Microsoft

TEALS
Best practices in computer science education

Do Now

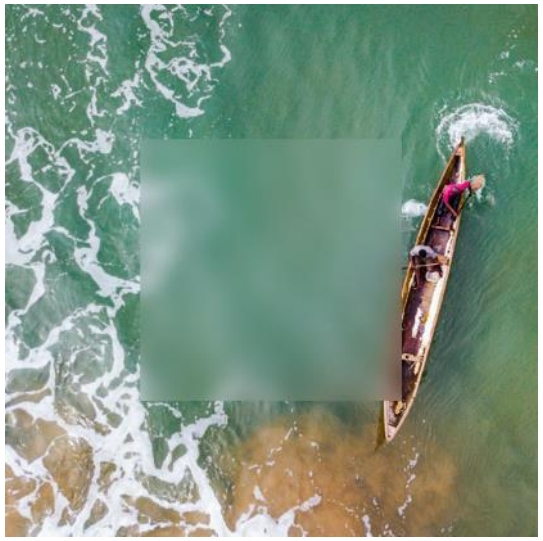
1. Download updated asset sheet

https://aka.ms/SummerTrainingMakeupAssets

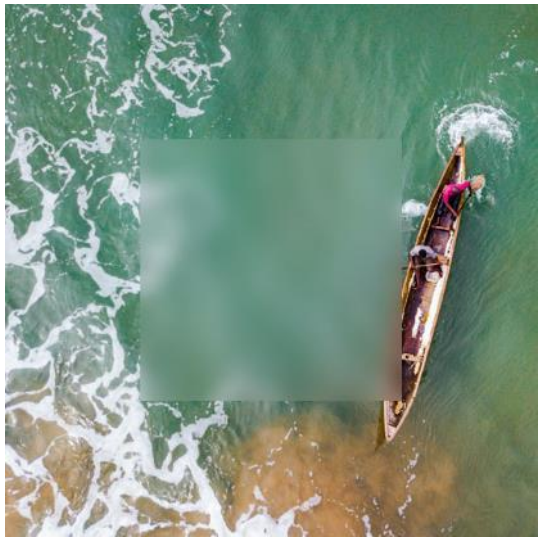
2. In Chat introduce yourself:
name, school, curriculum, role

- Respond to at least one other person not on your teaching team

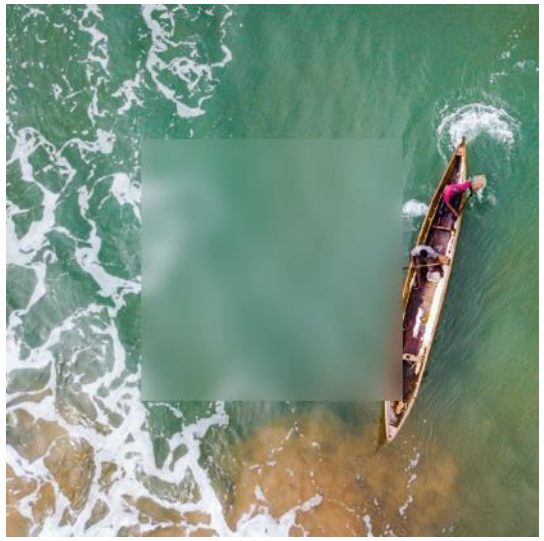
Meet the TEALS team



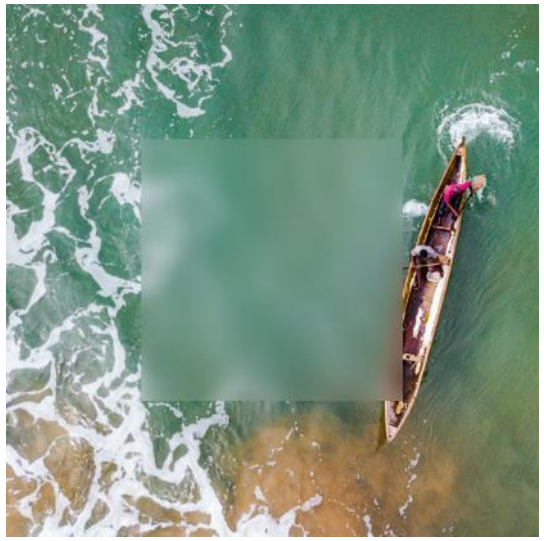
Presenter name
Title
Training role



Presenter name
Title
Training role



Presenter name
Title
Training role



Presenter name
Title
Training role

Agenda

Welcome/logistics

Your teaching team

Working as a teaching team

Student support scenarios

Computer science pedagogy

Pillar 1: The “Notional Machine”

Pillar 2: Problem solving

Pillar 3: Hierarchy of skills

Differentiated instruction

Wise feedback

Dealing with failure

Wrap-up/closing

Objectives

- Define “pedagogical content knowledge (PCK)” and list some examples of CS PCK.
- Identify four pillars of CS PCK and list elements from each pillar.
- Define differentiated instruction.
- Explain the importance of providing individualized support to students.
- List possible interventions for both struggling and advanced students.
- Use strategies to help students deal with frustration and failure in CS classes.
- You will be able identify the roles and responsibilities of the different TEALS teaching team members.
- You will be able to communicate efficiently with your teaching team.

Your teaching team

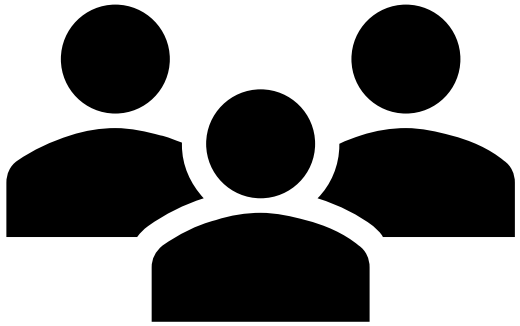
TEALS support models

	Co-Teach model	Lab support model	Graduation
Who's doing the teaching?	Teacher: 10% → 80% Volunteer: 90% → 20%	Teacher: 80% → 99% Volunteer: 20% → 1%	Teacher: 100%
Teacher's role	• Classroom and teaching team management • Learning computer science • Completing all assignments • Leading lessons at their capacity	• Classroom and teaching team management • Leading 80%+ of lessons • Continue refining CS understanding	• Teaching computer science independently of TEALS
Volunteer team engagement in the classroom	4-5 days a week	2-5 days a week	Stay in touch with volunteers! Online resources

The role of your regional manager

- Prepare teachers and volunteers for the school year.
- Facilitate quarterly Meetups for their region.
- Observe volunteers in class and offer feedback.
- Communicate action items for teaching teams throughout the year.
- Offer ongoing support for the teaching team.

Your teaching team roles



Classroom Teacher

- Manage classroom and school issues
- Lead the teaching team
- Give and take feedback
- Learn computer science

Volunteer(s)

- Classroom teacher support
- Communication
- Instruction
- Lab management

Working as a teaching team

Team communication

Plan on ***at least*** these two forms of communication:

Daily handoff

Asynchronous

Brief

Searchable

Weekly sync

Real-time

Bigger picture

Sample teaching rotations

Thursday is Alyssa's day to teach a lesson

Co-teaching sample

Monday	Tuesday	Wednesday	Thursday	Friday
Alyssa (CT)	Alyssa (CT)	Alyssa (CT)	Alyssa (CT)	Alyssa (CT)
Miya (VT)	Tho (VT)	Tho (VT)		Miya (VT)
Aidan (VTA)	Aidan (VTA)	Shaun (VTA)	Shaun (VTA)	Aidan (VTA)

Lab support sample

Monday	Tuesday	Wednesday	Thursday	Friday
Jerome (CT)	Jerome (CT)	Jerome (CT)	Jerome (CT)	Jerome (CT)
Alina (VTA)	Alina (VTA)	Alina/Suraj (VTA)	Suraj (VTA)	Suraj (VTA)

