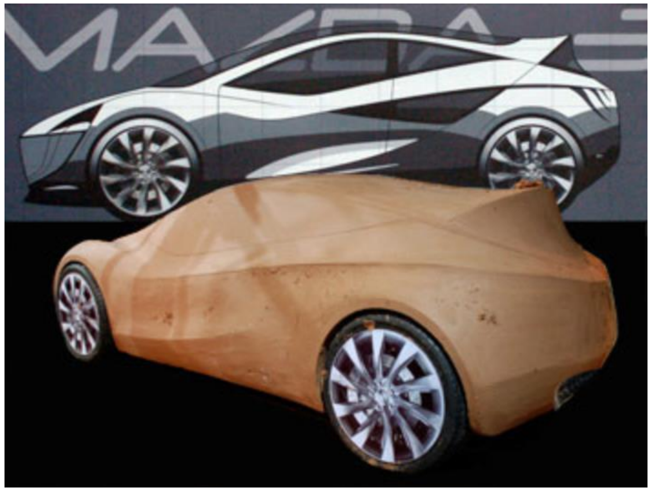
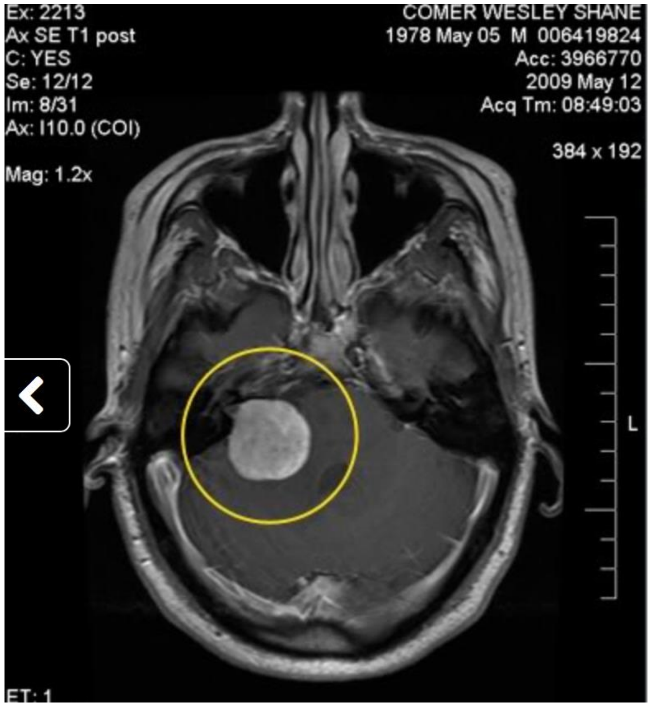
Barbara Ericson Georgia Tech Un. of Mich. (as of Sept 1, 2018)

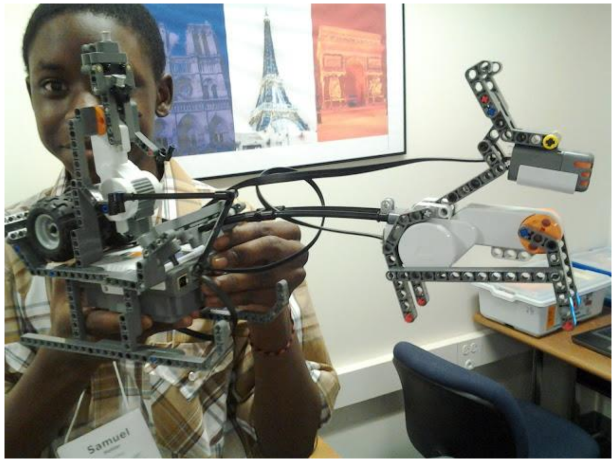
ericson@cc.gatech.edu
ericsonga1@gmail.com

Areas of Interest

- User Interface
- Artificial Intelligence
- Computer Science Education

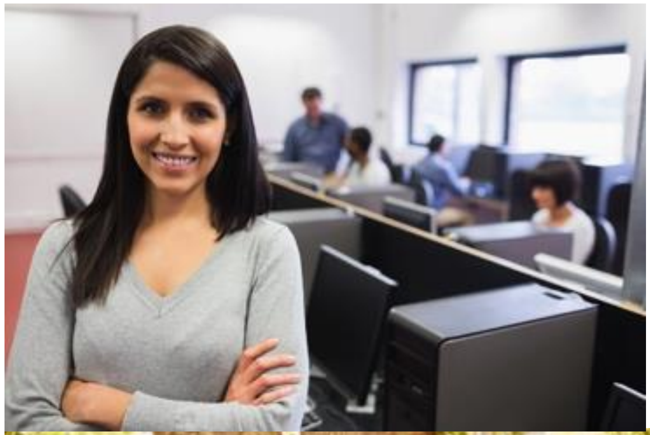






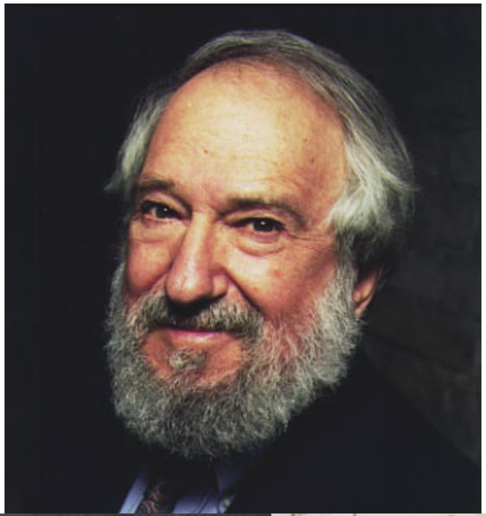
Current Questions

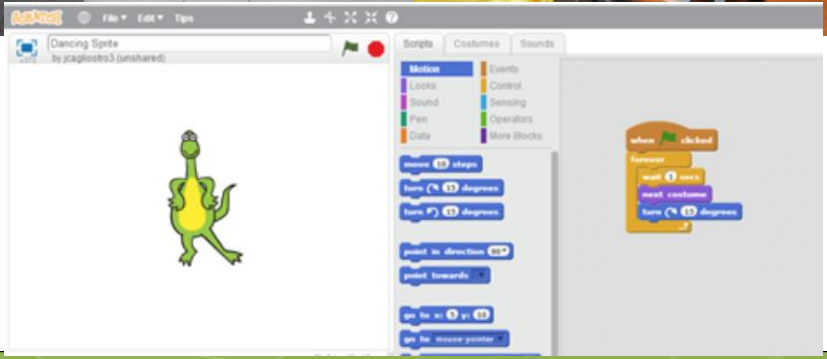
- How to improve computing education?
- How to increase diversity in computing?
- How to integrate computing and other subjects to improve learning?



