
Samiha Marwan  
samarwan@ncsu.edu

Dr. Joseph Williams  
williams@cs.toronto.edu

Dr. Thomas Price  
twprice@ncsu.edu
• Imagine Jill (a student) working on his first programming homework on Snap.

• Jill feels stuck.

• It is Midnight ..
• One way to help Jill is to allow the programming environment to provide him with **automated hints**.

• **Why automated hints?**
  – Adaptive to students’ code.
  – Can scale to new problems and contexts.
    • (Price at al., 2017, Rivers et al., 2017)
  – Can improve students’ performance and learning.
    • (Corbett et al., 2001, Fossati et al., 2015)
• Jill can ask for a hint from iSnap (Price et al., 2017)
  – But there are many design choices to consider....
• Should the hint show Jill only the next step to do? (eg. Rivers et al., 2017, Watson et al., 2012)
Should the hint give some text explaining why and how Jill needs to do this step? (e.g. Marwan et al., 2019)
• Should the hint ask Jill to \textit{self-explain} the hint? (e.g. Vihavainen et al., 2015)
Motivation

• While there is some evidence that programming hints can be helpful, there are many open questions:
  – How and when is each type of hint useful?
  – Do automated hints only get students ‘unstuck’? Or can they lead to learning as well?
  – Automated hints only show what to do – is this sufficient, or do we need human-authored explanations?
  – What are students’ perceptions on each hint design?
Primary Contributions

- Additional features to next-step code hints: *textual explanations*, and *self-explanation prompts*.

- **Study 1:**
  - Students’ perspectives on the value of code hints with different features.

- **Study 2:**
  - Impact of code hints with additional features on students’ behavior (e.g. performance, and learning).
Overview

• Hints Design
• Study 1
• Study 2
• Discussion
• Future Work
How do we design effective hints for programming?

DESIGN OF HINTS
iSnap already provides *Code hints* that suggest an *edit* the student can make to get close to a correct solution.
Limitation #1: Lack of Explanations

**Code hints** say *what to do*, but they do not say *why*.  
• This makes hints difficult to interpret *(Price et al., 2017, Gusukuma et al., 2018)*.
Solution: Add Textual Explanations

• To address this, in our prior work, we studied adding textual explanations to code hints (Marwan et al., 2019).

• **Results:**
  • Learners rate hints as more useful, and interpretable, compared to code hints only.
  • However, we did not find evidence that these hints improve learners’ performance.

Tip:
The repeat block under “Control” allows you to run the same code a fixed number of times, like moving and turning the sprite to draw each side of a polygon.
Limitation #2: Bottom-out Hints

**Code hints** are “bottom-out” hints that give away part of the solution

- Leads to learning *only* when students spontaneously self-explain the hint (Aleven et al., 2016, Shih et al., 2008).

- Self-explanations of examples can benefit learning (Williams et al., 2016, McNamara et al., 2017).
  - However, they may frustrate learners (Shin et al., 2018).

- Fewer studies have explored self-explanation prompts in programming (Vihavainen et al., 2015, Vieira et al., 2017).
Solution: Self-Explanation Prompts

**Explore:** the effect of adding self-explanations to hints in iSnap.

**Self-Explanation Prompt:**
Require students to reason about the information they have received.

Marwan, Williams, Price - ICER19
What are students’ perceptions on each hint design?

STUDY 1
Study 1

Goal:

• Understand *students’ subjective perceptions* of code hints, textual explanations and self-explanation prompts
  • Specifically: *when* and *how* they are helpful, and *how they can be improved*.
Study 1

Population:
• 10 Undergraduate students, who have attested to have no prior programming experience.
  • Limitation: all *males*; ages (18-20)

Procedure:
• 1-on-1 user-study.

<table>
<thead>
<tr>
<th>Step</th>
<th>Activity</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>iSnap tutorial</td>
<td>5-10 min</td>
</tr>
<tr>
<td>Step 2</td>
<td>Task 1</td>
<td>15-20 min</td>
</tr>
<tr>
<td>Step 3</td>
<td>Interview 1</td>
<td>5-10 min</td>
</tr>
<tr>
<td>Step 4</td>
<td>Task 2</td>
<td>15-20 min</td>
</tr>
<tr>
<td>Step 5</td>
<td>Interview 2</td>
<td>8-12 min</td>
</tr>
</tbody>
</table>
Programming Tasks

- Draw a regular polygon with any number of sides.