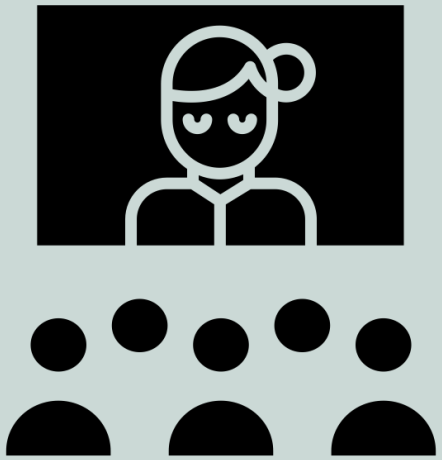
Four possible support scenarios

Students in school

Volunteer in person

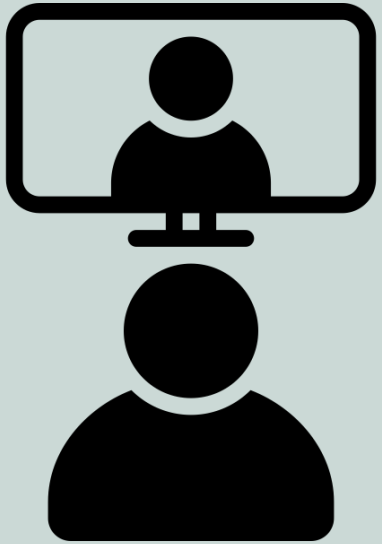
Students in school

Volunteer remote synchronous

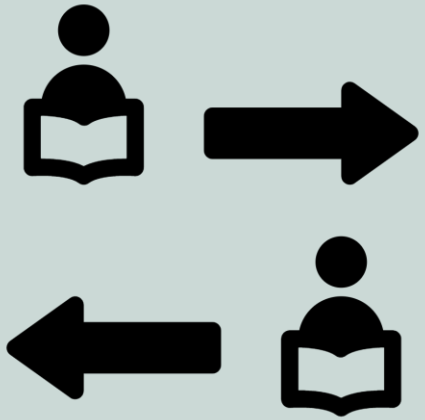


Students at home

Volunteer remote synchronous



Everyone remote asynchronous



All scenarios: Teacher/school responsibilities

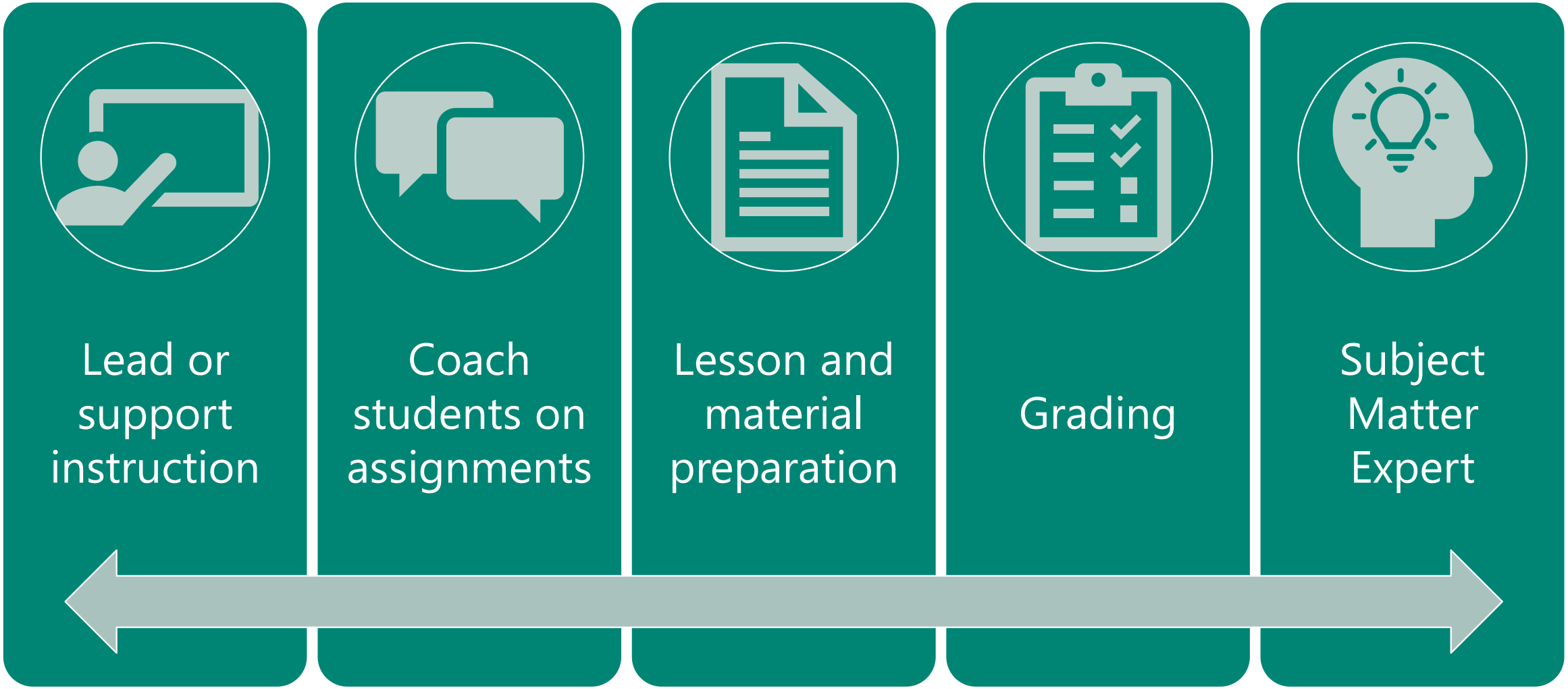
Lead and coordinate the teaching team.

Choose the tools (LMS, virtual classroom, other tools). Provide Volunteer access and training as allowed.

Be knowledgeable of school privacy, media, and student contact policies. Plan lesson activities accordingly.

Communicate clearly with volunteers on policy, procedure, and schedule changes

What volunteers do in TEALS classes



How TEALS will support you

- TEALS regional Manager
- Quarterly meetups
- Continuous training
- Best practices for remote teaching

Computer science pedagogy

Content and pedagogical knowledge

Content knowledge
"What to teach"

Syntax

Programming languages

Data representations

Algorithms

Abstraction

Tools

Real-world applications

…

Pedagogical Knowledge
"How to teach"

