Why do students think they're bad at programming? Understanding and Supporting Motivation and Learning in Introductory Computer Science (CS1)

Nell O'Rourke
Assistant Professor
Computer Science + Learning Sciences
Northwestern University

How do novice undergraduates approach learning to program?

How do novice undergraduates approach learning to program?

Goal: Explore, try, fail, discuss, read, practice, learn

Reality: Avoid errors, over/under utilize help, panic, flail

What drives novice students' approaches?
How does motivation come into play?

Motivational Context

- Computer science is a new field for most undergraduates
- Developing computer science identities
- Deciding whether or not to major
- Students consider their own assessments of programming ability when deciding whether or not to major [Lewis et al. 2011]
- They assess their own ability using previous performance, speed, and grades [Lewis et al. 2011]

Novices self-assess frequently

"I feel like ***I should remember the syntax*** *for basic things, such as lists, and both C++ and Python more closely than I currently do."*

"I'm particularly proud of that [assignment] because ***I was able to figure out most of that on my own****, and I didn't need as much TA help as I had anticipated."*

"That [assignment] wasn't too challenging, I guess… ***I got it done pretty quickly without that many errors****. So I was pretty happy with myself."*

How do novices define and assess programming ability?

Survey study of CS1 students

- 103 CS1 students
- Open-ended question:
  - *When watching someone program, how do you know if they are good at programming?*
  - Open coding to identify emergent themes

J. Gorson, E. O'Rourke "How do students talk about intelligence? An investigation of motivation, self-efficacy, and mindsets in computer science." ICER 2019.

When watching someone program, how do you know if they're good at programming?

Speed is important: *"If they can complete an assignment relatively quickly"*

Ease of debugging is important: *"They understand how to debug a program quickly based off of first glance"*

Knowing syntax is important: *"They know various functions like the back of their hand"*

J. Gorson, E. O'Rourke "How do students talk about intelligence? An investigation of motivation, self-efficacy, and mindsets in computer science." ICER 2019.

When watching someone program, how do you know if they’re good at programming?

Thinking and planning is good : *“They are able to plan out and structure their thoughts on how to approach the code before writing it”*

Thinking and planning is bad: *“They keep typing and don’t have to sit there and think”*

Novices may not agree on self-assessment criteria

J. Gorson, E. O’Rourke “How do students talk about intelligence? An investigation of motivation, self-efficacy, and mindsets in computer science.” ICER 2019.

Do students use a consistent set of self-assessment criteria?

Are there differences across university contexts?

Is there a relationship between self-assessments and student self-efficacy?

Identified a list of 13 self-assessment moments

- Getting a simple error
- Starting over
- Not understanding an error message
- Stopping programming to plan
- Getting help from others
- Spending a long time on a problem
- Not knowing how to start
- Using resources to look up syntax
- Spending time planning at the beginning
- Spending a long time looking for a simple error
- Struggling to fix errors
- Not able to finish in time expected
- Does not understand the problem statement

Survey Approach: Self-Assessment Vignettes

Arjun is working on a programming problem. He can't remember the syntax. He uses Google to look up the syntax. He is disappointed that he could not remember the syntax on his own.

- *I feel like I'm not doing well on a problem when I can't remember the syntax and have to look it up.*

Randomized question order, character name, and character gender for each participant

J. Gorson & E. O'Rourke "Why do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities." In ICER 2020

Survey Study Methods

Five self-efficacy questions, e.g.:

- *“Even if the work is hard in CS1, I can learn it”*
- *“I'm certain I can master the skills taught in my CS 1 class this term”*

Demographic Questions

Full survey available at: https://bit.ly/2B7irzC

J. Gorson & E. O’Rourke “Why do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities.” In ICER 2020

Survey Study Methods

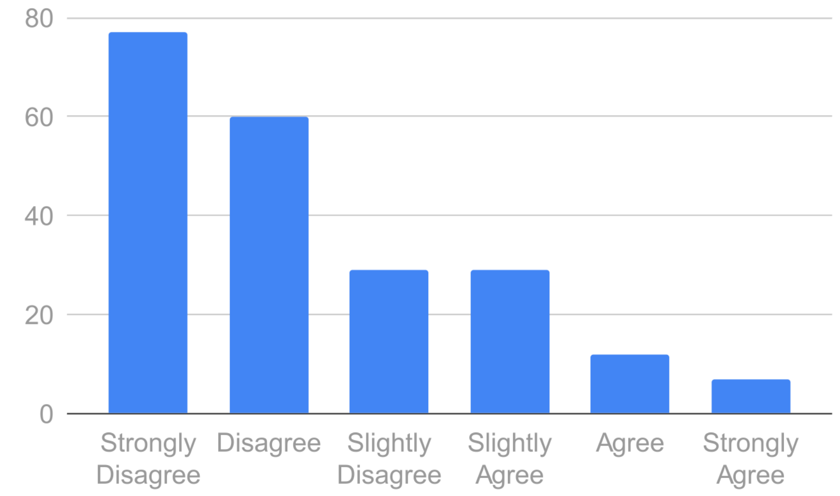
Participants

- 283 CS1 and CS2 students
- Three different universities in the Chicago metro area

