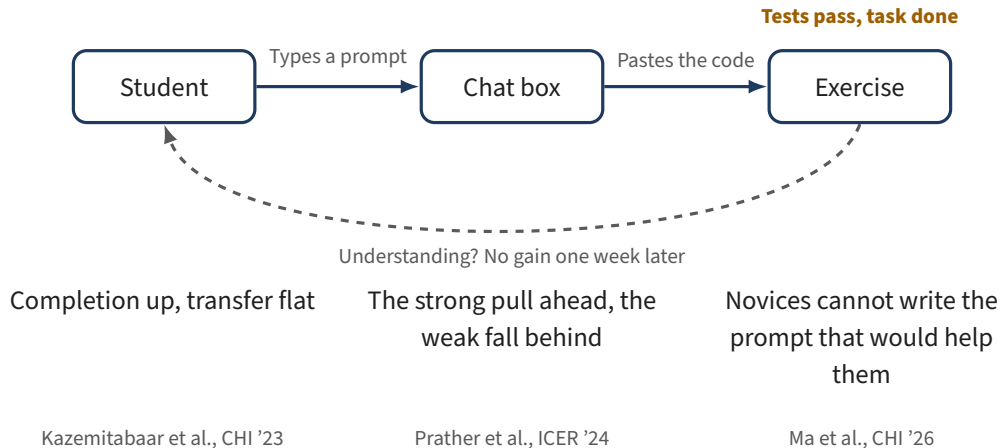
NordiCHI 2026 • Vaasa • 5 October 2026

TECFA, Technologies for Training and Learning
Faculty of Psychology and Educational Sciences, University of Geneva

When the chat box answers, novices lose



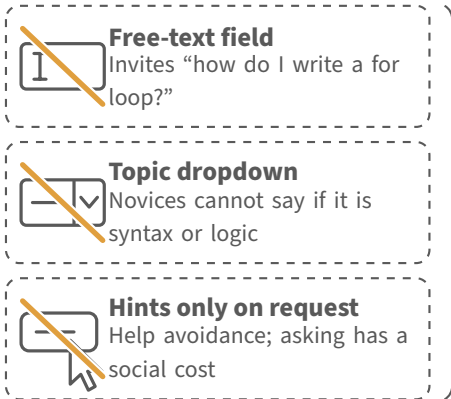
When the chat box answers, novices lose



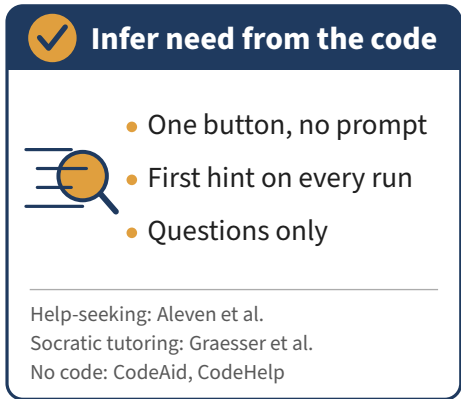
So we constrain the interface.

Design process and rejected alternatives

Rejected alternatives



Adopted design



A constraint set: it does not adapt, fade, or hand control back. The study asks what that costs.

Four constraints

C1 No free-form chat



One button, “Request hint”.

The system reads the code and the errors; the student cannot type a question.

Four constraints

C1 No free-form chat



One button, “Request hint”.
The system reads the code and the errors; the student cannot type a question.

C2 No code



No fix, no snippet, no pseudo-code, and no explanation of what is wrong.

Four constraints

C1 No free-form chat



One button, “Request hint”.
The system reads the code and the errors; the student cannot type a question.

C2 No code



No fix, no snippet, no pseudo-code, and no explanation of what is wrong.

C3 Automatic first hint



Every cell execution triggers one; follow-ups are voluntary.

Four constraints

C1 No free-form chat



One button, “Request hint”.
The system reads the code and the errors; the student cannot type a question.

C2 No code



No fix, no snippet, no pseudo-code, and no explanation of what is wrong.

C3 Automatic first hint



Every cell execution triggers one; follow-ups are voluntary.

C4 Socratic only



Two or three sentences, one question, one issue at a time.

The environment: a Jupyter notebook

The screenshot shows a Jupyter notebook environment. On the left is a file browser showing a directory structure with files like 'level1_beginner.ipynb'. The main area displays a notebook titled 'level1_beginner.ipynb' in 'Markdown' mode. It contains an 'Exercise' section with instructions to find 5 bugs in the provided Scala code. The code is a simple web analytics tracker. On the right, a 'Hint panel' is visible with a 'Request hint' button. Several orange callout boxes are overlaid on the image: 'Run a cell' points to the top of the code cell, 'Task' points to the exercise instructions, 'Starter code' points to the Scala code block, and 'Tests further down' points to the bottom of the code cell. The bottom status bar shows 'Scala | Connecting' and 'Ln 1, Col 1 level1_beginner.ipynb 2'.