Student mindsets

Classroom management

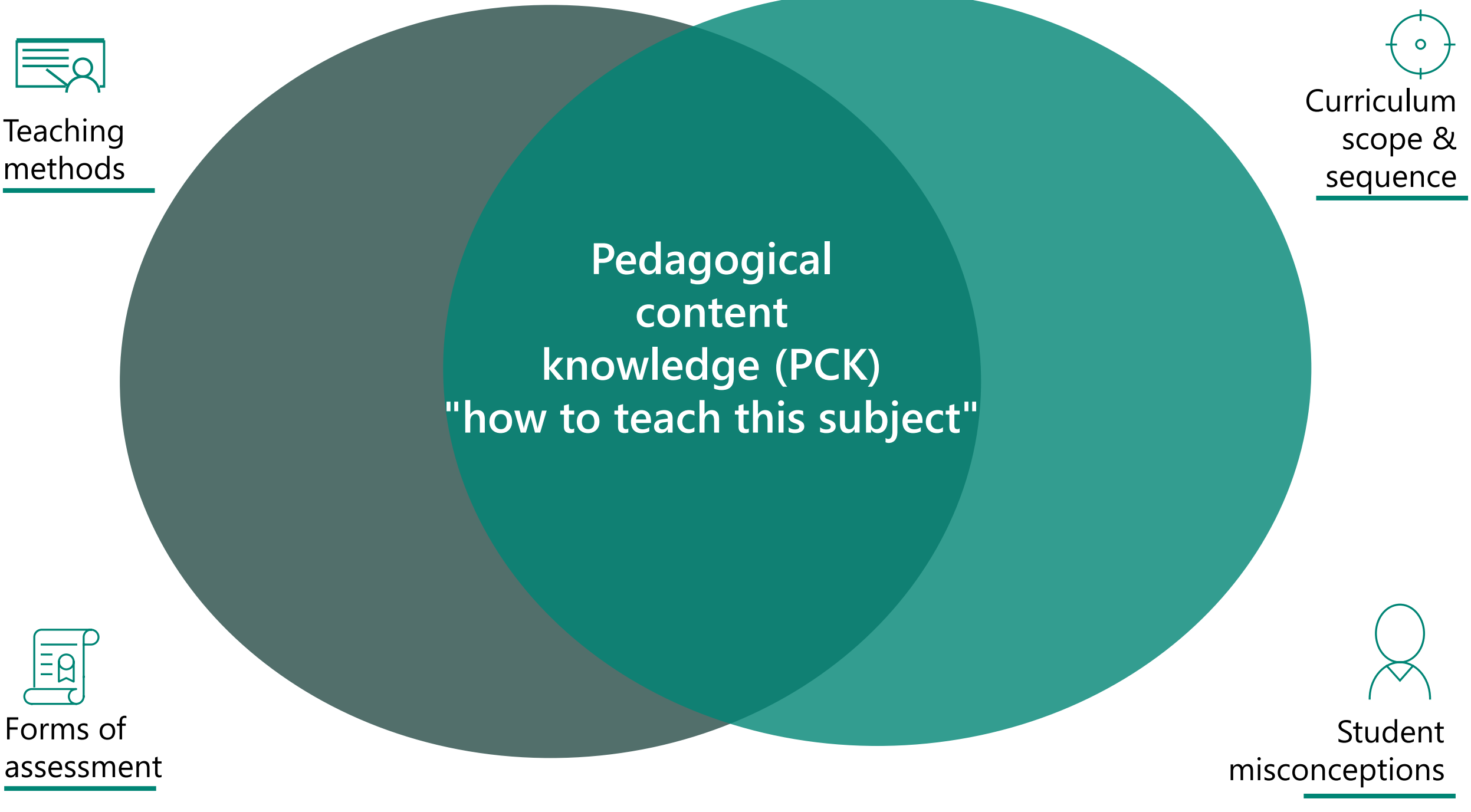
Theory of learning

Differentiated instruction

Lesson planning

Questioning techniques

…


Teaching methods
Curriculum scope & sequence
Pedagogical content knowledge (PCK)
"how to teach this subject"
Forms of assessment
Student misconceptions

TEALS computer science PCK pillars

Notional machine

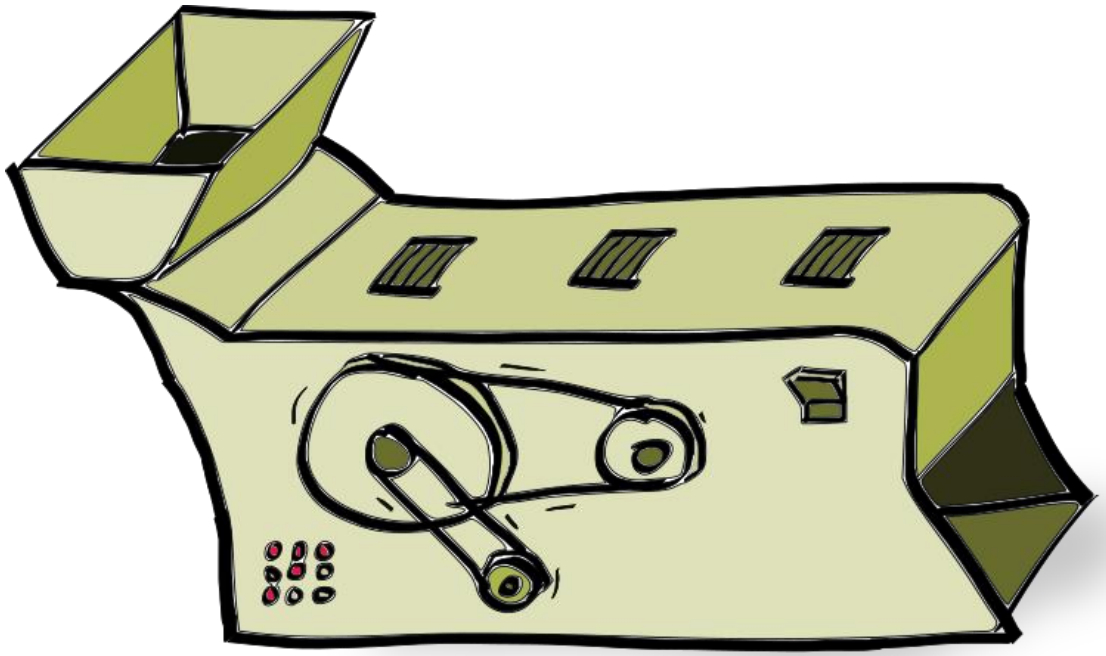
Problem solving

Hierarchy of skills

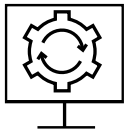
Diversity & inclusion

Pillar 1:
Notional machine

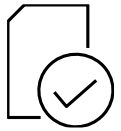
The purpose of a notional machine is to explain program execution.



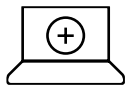
A notional machine is…



Not the hardware of your computer.

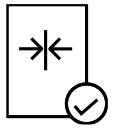


Referring to the software running on the computer.



Software written in different programming languages.

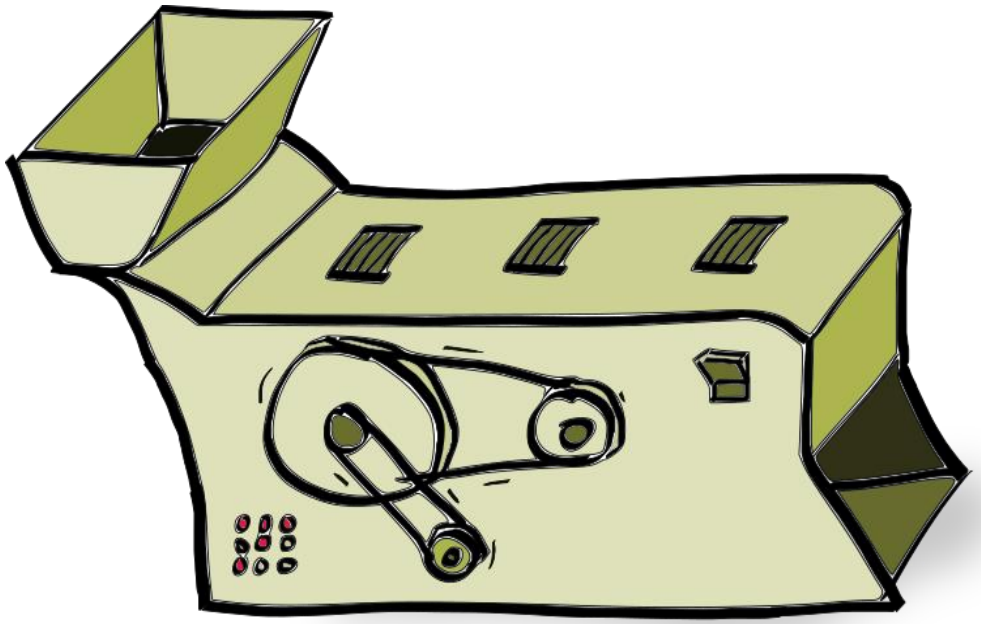
The way software handles information.



A reflection of what programs do when executed.

Pillar 1: The “Notional Machine”

- Memory diagrams
- Tracing code
- Worked examples
- Data-structure diagrams
- Class (OOP) diagrams
- Analogies and examples