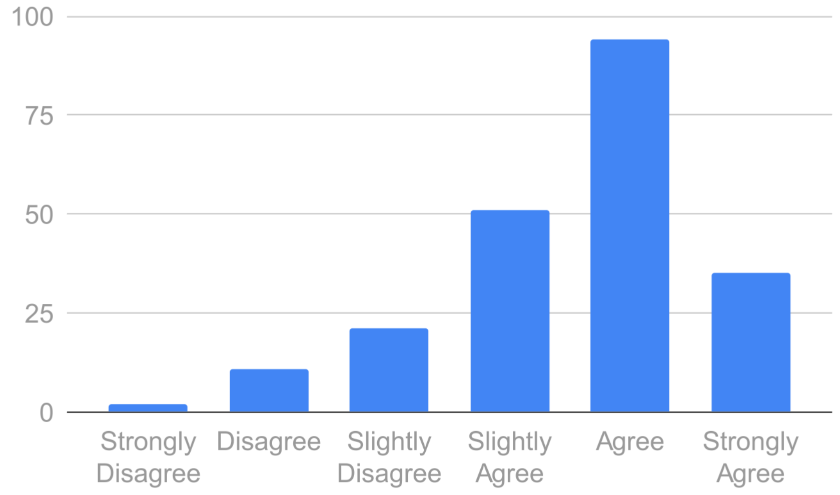
Selectivity	# Ugrads	# Subjects	% Female	% African American	% Asian	% Latin American	% White	% Other	% 2 or more
Highly selective	8,200	78	58%	6%	35%	4%	42%	4%	8%
Selective	14,500	57	32%	7%	23%	10%	47%	5%	5%
Less selective	6,400	78	17%	7%	24%	31%	24%	8%	6%

J. Gorson & E. O’Rourke “Why do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities.” In ICER 2020

Do students use a consistent set of self-assessment criteria?

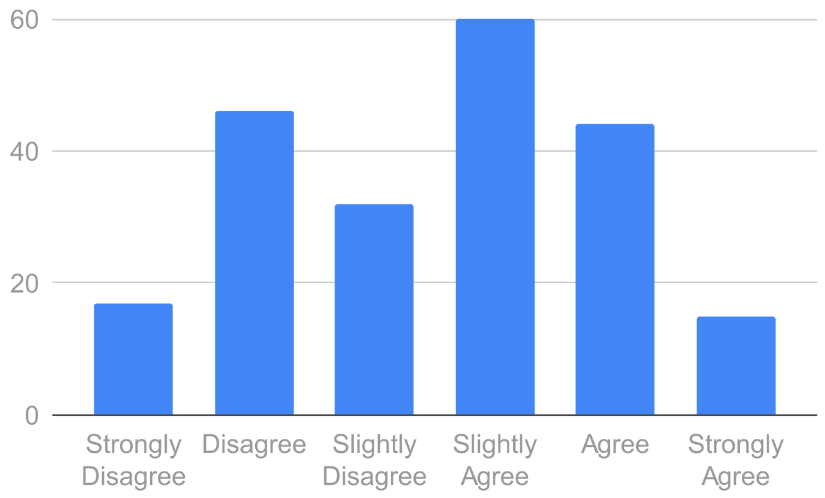


Isabella is working on a challenging problem. She runs into an error. She looks through the code but can't find it. After a long time, she realizes that it was a small typo. She thinks to herself: "Wow. I am so bad at programming. A good programmer would not take so long to find a simple error"

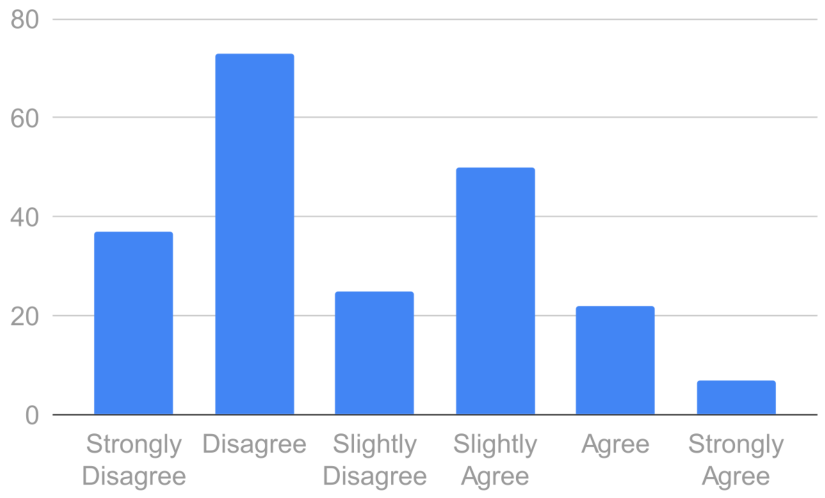


Miguel reads his programming homework assignment. He opens up the editor but has no idea where to start. Miguel feels disappointed in himself because he doesn't even know how to approach the problem.

Do students use a consistent set of self-assessment criteria?



Nadia is working on a hard homework problem. She plans out a solution. She writes a few lines of code. She realizes that her approach to the problem will not work. She decides to start over. Nadia feels frustrated that she wasted time. She erases all her code and starts again.



Tamyra is working really hard on a programming problem. She solves the problem. She is proud of herself. Tamyra looks at the clock and realizes how many hours she spent on the problem. She feels upset because it took her so long to finish it.

Do students use a consistent set of self-assessment criteria?

Self-assessment moment	% Students who report that they negatively self-assess
Getting a simple error	22%
Starting over	56%
Not understanding an error message	62%
Stopping programming to plan	16%
Getting help from others	29%
Spending a long time on a problem	37%
Not knowing how to start	84%
Using resources to look up syntax	30%
Spending time planning at the beginning	18%
Spending a long time looking for a simple error	35%
Struggling to fix errors	60%
Not able to finish in time expected	50%
Does not understand the problem statement	79%

Do students use a consistent set of self-assessment criteria?

