Programming:

What it is and how to teach it

Andrew J. Ko, Ph.D.

UNIVERSITY of WASHINGTON

Credit: Northwestern University

I love programming

- I'm guessing you do too!
- I've done 20 years of research on how to make it easier to **do**.
- This has mostly involved **inventing interactive tools** and studying **software engineering**.
- But then I become a co-founder and CTO of a **software startup**…

I quickly learned that tools **amplify** productivity, but they don't **cause** it.

(My startup, AnswerDash, in 2013)

I learned that **skills** cause productivity, and skills must be *taught* and *learned*.

This talk

- I'll review how we are failing to teach these skills, resulting in too few **great programmers**.

- I'll explain how the field of **Computing Education Research** (CER) is trying to solve this.

- I'll present my lab's research on **what programming is** and how to effectively **teach** *programming languages*, *APIs*, and *problem solving*.

100,000 CS majors globally

[NCES 2018, CRA 2017, Loyalka 2019]

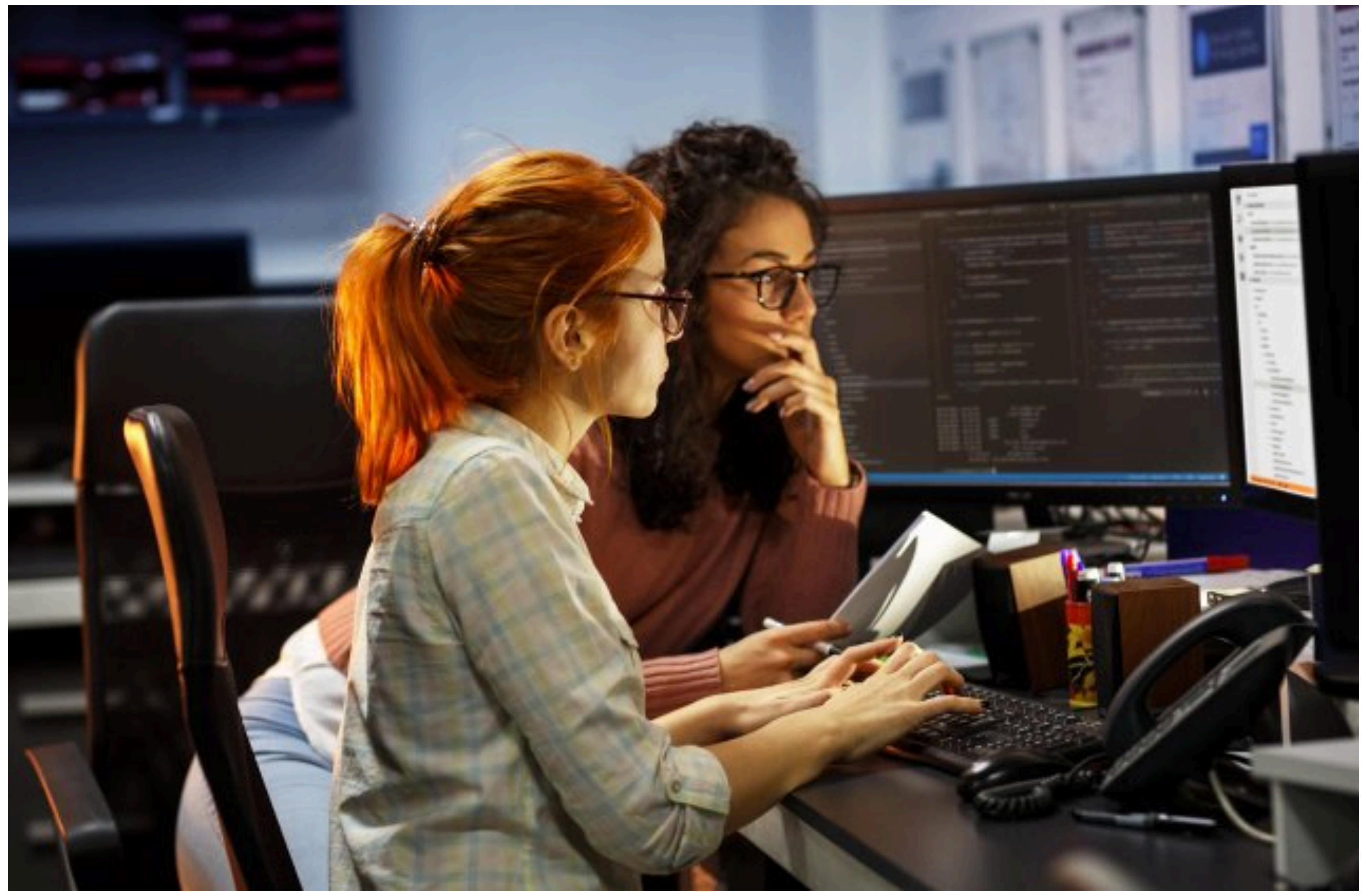
      

25,000 coding bootcamp students

[Course Report 2018]

10 million youth learning
CS in primary + secondary

[Code.org 2019]

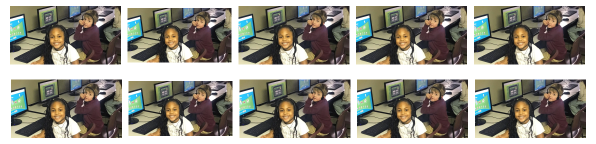


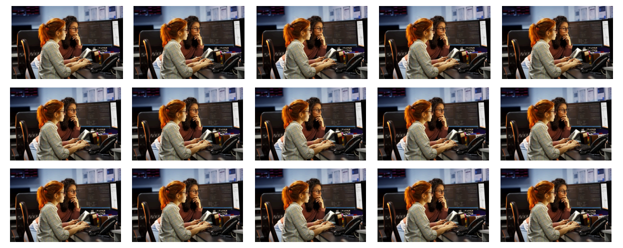
30 million developers learning languages + APIs

[Evans Data Corp, 2018]

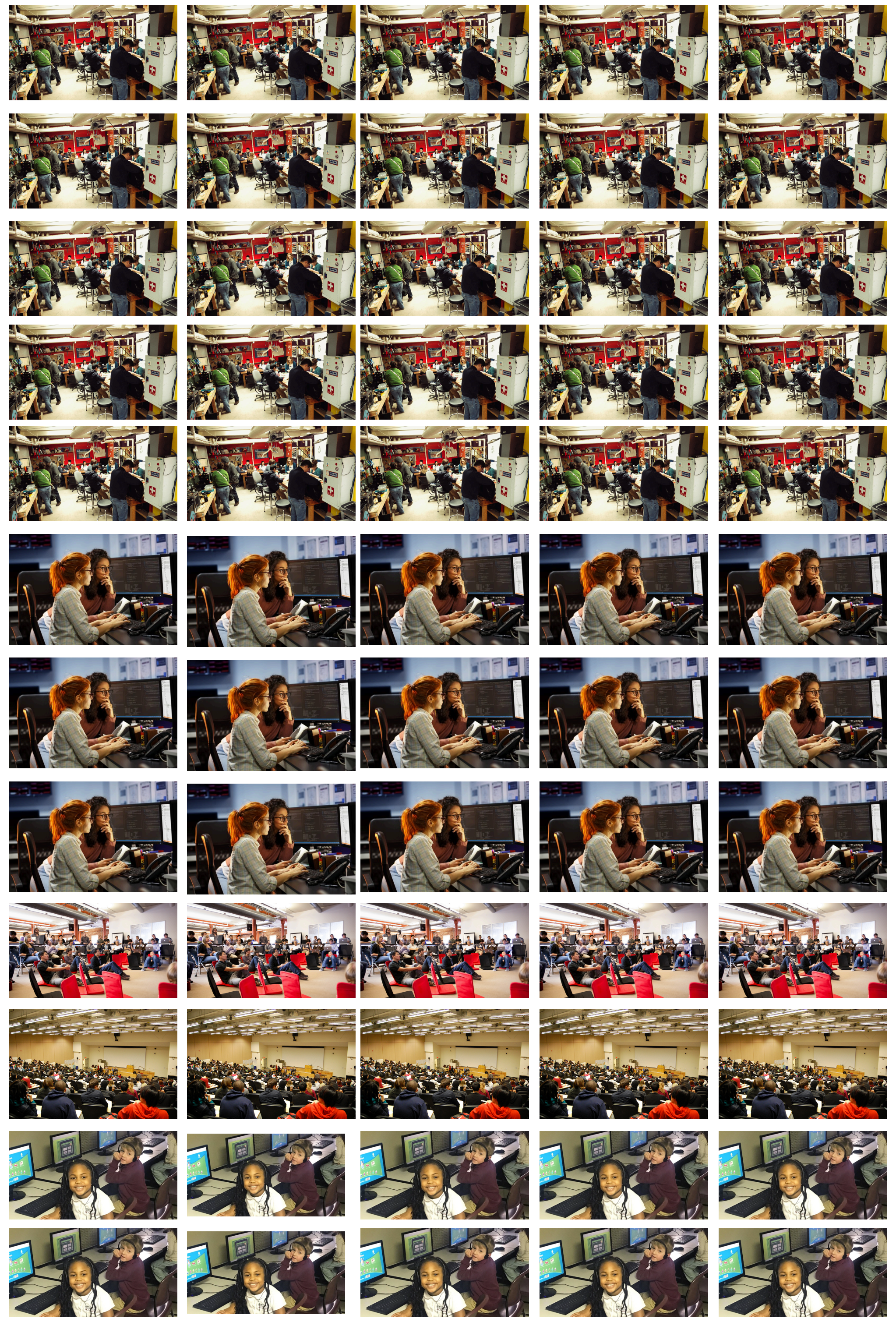
100 million programming to support their work and hobbies

[Scaffidi et al. 2005, Ko et al. 2011]

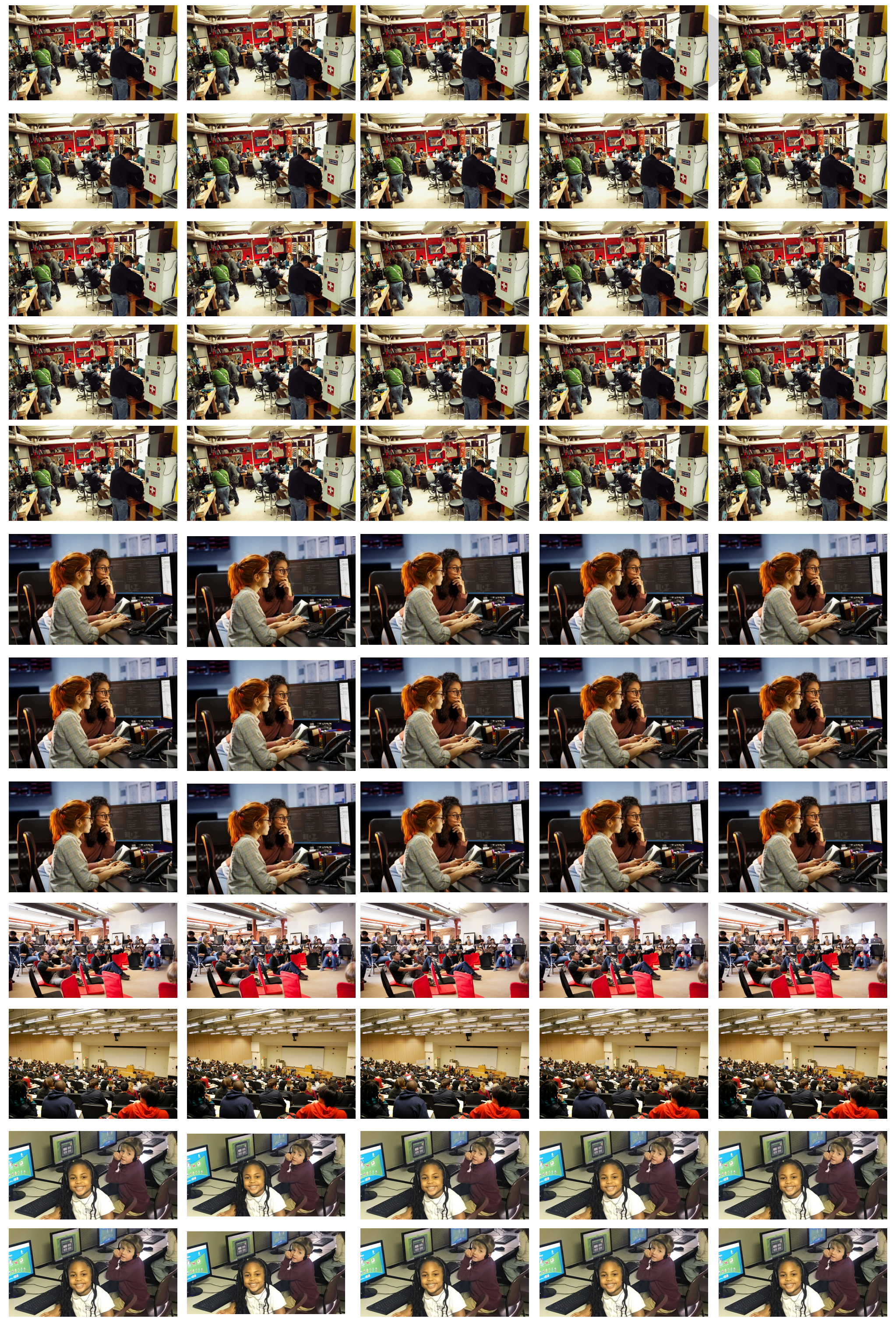




this is a lot of people learning programming!