- Draw a series of triangles.
Study 1

Procedure:
For both tasks, students received all types of hints randomly:

All Conditions: Code Hints

- Text Hint:
  The "repeat" block (under Control) allows you to run the same code a fixed number of times, like moving and turning the sprite to draw each side of a polygon.

- Why do you think Snap recommended this hint?
to reduce commands and draw the polygon
Study 1

Procedure:
For both tasks, students received all types of hints randomly:

All Conditions: Code Hints

[Only Code Hints]
Study 1

Procedure:
For both tasks, students received all types of hints randomly:

All Conditions: Code Hints

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Procedure:
*For both tasks, students received all types of hints randomly:*

All Conditions: Code Hints

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ SE Prompts</td>
<td></td>
</tr>
</tbody>
</table>
Procedure:

For both tasks, students received all types of hints randomly:

All Conditions: Code Hints

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ SE Prompts</td>
<td>+ Text Expl. + SE Prompts</td>
</tr>
</tbody>
</table>
Study 1

Data:

- Complete traces of students’ work.
- Audio-recordings of students’ interviews

Qualitative Analysis

- All 10 students responses were analyzed by:
  - Identifying and grouping both positive and negative themes that emerged for each type of hint and how they can be improved.
Study 1

**Code hints:**

- Helped students to see a clear next action they could take.

  “it showed you your code next to what you should do”

  P10

  ➢ Specifically useful for students who are using iSnap for the first time.

  “it gives you something that you had not thought about.”

  P6
Study 1

Code hints:

– *However*, Code Hints only said what to do, but not why

“*it just told me what to do but I did not know what the problem is or how can I fix it up*”

“*code hint itself cannot provide enough information*”
Study 1

**Code hints with Textual Explanations:**

- Gave useful but different information from a code hint.
  
  “[the code hint] shows which block to use and the text gives an idea of what to use it for.”

- No downside for the student
  (easy enough to ignore)

  “[Adding text hints to code hints] can not be not helpful.”

---

Marwan, Williams, Price - ICER19
Study 1

**Self Explanation Prompts with Hints:**

- Help students stop and think more deeply about the hint.

  "it made me think and take a step back about the whole process."

  - **P8**

- **However**, they can be frustrating and confusing.

  "it is not giving me anything back, it is just asking me if I understood it."

  - **P5**
Study 1

Our results suggested that:

• Students see the benefits in all three types of hint support, which offer complementary benefits.

• However, some students did find self-explanation prompts to be confusing or irritating.

• Therefore, it is important to investigate how they impact students’ outcomes.

Study 2
What is the effect of hints on learners’ performance and learning?

STUDY 2
Study 2

Goal:
• Investigate the effect of hints on *performance* and *learning* in a *larger-scale study*.

Research Questions:
What is the effect of hints *with and without* self-explanation prompts on:
1. learners’ *performance*?
2. their *learning* (performance on future tasks without hints)?
3. their *hint usage* during programming?
Study 2

Population:
• 201 total crowd workers through Amazon’s Mechanical Turk (MTurk) platform.
  • Have no prior programming experience.
  
  An effective form of conducting large-scale user studies in lieu of using university participants (Behrend et al., 2011, and Kittur et al., 2008)

• Demographically similar in age and education level (84.5% B.S. degree).
Study 2

Procedure:
• Similar to study 1 (iSnap tutorial, task 1, task 2), except:

| 1- Each learner is assigned to one condition: | 2- Task 2 was used as a post-test: |
| - **CT**: Code + text | - No hints were given to all groups. |
| - **CTE**: Code + Text + Prompt | - Used to assess learning |
| - **Control**: No Hint |

| 3- Study 2 was online study, **without interviews** | 4- Both Tasks were only for 15 min. |
Measures:

• Performance:
  • Each task is divided into 4 objectives

Ask for sides, and repeat answer times
Measures:

• Performance:
  • Each task is divided into 4 objectives
Measures:

• Performance:
  • developed an automatic grader to determine the number of objectives completed by each learner.
  • analyzed time taken to complete each objective.
RQ1: What is the impact of hints on learners’ performance?

- No Hint: 22.2% completed Task 1
- CT: 27.8% completed Task 1
- CTE: 45.8% completed Task 1
**RQ1:** What is the impact of hints on learners’ *performance*?

<table>
<thead>
<tr>
<th></th>
<th>Control</th>
<th>CT</th>
<th>CTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2</td>
<td>2.53</td>
<td>2.8</td>
</tr>
<tr>
<td>SD</td>
<td>1.40</td>
<td>1.24</td>
<td>1.31</td>
</tr>
<tr>
<td><em>P</em>-value</td>
<td></td>
<td>0.001</td>
<td></td>
</tr>
</tbody>
</table>

Both conditions with code hints *completed significantly more objectives* than the control condition.