Self-assessment moment	% Students who report that they negatively self-assess
Getting a simple error	22%
Starting over	56%
Not understanding an error message	62%
Stopping programming to plan	16%
Getting help from others	29%
Spending a long time on a problem	37%
Not knowing how to start	84%
Using resources to look up syntax	30%
Spending time planning at the beginning	18%
Spending a long time looking for a simple error	35%
Struggling to fix errors	60%
Not able to finish in time expected	50%
Does not understand the problem statement	79%

How do students think about the vignettes?

- Follow-up interviews with 13 students
- Showed them their survey results and asked them to explain their reasoning

"[The question was] kinda cool for me because Diego had to get help from the instructor and I often have to get help from the TAs a lot and I don't feel like I'm doing bad. Like I feel like I'm just improving myself."

Are there differences in student self-assessments at different universities?

Self-assessment moment	Kruskal Wallis test comparing responses by university
Getting a simple error	H(2) = 0.27, p = 0.88
Starting over	H(2) = 1.33 , p = 0.52
Not understanding an error message	H(2) = 0.37, p = 0.83
Stopping programming to plan	H(2) = 3.86, p = 0.15
Getting help from others	H(2) = 4.83, p =0.09
Spending a long time on a problem	H(2) = 1.03, p = 0.60
Not knowing how to start	H(2) = 2.50, p = 0.29
Using resources to look up syntax	H(2) = 1.23, p = 0.54
Spending time planning at the beginning	H(2) = 4.47, p = 0.11
Spending a long time looking for a simple error	H(2) = 4.05, p = 0.13
Struggling to fix errors	H(2) = 1.83, p = 0.40
Not able to finish in time expected	H(2) = 10.16, p = 0.006
Does not understand the problem statement	H(2) = 1.17, p = 0.56

Is there a relationship between self-assessments and student self-efficacy?

Computed a self-assessment compound score (average of all 13 vignette Likert-scale responses)

Computed a self-efficacy compound score (average of all 5 Likert-scale responses)

Significant correlation between self-assessment compound score and self-efficacy compound score (r_s (212) = -0.418, $p < 0.001$)

Students who negatively self-assess more frequently and strongly have a lower self-efficacy on average

Where do these self-assessments come from?

Interview participants cited a few different sources for their responses and reasoning

More experienced programmers: *“I [think] that programmers need to know every single little piece of syntax . . . so the fact that I don’t have it memorized just made me feel bad.”*

Peers: *“Yeah I definitely feel badly on a problem if I don’t know where to start. Just like I said before, just seeing everyone else around me being able to solve it. I don’t know if I’m just struggling and everyone else is breezing through it, then I feel badly about myself and I feel like I’m not as smart as everyone else.”*

Instructors: *“I put disagree, mainly because of our professors. They always tell us that planning should be first and once you have your plan then you start coding.”*

How do novice undergraduates approach learning to program?

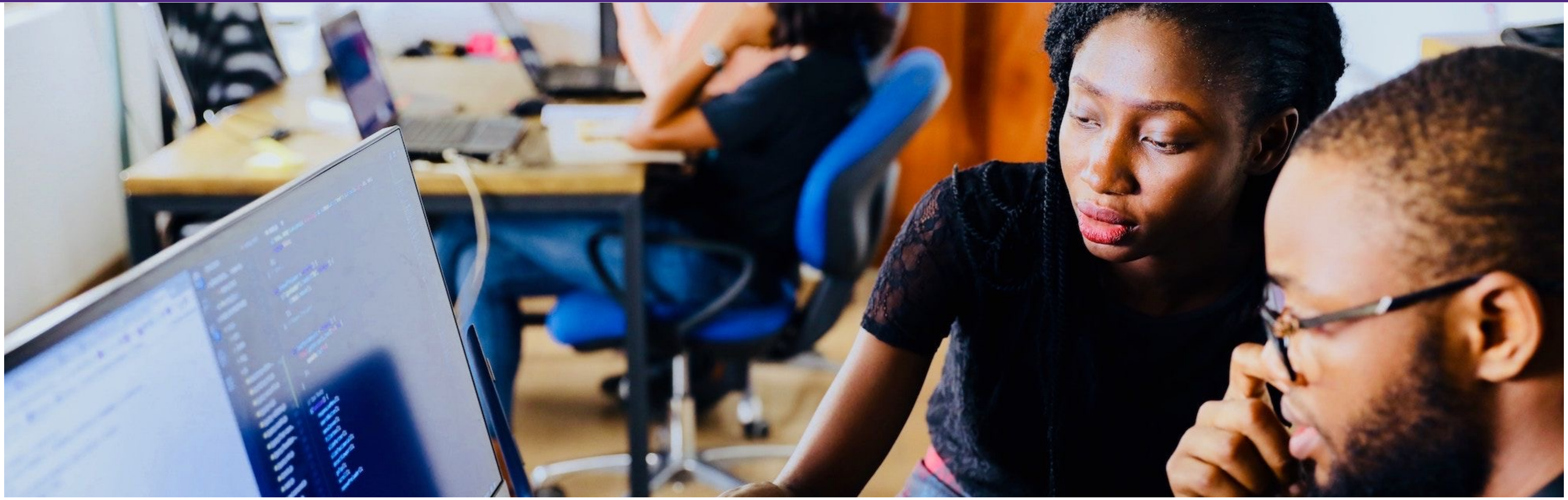
Novices develop their own values and expectations.

These may not align with instructor values.

How do these beliefs shape learning experiences?