…but this excludes everyone **afraid** to learn



many quit because teaching is **decontextualized**, thus boring

[Guzdial 2003, Margolis 2003]

many quit because of **racism, sexism, ableism, ageism**

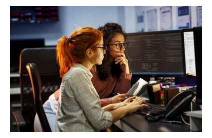
[Margolis 2003, Margolis 2008, Baker 2017, Xia 2001]

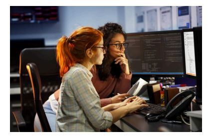
many quit because of **poor teaching**

[Margolis 2003, Kinnunen 2006, Margolis 2008, Patitsas et al. 2016, Kim 2017]

…and *because* of poor teaching, few become **great programmers**.

[Li 2015]





How do we solve these problems, cultivating more great programmers in school and at work?

Computing education research (CER)

An international community of hundreds of outstanding researchers, driving innovation in CS teaching, learning, and educational technology.

ICER 2018, Espoo, Finland

Computing education **practice** vs Computing education **research**

Who: Faculty, teachers, documentation writers, Stack Overflow contributors, developers helping coworkers.

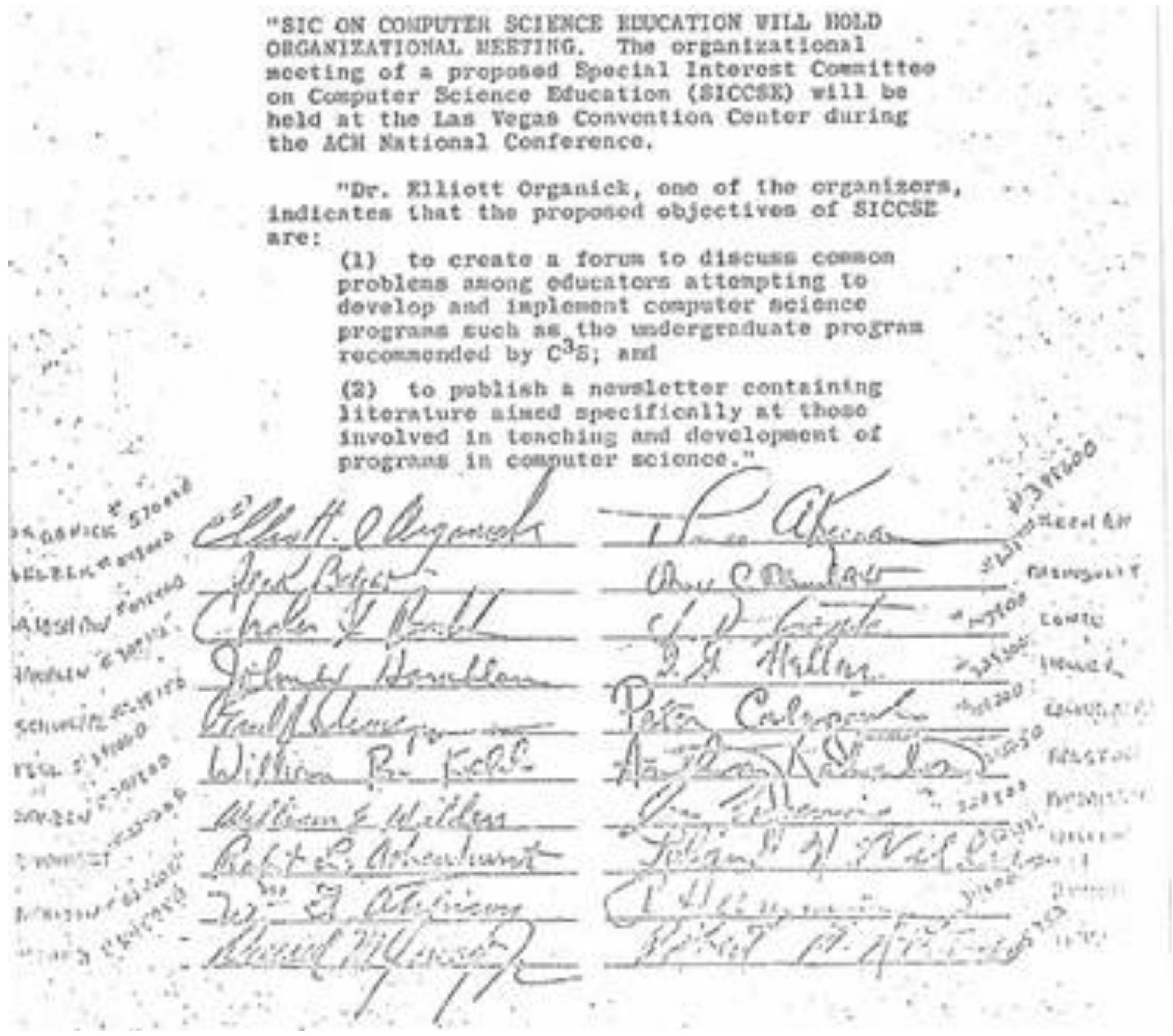
What: teaching classes, developing learning materials, mentoring students, assessing learning, developing academic programs for learning, etc.

Who: Globally, 500+ faculty and doctoral students in Computing and Information Science, Education.

What: The *science* of how people teach, learn, and develop interest in computing; theories, empirical studies, and innovations in teaching.

One of the oldest CS research areas

- ACM SIGCSE was the first SIG in **1968**
- ACM SIGCSE's Technical Symposium was one of the first ACM conferences in **1970**
- Up until about 2000, the CS education community was a **practical** community, mostly writing experience reports about classes they taught and challenges they faced.



"SIC ON COMPUTER SCIENCE EDUCATION WILL HOLD ORGANIZATIONAL MEETING. The organizational meeting of a proposed Special Interest Committee on Computer Science Education (SICCSE) will be held at the Las Vegas Convention Center during the ACM National Conference.

"Dr. Elliott Organick, one of the organizers, indicates that the proposed objectives of SICCSE are:

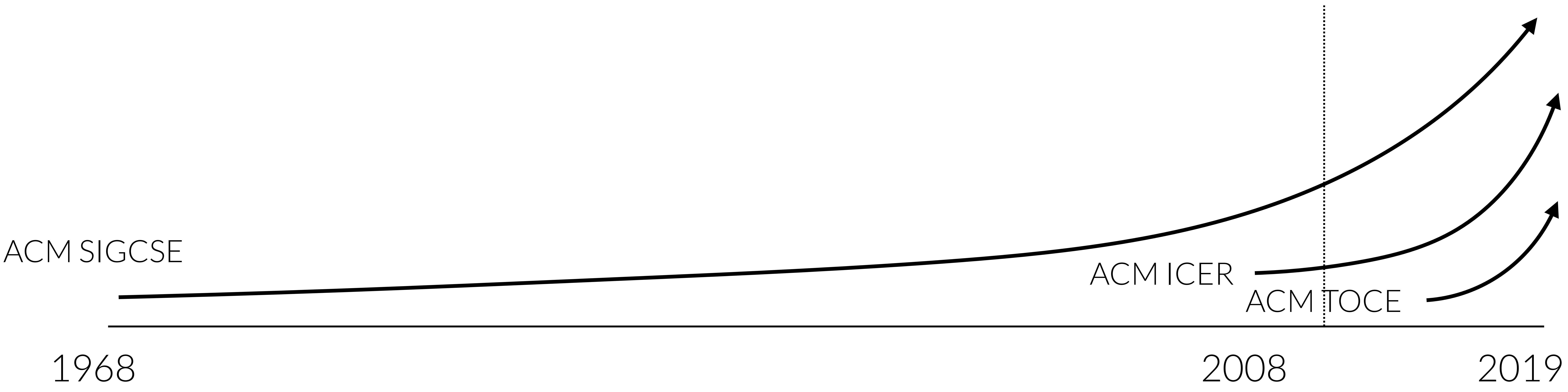
(1) to create a forum to discuss common problems among educators attempting to develop and implement computer science programs such as the undergraduate program recommended by C^3S; and

(2) to publish a newsletter containing literature aimed specifically at those involved in teaching and development of programs in computer science."

The 1968 SIGCSE formation petition

CER publication activity

U.S. National Science Foundation, MacAurthur,
Microsoft, Google begin funding CER



So many questions!

How can we effectively teach PL?

Why do students quit CS?

Does knowledge of one PL transfer to another?

How can we effectively teach APIs?

Why is there so little gender and racial diversity?

How does culture affect CS learning?

How can we motivate people to learn to code?

How can we accurately assess CS knowledge?

How can we teach programming online?

Why are particular concepts hard to learn?

How can we improve access to computing education?

How can we effectively prepare CS teachers?

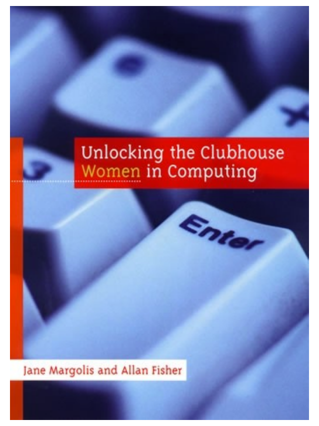
What can be taught about computing to learners of different ages?

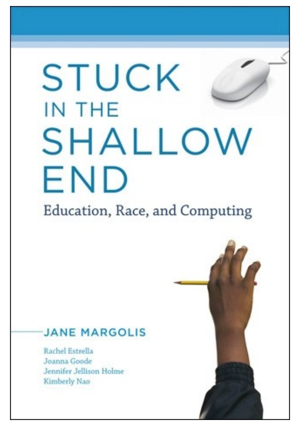
So many contexts!

Primary Secondary College Bootcamps Work

So many discoveries!

How do people develop interest in computing?

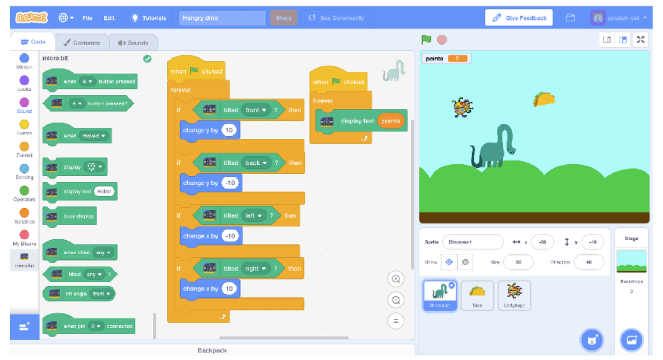




Jane Margolis and many others have shown through a series of studies that interest is shaped *not* by something innate in people, but by **access** to opportunities to *develop* interest.

This is impacting *policy* globally.

How can we lower barriers to learning to code?



Cornell Program Synthesizer (1979) → *Alice* (2000) → *Scratch* (2009) → "Blocks" editors. These have eliminated syntax and type errors as a barrier to learning to code for *hundreds of millions* of learners.

This is impacting *teaching* globally.

My lab


GUEST
ELEVATORS
RESTROOMS
ATM
MINNEAPOLIS, MN
FEBRUARY 27 - MARCH 2

My lab's research

- We study **effective**, **equitable**, and **scalable** ways to teach *hard concepts and skills* in CS.
- Central to discovering ways of teaching hard concepts is to understand the concepts themselves.
- We've recently focused on one big question: **what is programming**?