Seymour Papert - Constructionist

- Use computers to enhance learning and creativity
  - Apply Piaget's theories of learning
- Cofounder of the AI lab at MIT
- Created the Logo programing language
  - Turtle Graphics
- LEGO Mindstorms robots and Scratch





Ann Brown – Design Based Res.

- Educational psychologist
- Moved away from lab science to working in schools
  - Created reciprocal teaching method
  - Used both quantitative data and in-depth qualitative data

The vision of Papert with the rigor of Brown

Past Work

- Summer Camps for rising 4th – 13th graders
  - Increases in content knowledge
  - Increases in self-efficacy
- Weekend workshops with youth serving organizations
  - Positive attitudes towards computing
- Competitions
  - Scratch
  - Alice
  - Advanced Placement Computer Science

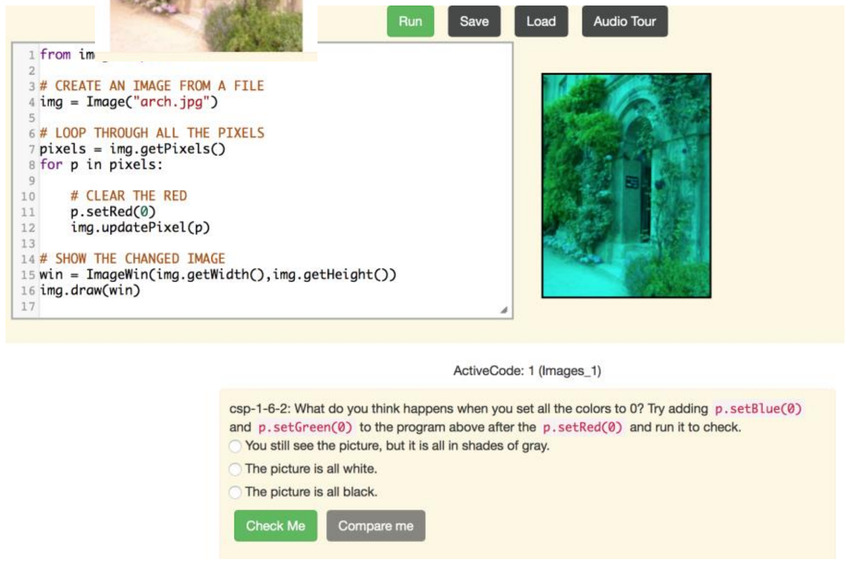
Teacher Professional Development

- 500+ teachers in face-to-face workshops
  - Mostly secondary teachers
  - One week
  - One day
- Created curriculum
  - 4 courses in Georgia
  - Advanced Placement Computer Science A
    - Picture lab

Ebook Design

- Educational Psychology
  - Interleaved worked examples plus practice problems
    - Multiple modalities
    - Assessment with immediate feedback
    - Subgoal labels
- Run, Edit, and Modify Code
  - Not write lots of code

Original Image



```
from im

# CREATE AN IMAGE FROM A FILE
img = Image("arch.jpg")

# LOOP THROUGH ALL THE PIXELS
pixels = img.getPixels()
for p in pixels:

    # CLEAR THE RED
    p.setRed(0)
    img.updatePixel(p)

# SHOW THE CHANGED IMAGE
win = ImageWin(img.getWidth(),img.getHeight())
img.draw(win)
```

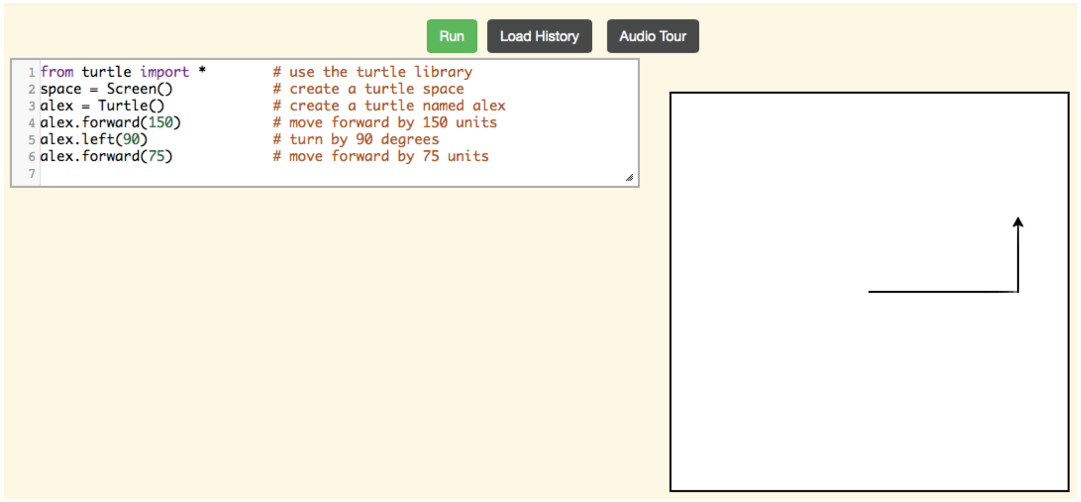
ActiveCode: 1 (Images_1)

csp-1-6-2: What do you think happens when you set all the colors to 0? Try adding `p.setBlue(0)` and `p.setGreen(0)` to the program above after the `p.setRed(0)` and run it to check.

- ◯ You still see the picture, but it is all in shades of gray.
- ◯ The picture is all white.
- ◯ The picture is all black.

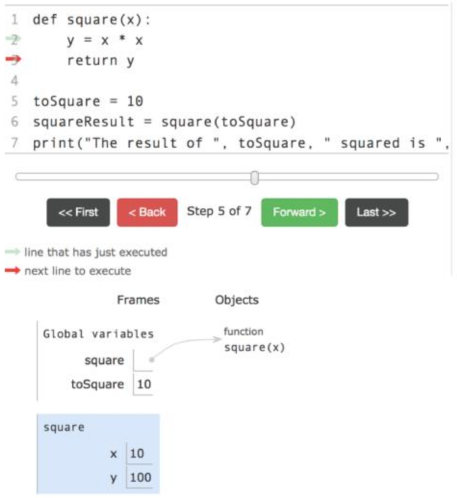
Code Tools

Runnable and Editable Python Code – runs as JavaScript



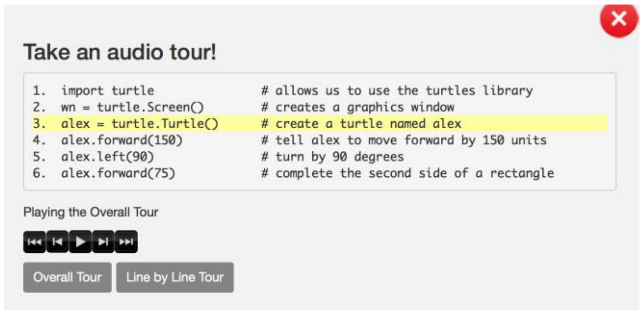
```
from turtle import *         # use the turtle library
space = Screen()             # create a turtle space
alex = Turtle()              # create a turtle named alex
alex.forward(150)            # move forward by 150 units
alex.left(90)                # turn by 90 degrees
alex.forward(75)             # move forward by 75 units
```