File Run Kernel

/ exercises / scala /

Name	Modified
Ys: learningtrace_co...	7mo ago
level1_beginner.ipynb	7mo ago
level2_intermedi...	3mo ago
level3_advanced...	7mo ago

level1_beginner.ipynb

Markdown Notebook Scala

Run a cell

Task

Hint panel

Starter code

Tests further down

Exercise

The code below tracks a website's page views, but it has **5 bugs**. Some compile errors; others compile fine but produce **wrong behavior**.

Your task:

1. Run the code and read any error messages carefully
2. Fix the bugs one by one
3. Run the tests to verify

```
[ ]: // Bug Hunt: Fix the 5 bugs below

// --- Page view counter ---
val pageViews: Int = 0

def recordView(): Unit = {
  pageViews = pageViews + 100
}

// --- Site configuration ---
val siteName = "My Awesome Site"
val maxViews: String = "1"

// --- Analytics ---
var bounceRate = 45
```

Run a code cell to get started.

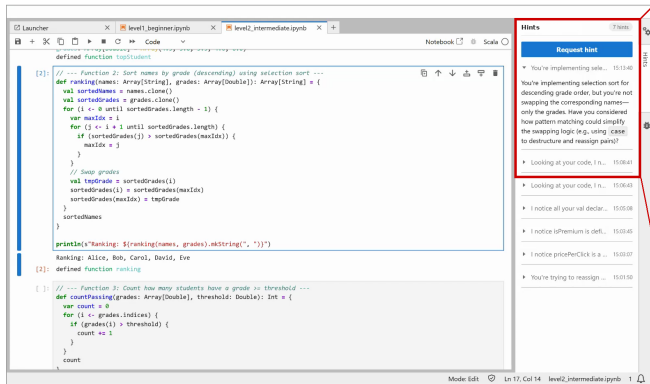
Request hint

JupyterLab Hints

Simple 1 Scala | Connecting Mode: Command Ln 1, Col 1 level1_beginner.ipynb 2

Task 1: immutability bug hunt, Scala.

Our addon: the hint panel



The screenshot shows a code editor with two tabs: 'level1_beginner.ipynb' and 'level2_intermediate.ipynb'. The active tab is 'level2_intermediate.ipynb', which contains Scala code for a selection sort function and a count function. A red box highlights the 'Hints' panel on the right side of the editor. The 'Hints' panel has a 'Request hint' button and a list of hints. The first hint is highlighted and reads: 'You're implementing selection sort for descending grade order, but you're not swapping the corresponding names—only the grades. Have you considered how pattern matching could simplify the swapping logic (e.g., using `case` to destructure and reassign pairs)?'.

```
defined function topStudent

[2]: // --- Function 2: Sort names by grade (descending) using selection sort ---
def ranking(names: Array[String], grades: Array[Double]): Array[String] = {
  val sortedNames = names.clone()
  val sortedGrades = grades.clone()
  for (i <- 0 until sortedGrades.length - 1) {
    var maxIdx = i
    for (j <- i + 1 until sortedGrades.length) {
      if (sortedGrades(j) > sortedGrades(maxIdx)) {
        maxIdx = j
      }
    }
    // Swap grades
    val tmpGrade = sortedGrades(i)
    sortedGrades(i) = sortedGrades(maxIdx)
    sortedGrades(maxIdx) = tmpGrade
  }
  sortedNames
}

println(s"Ranking: ${ranking(names, grades).mkString(", ")}")

Ranking: Alice, Bob, Carol, David, Eve

[2]: defined function ranking

[ ]: // --- Function 3: Count how many students have a grade >= threshold ---
def countPassing(grades: Array[Double], threshold: Double): Int = {
  var count = 0
  for (i <- grades.indices) {
    if (grades(i) >= threshold) {
      count += 1
    }
  }
  count
}
```