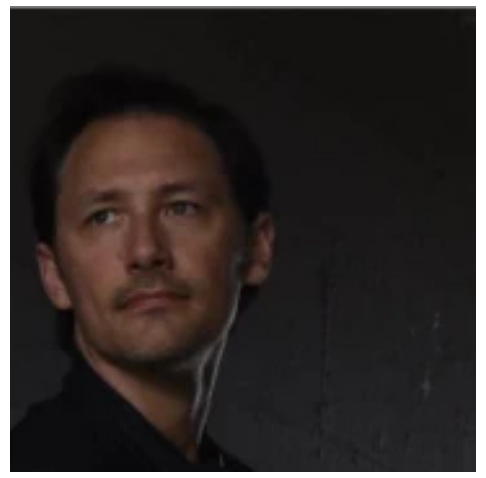
Don’t we know what programming is?

- Isn’t programming *a **logical activity** of designing algorithms + data structures and encoding them in a formal notation?*
- This definition implies that all someone needs to know is **logic** and a **notation**.
- My lab’s discoveries have shown that this definition is **too narrow**, excluding key cognitive and social processes.

+

Programming is a set of *activities*

[Ko & Myers 2015, Loksa et al, 2015, Li et al. 2015, Xie et al. 2019]

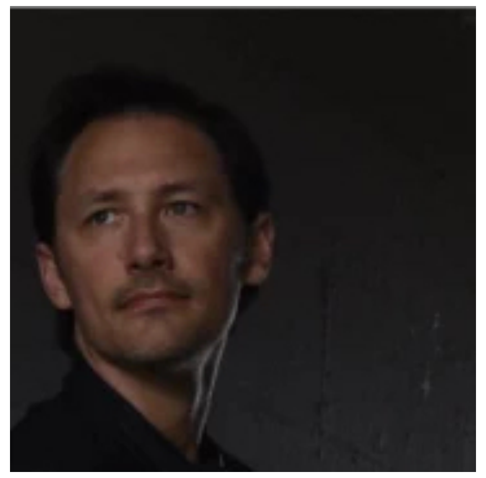


Dastyni Loksa

Problem → **Interpret the problem** → **Search for analogous problems** → **Search for solutions to those problems** → **Verifying that solution solves the problem** → **Implementing the solution** → **Verifying the implementation** → Solution

Programming is a *iterative process*

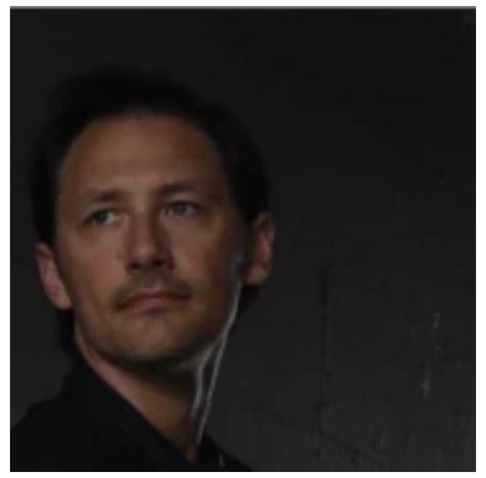
[Ko & Myers 2015, Loksa et al, 2015, Li et al. 2015, Xie et al. 2019]



Dastyni Loksa

Problem → Interpret the problem → Search for analogous problems → Search for solutions to those problems → Verifying that solution solves the problem → Implementing the solution → Verifying the implementation → Solution

Each activity requires *skills*



Dastyni Loksa

Strategies for all of these skills

Translating into formal problem definitions

Describe the problems to people and search engines

How to find existing solutions and interpret their relevance

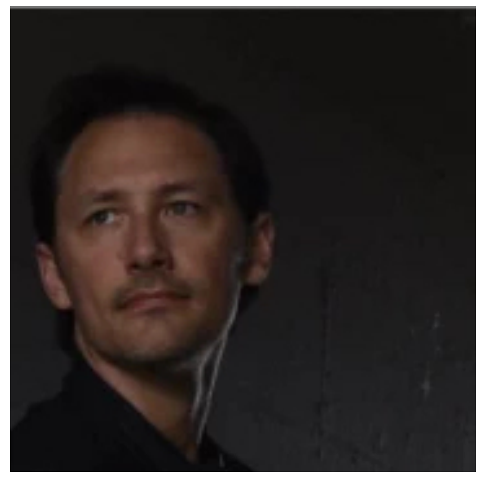
How to analyze algorithms

Programming language syntax and semantics + API knowledge

How to verify and fix with testing, debugging

Problem → Interpret the problem → Search for analogous problems → Search for solutions to those problems → Verifying that solution solves the problem → Implementing the solution → Verifying the implementation → Solution

Productivity requires *process*



Dastyni Loksa

Strategies for all of these skills

Translating into formal problem definitions

Describe the problems to people and search engines

How to find existing solutions and interpret their relevance

How to analyze algorithms

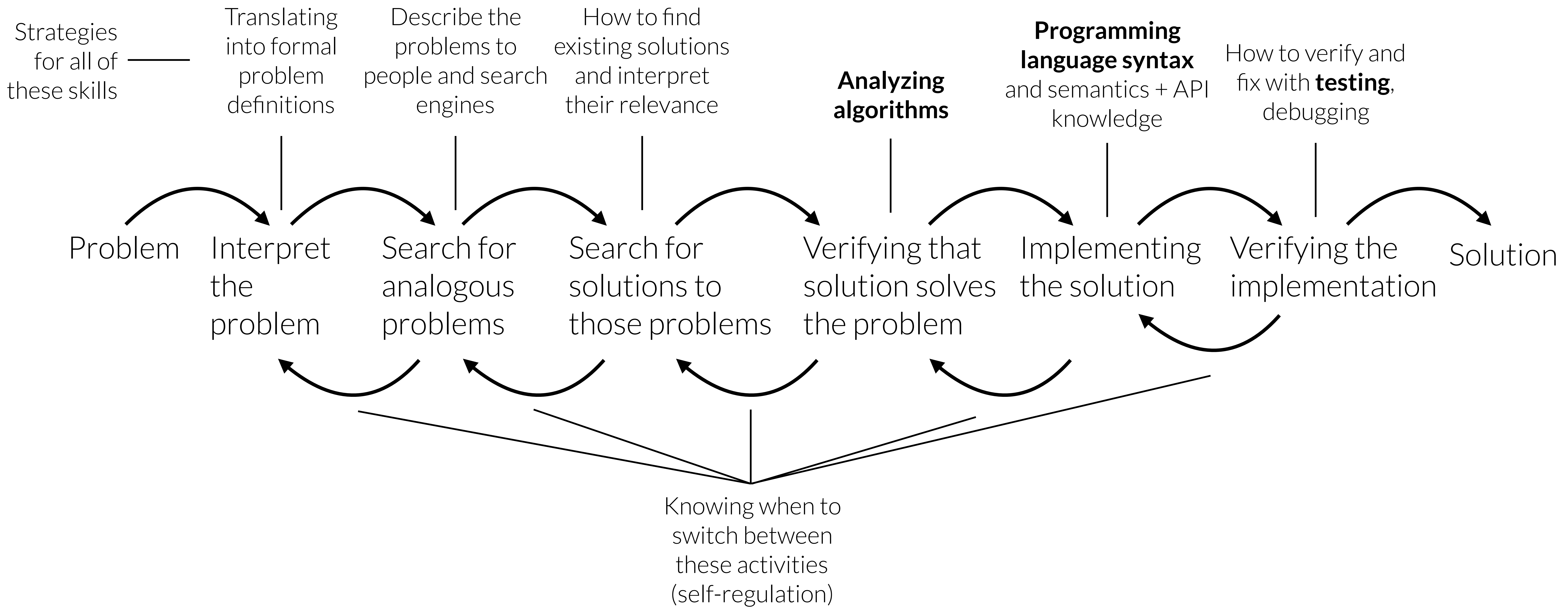
Programming language syntax and semantics + API knowledge

How to verify and fix with testing, debugging

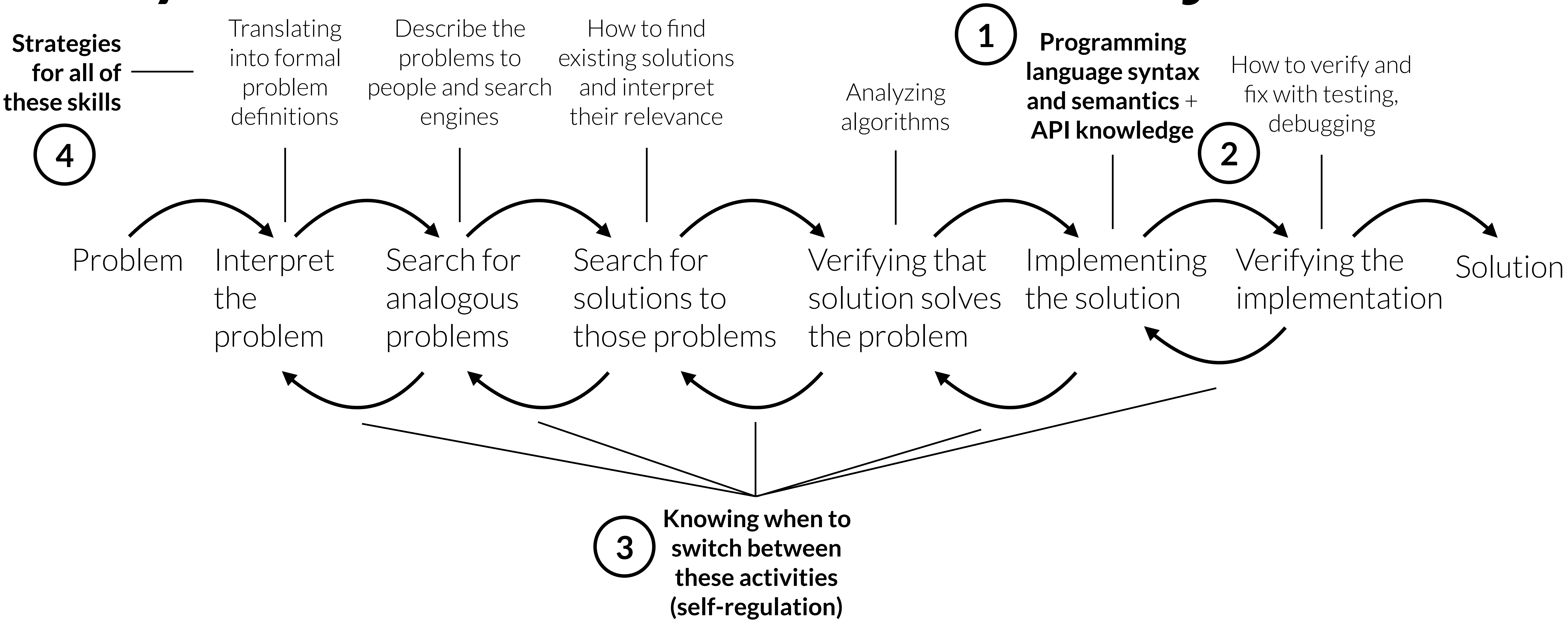
Problem → Interpret the problem → Search for analogous problems → Search for solutions to those problems → Verifying that solution solves the problem → Implementing the solution → Verifying the implementation → Solution

Knowing when to switch activities and try new strategies (self-regulation)

We teach *few* of these skills



My lab has studied four major skills



The rest of this talk

1. **Programming language** knowledge
2. **API** knowledge
3. **Self-regulation** skills
4. **Strategic** knowledge
5. **Implications** of these discoveries for teaching.

Programming language knowledge

- Most approaches to teaching programming languages proceeds as follows:

For all language semantics:

1. Show **syntax examples**
2. Explain **semantics** in natural language
3. Ask learners to **write** programs

A Pedagogical Analysis of Online Coding Tutorials. Ada Kim and Andrew J. Ko (2017). *ACM Technical Symposium on Computer Science Education (SIGCSE).*


This approach overlooks *reading*

- We argue reading is different from writing

1. **Reading semantics.** *How will this conditional execute?*
2. **Writing semantics.** *How do I construct a syntactically valid conditional statement?*

- We also argue that writing depends critically on robust reading skills

A Theory of Instruction for Introductory Programming Skills. Benjamin Xie, Dastyni Loksa, Greg L. Nelson, Matthew J. Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, Andrew J. Ko (2019). *Computer Science Education*.