Code Visualizer



```
def square(x):
    y = x * x
    return y

toSquare = 10
squareResult = square(toSquare)
print("The result of ", toSquare, " squared is ",
```

Audio tours of code. Highlights line(s) as audio plays



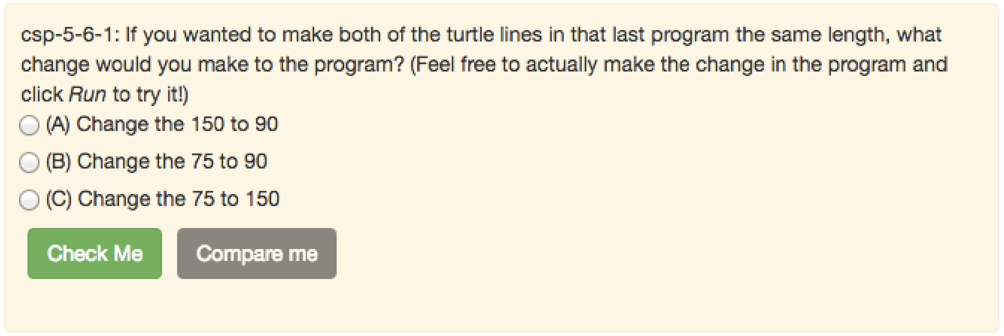
Take an audio tour!

```
import turtle               # allows us to use the turtles library
wn = turtle.Screen()        # creates a graphics window
alex = turtle.Turtle()      # create a turtle named alex
alex.forward(150)           # tell alex to move forward by 150 units
alex.left(90)               # turn by 90 degrees
alex.forward(75)            # complete the second side of a rectangle
```

Playing the Overall Tour

Practice Problems

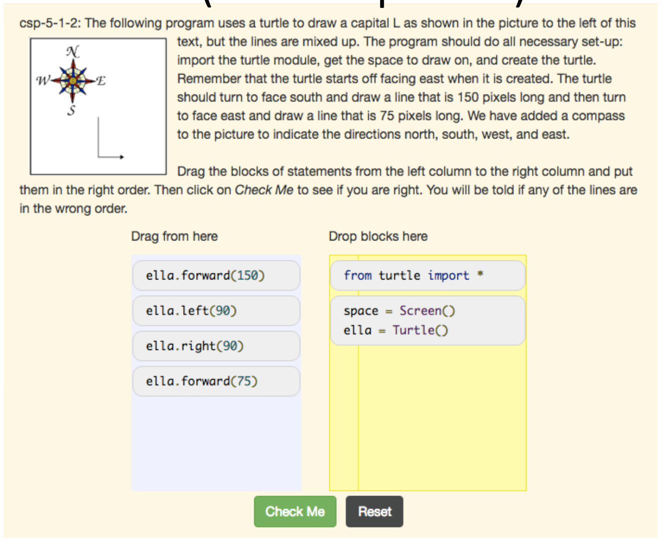
Multiple Choice with Multiple Feedback



csp-5-6-1: If you wanted to make both of the turtle lines in that last program the same length, what change would you make to the program? (Feel free to actually make the change in the program and click *Run* to try it!)

- (A) Change the 150 to 90
- (B) Change the 75 to 90
- (C) Change the 75 to 150

Parsons Problem (Mixed Up Code)



csp-5-1-2: The following program uses a turtle to draw a capital L as shown in the picture to the left of this text, but the lines are mixed up. The program should do all necessary set-up: import the turtle module, get the space to draw on, and create the turtle. Remember that the turtle starts off facing east when it is created. The turtle should turn to face south and draw a line that is 150 pixels long and then turn to face east and draw a line that is 75 pixels long. We have added a compass to the picture to indicate the directions north, south, west, and east.

Drag the blocks of statements from the left column to the right column and put them in the right order. Then click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order.

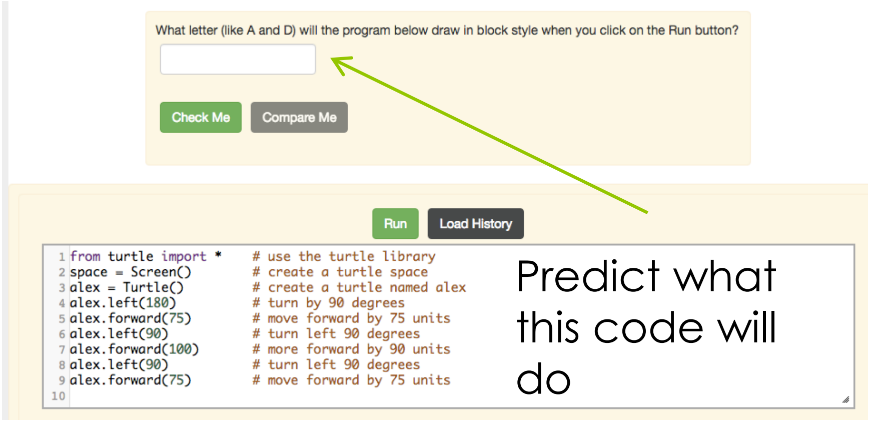
Drag from here

```
ella.forward(150)
```
```
ella.left(90)
```
```
ella.right(90)
```
```
ella.forward(75)
```

Drop blocks here

```
from turtle import *
```
```
space = Screen()
ella = Turtle()
```