- Dunn’s test with Benjamini-Hochberg correction for multiple comparisons:
  - Control group and CT learners (*p* = 0.045)
  - Control group and CTE learners (*p* = 0.001)
  - CT and CTE learners (*p* = 0.104).
Study 2

Measures:

• Learning:
  • Task 2 served as our measure of learning
    • No hints on this task.
  • Consists of 4 objectives.
    • First two objectives were identical to the first two objectives in Task 1
      • Measure how well participants learned to repeat these steps in a new context.
    • Last two objectives were different
      • measured learners’ ability to apply the same programming constructs in a new way (e.g. use of nested loops).
RQ2: What is the impact of hints on learners’ learning?

Kruskal-Wallis test showed that this difference was not significant ($p = 0.06$)

- 26.9% completed Task 2
- 25.3% completed Task 2
- 40.6% completed Task 2

Kruskal-Wallis test showed that this difference was not significant ($p = 0.06$)
RQ2: What is the impact of hints on learners’ *learning*?

• Performance on the first two objectives on Task 2 (*isomorphic to Task 1*):
  • A Kruskal-Wallis test shows a significant difference across groups (*p* = 0.013).
  • A post-hoc Dunn’s test with Benjamini-Hochberg correction for multiple comparisons:
    • CTE learners and the Control group (*p* = 0.01)
    • CT learners and the Control group (*p* = 0.59).
    • CTE learners and CT learners (*p* = 0.028)
Study 2

**RQ3:** What is the impact of hints on learners’ *Hints Request Rate*, and *Follow rate* in Task 1?
Study 2

**RQ3:** What is the impact of hints on learners’ *Hints Request Rate*, and *Follow rate* in Task 1?

1. **Hints Request Rate:** Average number of hints requested by each learner.
   1. *CT requested significantly more* hints than CTE.

<table>
<thead>
<tr>
<th></th>
<th>CT</th>
<th>CTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>P-value</td>
<td></td>
<td>0.003</td>
</tr>
</tbody>
</table>

2. **Hints Follow Rate:** The average number of requested hints that have been followed by each learner.
   1. *CTE followed significantly more* hints than CT.

<table>
<thead>
<tr>
<th></th>
<th>CT</th>
<th>CTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median</td>
<td>0.57</td>
<td>0.71</td>
</tr>
<tr>
<td>P-value</td>
<td></td>
<td>0.03</td>
</tr>
</tbody>
</table>
DISCUSSION

Implications and relation to prior work
Summary of the Results

*Code hints with or without self-explanation improve learners’ immediate programming performance.*

- Aleven et al.: next-step hints’ primary role is to help students when they get stuck *(2016)*.
  - From *Study 1*, students appreciated the hints’:
    “it just puts me in the right direction”. [P2]

- From *Study 2*:
  - Learners in both *CT* and *CTE* conditions completed significantly more objectives than the control group.
Code hints only improved learning when accompanied by self-explanation prompts.

- In the KLI framework, Koedinger et al., argued that “sense making” is necessary for learning from hints (2012).

- From *Study 1*, 70% of participants appreciated self-explanation prompts “[it helps] to interpret what... the picture [hint] mean[s]” [P2].

- From *Study 2*:
  - CT learners *performed no better* than the control condition.
  - CTE learners *performed significantly better*, only on isomorphic objectives with Task 1.
Self-explanation prompts changed the way that students interacted with code hints.

• Vieira et al., argue that adding self-explanation to worked examples increased students’ engagement and awareness of the worked-example (2017).

  • From study 1, most participants noted that self-explanation prompts caused them to:
    • “think and take a step back about the whole process” [P6].

  • From study 2:
    • Learners in the CTE condition:
      • Spent more time viewing each hint than CT learners, requested fewer hints, but were more likely to follow them.
CONCLUSION & FUTURE WORK
Conclusion

Code hints can improve performance and learning, but depending on how we present them.
Future Work

- Investigate whether our results would generalize to a classroom context with more complicated programming tasks.

- Explore other forms of hints on students’ behavior. (e.g. positive feedback).

- How textual explanations can be reused across several programming exercises.
Thank You!!
Questions?

Samiha Marwan
samarwan@ncsu.edu

Dr. Joseph Williams
williams@cs.toronto.edu

Dr. Thomas Price
twprice@ncsu.edu