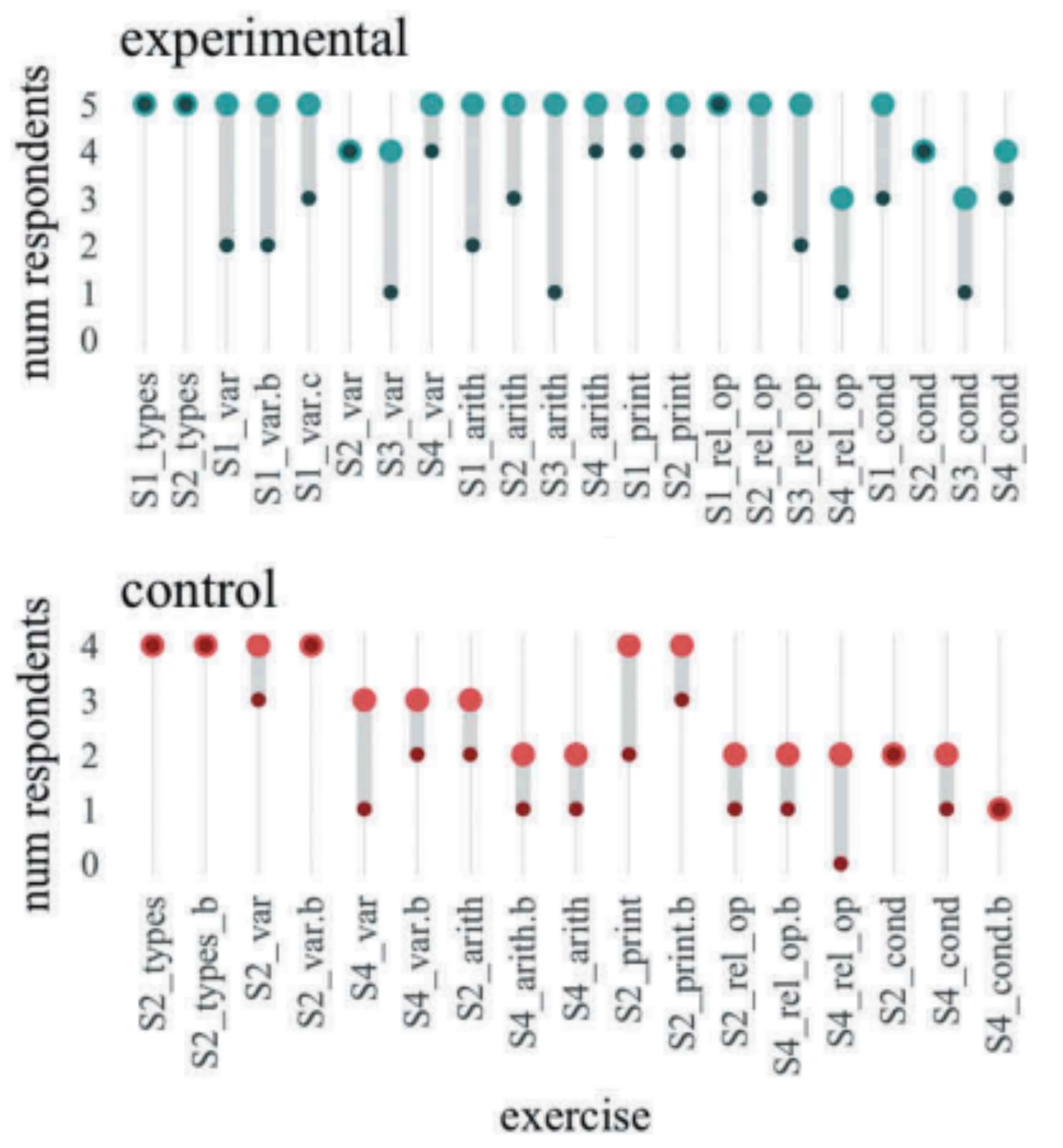
Benji Xie

Greg Nelson

Teaching reading *first* helps

- We designed 2 versions of a 4-hour Python lesson
  - **Control**: 1) show syntax, 2) explain semantics, 3) practice writing semantics
  - **Treatment**: 1) show syntax, 2) explain semantics, 3) **practice reading** semantics, 4) practice writing semantics
- The treatment group:
  - **Completed more practice** in the same amount of time
  - Made **fewer errors**
  - Had a **more robust understanding** of their errors

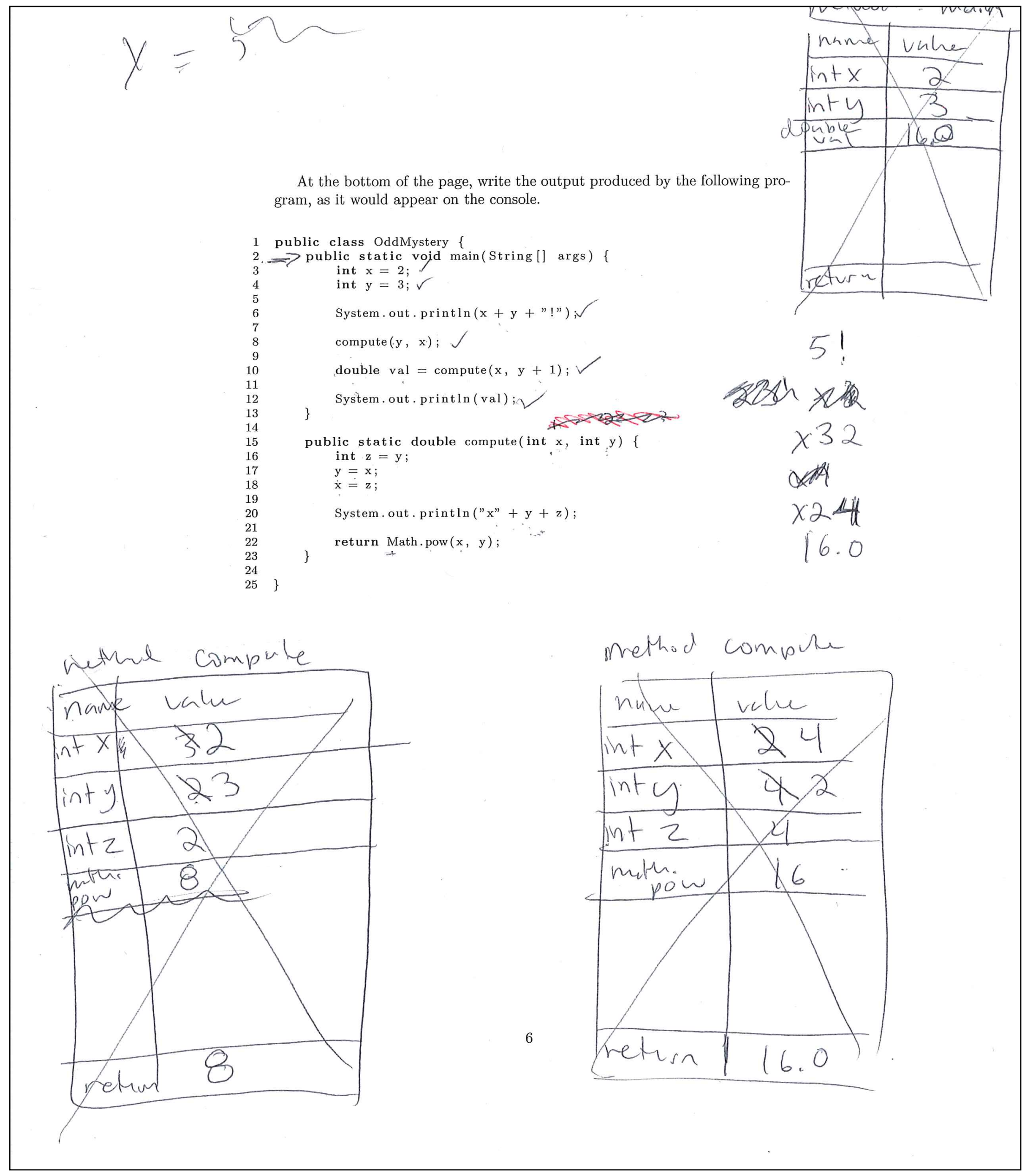
A Theory of Instruction for Introductory Programming Skills. Benjamin Xie, Dastyni Loksa, Greg L. Nelson, Matthew J. Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, Andrew J. Ko (2019). *Computer Science Education*.



Teaching *how* to read helps

- In a lab experiment, we spent 5-minutes teaching a strategy for tracing program execution: *line by line, follow the semantics rules, update a memory table.*

- Students who used the strategy:
  - Scored on average **15% higher** on a post-test
  - Based on think-aloud data, were **more systematic**
  - Scored on average **7% higher** on the course midterm

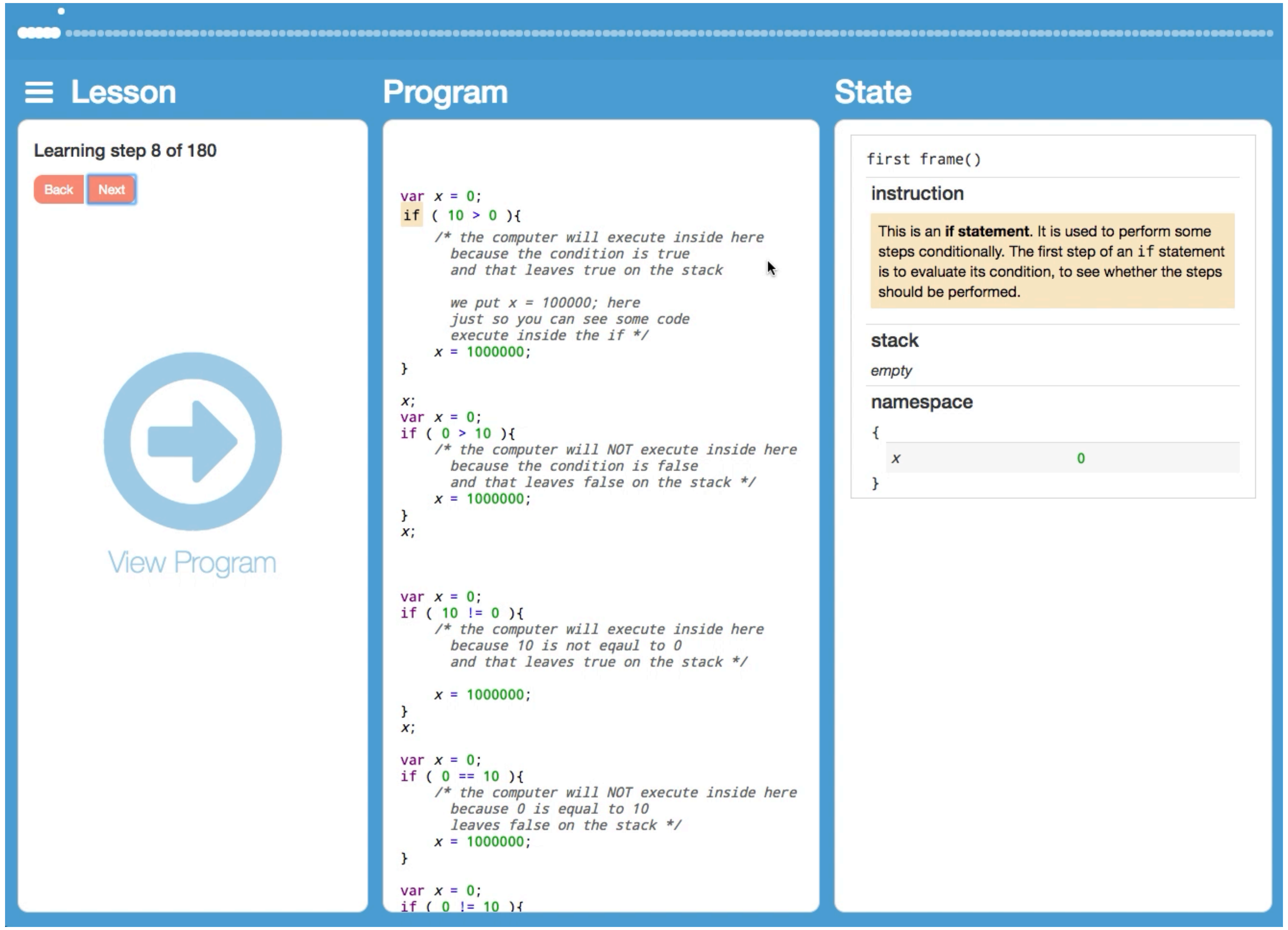


An Explicit Strategy to Scaffold Novice Program Tracing. Benjamin Xie, Greg Nelson, and Andrew J. Ko (2018). ACM Technical Symposium on Computer Science Education (SIGCSE), Research Track.