Fill in the Blank

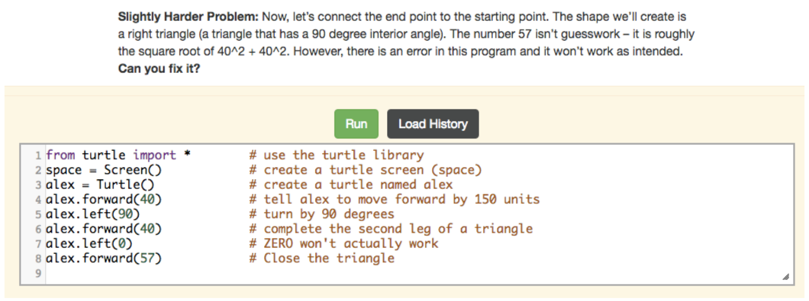


What letter (like A and D) will the program below draw in block style when you click on the Run button?

```
from turtle import *       # use the turtle library
space = Screen()           # create a turtle space
alex = Turtle()            # create a turtle named alex
alex.left(180)             # turn by 90 degrees
alex.forward(75)           # move forward by 75 units
alex.left(90)              # turn left 90 degrees
alex.forward(100)          # more forward by 90 units
alex.left(90)              # turn left 90 degrees
alex.forward(75)           # move forward by 75 units
```

Predict what this code will do

Edit Code Problem



Slightly Harder Problem: Now, let's connect the end point to the starting point. The shape we'll create is a right triangle (a triangle that has a 90 degree interior angle). The number 57 isn't guesswork – it is roughly the square root of 40^2 + 40^2. However, there is an error in this program and it won't work as intended. **Can you fix it?**

```
from turtle import *         # use the turtle library
space = Screen()             # create a turtle screen (space)
alex = Turtle()              # create a turtle named alex
alex.forward(40)             # tell alex to move forward by 150 units
alex.left(90)                # turn by 90 degrees
alex.forward(40)             # complete the second leg of a triangle
alex.left(0)                 # ZERO won't actually work
alex.forward(57)             # Close the triangle
```

Ebook Research Studies

- Teacher Observations
  - Think aloud
- Log File Analyses
  - Both student and teacher use of the ebooks
- Usability Study
  - CS Circles, Zyante, Runestone
- Teacher Studies
  - Pilot Study with 10 Teachers
  - Larger Study with 130 teachers
  - Online feedback
  - Interviews with teachers

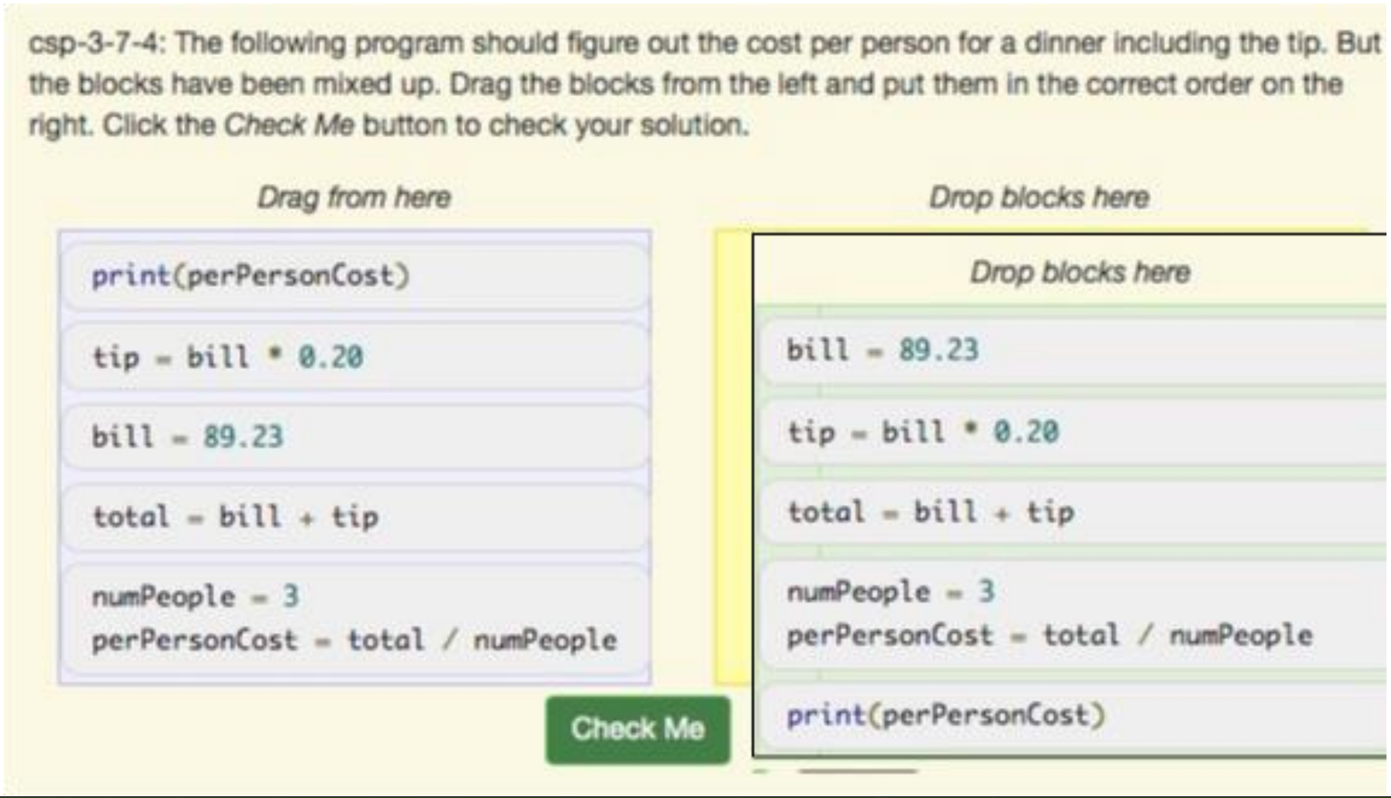
Design Principles for Teacher Ebooks

- Use worked examples plus practice
  - With subgoal labels
- Use low-cognitive load practice problems
  - Multiple-choice, fill in the blank, Parsons problems,
- Provide lots of content and associated materials
- Provide features to save teacher's time
  - Small sections and a bookmark
- Provide answers

Ericson, Moore, Morrison, Guzdial, ICER, 2015
Ericson, Rogers, Parker, Morrison, Guzdial, ICER, 2016

Parsons Problems

- Completion task
  - Place mixed up code blocks in order



csp-3-7-4: The following program should figure out the cost per person for a dinner including the tip. But the blocks have been mixed up. Drag the blocks from the left and put them in the correct order on the right. Click the *Check Me* button to check your solution.