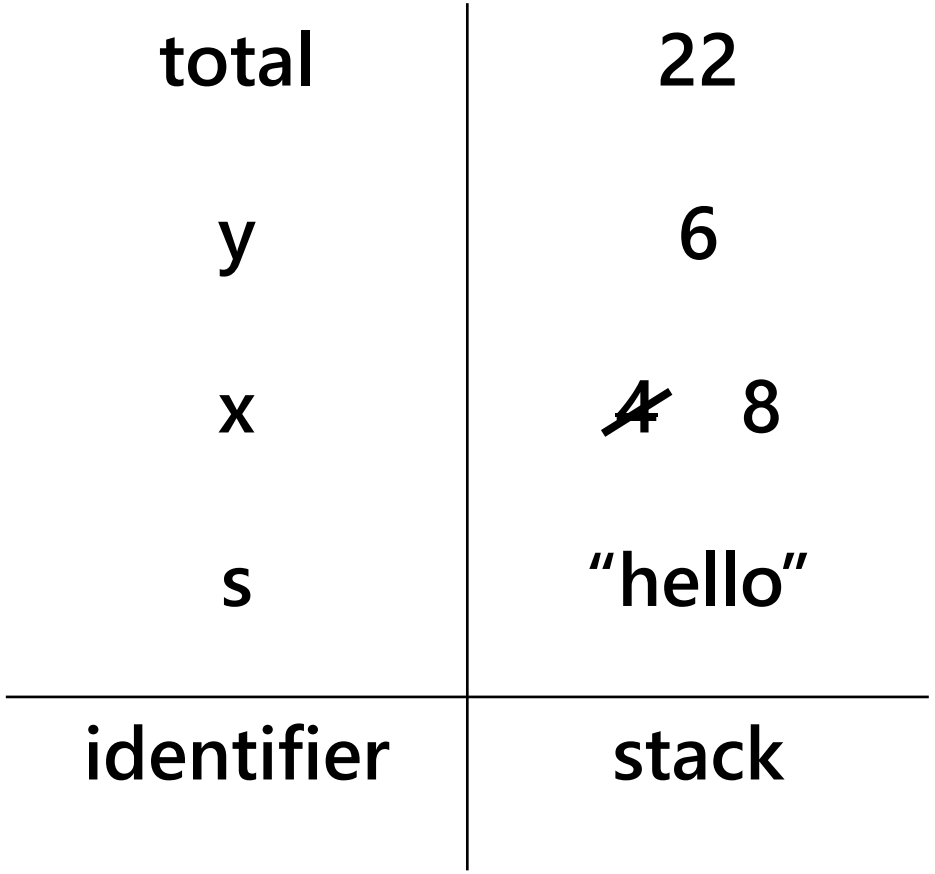


Memory diagrams

A visual representation of the state of the computer memory during the execution of a program.

Typically used while tracing through a code example.

Mimics a “debugger”



Output
22

Expert bias

> Perceiving something as easier or simpler than it is because one's own experience or knowledge of the subject.

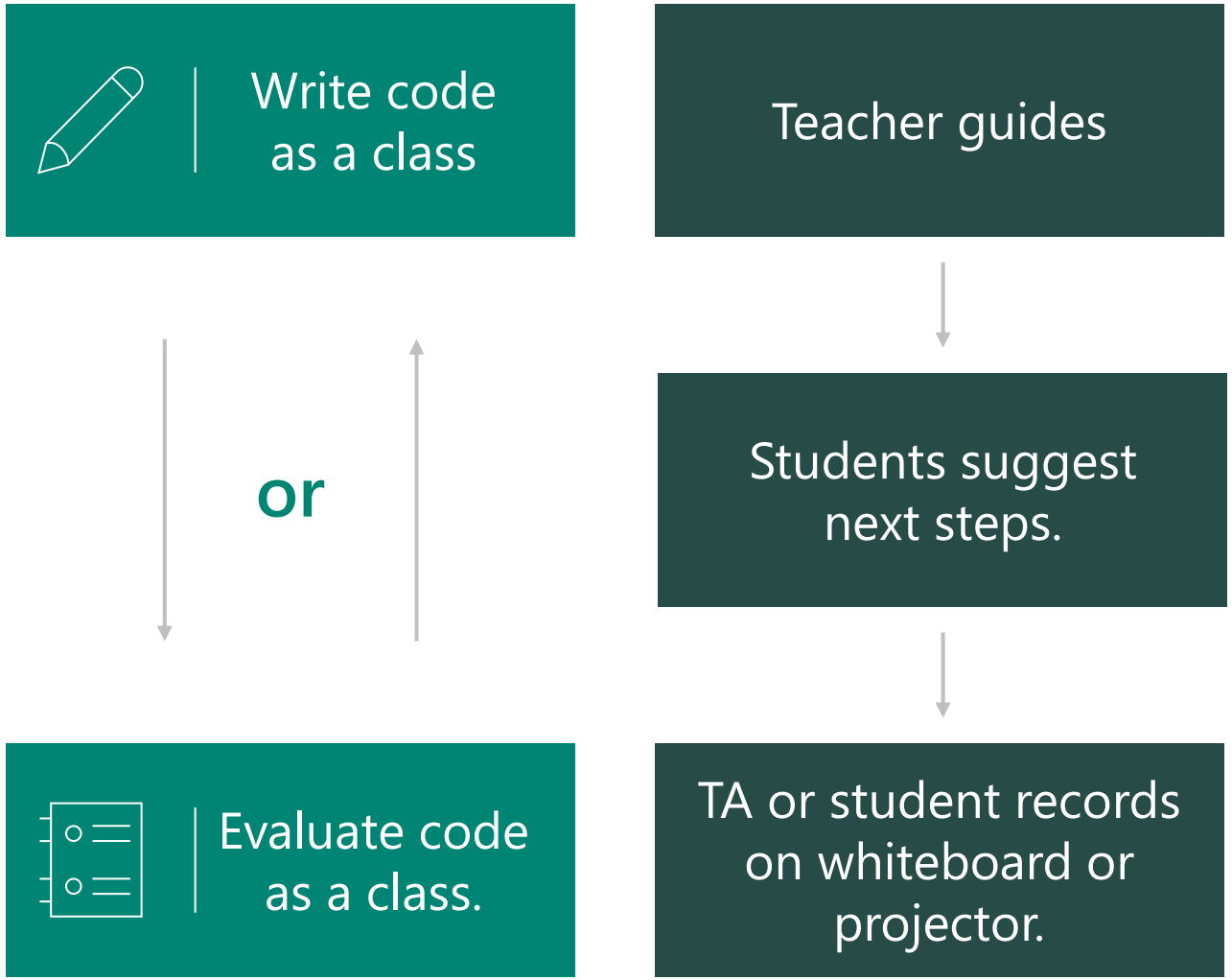
Examples

- → Lack of abstraction
- → Jargon
- → Unrealistic time expectations
- → Giving few examples

Classroom teachers

Help catch and mitigate expert bias.

Worked examples



"We do" then "you do"

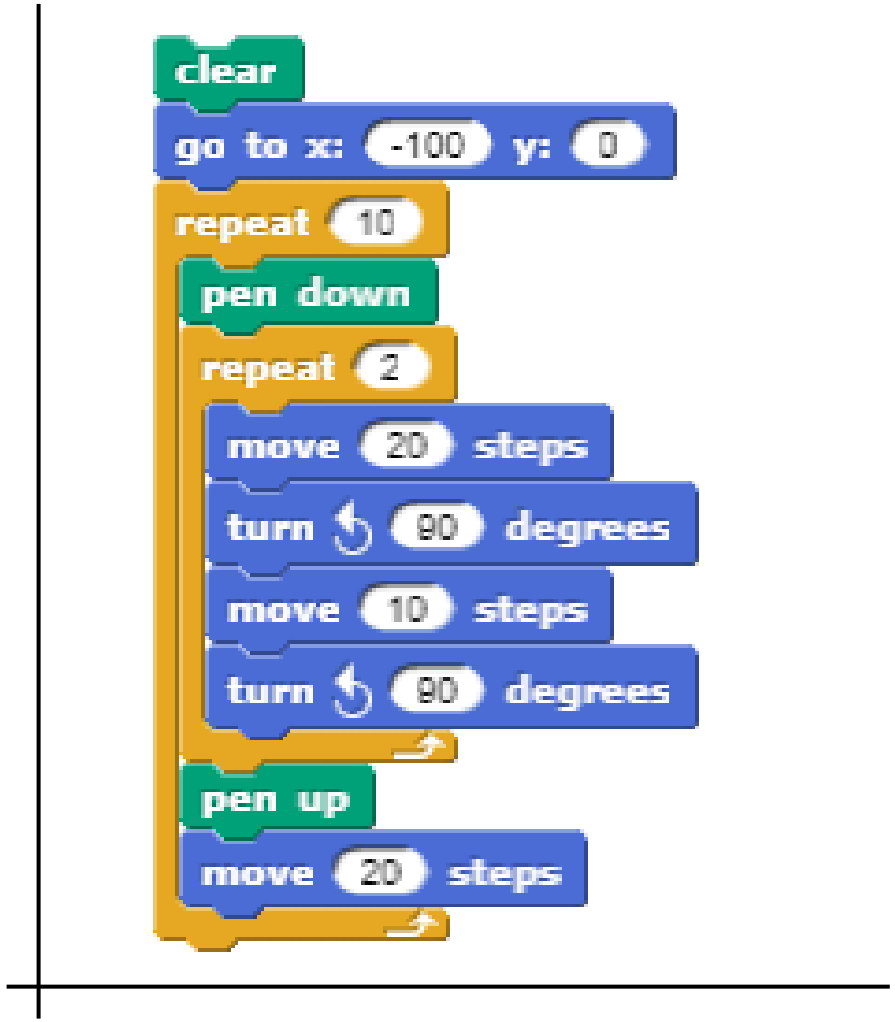
- Have a clear transition ("Now it's your turn." "OK, see if you can…" etc.)
- Students can reimplement, extend the example, or do a variation.
- Be careful of chunking; provide hints (including pseudocode).

Worked example preparation

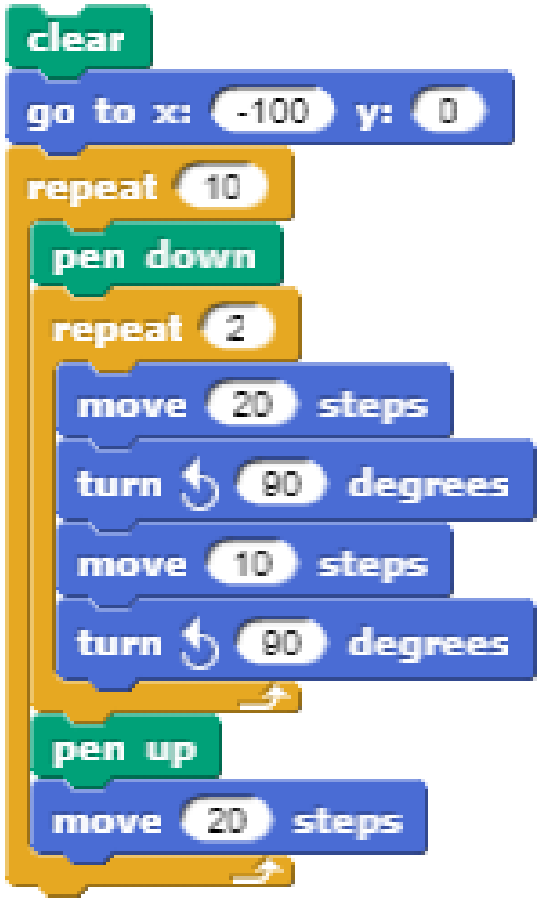
- Have a clear start and end point.
- Decide what is in and out of scope.
- Word questions carefully.
- Focus on learning objectives.
- Reference patterns and idioms.
- Consider multiple possible approaches.
- Know what comes next.

Worked example activity: Code tracing in Snap!

Let’s step through this code together.

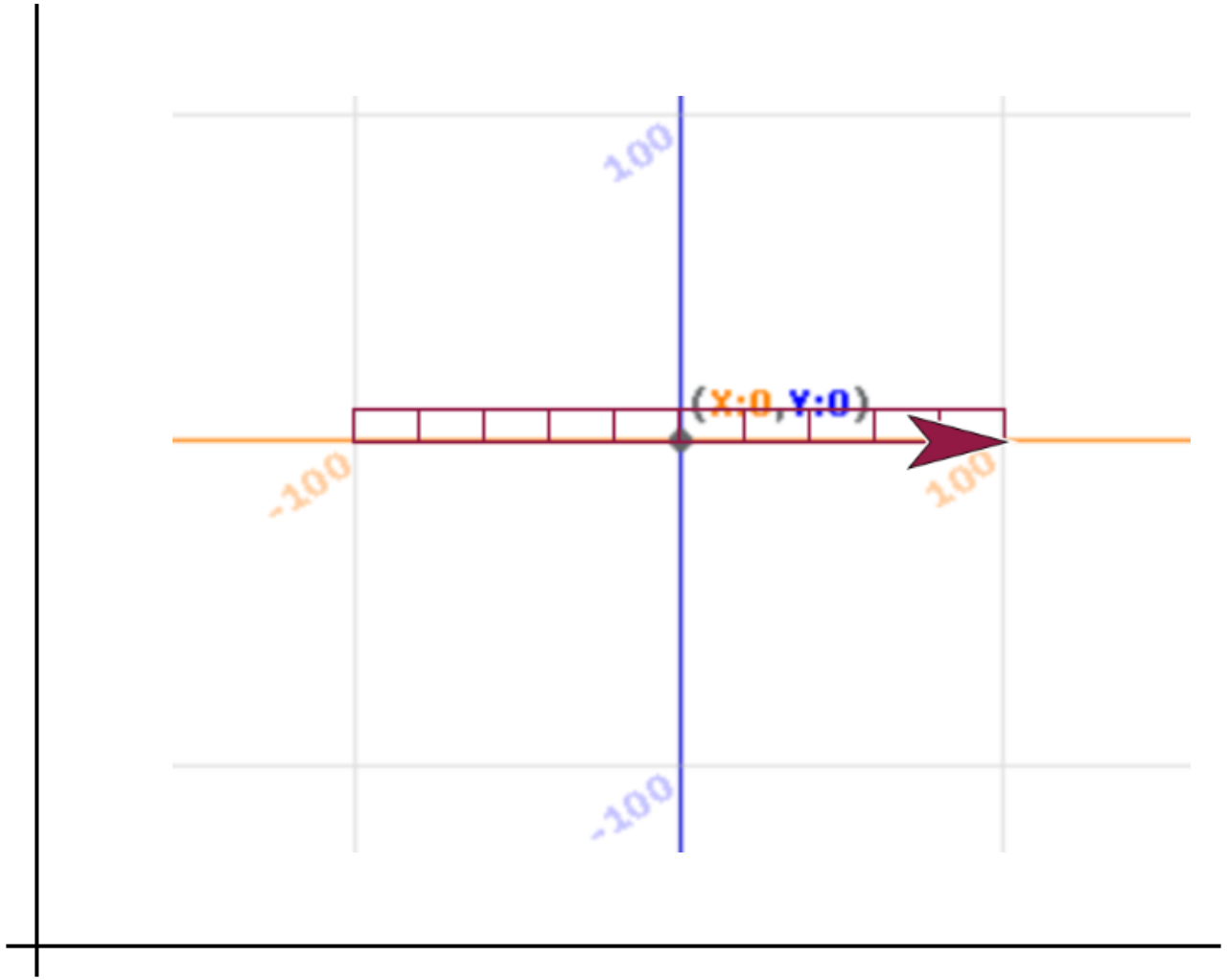


Code tracing in Snap! solution: Code



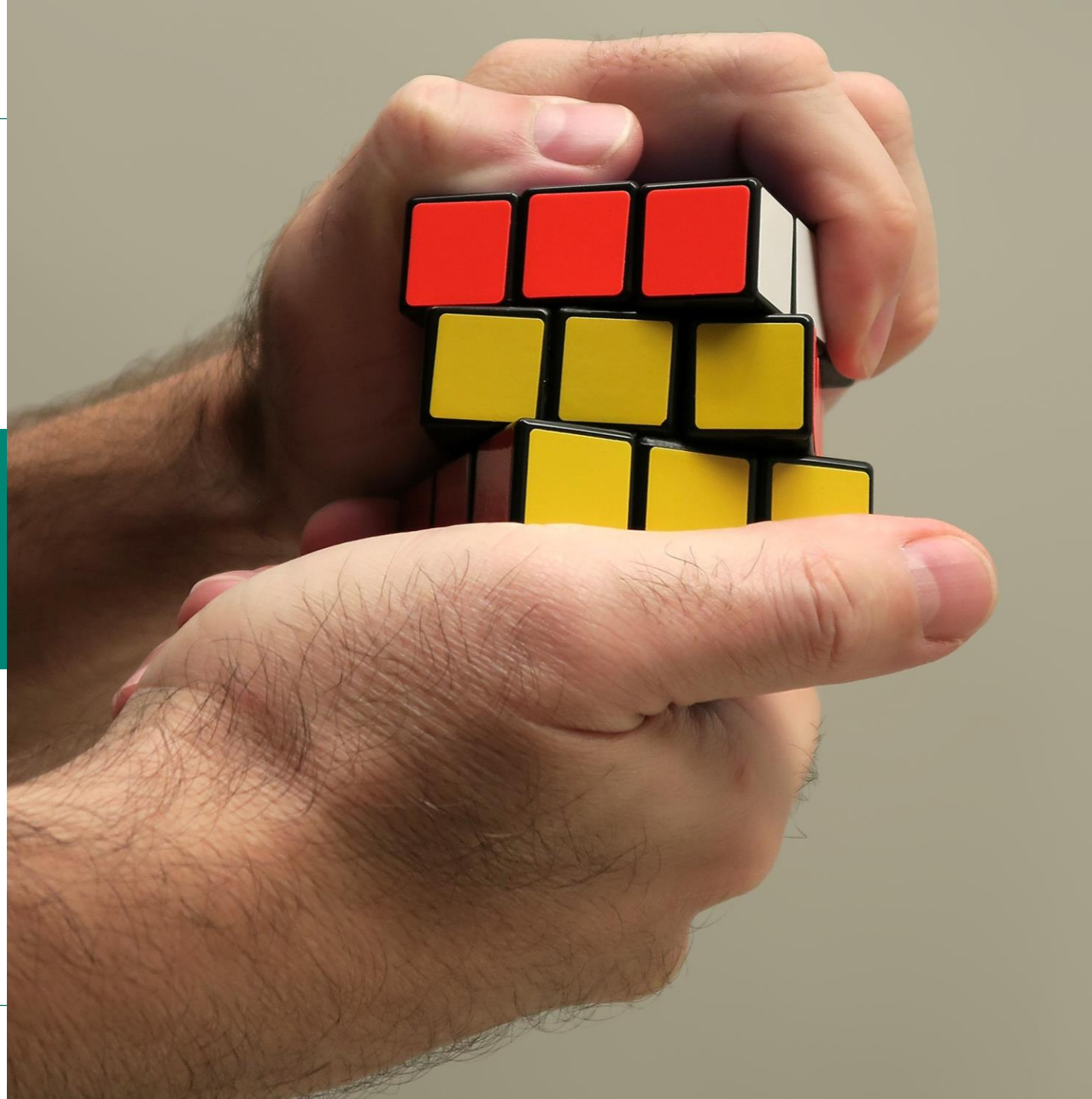
1	Clears the Screen
2	Puts the sprite 100 to the left of center horizontally
3	Repeats the enclosed block of code 10 times
4	Begins leaving trail
5	Repeats the enclosed block of code 2 times
6	Moves the sprite 20 steps
7	Turns the sprite left/counterclockwise 90 degrees
8	Moves the sprite 10 steps
9	Turns the sprite left/counterclockwise 90 degrees
10	This is the end of the Repeat Two code
11	This stops making a trail
12	Move the sprite 20 steps
13	This is the end of Repeat 10 code

Code tracing in Snap! solution: Output



Pillar 2:
Solving problems

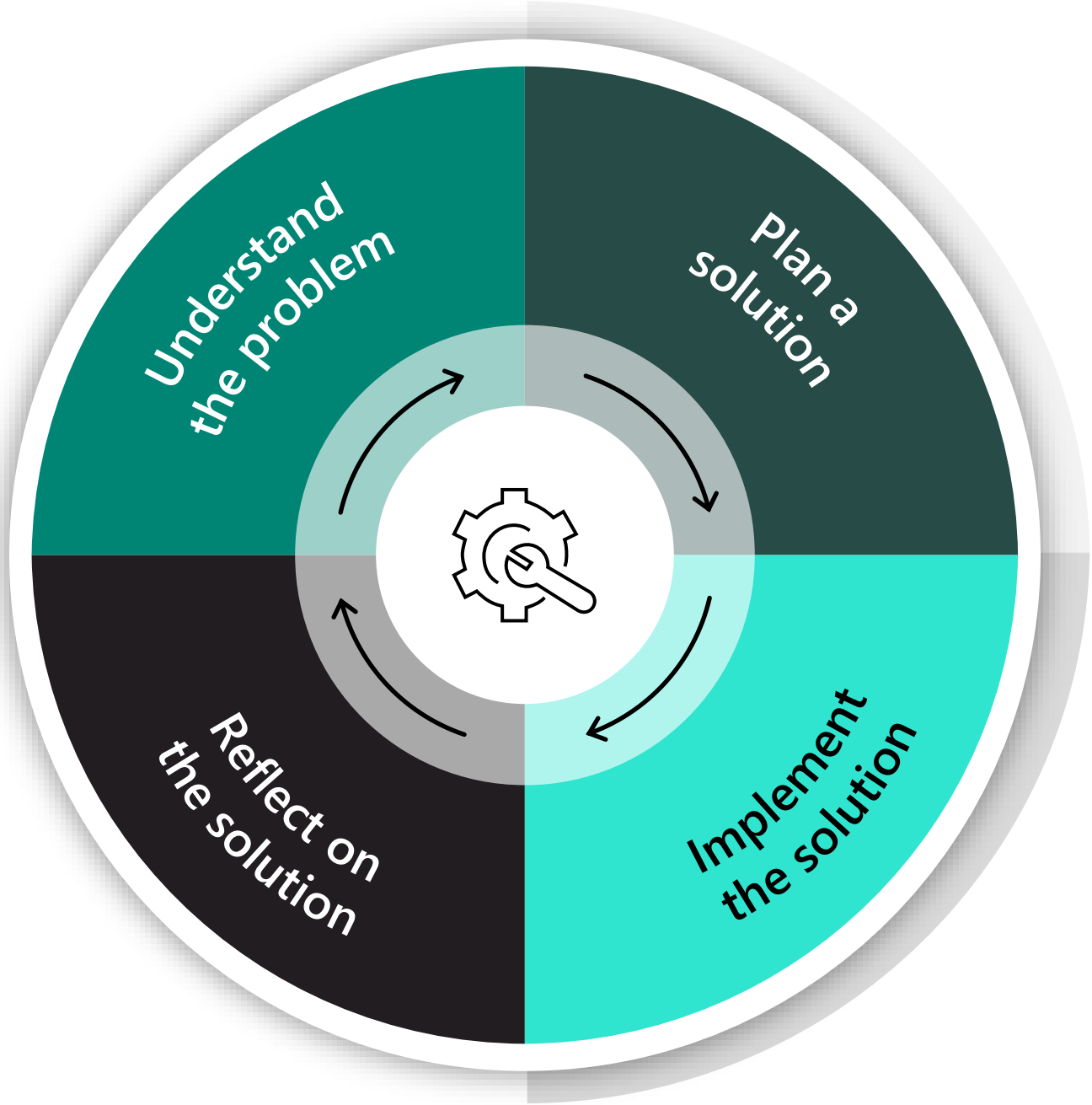
Problem solving is an integral part of what students need to learn and do in computer science.



Pillar 2: Solving problems

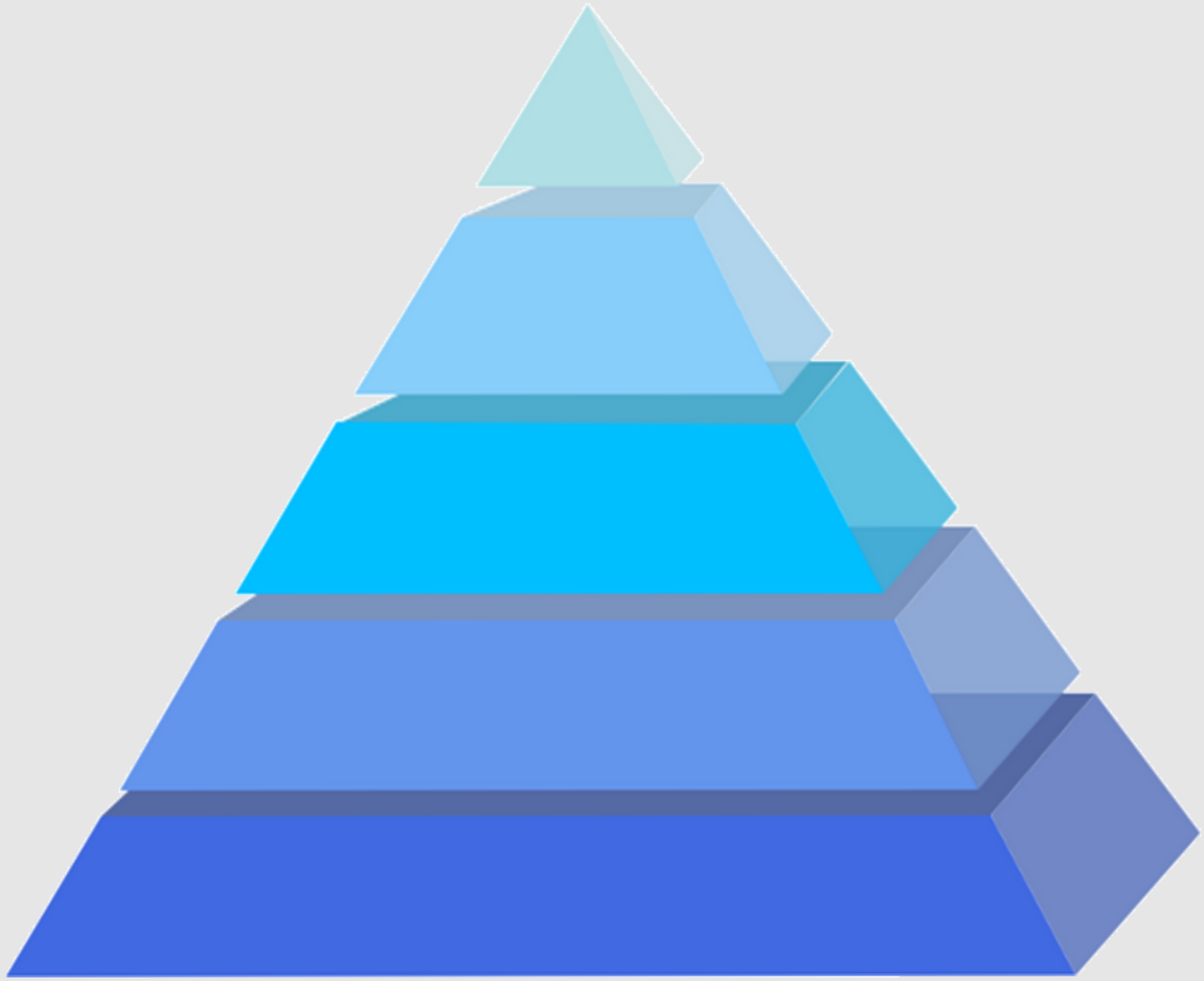
- **Four steps to solve any CS problem**
- Subgoal labeling (from Pre-Work)
- Notebooks
- Debugging techniques
  - Print statement
  - Isolation
  - Debugger tool
- Socratic method

Four steps to solving any problem.



Pillar 3:
Hierarchy of skills

Learning to write programs is a many-layered skill. Lessons and assessments should progress through the hierarchies of skills and knowledge.



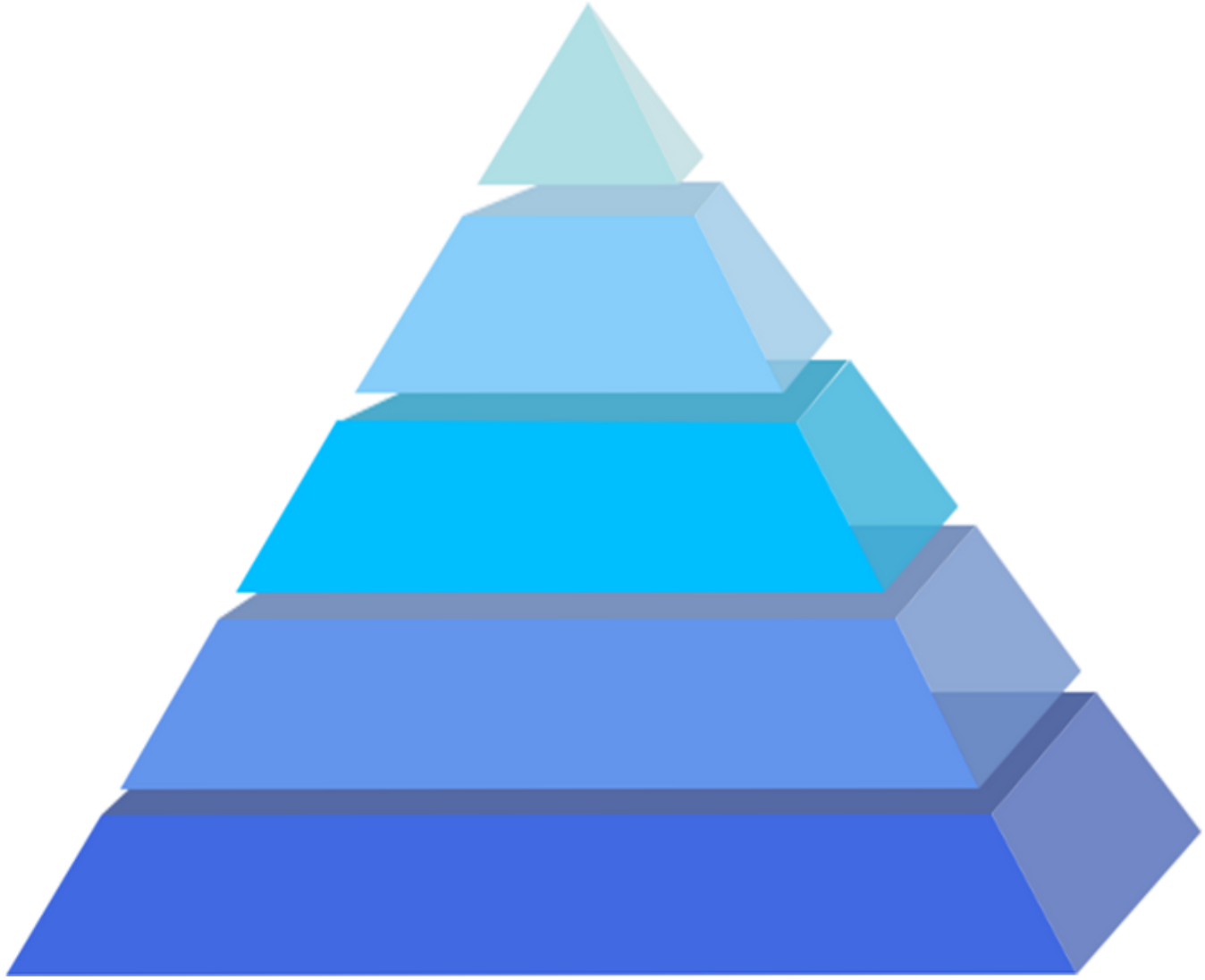
Pillar 3: Hierarchy of skills

Bloom's taxonomy

Levels of abstraction in defining CS concepts.

- Definitions
- Syntax
- Patterns
- Generalization

Assessment tasks that map to these hierarchies.



Common programming assignment tasks

Code Completion

Insert the missing expression so that this program prints the maximum value from a list stored in the variable numList:

```
max = numList[0]
foreach n in numList:
 if {MISSING CODE}:
 max = n
print(max)
```

Code Tracing

What value will be printed?

```
numList = [32, 100, 31, 5]
max = numList[0]
foreach n in numList:
 if n > max:
 max = n
print(max)
```

Literal Translation

Translate each of the following pseudocode statements into Python code:

1. Initialize variable “max” to 0
2. Iterate through each value in the list “numList”
3. Print the variable “max”

Parsons Problem

Reorder the following lines of code so that this program prints the maximum value from a list stored in the variable numList:

```
1. max = n
2. print(max)
3. foreach n in numList:
4. max = numList[0]
5. if n > max:
```

Summary

In words, describe what this code does:

```
max = numList[0]
foreach n in numList:
 if n > max:
 max = n
print(max)
```

Synthesis

Write a program that prints the maximum value from a list stored in the variable numList:

Common programming assignment tasks

Literal translation (Remember)

Translate each of the following pseudocode statements into Python code:

1. Initialize variable “max” to 0
2. Iterate through each value in the list “numList”
3. Print the variable “max”

Summary (Understand)

In words, describe what this code does:

```
max = numList[0]
foreach n in numList:
  if n > max:
  max = n
print(max)
```

Code tracing (Apply)

What value will be printed?

```
numList = [32, 100, 31, 5]
max = numList[0]
foreach n in numList:
    if n > max:
        max = n
print(max)
```

Code completion (Analyze)

Insert the missing expression so that this program prints the maximum value from a list stored in the variable numList:

```
max = numList[0]
foreach n in numList:
  if {MISSING CODE}:
  max = n
print(max)
```

Parsons problem (Evaluate)

Reorder the following lines of code so that this program prints the maximum value from a list stored in the variable numList:

1. `max = n`
2. `print(max)`
3. `foreach n in numList:`
4. `max = numList[0]`
5. `if n > max:`

Synthesis (Create)

Write a program that prints the maximum value from a list stored in the variable numList:

Microsoft

Differentiated instruction

Some students are ahead, others are behind.

What should I do?

What it really means

Differentiated instruction is

- *Planned*, not an afterthought.
- A *value-add*, not a time-killer.
- Focused on helping students *grow and learn*, not "catch up" or "slow down".
- Tailored to students' strengths, interests, background, home life, and lived experiences.

Differentiated instruction is ***not***

- Extra or reduced work (without shifting complexity, style, etc.)
- Providing answers when "time is up".

Activity: Classroom example

During lab you find that:

- 5 students don't understand loops at all.
- 15 are handling the lab just fine.
- 5 finished the lab in 10 minutes and are now bored.

How can you help those students who are **behind?**

How can you help those students who are **ahead?**

What if a particular student is **frequently** ahead or behind?

Microsoft

Wise feedback

Emotions affect learning

Wise feedback

Emotions affect learning

Effective feedback:

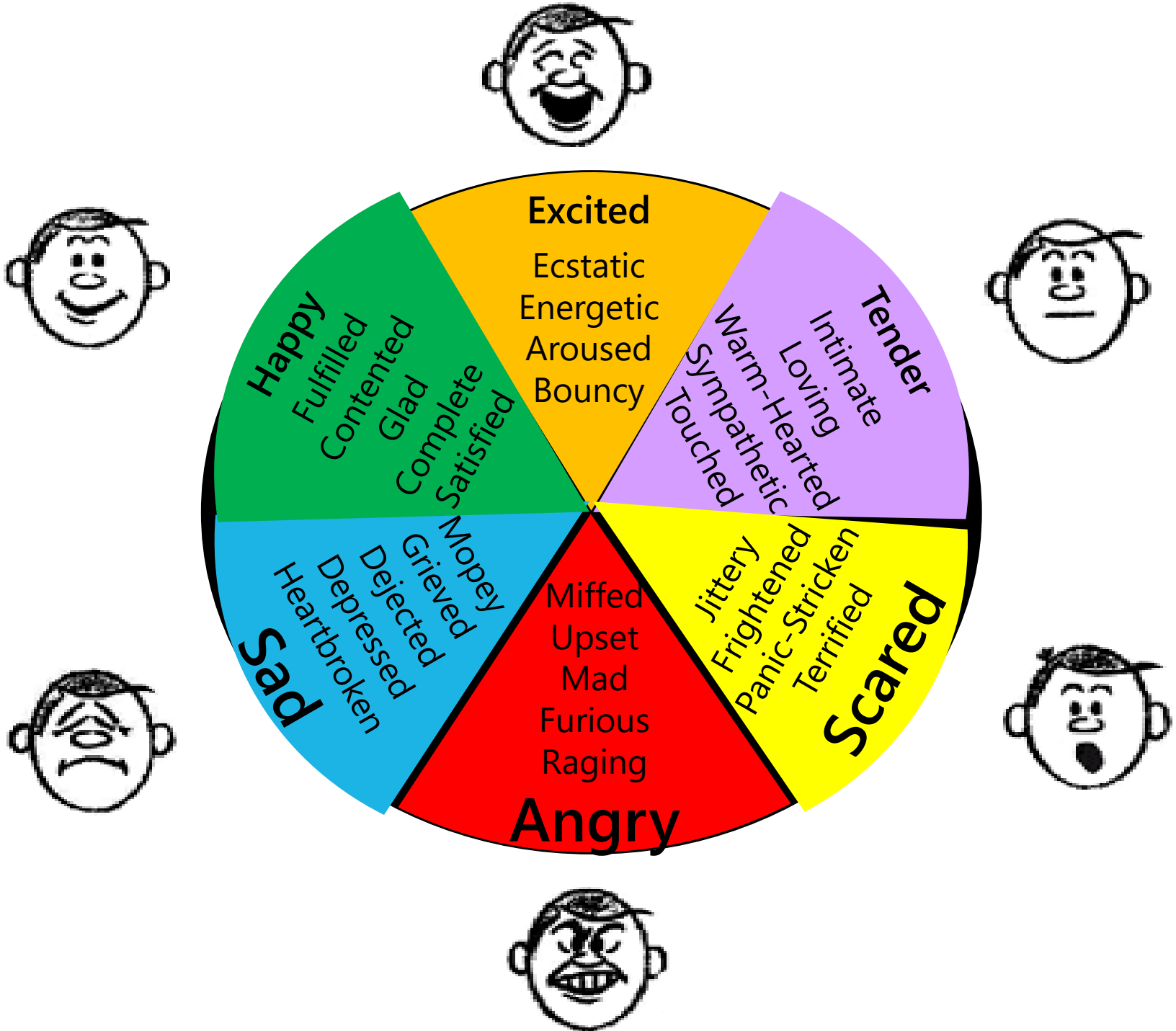
- Is unique to the student's learning situation.
- Balances faith in student potential with honesty about current performance.
- Reassures students that they will not be stereotyped or doubted as less capable.

Wise feedback is most effective with students who lack trust

Microsoft

Basic emotions

Students come to class with a variety of emotional states.



Three elements of “Wise Feedback”

High standards

- Affirm holding high standards.
- Mistakes are a sign of the high demands of the task/class, not of (perceived) low capability.

Personal assurance

- Assure the student that he or she is capable and can improve with effort.
- Reference past successes or progress.

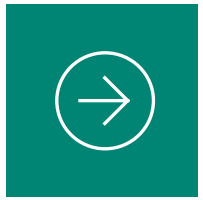
Actionable steps

- Give specific, actionable steps to work on.
- Instructive rather than evaluative.
- Corrective rather than vague.

Example feedback

High standards

“Developers are often asked to create programs like this.”



Actionable next steps

“One thing computer scientists do on a regular basis is look for ways to make their code more efficient. Can you think of a way to produce the same output with fewer lines of code?”

“You may be asked to make this program do something different in the future. What could you change in your code to allow you to find all the even numbers up to 50? All the odd numbers?”

Personal assurance

“I really appreciated how you solved the problem as specified. I’m certain you can improve your program if you take another look.”

Word choice

Feedback	WISE feedback
“That’s wrong.”	“Good first step. Let’s see how we can get closer to solving the problem”
“Fix it.”	“Some of the code works, let’s fix the parts that don’t”
“Who’s not done yet?”	“Who’s still working?”
“It’s much better than it was.”	“You’re making great progress, keep up the effort.”
“Raise your hand if you don’t get it.”	“Raise your hand if you’d like to hear that again.”
“Here’s a better way.”	“There’s many ways to solve a problem, may I suggest a different way?”
“What are you struggling with?”	“What are you working on that I can help with?”

Inclusive assessment examples

Assignment	Assessment modification example
Lab/practice assignments:	Provide several ways to demonstrate understanding of new content: • Exercises using a practice tool. • Write a program of your own design that meets a checklist of criteria (i.e. uses methods X, Z, has 10 lines of code). • Creation of posters or songs to review/reinforce new material.
Tests/HW:	Students select 4 out of 8 of homework or quiz questions.
Final projects/ presentations	Student work portfolio with documented progress, peer review, frequent feedback and rubrics, student contracts with predetermined learning goals.

Dealing with failure

CS is different

No single right answer

Rarely right the first time

Completely new concepts
(for most students)

AND

Not completely open-ended

Clear criteria for completeness

High expectations

Growth Mindset

According to Carol Dweck, Growth Mindset is the belief that rather than being fixed or innate, abilities can be acquired through study and effort.

Student behaviors and mindset

I'm a good student, I should be better at this.

I'm bad at math; I'll be bad at this too.

I don't even know where to start.

What they're thinking

What they're doing

- Frustration
- Not seeking help
- Cheating
- Arguing over grades
- Showing off

- Shutdown
- Not seeking help
- Over-reliance on help

- Paralysis
- Not seeking help.

Wrap up

What’s next?

- Continue working through your eLearning modules.
- Reach out to your teaching team or regional manager with questions.

Exit ticket

https://aka.ms/BestPracticesMakeupReview