Visualizing semantics helps

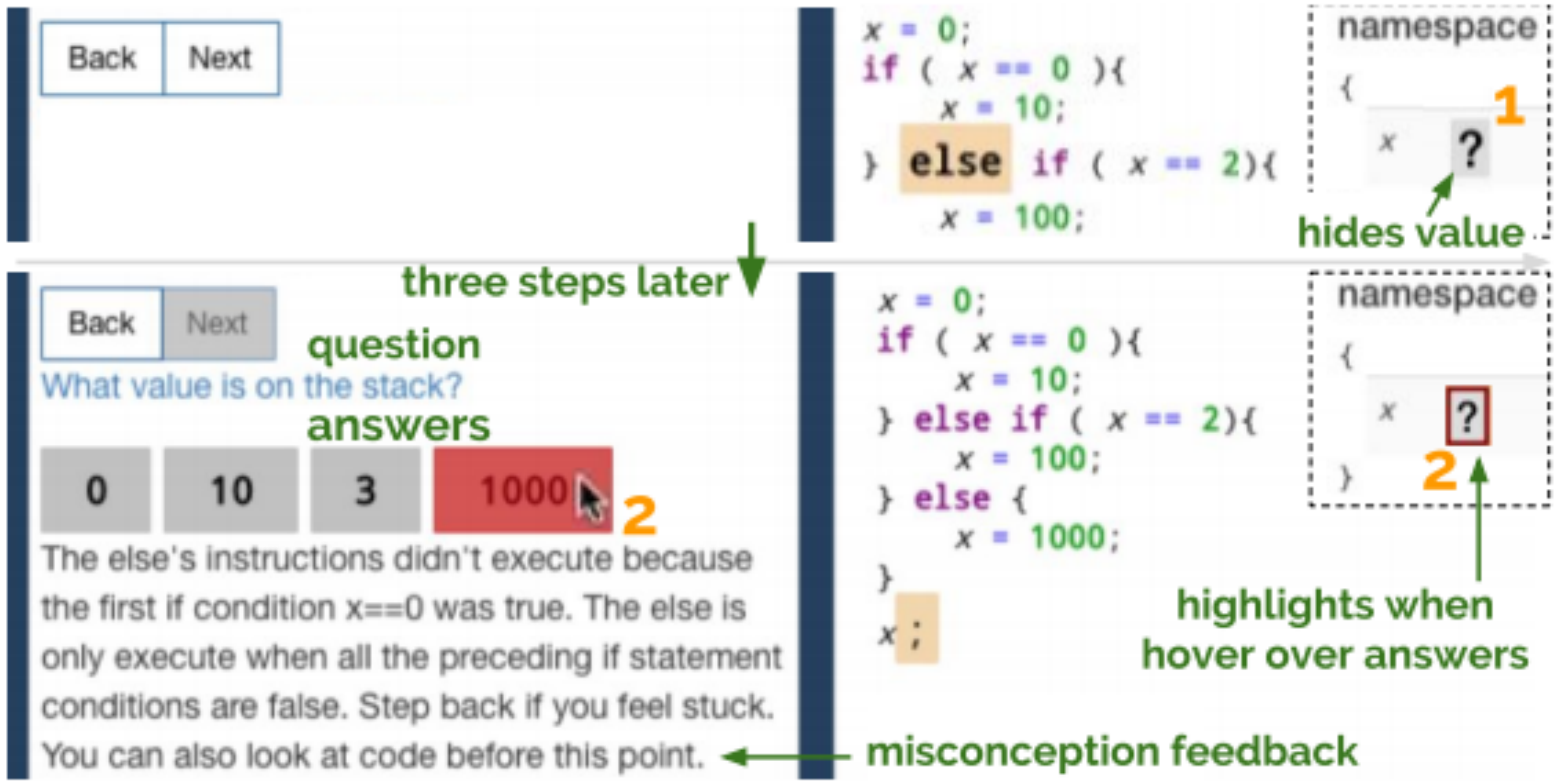
- **PLTutor**: teach JavaScript semantics by visualizing execution one instruction at a time, linking syntax to control and data side effects
  - **60% higher learning** gains than a Codecademy tutorial
  - PLTutor associated with **higher midterm** grades.

Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. Greg Nelson, Benjamin Xie, and Andrew J. Ko (2017). *ACM International Computing Education Research Conference (ICER).*



PLTutor’s hidden complexity

- We had to redesign the **entire** JavaScript language stack to support:
  - Provenance of data values
  - Bi-directional mapping from instructions to tokens
  - Granular execution and reverse-execution
  - Annotated program execution histories



Comprehension First: Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. Greg Nelson, Benjamin Xie, and Andrew J. Ko (2017). *ACM International Computing Education Research Conference (ICER).*

Future work on PL learning

- Many of these ideas are being integrated into code.org's curriculum used by 10 million learners.
- We're building a version of PLTutor that models learner knowledge, **adapting** itself to what a learner knows
- We're building an **ecosystem** of tutors for different programming languages, building upon prior PL knowledge
- We envision a world in which learning a PL is the *easiest* part of learning programming.

The rest of this talk

1. ✓ **Programming language** knowledge | Robust ability to **read** semantics is key to writing
2. **API** knowledge
3. **Self-regulation** skills
4. **Strategic** knowledge
5. **Implications** of these discoveries for teaching.

API knowledge

- Most API learning involves:
  - Reading API documentation
  - Finding and adapting code examples (e.g., StackOverflow)
- Such learning results in **brittle** API knowledge, where weak knowledge of API behavior results in resulting in difficulty modifying, fixing, or correctly using APIs.
- How do we teach **robust** API knowledge?

Six Learning Barriers in End-User Programming Systems. Andrew J. Ko, Brad A. Myers, and Htet Htet Aung (2004). *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).*

Three components of API knowledge

- **Domain concepts**, which come from the world, and how the API models those concepts
  - e.g., typography in graphic design versus typography in LaTeX
- **Parameterized code templates**, which describe how to coordinate API features to achieve a range of related functionality
  - e.g., *a two-tiered bulleted list example in LaTeX*
- **Execution facts**, which describe the runtime behavior and dependencies of API functionality
  - e.g., *knowing how LaTeX chooses the bullet symbol for lists*

A Theory of Robust API Knowledge. Kyle Thayer, Sarah Chasins, and Andrew J. Ko. *In review.*

Kyle Thayer

LaTeX nested bullets model concepts from typography and graphic design such s baselines and whitespace.

- First item
  - First subitem
  - Second subitem
  - Third subitem
- Second item
- Third item

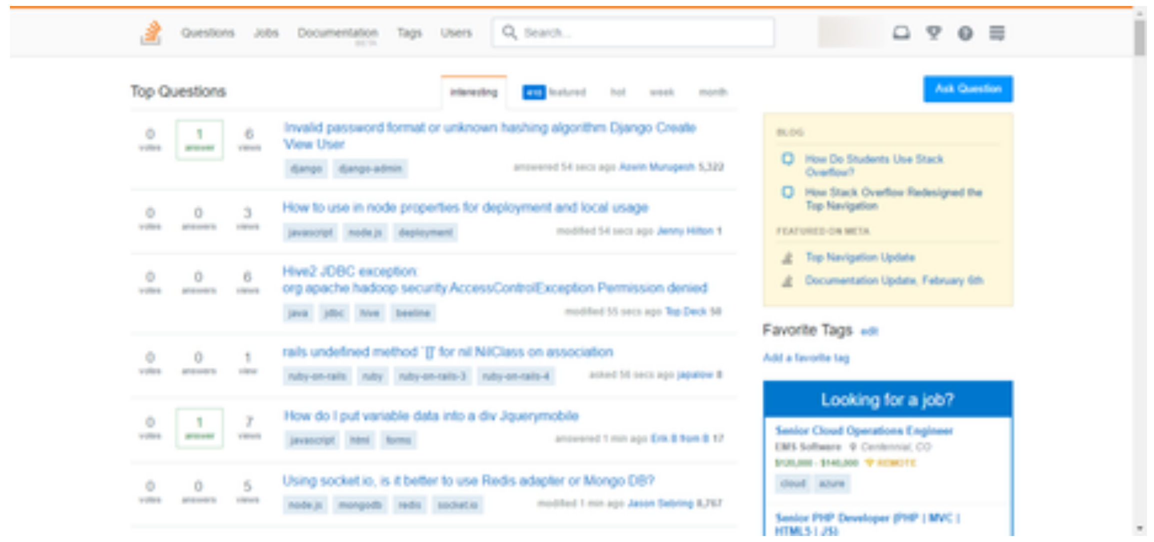
- First item
  - First subitem
  - Second subitem
  - Third subitem
- Second item
- Third item

These account for the content of most StackOverflow answers

- We selected 10 APIs, then the 10 Q&A pairs with the most votes on StackOverflow
- **90% of answers** were composed of explanations of domain concepts, parameterized templates, and execution facts.
  - The remaining 10% were comparisons of alternatives, clarifications, and thank yous.
- The majority of replies were **requests** for one of these three types of information.

A Theory of Robust API Knowledge. Kyle Thayer, Sarah Chasins, and Andrew J. Ko. *In review.*

StackOverflow content is predominantly concepts, templates, and facts.



Explicitly teaching this content helps

- Between-subjects experiment of 4 APIs providing **one** of concepts/templates/facts, **all** three, or **none**.
  - Learners **requested** these three types of knowledge when they were not available
  - For most tasks, the more of these three the learner had, the more **correct** and **complete** their solution.
  - Success depended highly on learners' ability to 1) **find** the instruction and 2) **comprehend** it.

A Theory of Robust API Knowledge. Kyle Thayer, Sarah Chasins, and Andrew J. Ko. *In review.*

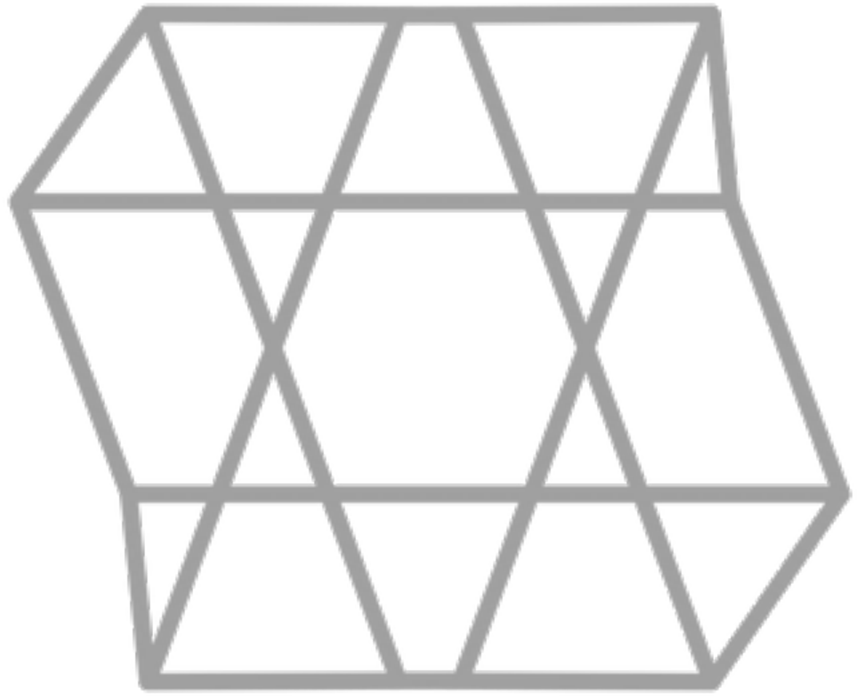
Examples of content we provided in the study.

Future work on API learning

- We're building tools for **automatically extracting templates and facts**, so learning materials can quickly adapt to API evolution

- We're building tools for **automatically generating API tutorials** to optimize discovery and learning of API knowledge

- We envision a world in which robustly learning an API is about careful reasoning, not copy and paste.