Drag from here

```
print(perPersonCost)
```
```
tip = bill * 0.20
```
```
bill = 89.23
```
```
total = bill + tip
```
```
numPeople = 3
perPersonCost = total / numPeople
```

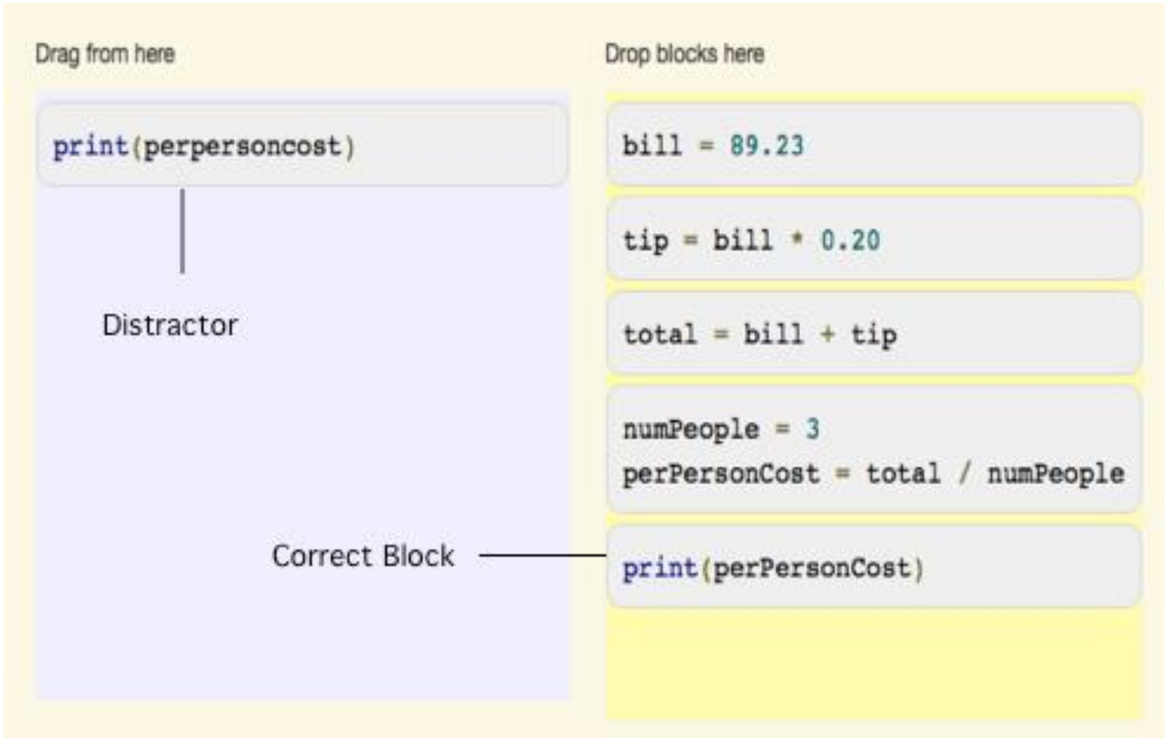
Drop blocks here

Drop blocks here

```
bill = 89.23
```
```
tip = bill * 0.20
```
```
total = bill + tip
```
```
numPeople = 3
perPersonCost = total / numPeople
```
```
print(perPersonCost)
```

Distractors

- Incorrect code blocks not needed in a solution



```
print(perpersoncost)
```

```
bill = 89.23
```

```
tip = bill * 0.20
```

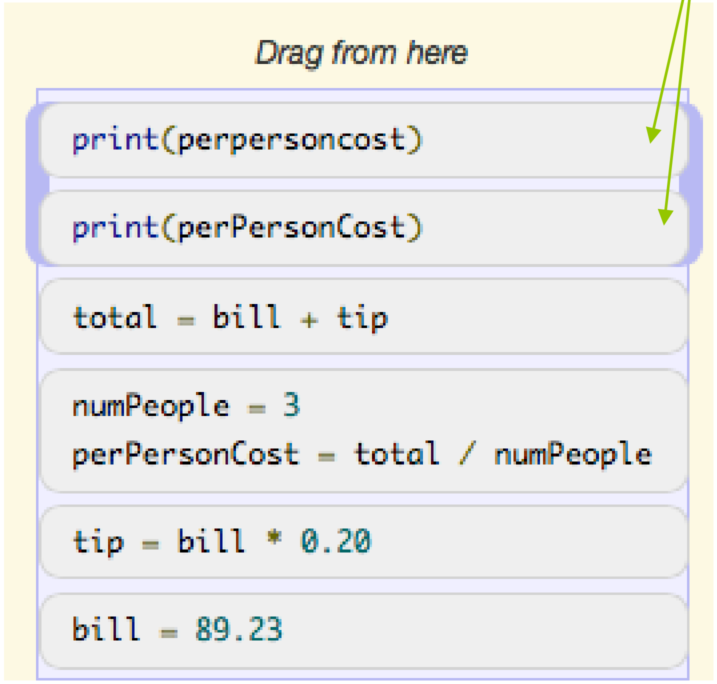
```
total = bill + tip
```

```
numPeople = 3
perPersonCost = total / numPeople
```

```
print(perPersonCost)
```

Distractor Types

Paired – shown above or below the correct block



Drag from here

```
print(perpersoncost)
```

```
print(perPersonCost)
```

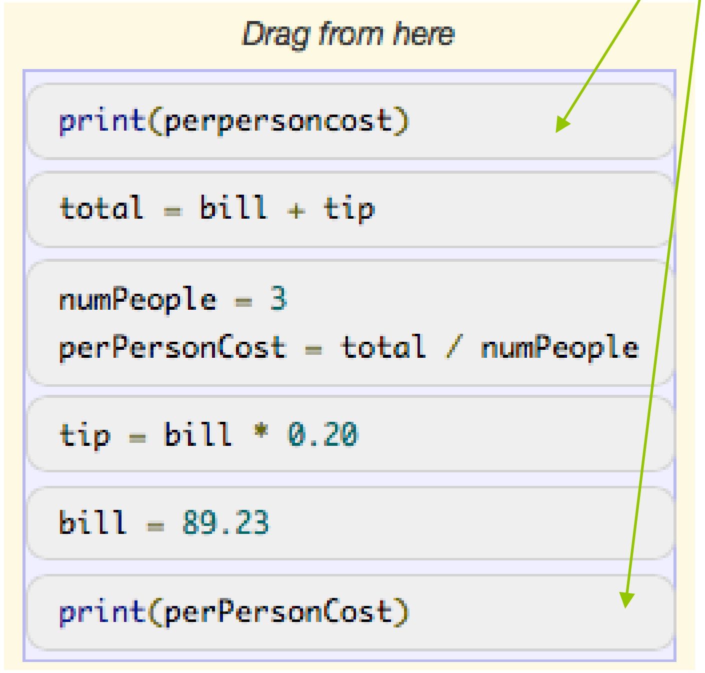
```
total = bill + tip
```

```
numPeople = 3
perPersonCost = total / numPeople
```

```
tip = bill * 0.20
```

```
bill = 89.23
```