Hints

7 hints

Request hint

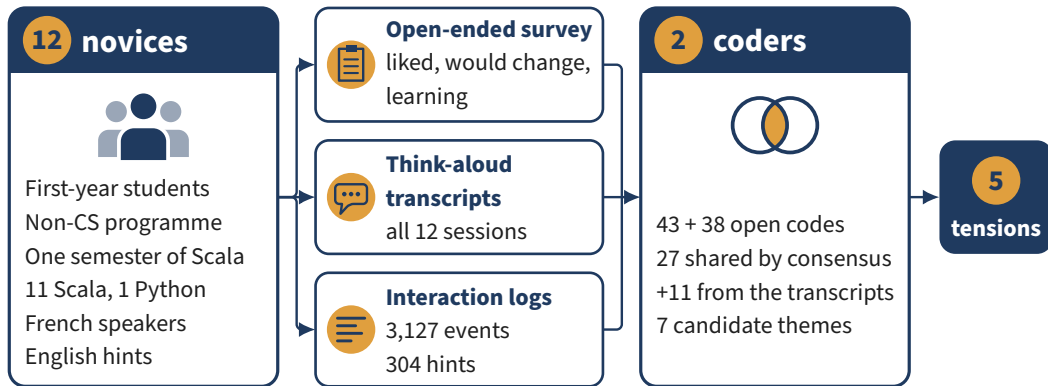
▼ You're implementing sele... 15:13:40

You're implementing selection sort for descending grade order, but you're not swapping the corresponding names—only the grades. Have you considered how pattern matching could simplify the swapping logic (e.g., using `case` to destructure and reassign pairs)?

► Looking at your code, I n... 15:08:41

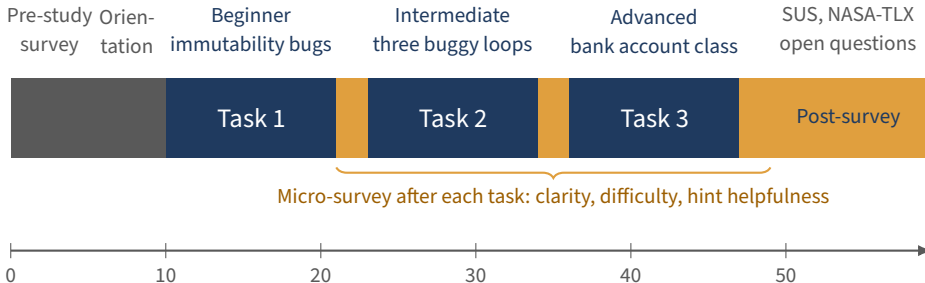
Real session, Task 2. Hints stack as cards; code names are highlighted, code is never written.

Twelve novices, three data streams, five tensions

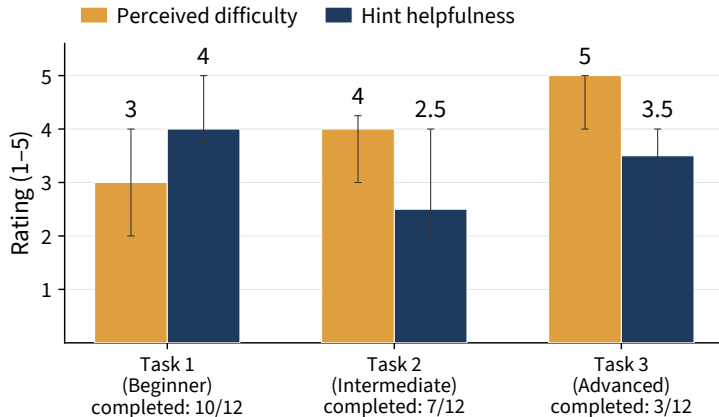


One session, 55 minutes

Think-aloud throughout • screen and audio recorded • every hint and run logged



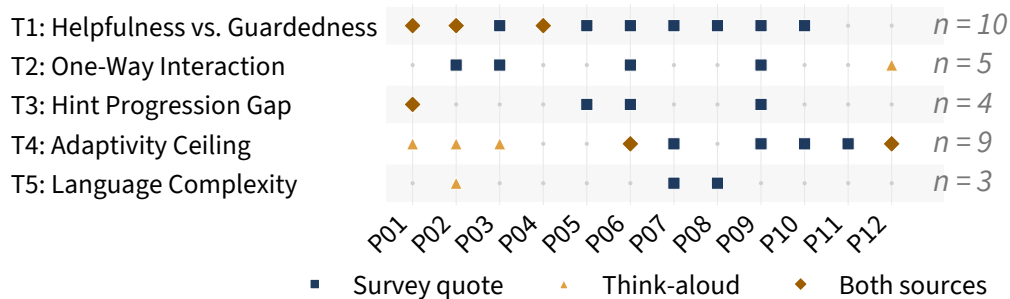
Harder tasks, less help



- Completion 10, 7, then 3 of 12
- Helpfulness falls, then only half recovers
- SUS 72.5: acceptable, not loved

Much of the frustration below comes from a task most could not finish.