The rest of this talk

✓ 1. **Programming language** knowledge

Robust ability to **read** semantics is key to writing

✓ 2. **API** knowledge

Robust knowledge of **concepts**, **templates**, and **facts** is key to correct API use

3. **Self-regulation** skills
4. **Strategic** knowledge
5. **Implications** of these discoveries for teaching.

Self-regulation skills

- *Self-regulation* is the ability to **monitor** one's comprehension, processes, and decisions
- Programming requires self-regulation to make decisions about when to switch **activities**, when to seek new **resources**, when to try a new **strategy**
- Strong self-regulation skills correlate with **fewer defects**, **higher productivity**, **better learning**
- But how do we **teach** it?

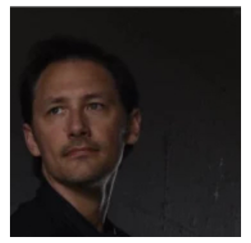
The Role of Self-Regulation in Programming Problem Solving Process and Success
Dastyni Loksa and Andrew J. Ko (2016)
ACM International Computing Education Research Conference (ICER).

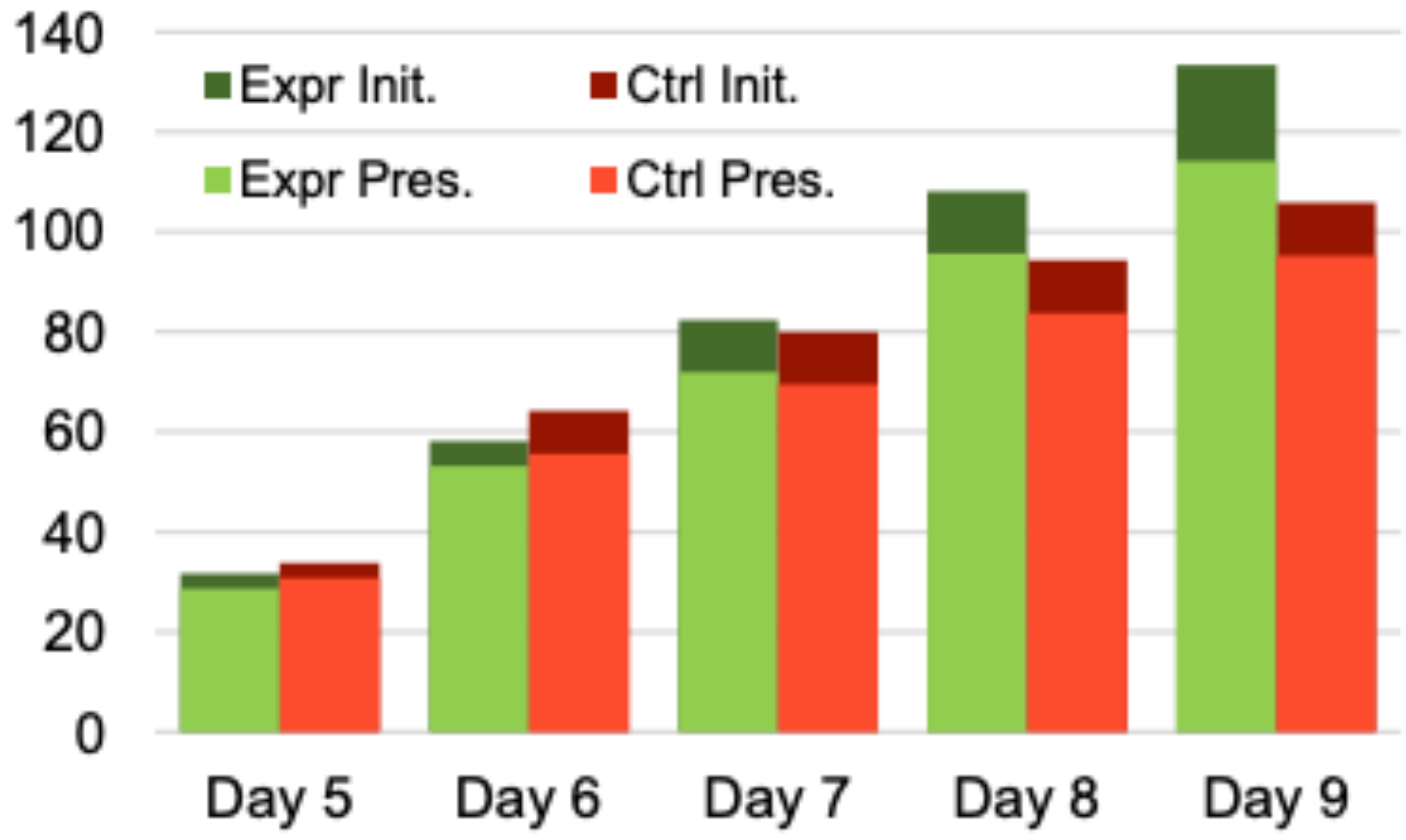
Dastyni Loksa

Self-regulation *prompting* helps

- We ran a classroom experiment with two groups of 40 secondary novice programming students.
- When students asked for help:
  - **Control**. Teachers provided help.
  - **Treatment**. Teachers asked 1) *what are you doing?* 2) *why are you doing it?* 3) *is it helping?* 4) then provided help.
- This increased **productivity**, **independence**, **programming self-efficacy**.

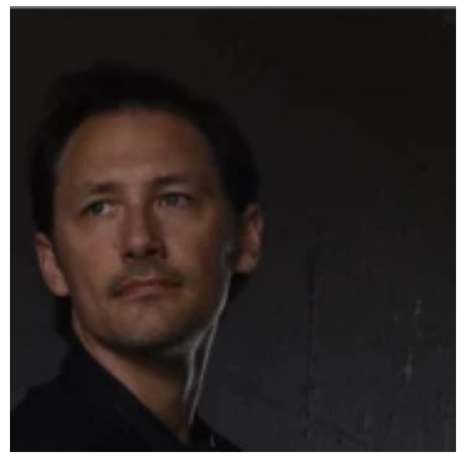
Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance
Dastyni Loksa, Andrew J. Ko, William Jernigan, Alannah Oleson, Chris Mendez, Margaret M. Burnett(2016)
ACM Conference on Human Factors in Computing Systems (CHI)





Modeling self-regulation helps

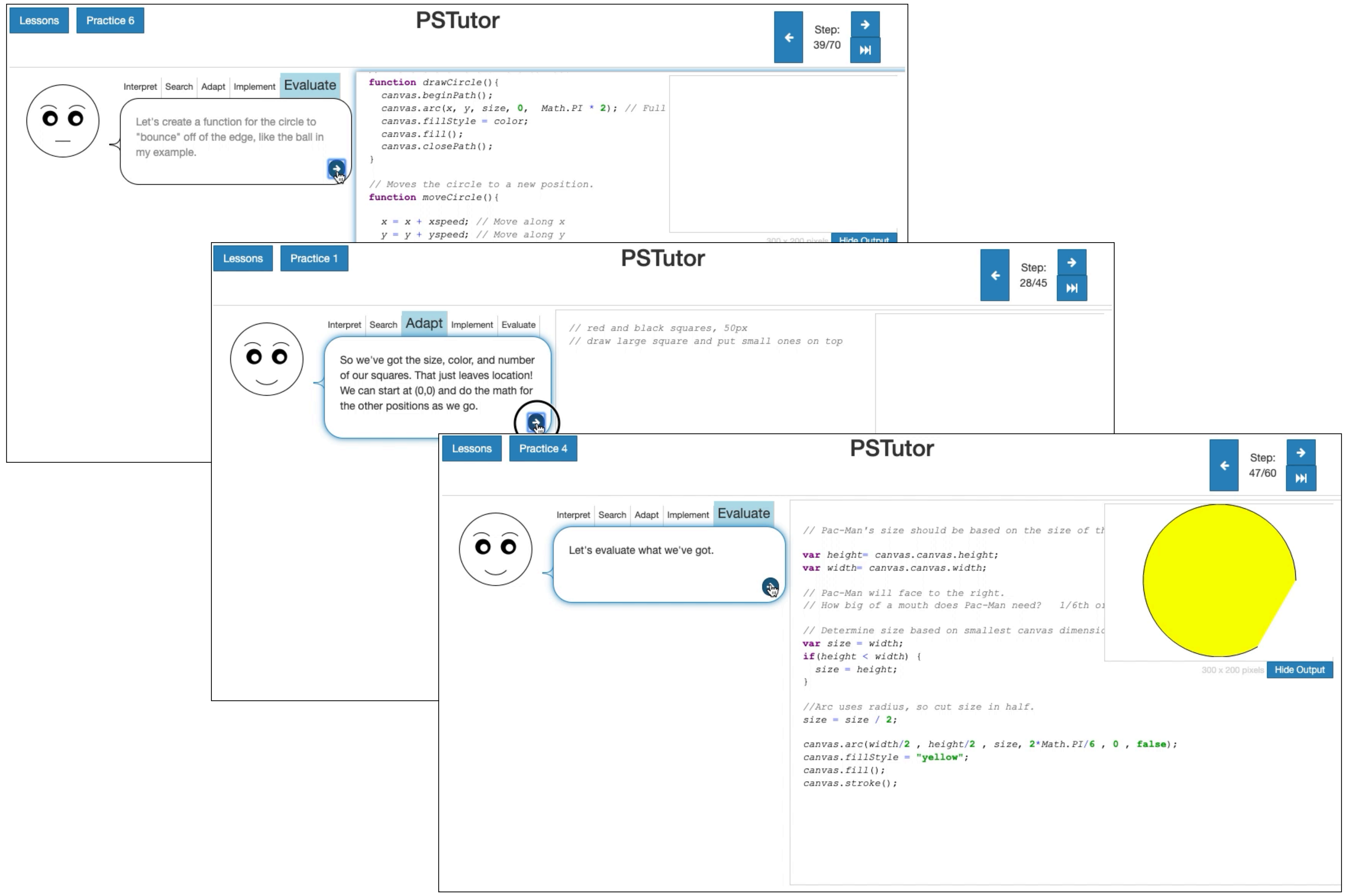
- **PSTutor**: teach self-regulation by showing examples of an expert self-regulating their programming.
- A classroom experiment showed that providing this tutor before a programming project
    - Increased self-regulation activity
    - Increase the difficulty of problems students independently chose.



Modeling Programming Problem Solving Through Interactive Worked Examples
Dastyni Loksa and Andrew J. Ko (2017)
Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU).

PSTutor's hidden complexity

- We invented an entire platform for authoring **instructional programming sessions** to support
  - Character-level revision histories
  - Real-time visualization of programming actions such as testing, debugging
  - Self-regulation annotations on every action in a script
  - Authoring tools for creating examples



Modeling Programming Problem Solving Through Interactive Worked Examples. Dastyni Loksa and Andrew J. Ko (2017). Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU).

Future work on self-regulation

- Many of these ideas are being integrated into code.org's curriculum, used by 10 million learners
- We're exploring new ways of measuring and teaching self-regulation skills **at scale**
- We're exploring the many challenges to preparing teachers to model self-regulation and author **PSTutor** worked examples
- We envision a world in which *every* learner has strong self-regulation skills

The rest of this talk

1. **Programming language** knowledge ✓ — Robust ability to **read** semantics is key to writing
2. **API** knowledge ✓ — Robust knowledge of **concepts**, **templates**, and **facts** is key to correct API use
3. **Self-regulation** skills ✓ — **Modeling** self-regulation skills helps develop them, improving independence
4. **Strategic** knowledge
5. **Implications** of these discoveries for teaching.

Strategic knowledge

- Strong self-regulation skills are useless if a learner has **poor strategies** for solving programming problems.
  - Knowing you're struggling to debug something doesn't help if you don't have a better debugging strategy
- How can we help people learn effective strategies for **all** of the programming problems they might encounter?