Unpaired – randomly mixed in with the correct blocks



Drag from here

```
print(perpersoncost)
```

```
total = bill + tip
```

```
numPeople = 3
perPersonCost = total / numPeople
```

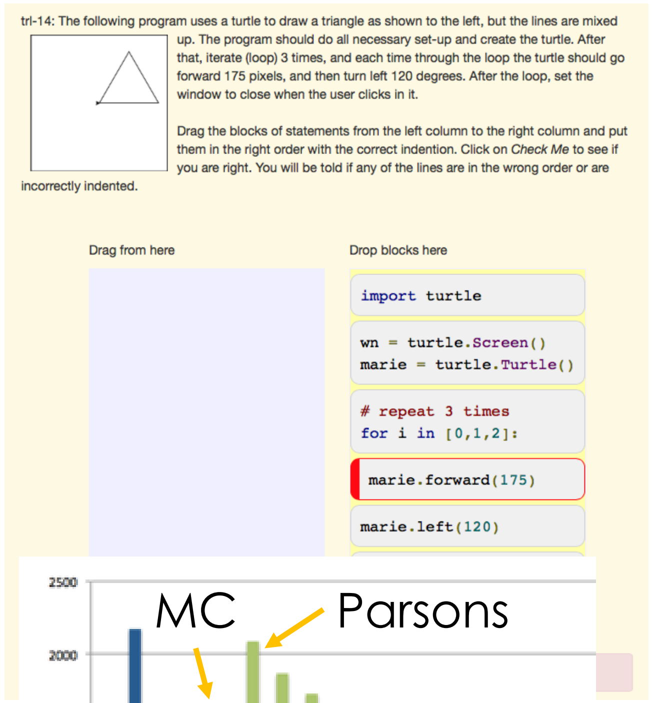
```
tip = bill * 0.20
```

```
bill = 89.23
```

```
print(perPersonCost)
```

Initial Investigations

- Four teachers working through 11 Parsons problems (no distractors)
  - Parsons problems interesting, but too easy
- Log file analysis of students
  - Some students struggled (> 100 attempts), and some gave up
  - More students attempted to solve the Parsons problems than nearby multiple-choice questions

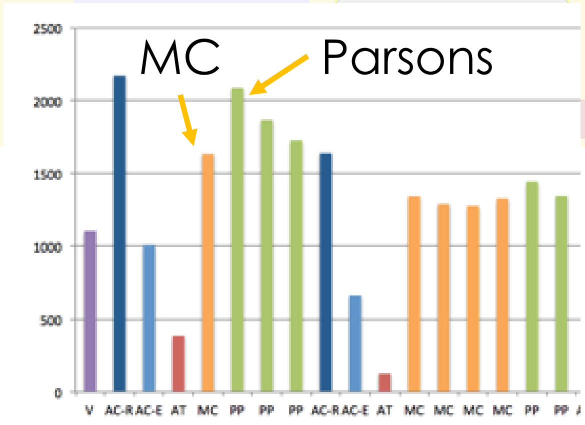

trl-14: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 175 pixels, and then turn left 120 degrees. After the loop, set the window to close when the user clicks in it.

Drag the blocks of statements from the left column to the right column and put them in the right order with the correct indention. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are incorrectly indented.

Drag from here

Drop blocks here

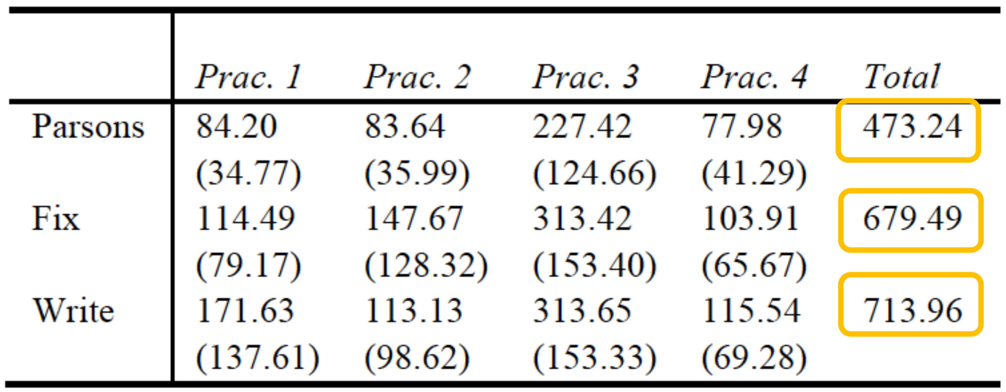
```
import turtle

wn = turtle.Screen()
marie = turtle.Turtle()

# repeat 3 times
for i in [0,1,2]:

 marie.forward(175)

marie.left(120)
```



Efficiency vs Write and Fix

- The Parsons condition completed the four practice problems ***significantly faster*** – no significant difference between fix and write

Table 1: Mean Time in Seconds (and Standard Deviation) to Complete each Practice Problem by Condition



	Prac. 1	*Prac. 2*	*Prac. 3*	*Prac. 4*	*Total*
Parsons	84.20	83.64	227.42	77.98	473.24
	(34.77)	(35.99)	(124.66)	(41.29)	
Fix	114.49	147.67	313.42	103.91	679.49
	(79.17)	(128.32)	(153.40)	(65.67)	
Write	171.63	113.13	313.65	115.54	713.96
	(137.61)	(98.62)	(153.33)	(69.28)	

ANOVA F(2,133) = 10.835, p < 0.001

LSD post-hoc test
Parsons < Fix p < 0.001
Parsons < Write p < 0.001

Learning Performance

- ***Significant difference*** from pretest to immediate posttest
  - Fix Code (*p* = *.001*)
  - Write Code (*p* =*.018*)
- Not for multiple-choice or Parsons
  - Lack of feedback on multiple-choice
  - Ceiling effect on Parsons
- No significant difference in performance gains based on the condition