Example Domain: Pair Programming

Pair Programming

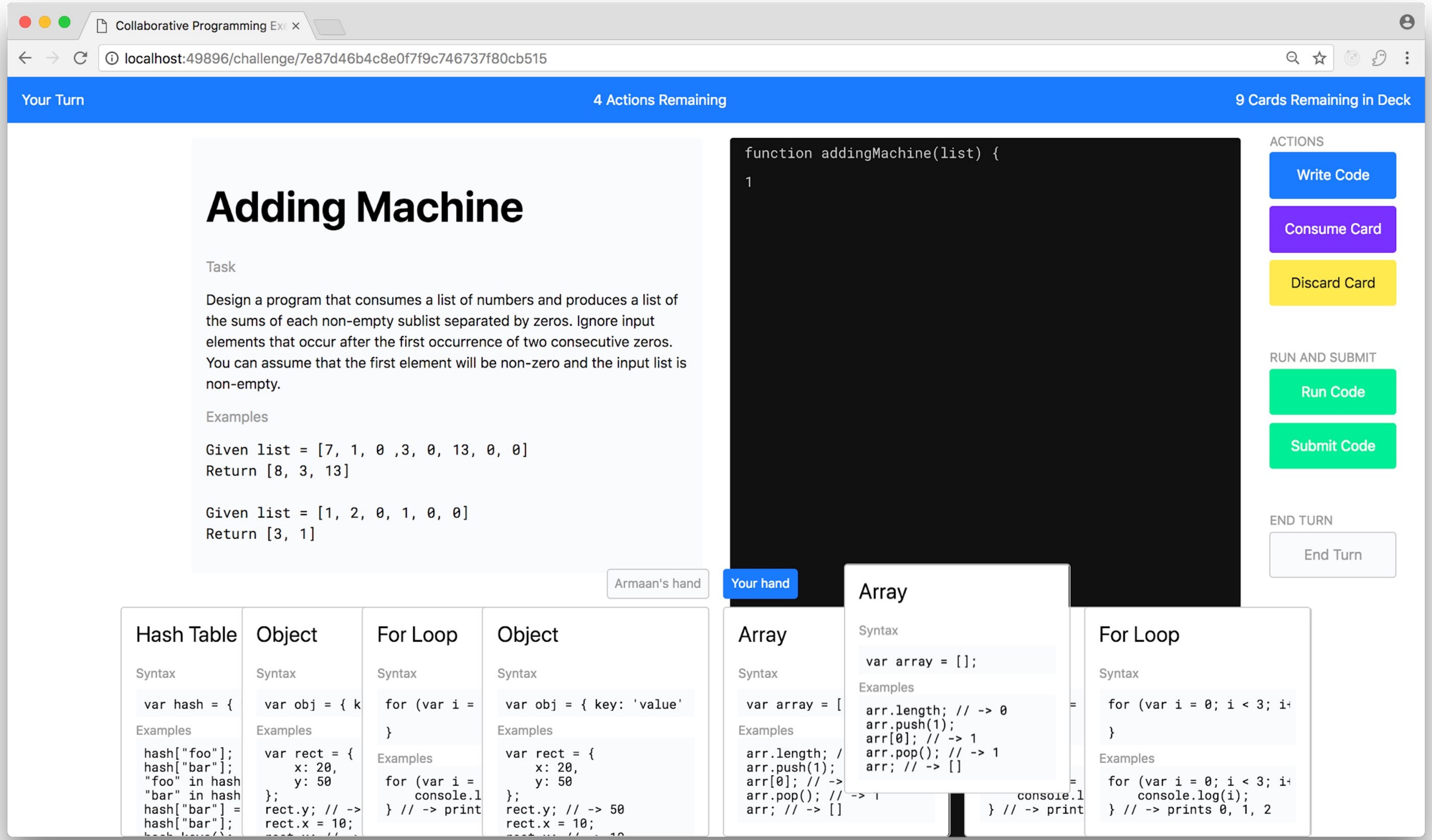


Goal: Explain thinking, compare approaches, co-regulate

Reality: Divide work, pilot dominates, co-pilot checks syntax

How can we encourage pairs to collaborate more equally and discuss their solution plans more deeply?

Pyrus: A Collaborative Programming Game



Adding Machine

Task

Design a program that consumes a list of numbers and produces a list of the sums of each non-empty sublist separated by zeros. Ignore input elements that occur after the first occurrence of two consecutive zeros. You can assume that the first element will be non-zero and the input list is non-empty.

Examples

```
Given list = [7, 1, 0 ,3, 0, 13, 0, 0]
Return [8, 3, 13]

Given list = [1, 2, 0, 1, 0, 0]
Return [3, 1]
```

```
function addingMachine(list) {
```

Hash Table

Syntax

```
var hash = {
```

Examples

```
hash["foo"];
hash["bar"];
"foo" in hash
"bar" in hash
hash["bar"] =
hash["bar"];
```

Object

Syntax

```
var obj = { k
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // ->
rect.x = 10;
```

For Loop

Syntax

```
for (var i =

}
```

Examples

```
for (var i =
    console.l
} // -> print
```

Object

Syntax

```
var obj = { key: 'value'
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // -> 50
rect.x = 10;
```

Array

Syntax

```
var array = [
```

Examples

```
arr.length; /
arr.push(1);
arr[0]; // ->
arr.pop(); //
arr; // -> []
```

Array

Syntax

```
var array = [];
```

Examples

```
arr.length; // -> 0
arr.push(1);
arr[0]; // -> 1
arr.pop(); // -> 1
arr; // -> []
```