Five design tensions



T1 Valued on easy tasks, broke on hard ones

Ten of twelve praised the Socratic approach in principle.

“if I’m completely blocked it should give me maybe one answer to help go forward, whereas if I have a little issue, it should guide me” (P08)

- The breakdown is a **threshold**: it appears where the task passes the student’s competence
- The problem sits in task–learner calibration; the Socratic format itself was endorsed

T2 The system sees the code, not the confusion

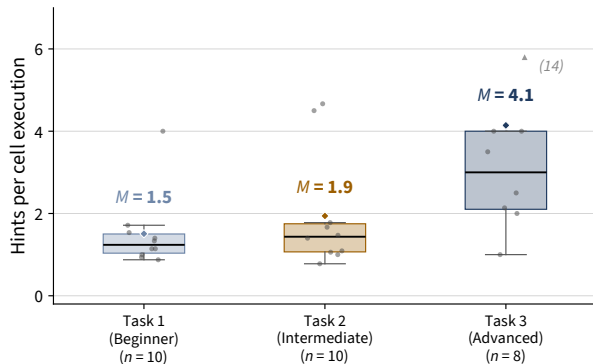
Five of twelve wanted to talk back.

“you cannot communicate with the assistance. It’s just a one-way road.” (P03)

“there is no way to tell it what seems to be blocking us” (P09)

Code and errors are a lossy picture of understanding. A student can be stuck on something that leaves no trace in the cell.

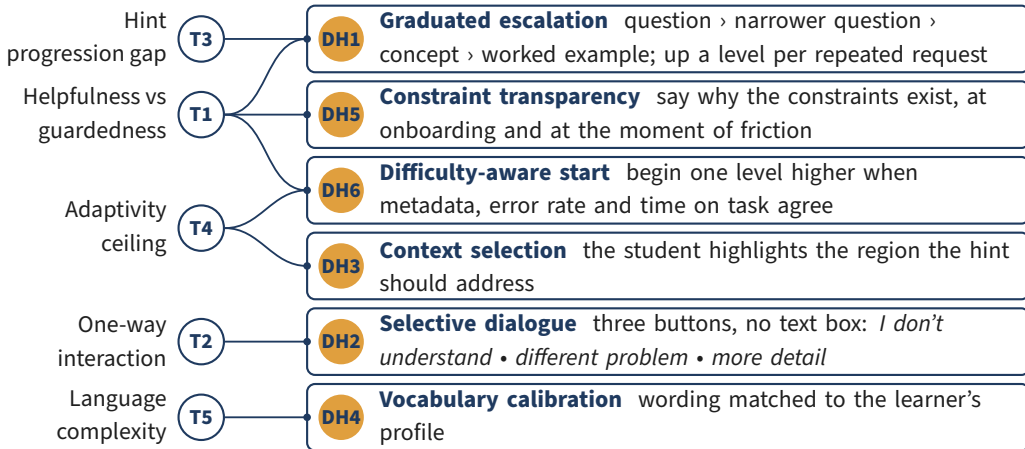
T3 and T4 No new signal, no escalation



- 47 of 203 follow-ups (23%) came within 30 s with no edit; two in three repeated the same issue
- Hints per cell rise with difficulty, and so does the spread between students
- A question cannot bridge a gap the student lacks the words for

“the AI gives us no information about what a function means” (P09)

Six design hypotheses



What we did not foresee

1 A threshold



praised, then abrupt failure

What we did not foresee

1 A threshold



praised, then abrupt failure

2 Measurable traces



23% rapid re-asks

What we did not foresee

1 A threshold



praised, then abrupt failure

2 Measurable traces



23% rapid re-asks

3 Automatic hints were welcome



intrusion expected, none felt

What we did not foresee

1 A threshold



praised, then abrupt failure

2 Measurable traces



23% rapid re-asks

3 Automatic hints were welcome



intrusion expected, none felt

4 Confusion the code can't show



no chat, no way to say so

Constrained should not mean rigid

The constraint level is a design parameter.

DH1: one level more direct per repeated request, if nothing changed

DH6: start one level higher on hard tasks



Keep the box closed, open a narrow channel back (DH2, DH3). Next: measure learning, against open chat and no AI.