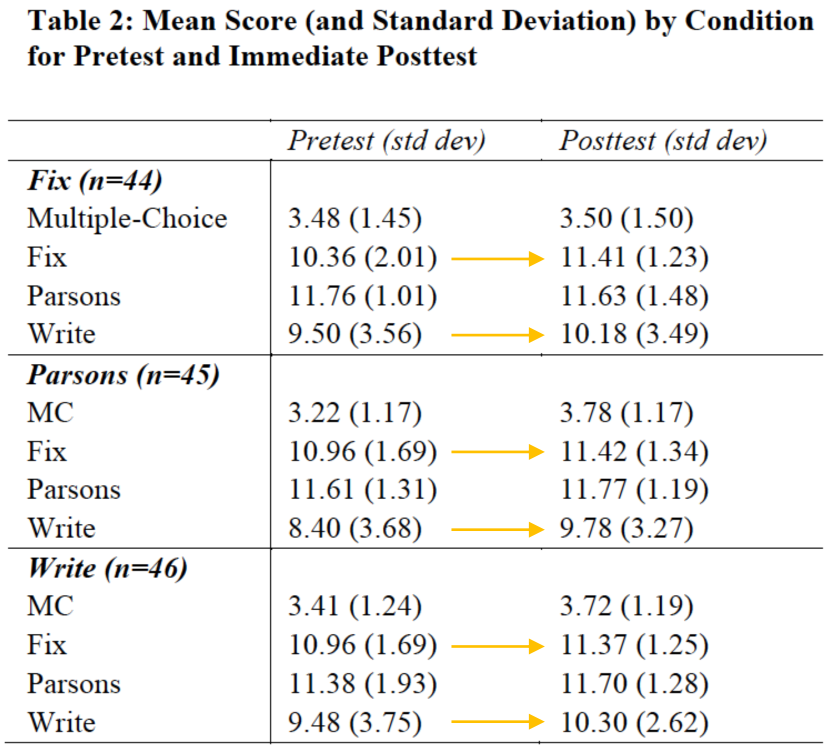
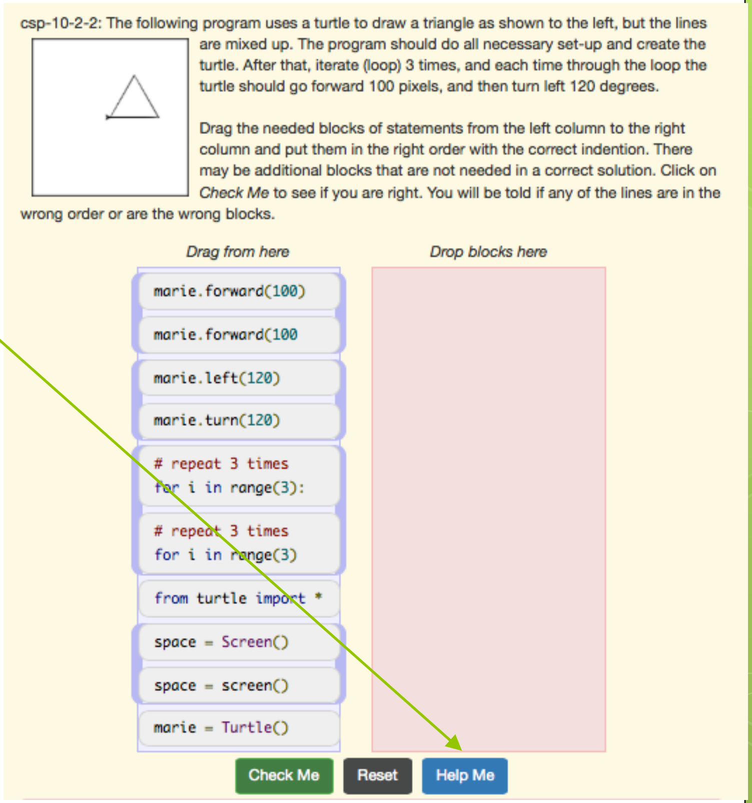


Table 2: Mean Score (and Standard Deviation) by Condition for Pretest and Immediate Posttest

	Pretest (std dev)		*Posttest (std dev)*
Fix (n=44)			
Multiple-Choice	3.48 (1.45)		3.50 (1.50)
Fix	10.36 (2.01)	→	11.41 (1.23)
Parsons	11.76 (1.01)		11.63 (1.48)
Write	9.50 (3.56)	→	10.18 (3.49)
Parsons (n=45)			
MC	3.22 (1.17)		3.78 (1.17)
Fix	10.96 (1.69)	→	11.42 (1.34)
Parsons	11.61 (1.31)		11.77 (1.19)
Write	8.40 (3.68)	→	9.78 (3.27)
Write (n=46)			
MC	3.41 (1.24)		3.72 (1.19)
Fix	10.96 (1.69)	→	11.37 (1.25)
Parsons	11.38 (1.93)		11.70 (1.28)
Write	9.48 (3.75)	→	10.30 (2.62)

Adaptive Parsons Problems

- Intra-problem adaptation
  - Make the current problem easier if the learner clicks on the *Help Me* button *after* 3 full attempts
- Inter-problem adaptation
  - Make the *next* problem easier or harder depending on the performance on the *current* problem


csp-10-2-2: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 100 pixels, and then turn left 120 degrees.

Drag the needed blocks of statements from the left column to the right column and put them in the right order with the correct indention. There may be additional blocks that are not needed in a correct solution. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

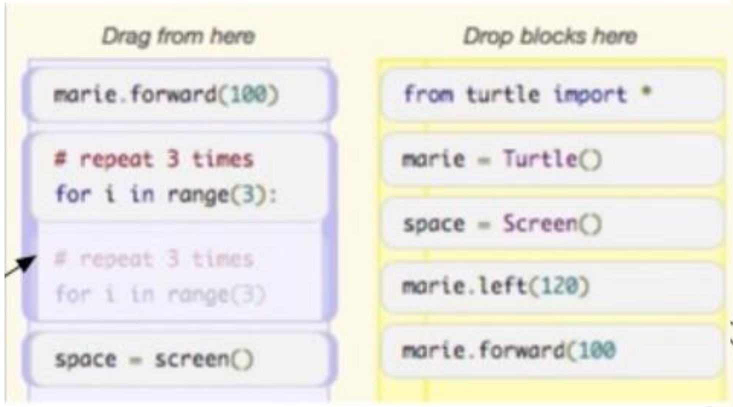
Drag from here *Drop blocks here*

```
marie.forward(100)

marie.forward(100

marie.left(120)

marie.turn(120)

# repeat 3 times
for i in range(3):

# repeat 3 times
for i in range(3)

from turtle import *

space = Screen()

space = screen()

marie = Turtle()
```

Intra-Problem Adaptation

- To make the current problem easier
  - Disable a distractor
  - Provide indentation
  - Combine two blocks –till 3 blocks remain

Disable a distractor



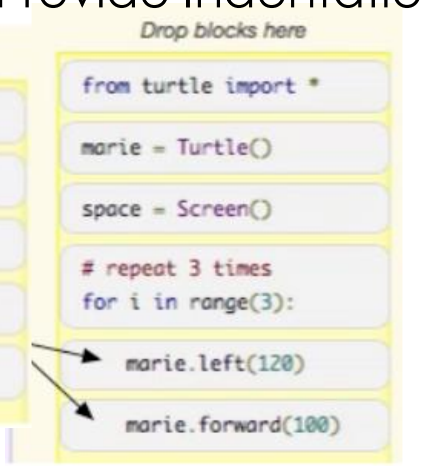
Drag from here

```
marie.forward(100)

# repeat 3 times
for i in range(3):

# repeat 3 times
for i in range(3)

space = screen()
```

Drop blocks here

```
from turtle import *

marie = Turtle()

space = Screen()

marie.left(120)

marie.forward(100
```

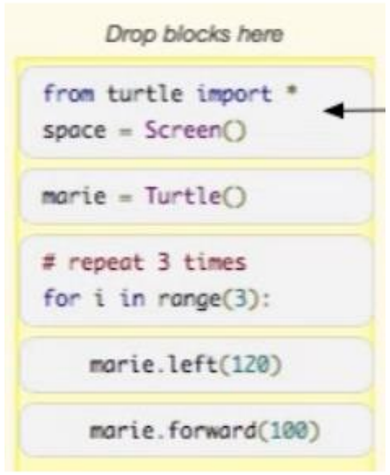
Provide Indentation



Drop blocks here

```
from turtle import *

marie = Turtle()

space = Screen()

# repeat 3 times
for i in range(3):

    marie.left(120)

    marie.forward(100)
```

Combine Blocks



Drop blocks here

```
from turtle import *
space = Screen()

marie = Turtle()

# repeat 3 times
for i in range(3):

    marie.left(120)

    marie.forward(100)
```

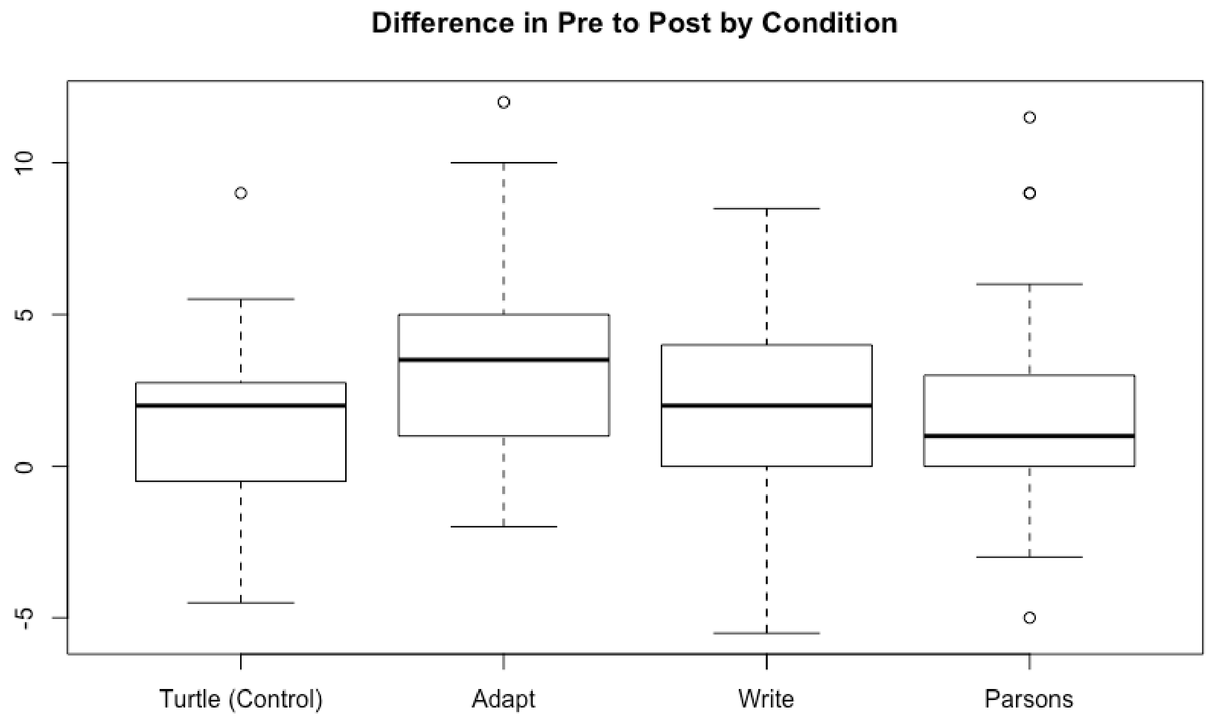
Inter-Problem Adaptation

- Modify the difficulty of the *next* problem
  - Solved last in 1 attempt
    - Make next harder: unpair distractors
  - Solved last in 4-5 attempts
    - pair distractors
  - Solved last in 6-7 attempts
    - Remove 50% of distractors and show remaining paired
  - Solved last in 8+
    - remove all distractors

Results

Learners who solve on-task adaptive and non-adaptive Parsons problems will finish the instructional problems ***significantly faster*** than the learners who write code.	Supported
Learners who solve adaptive Parsons problems with distractors will achieve ***similar learning gains*** from pretest to posttest than learners who solve non-adaptive Parsons problems or learners who write code.	Supported
Learners who solve off-task adaptive Parsons problems (the control group) will have ***lower learning gains*** than those who solve on-task problems.	Mixed

A Mann-Whitney U Test – 27 randomly picked from each condition (*p* = *.007882*) for Control vs Adaptive



Rise Up 4 CS

- Help more underrepresented students succeed in Advanced Placement Computer Science
- Attract more underrepresented students to computing careers
  - Self-efficacy
  - Interest in computing
  - Near-peer role models
  - Community of practice

Bandura, Albert, 1997. Self-efficacy: The exercise of control. Macmillan.

One Student's Story – Spring 2013

Project Rise UP 4 CS gave me the direction I wanted to take in life. I had no interest in CS, but taking the program I desired to take a computer related field. The program helped me out because I learned the basic concepts of coding that allowed me to succeed in my coding class.

Alumni Survey Results

- 79% of the respondents in college have taken more computing courses
- 63% of the respondents in college *are majoring* in computing
- 62% of the respondents in high school *intend to major* in computing
- 61% of respondents said the project *increased their interest* in computing
- 24% said Rise Up or Sisters Rise Up *changed their major* to computing

"I was planning to go into prosthetic research with Biomedical Engineering, but I realized that Computer Science is much more fitting for me."

Future Work

- Multi-institution study of Parsons problems
- Peer instruction with Parsons problems
  - Support groups
- Scale Rise Up 4 CS
- Allow easier reuse of content from ebook
- Add machine learning to identify struggling students
- Integrate Scratch
  - 3rd and 4th grade math - fractions