For Loop

Syntax

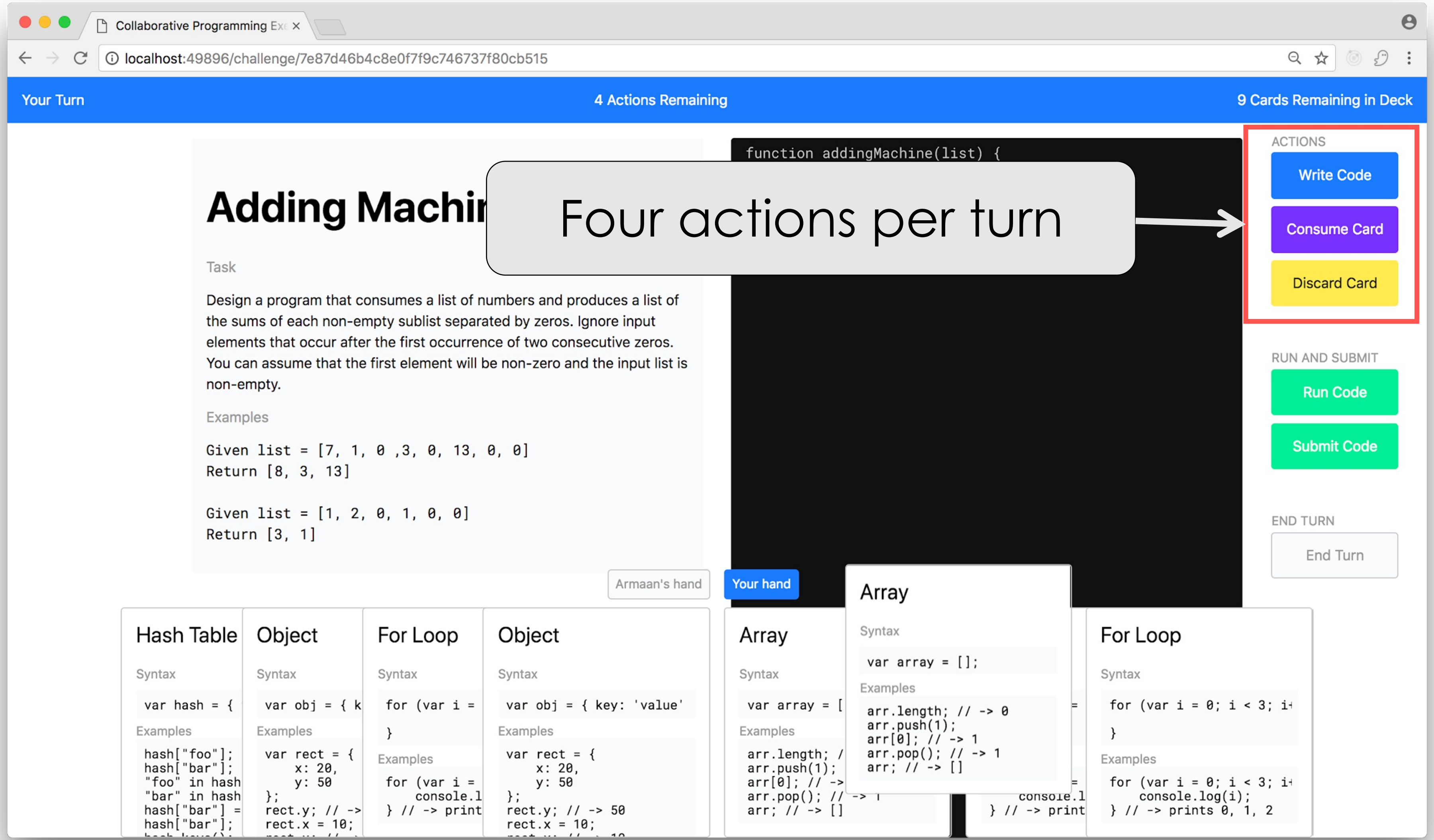
```
for (var i = 0; i < 3; i

}
```

Examples

```
for (var i = 0; i < 3; i
    console.log(i);
} // -> prints 0, 1, 2
```

Pyrus: A Collaborative Programming Game



Collaborative Programming Ex

localhost:49896/challenge/7e87d46b4c8e0f7f9c746737f80cb515

Your Turn | 4 Actions Remaining | 9 Cards Remaining in Deck

Adding Machi

Task

Design a program that consumes a list of numbers and produces a list of the sums of each non-empty sublist separated by zeros. Ignore input elements that occur after the first occurrence of two consecutive zeros. You can assume that the first element will be non-zero and the input list is non-empty.

Examples

```
Given list = [7, 1, 0 ,3, 0, 13, 0, 0]
Return [8, 3, 13]

Given list = [1, 2, 0, 1, 0, 0]
Return [3, 1]
```

```
function addingMachine(list) {
```

Four actions per turn

Hash Table

Syntax

```
var hash = {
```

Examples

```
hash["foo"];
hash["bar"];
"foo" in hash
"bar" in hash
hash["bar"] =
hash["bar"];
```

Object

Syntax

```
var obj = { k
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // ->
rect.x = 10;
```

For Loop

Syntax

```
for (var i =

}
```

Examples

```
for (var i =
    console.l
} // -> print
```

Object

Syntax

```
var obj = { key: 'value'
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // -> 50
rect.x = 10;
```

Array

Syntax

```
var array = [
```

Examples

```
arr.length; /
arr.push(1);
arr[0]; // ->
arr.pop(); //
arr; // -> []
```

Array

Syntax

```
var array = [];
```

Examples

```
arr.length; // -> 0
arr.push(1);
arr[0]; // -> 1
arr.pop(); // -> 1
arr; // -> []
```

```
    console.l
} // -> print
```

For Loop

Syntax

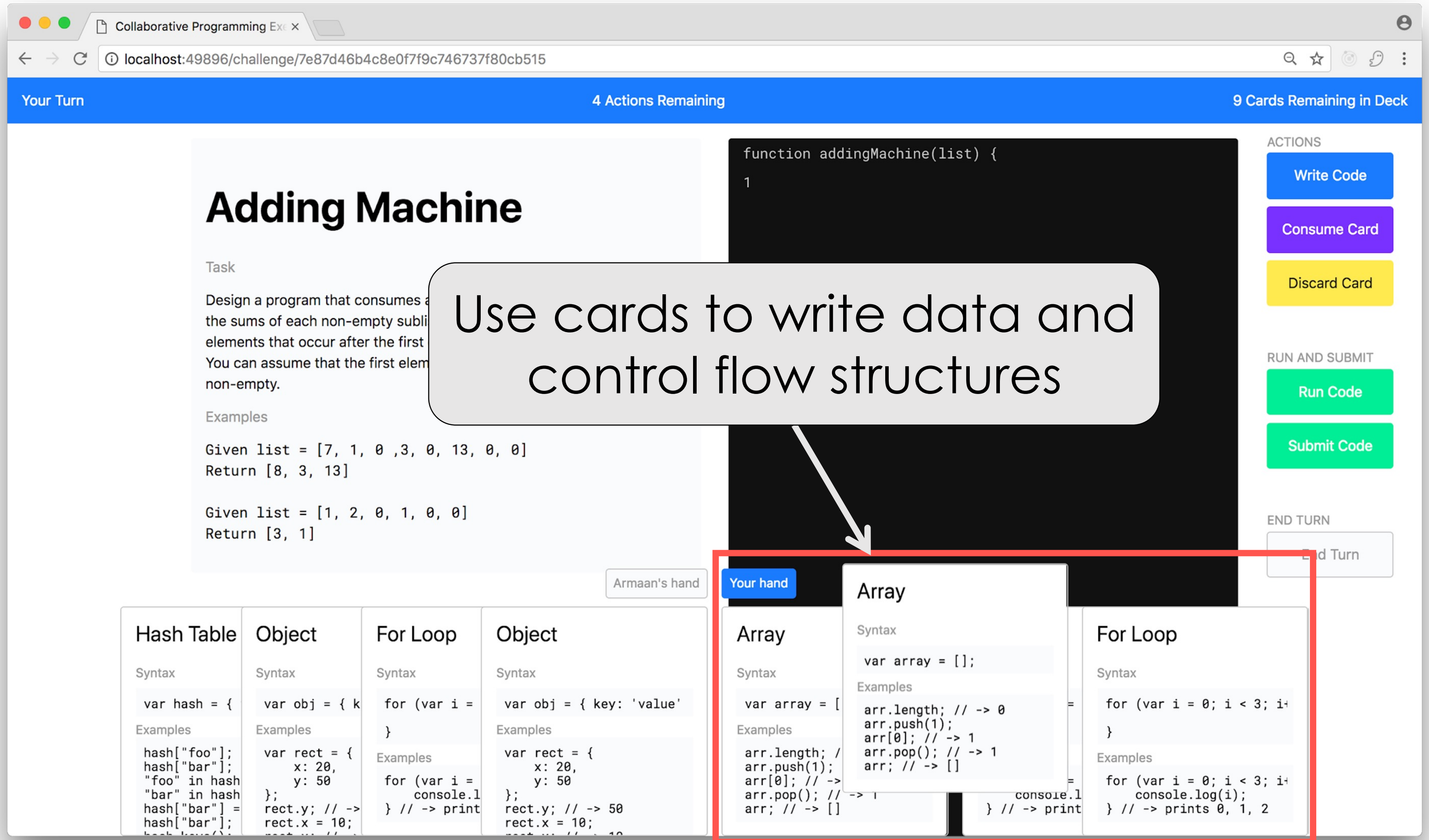
```
for (var i = 0; i < 3; i

}
```

Examples

```
for (var i = 0; i < 3; i
    console.log(i);
} // -> prints 0, 1, 2
```

Pyrus: A Collaborative Programming Game



Collaborative Programming Ex

localhost:49896/challenge/7e87d46b4c8e0f7f9c746737f80cb515

Your Turn — 4 Actions Remaining — 9 Cards Remaining in Deck

Adding Machine

Task

Design a program that consumes a
the sums of each non-empty subli
elements that occur after the first
You can assume that the first elem
non-empty.

Examples

```
Given list = [7, 1, 0 ,3, 0, 13, 0, 0]
Return [8, 3, 13]

Given list = [1, 2, 0, 1, 0, 0]
Return [3, 1]
```

```
function addingMachine(list) {
```

Hash Table

Syntax

```
var hash = {
```

Examples

```
hash["foo"];
hash["bar"];
"foo" in hash
"bar" in hash
hash["bar"] =
hash["bar"];
```

Object

Syntax

```
var obj = { k
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // ->
rect.x = 10;
```

For Loop

Syntax

```
for (var i =
}
```

Examples

```
for (var i =
    console.l
} // -> print
```

Object

Syntax

```
var obj = { key: 'value'
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // -> 50
rect.x = 10;
```

Array

Syntax

```
var array = [];
```

Examples

```
arr.length; // -> 0
arr.push(1);
arr[0]; // -> 1
arr.pop(); // -> 1
arr; // -> []
```

For Loop

Syntax

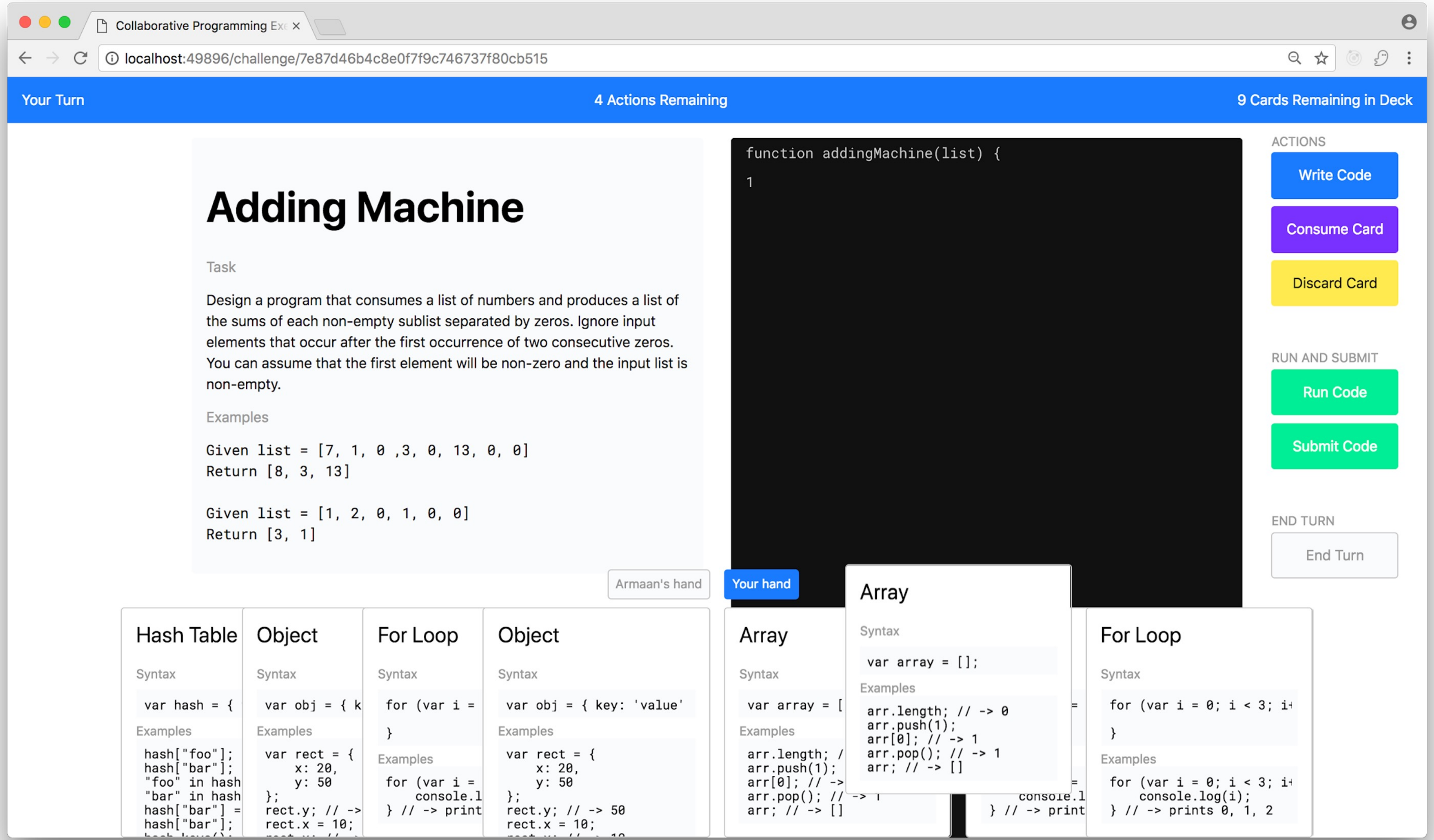
```
for (var i = 0; i < 3; i+
}
```

Examples

```
for (var i = 0; i < 3; i+
    console.log(i);
} // -> prints 0, 1, 2
```

Use cards to write data and control flow structures

Pyrus: A Collaborative Programming Game



Adding Machine

Task

Design a program that consumes a list of numbers and produces a list of the sums of each non-empty sublist separated by zeros. Ignore input elements that occur after the first occurrence of two consecutive zeros. You can assume that the first element will be non-zero and the input list is non-empty.

Examples

```
Given list = [7, 1, 0 ,3, 0, 13, 0, 0]
Return [8, 3, 13]

Given list = [1, 2, 0, 1, 0, 0]
Return [3, 1]
```

```
function addingMachine(list) {
1
```

Hash Table

Syntax

```
var hash = {
```

Examples

```
hash["foo"];
hash["bar"];
"foo" in hash
"bar" in hash
hash["bar"] =
hash["bar"];
```

Object

Syntax

```
var obj = { k
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // ->
rect.x = 10;
```

For Loop

Syntax

```
for (var i =
}
```

Examples

```
for (var i =
    console.l
} // -> print
```

Object

Syntax

```
var obj = { key: 'value'
```

Examples

```
var rect = {
    x: 20,
    y: 50
};
rect.y; // -> 50
rect.x = 10;
```

Array

Syntax

```
var array = [
```

Examples

```
arr.length; /
arr.push(1);
arr[0]; // ->
arr.pop(); // ->
arr; // -> []
```

Array

Syntax

```
var array = [];
```

Examples

```
arr.length; // -> 0
arr.push(1);
arr[0]; // -> 1
arr.pop(); // -> 1
arr; // -> []
```

```
    console.l
} // -> print
```