Six Learning Barriers in End-User Programming Systems. Andrew J. Ko, Brad A. Myers, and Htet Htet Aung (2004). *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).*

Explicit programming strategies

- **Roboto** is a notation for *explicitly represent* expert strategies for solving problems

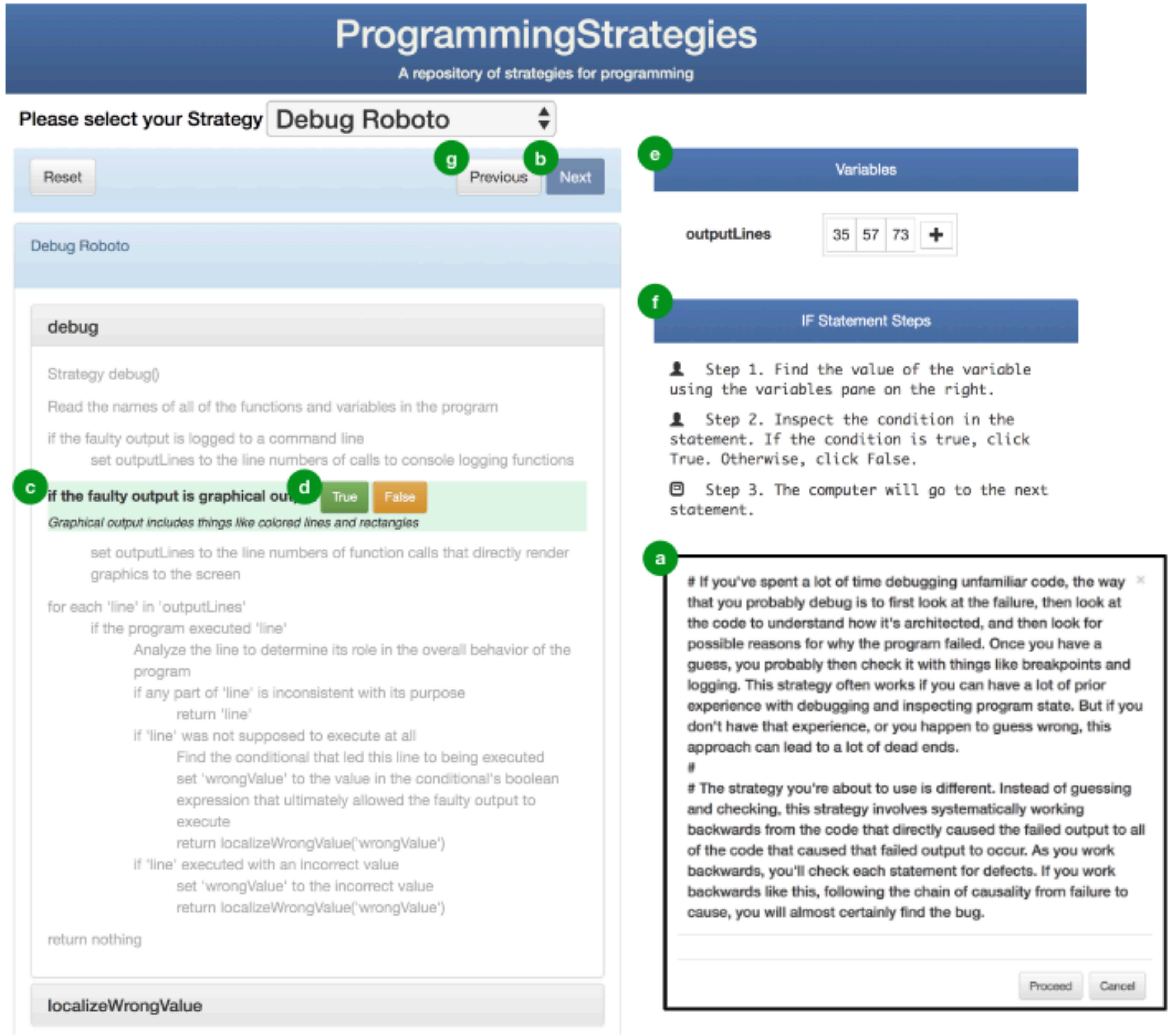
STRATEGY **renameVariable** (name)
  SET **codeLines** TO all lines of source code that contain ’**name**’
  FOR EACH ’**line**’ IN ‘**codeLines’**
    IF the ‘**line**’ contains a valid reference to the variable
        Rename the reference
  SET **docLines** TO all lines of documentation that contain the name ’name’
  FOR EACH ’**line**’ IN ‘**docLines’**
    IF ‘**line**’ contains a reference to the name
      Rename the name

Explicit Programming Strategies. Thomas LaToza, Andrew J. Ko, Miryam Arab, et al.. In review.

Thomas LaToza

Scaffolded strategy execution

- The **developer** makes judgements, gathers information, takes action.
- The **tracker** ensures the developer follows the steps and helps them store information they gather.
- The tracker behaves like a debugger, but with reverse execution and fix-and-continue state editing.



Strategies make *experts* and *novices* more effective

- An experiment with 28 developers working on two tasks: test-driven development (TDD) and debugging
  - **Control**. Chose strategies *independently*.
  - **Treatment**. *Required* to use the TDD and debugging strategy we provided.
- Developers of all expertise using explicit strategies were **more successful** at TDD and debugging
- **Novices** using strategies > **experts** who didn't

Task	Param	Diff	P-value
Design-Implementation	Expertise	87.0	0.3021
	Guided	82.5	0.2325
Design-Tests	Expertise	72.0	0.1036
	Guided	48.0	0.0076*
Debug	Expertise	92.5	0.4779
	Guided	39.5	0.0008*

Strategies make *novices* more effective

- In a classroom study of 20 novice adolescents, we taught a design and debugging strategy
- Learners who used it were **more productive** and **more independent**
- However, many learners struggled to use it because of **weak self-regulation skills**

```
# If you need help finding the problem, ask for help.
Find what your program is doing that you do not want it to do
# Write the line number inside of the program
# and separate with commas.
SET 'possibleCauses' to any lines of the program that
    might be responsible for causing that incorrect 'behavior'
FOR EACH 'cause' IN 'possibleCauses'
    Navigate to 'cause'
    # Ask for help if you need guidance on how.
    Look at the code to verify if it causes the incorrect behavior
    IF 'cause' is the cause of the problem
        # If you need help finding the problem, ask for help.
        Find a way to stop 'cause' from happening
        # Ask for help if you need guidance on how.
        Change the program to stop the incorrect behavior
        # Ask for help if you need guidance on how.
        Mark the task as finished
        RETURN nothing
IF you did not find the cause
    Ask for help finding other possible causes
    Restart the strategy
RETURN nothing
```

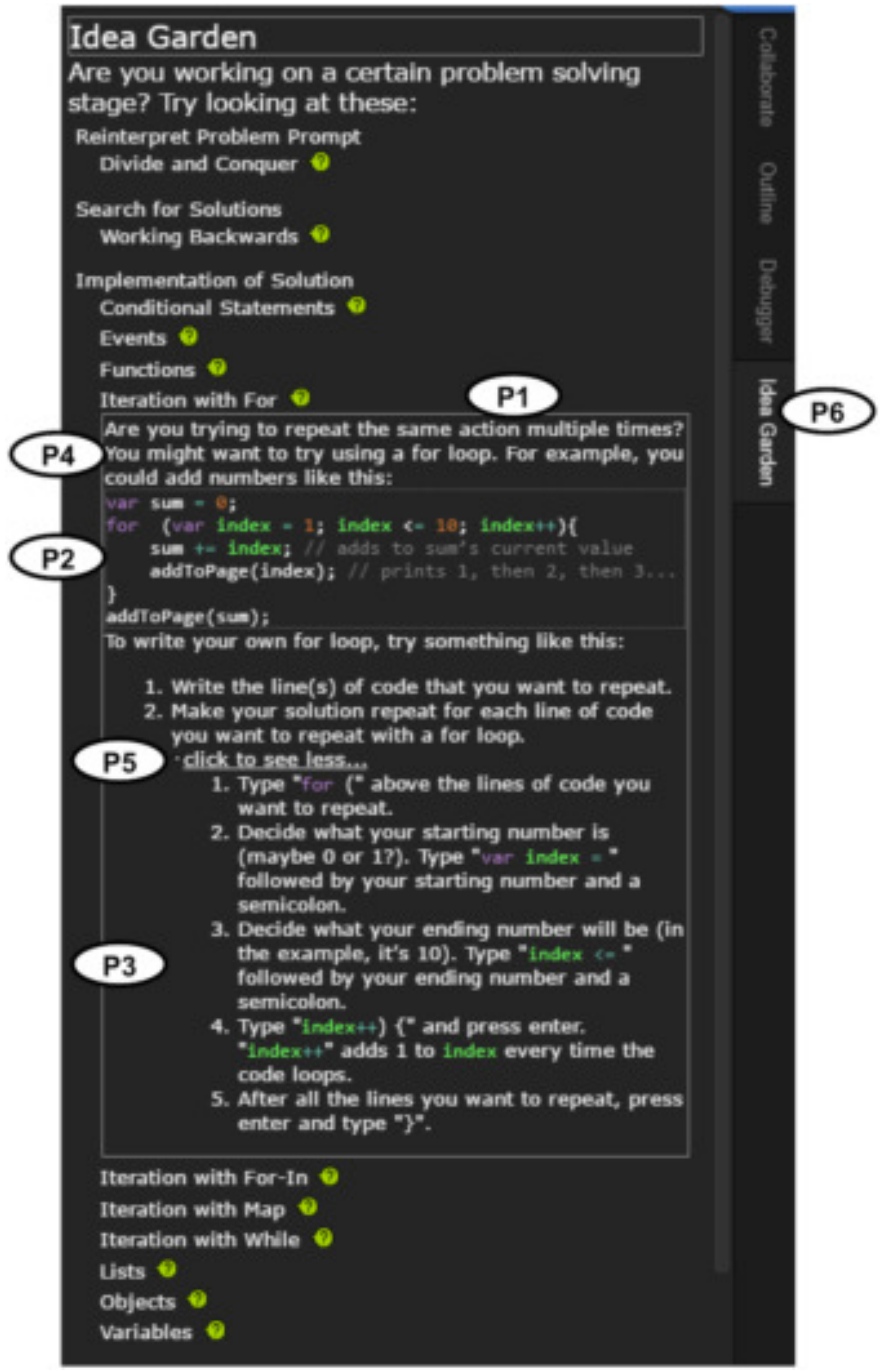
"They're like a formula for when you get stuck.

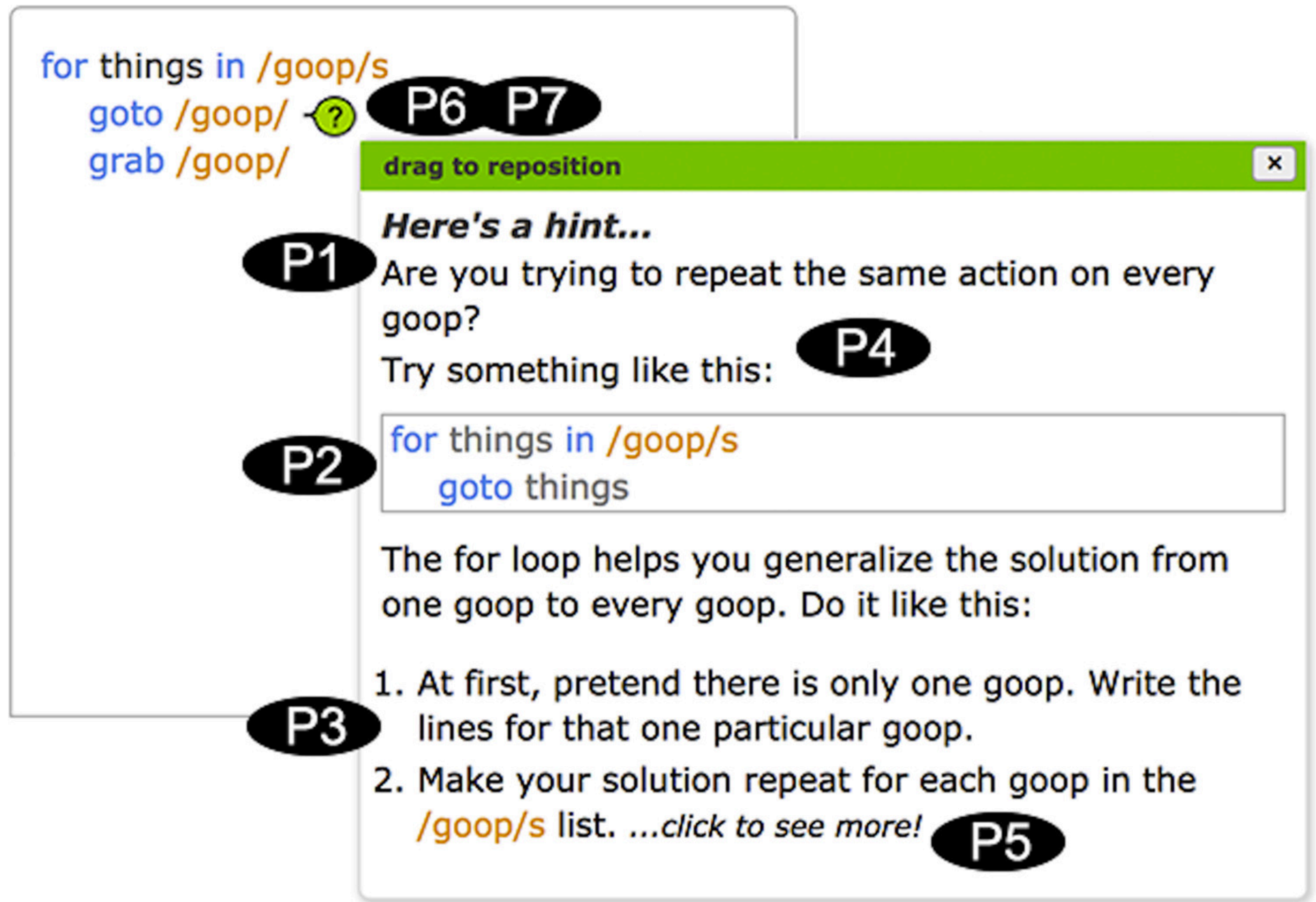
"It forces us to actually look at our code instead of adding random stuff."

