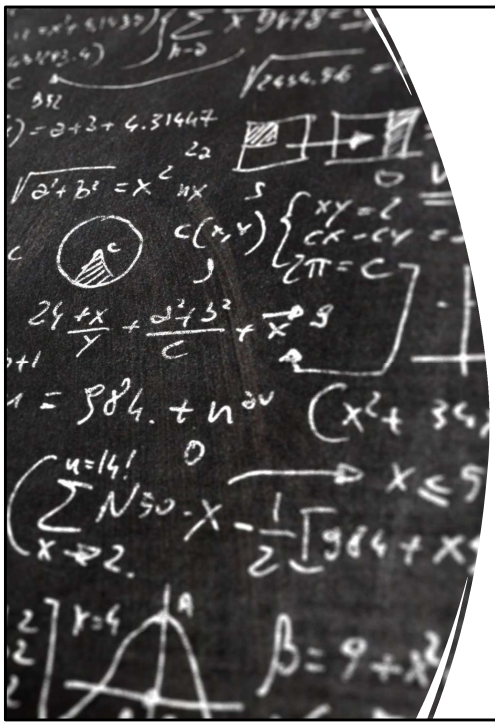


What (Usually) Works in Teaching CS

Alfred Thompson
https://blog.acthompson.net
act2@acthompson.net

Who Am I?

- Bishop Guertin High School – Retired
- 15+ years teaching high school computer science
- 27 years working in the computer industry
- Former member of the CSTA Board
- Member of the ACM/IEEE CS 2013 Curriculum Task Force
- Member of the CS K12 Framework development team



Techniques for Today

- Sub Goal Labels
- Live Coding
- Parsons Problems
- Rubber Duck Debugging

Subgoal labels explain the purpose of groups of steps in a worked example. This introductory example gro

Illustrative Python example assumes two valid integer inputs.

[Sources]
Concept: the existing What Is Subgoal Labeling and Subgoal Labeling slides in the user-provided presentation.
Visual: AI-generated conceptual code illustration.
[/Sources]

What Is Subgoal Labeling

What Is Subgoal Labeling	Why?
Subgoal labels are structure-based instructional explanations that describe the subgoal of a set of steps of a worked example to the learner **In a word, scaffolding**	• Better learning • Better transfer • Works in different languages • Works in both text and block-based programming languages • Works in Parsons Problems

[Subgoal labeling – Wikipedia]

Note that is not the same as pseudo code. Pseudo code is more specific.

Subgoal Labeling

Input a list of numbers one at a time, stopping when -999 is entered. Display the mean of the numbers prior to that point

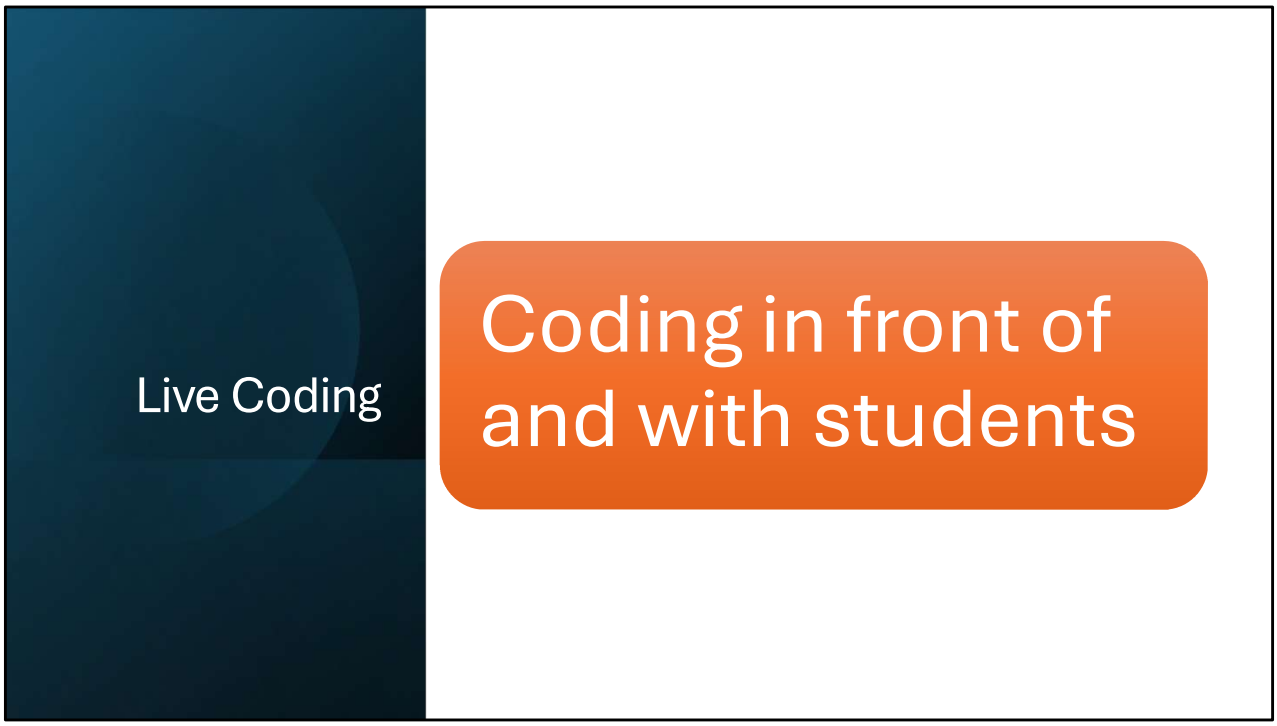
Subgoal Labeling

```
# Initialize variables
total = 0
count = 0
# Loop to input data
while True:
    item = float(input("? ")
# test if end of list
    if item == -999:
        break
# if not, update total
    total = total + item
    count += 1
# Print the mean
print (total / count)
```

Introduce live coding as writing code in front of and with students while explaining your thinking. Pause to

The classroom scene and dialogue are illustrative. For x = 3, print(x + 2) outputs 5.

[Sources]
Content: existing Live Coding slides in the user-provided presentation.
Visual: AI-generated conceptual classroom illustration, not a photograph of an actual lesson.
[/Sources]


Live Coding
Coding in front of and with students

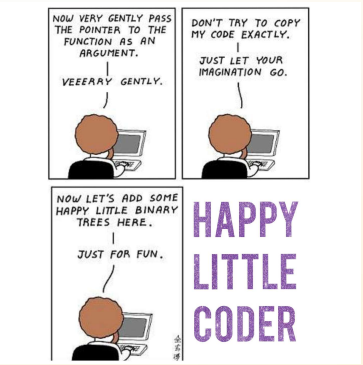
Live Coding

Think out loud to share your thinking

- Talk about what you are doing
- Making mistakes is a good thing
- Ask for predictions

Don’t emulate professional developer presenters

- No prewritten code
- A teacher’s priority is different



Live Coding Options



Code alone

Moves faster

Students are focused on you

Easier to ask for predictions



Have Students code along with you

Students come away with sample code

Students may make errors and must deal with them

Ask the audience to reorder the lines so the program asks for a name, builds a greeting, and prints it. The s

Parsons problems give learners scrambled code to reorganize into a correct sequence. Some variants include distractor lines or require learners to adjust indentation.

[Sources]
National Centre for Computing Education, Pedagogy Quick Reads: Improving program comprehension through Parson's Problems. https://static.teachcomputing.org/pedagogy/QR13-Parsons-problems.pdf
[/Sources]

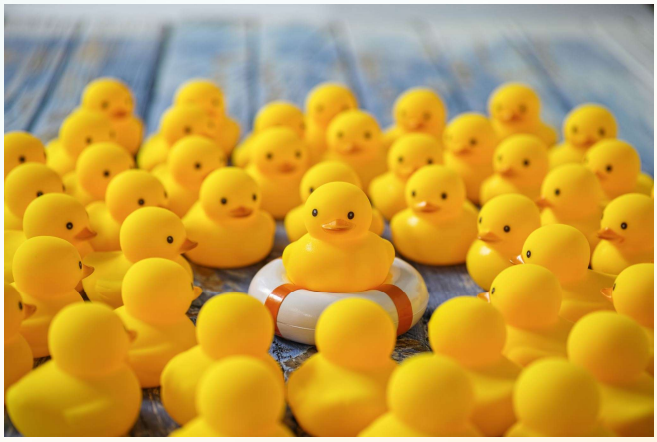
Parsons problems ask learners to arrange scrambled code into a sequence that completes a task. The illu

[Sources]
National Centre for Computing Education: Improving program comprehension through Parson’s Problems. https://static.teachcomputing.org/pedagogy/QR13-Parsons-problems.pdf
Visual: AI-generated conceptual illustration.
[/Sources]

https://js-parsons.github.io/

Graphical Parson’s Problem Creator | parsons-puzzle-ui
S4xE5: Parson's Problems - CS-Ed Podcast Home Page
Parsons Problem - Kapor Foundation

Rubber Duck Debugging



Rubber Duck Debugging

What?

- Talk to the duck
  - Explain the problem
  - What is supposed to be happening
  - Explain what the code says



Why?

- Talking out the problem often suggests the fix

[Rubber Duck Debugging – Rubber Duck Debugging – Debugging software with a rubber ducky]

Illustrative classroom exercise, not an observed student quotation.

Ask students to predict the output before running the code. Give them a minute to explain each line to a rubber duck or another inanimate object. Prompt: Which values does n take? In Python, range excludes its stop value, so range(1, 5) produces 1, 2, 3, 4 and the sum is 10. Changing the stop value to 6 includes 5 and yields 15. Ask students to run the corrected program and check the result.

Rubber Duck Debugging Options

- Give each student their own duck
- Assign a duck to each workstation
- Build a duck pond
- Find what works for you and your students

Questions?



The video of when I gave a version of this talk at the national conference a few years ago https://www.youtube.com/watch?v=vHmnGD1aJM4