For Loop

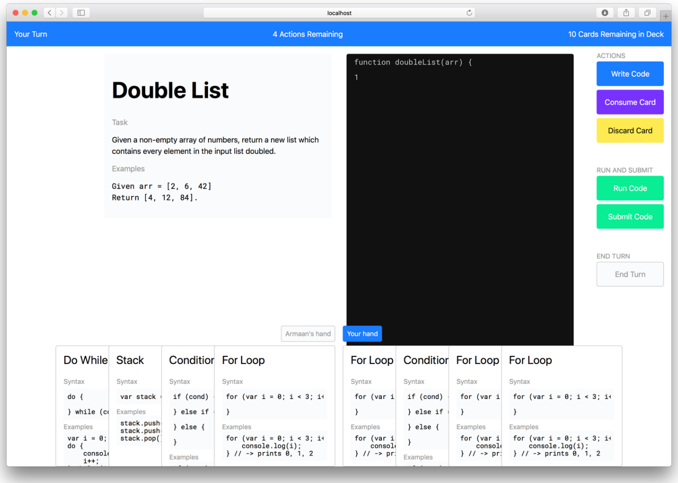
Syntax

```
for (var i = 0; i < 3; i+
}
```

Examples

```
for (var i = 0; i < 3; i+
    console.log(i);
} // -> prints 0, 1, 2
```

Do the game mechanics encourage planning and collaboration?



Double List

Task

Given a non-empty array of numbers, return a new list which contains every element in the input list doubled.

Examples

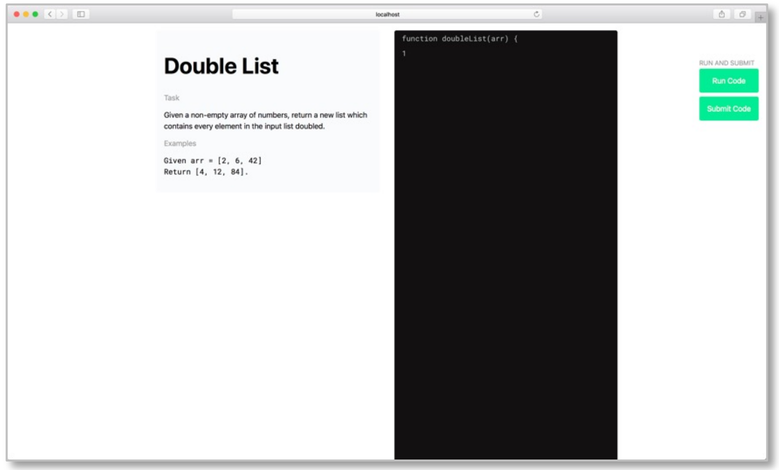
```
Given arr = [2, 6, 42]
Return [4, 12, 84].
```

```
function doubleList(arr) {
```

```
for (var i = 0; i < 3; i
}
```

```
for (var i = 0; i < 3; i
    console.log(i);
} // -> prints 0, 1, 2
```

Pyrus Condition



Double List

Task

Given a non-empty array of numbers, return a new list which contains every element in the input list doubled.

Examples

```
Given arr = [2, 6, 42]
Return [4, 12, 84].
```

```
function doubleList(arr) {
```

Pair Programming Condition

Novices spend 14 minutes planning in Pyrus on average, compared to 7 minutes in the control (F(1,18)=3.48, $p<0.001$)

Difference in time spent in pilot role is 6 minutes in Pyrus, compared to 18 minutes in the control (F(1,18)=1.17, $p<0.05$)

A. Shah, J. Shi, G. Hedman, E. O'Rourke "Pyrus: Designing A Collaborative Programming Game to Support Problem Solving Behaviors." CHI 2019.

Do the game mechanics encourage planning and collaboration?

Roles in Pair Programming: *"We each implemented our own solutions when it was our time to [be the pilot] because we thought it would be easier to do our own solutions than explain our solution to the other person."*

Roles in Pyrus: *"We talked about what we needed to do first, and then whoever's turn it was just wrote it. There wasn't much of a difference between whose turn it was."*

The game changed how students approached planning and collaboration

A. Shah, J. Shi, G. Hedman, E. O'Rourke "Pyrus: Designing A Collaborative Programming Game to Support Problem Solving Behaviors." CHI 2019.

Did students like the experience?

“I don't know. I guess I liked that it made me more deliberate and creative, I guess in the way I was doing things, but ***I didn't like that it took so much longer*** *than it would have otherwise.”*

“I think as far as being on the same page and having to explain yourself, ***it slowed things down*** *a bit.”*

“It can be a bit slower … ***you got to like slow down and now I have to like explain everything****, and just like teach so that the other person is on the same page, like, that just takes up more time.”*

A. Shah, J. Shi, G. Hedman, E. O’Rourke “Pyrus: Designing A Collaborative Programming Game to Support Problem Solving Behaviors.” CHI 2019.

How do novice undergraduates approach learning to program?

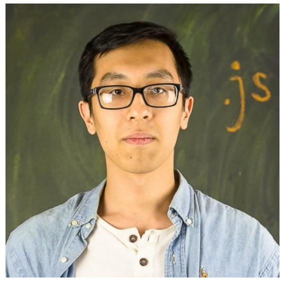
Novices may have competing goals of assessing their programming ability and learning content.

Many implications for instruction and design

Can we make changes to our instructional practices to reduce overly negative self-assessments?

Can we provide students with real-time feedback to help them reframe potential self-assessment moments while programming?

Thank You!



NSF CRII: Automatically Praising Learning Process to Promote the Growth Mindset in Computer Science

NSF CAREER: Towards Intelligent Learning Environments that Support the Practice of Programming

NSF Cyberlearning: Context-Aware Metacognitive Practice: Instrumenting Classroom Ecosystems to Help Introductory Computer Science Students Develop Effective Learning Strategies