Teaching Explicit Programming Strategies to Adolescents
Andrew J. Ko, Thomas LaToza, et al (2019)
ACM Technical Symposium on Computer Science Education (SIGCSE), Research Track.


Embedded strategies make everyone more effective

- **Idea Garden**: embeds hints about how to approach a problem into an IDE
- A series of studies show **improved productivity**, **independence**.





General Principles for a Generalized Idea Garden. Will Jernigan et al. Journal of Visual Languages and Computing (JVLC),.

Future work on strategies

- We're partnering with code.org to write debugging strategies for secondary education.
- We're exploring barriers to **authoring strategies** and barriers to **learning strategies**.
- We envision a world in which there are **strategies for every problem** a programmer might encounter, and a StackOverflow-like site for finding and learning them.

The rest of this talk

✓1. **Programming language** knowledge — Robust ability to **read** semantics is key to writing

✓2. **API** knowledge — Robust knowledge of **concepts**, **templates**, and **facts** is key to correct API use

✓3. **Self-regulation** skills — **Modeling** self-regulation skills helps develop them, improving independence

✓4. **Strategic** knowledge — **Step-by-step representations of strategies** improve effectiveness when used.

5. **Implications** of these discoveries for teaching.

1. Programming is more than logic

- It requires **planning**, **self-awareness**, and dozens of **sub-skills**
- All *require* logic, but they also require systematic behavior and continuous learning
- By ignoring these skills, we ensure that most who *try* to learn programming will fail

2. Teach self-regulation

- **Poor self-regulation** = **poor programming**
- If learners aren't aware of their process, their comprehension, and their decisions, they can't improve them
- Show learners how to think about their thinking by **showing them your thinking** (or use our PSTutor when when release it)

3. Teach strategic knowledge

- **Programming skill** = hundreds of different strategies for solving hundreds of different problems
- Teach these strategies by **writing them down** and having learners **practice** them.
    - No different than any other field of engineering, where there are entire handbooks that describe how to solve every known class of problems.

4. Teach how to read code

- Without a robust ability to **read program semantics**, learners will fail
- Teach learners **reading strategies** and give learners extensive **practice and feedback** (or use PLTutor when we release it)
- Do this *before* you ask them to **write** programs

5. Teach robust API knowledge

- It's easy to imagine that **Stack Overflow** and documentation is everything a learner needs.
- It's not: most answers are **missing** key conceptual and semantic knowledge, and missing key information about the design space in which a code example sits.
- Provide **explicit instruction** on API concepts, templates, and execution to ensure correct API use.

Is there really time for all this?

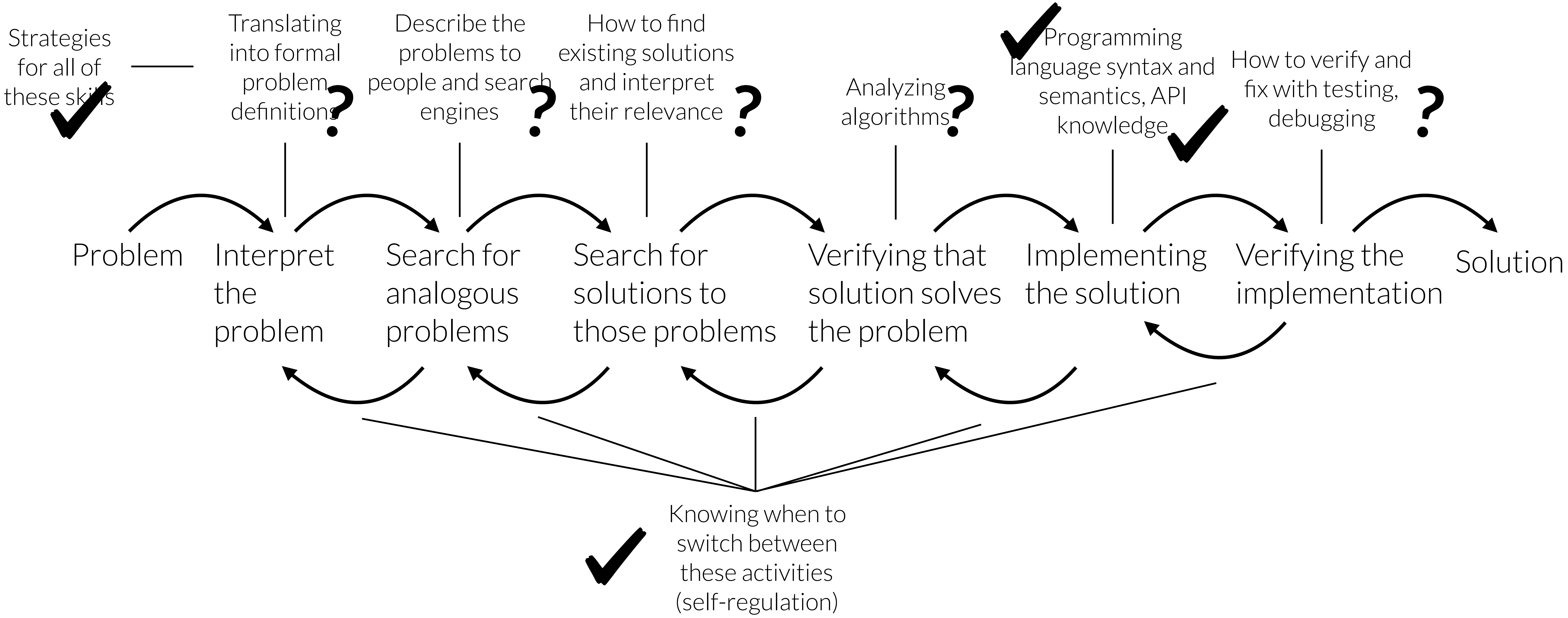
- Teachers get to choose one of two paths:

1. Cover "**all the material**" but produce low-skill programmers, OR

2. Develop **robust foundational skills**, and the ability to independently learn new skills, producing high-skill continuously-learning programmers

- I argue the world prefers #2.

Many open questions about programming...



Many *new* questions about programming...

h

How to find existing solutions and interpret their relevance

Search for solutions to those problems

Analyzing algorithms

Verifying that solution solves the problem

Programming language syntax and semantics, API knowledge

Implementing the solution

How to verify and fix with testing, debugging

Verifying the implementation

Solution

Knowing when to switch between these activities (self-regulation)

- Data programming skills?
- Machine learning skills?
- Software design skills?
- Algorithm design skills?
- Data structure design skills?

Alannah Oleson

Yim Register

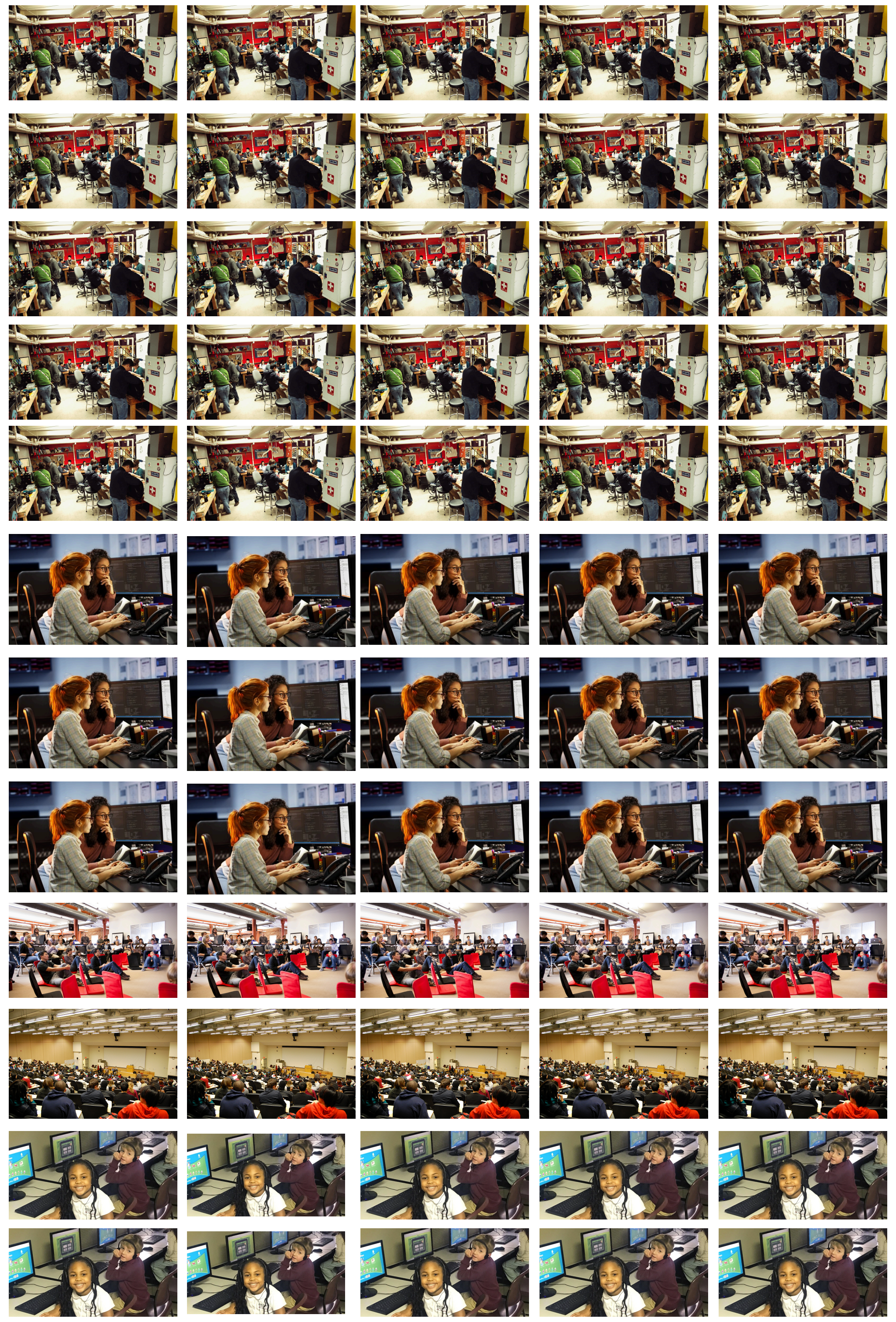
What and how do we teach in different contexts?

Primary Secondary College Bootcamps Work

How do we make our learning contexts **inclusive**, **equitable**, and **diverse?**

If we don't, few will want to learn these skills.

Computing education research is working on all of these problems, helping **everyone** succeed.



Questions?

- Programming is more than logic and notation, it's *planning*, *self-awareness*, *strategy*, *robust PL and API knowledge*, and many other skills.
- Explicit instruction of all of these can improve learning, productivity, confidence, and independence.
- Learners of all kinds—primary, secondary, post-secondary, professional, and hobbyist—need help.
- *Computing Education Research* (CER) is the field solving these problems.

This work was supported by the National Science Foundation, Microsoft, Google, Adobe, and the University of Washington.

Learn about **CER**:
http://faculty.uw.edu/ajko/cer

Read my **blog**:
https://medium.com/bits-and-behavior

Meet my **students**: