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]
"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

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”

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

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
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
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


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

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

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
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.
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.

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?

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

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

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

Pyrus: A Collaborative Programming Game
Pyrus: A Collaborative Programming Game

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 \([3, 1]\)

### Hash Table
**Syntax**
```
var hash = {
    "foo": 1,
    "bar": 2
}
```

**Examples**
- \(x = 20;\)
- \(y = 50;\)
- \(rect.x = 10;\)

### Object
**Syntax**
```
var obj = {
    key: value
}
```

**Examples**
- \(var rect = {\}
- \(for (var i = 0; i < arr.length; i++)\)
- \(console.log(1);\)

### For Loop
**Syntax**
```
for (var i = 0; i < arr.length; i++)
```

**Examples**
- \(arr.push(1);\)
- \(arr[i];\)
- \(arr.pop();\)
- \(arr.length;\)

### Array
**Syntax**
```
var array = [ ];
```

**Examples**
- \(arr.length;\)
- \(arr.push(1);\)
- \(arr.pop();\)
- \(for (var i = 0; i < arr.length; i++)\)
  - \(console.log(1);\)

---

**Your hand**

**Array**

**Syntax**
```
var array = [ ];
```

**Examples**
- \(arr.length;\)
- \(arr.push(1);\)
- \(arr.pop();\)
- \(for (var i = 0; i < arr.length; i++)\)
  - \(console.log(1);\)

---

**For Loop**

**Syntax**
```
for (var i = 0; i < arr.length; i++)
```

**Examples**
- \(for (var i = 0; i < 3; i++)\)
  - \(console.log(1);\)
  - \(console.log(2);\)
  - \(console.log(3);\)

---

**Adding Machine**

**Function**
```
function addingMachine(list) {
    1
}
```

**Examples**
- Given list = \([7, 1, 0, 3, 0, 13, 0, 0]\)
  - Return \([8, 3, 13]\)
- Given list = \([1, 2, 0, 1, 0, 0]\)
  - Return \([3, 1]\)
Do the game mechanics encourage planning and collaboration?

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 \((F(1,18)=1.17, p<0.05)\).

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.

Did students like the experience?

“\textquote{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.”

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