Alexandre De Masi

alexandre.demasi@unige.ch



ademasi.ch

Thank you

**Designing Constrained AI Assistance for Novice
Programmers: Tensions and Hypotheses from a
JupyterLab Hint System**

with Chen Wang and Laurent Moccozet

**Beyond the Chatbot • Swiss National Science
Foundation
TECFA, University of Geneva**



The paper

doi.org/10.1145/3829807.3829904

Backup: limitations

Small sample



12 participants, one institution, one 55-minute session

No learning outcomes



Surfaces tensions; a controlled study tests them

No comparison arm



Some tensions may appear with any hint system

One model



Qwen3 Coder Next; hint quality may differ elsewhere

Notebook cells



Out-of-order runs and stale state add their own load

Designers as analysts



Independent coding, disconfirming evidence, verbatim quotes

Backup: the Scala confound



11 of 12 chose Scala, the language of their course; **one used Python**

Harder tasks drew on end-of-semester material, not yet consolidated

Constraint-specific

T1

T2

T3

Would appear in any
language, at other
thresholds

Amplified

T4

By recently taught
constructs

```
grades.clone()
```

Context-dependent

T5

English hints

French speakers

Scala vocabulary

May affect all five tensions

Only one participant used Python; the classification is a working hypothesis.

Backup: T5 language complexity

 Raised by three participants, two of them answering the survey in French

 *“The answers given by the LLM could be easier to understand (use simpler words)”*

P08

Direct-answer system



Hard words are offset by the clarity of the fix

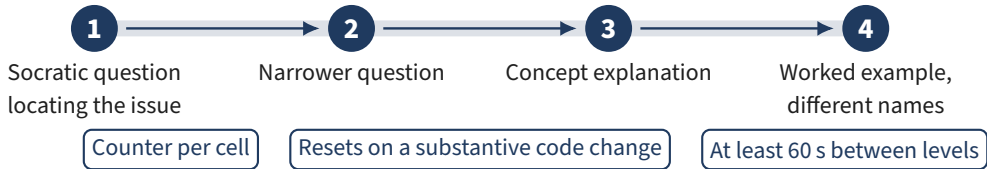
Socratic system



The question *is* the hint: an opaque question ends the interaction

Backup: DH1 and DH6 in detail

DH1 Graduated escalation: four levels



DH6 Difficulty-aware start



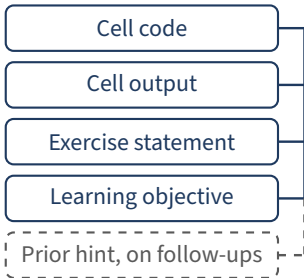
Backup: where the system sits

System	Chat	Code	Trigger	Scaffolding	Student model
CodeAid	Free-form	Pseudo-code	On demand	Explanation	No
CodeHelp	Free-form	None	On demand	Guiding text	No
DBox	Structured	None	On demand	Decomposition	Implicit
Khanmigo	Multi-turn	None	On demand	Socratic	Implicit
GitHub Copilot	Inline	Full code	Automatic	Completion	No
Ours	None	None	Automatic + on demand	Socratic	No

The most restrictive point: no chat, no code, automatic trigger, no student model

Backup: the system prompt, in short

Context assembled



System prompt, key rules

"Questions, never answers."

2-3 sentences, under 50 words.

One issue: the first the student would hit.

No code, no fixes, no "the fix is...".

No explanation of what is wrong.

Follow-up: more specific, still a question.

One flat prompt

{ No chain of thought }

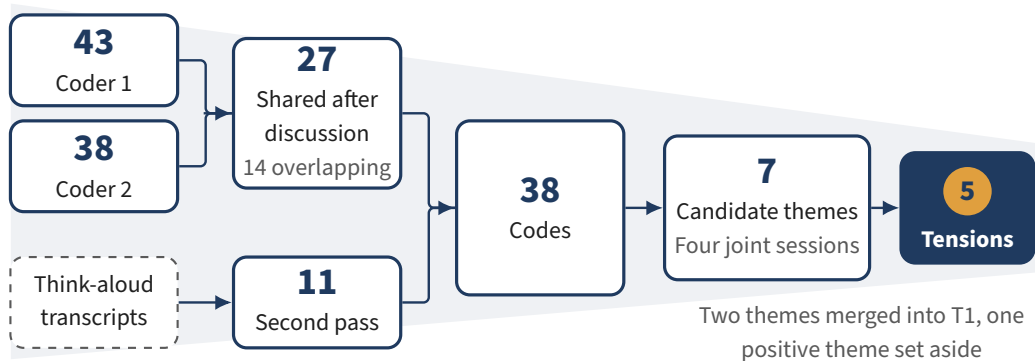
{ No self-critique }

{ No validator }

Qwen3 Coder Next via OpenRouter; Qwen's recommended sampling: temperature 0.7, top-p 0.8, top-k 20.

Backup: analysis

Content analysis, interpretive orientation, thematic-analysis process



No kappa: consensus coding, not independent assignment