How should we teach debugging to secondary school students?

Laurie Gale

July 2024
Programming is not always plain sailing

```python
first_name = "Laurie"
surname = "Gale"
print("Hello world and welcome to my presentation my name is," + first_name + " " + surname)
```
Debugging is necessary

**MILLIONS** will learn text-based programming at secondary school

**MILLIONS** will be exposed to errors
Debugging is...

- Necessary if we want meaningful programs
- A useful practice of troubleshooting
- An enjoyable/infuriating challenge
Debugging is challenging

“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it?” - Kernighan's Law
The challenges of learning to debug

Part I: Debugging strategy

Cognitive challenges

Affective challenges

Part II: Attitudes and Emotions

In-class challenges
Part I: Debugging Strategy
Lots to think about when programming

Design of the program

Managing variables

Program syntax

Interaction of programming concepts

Using built-in functions

Control structures

Sequential execution
Learners have misconceptions

“Several lines of a (simple non-concurrent) program can be simultaneously active.”\textsuperscript{1}, p. 261
Learners have misconceptions

Managing variables

\[ \alpha = "Hello\ world" \] \(^{1, p. 261}\)

"Hello world" = \alpha
Error messages are confusing

Error at line 132 but to fix it you add a parenthesis at line 120
Learning to debug is **HARD**

**Cognitive challenges**
- Lots to store in working memory
- Presence of misconceptions
- Hard to interpret error messages
- Lack of necessary knowledge & experience

**Ineffective debugging strategy**

What do these look like for school students learning Python?

How can we improve them?
Investigating debugging behaviours

What **behaviours** do lower secondary students exhibit when **debugging**?
Study outline

73 lower secondary students → Debugging exercises → Survey → Analysis
Programming Exercise 3

This program checks if someone should apply to be a computing teacher using the steps below:

- Input the user's age.
- Input the user's response to the question "Do you have a passion for teaching computing? Enter 'yes' or 'no': "
- If the user is 21 or over and does have a passion for teaching computing, the check should be a success. Otherwise, the check should be unsuccessful.
- Print the result of the check.

This program has 4 errors - have a go at fixing them all.

```python
# Question 3
print("This program will check if you should apply to be a computing teacher")
age = int(input("What is your age? "))
computing_degree = input("Do you have a passion for teaching computing? Enter 'yes' or 'no': ")

if age >= 21 and computing_degree == "yes":
    allowed_to_apply = "Successful"
else:
    allowed_to_apply = "Unsuccessful"
print("Result of check:", allowed_to_apply)
```
Results
WIDE RANGE of debugging behaviours

73 students

346 exercise attempts

7,247 runs

40% were successfully completed

36% of students didn’t successfully complete any

The results in numbers
An overview

Less successful students...
• Often added errors
• Tinkered/used trial and error
• “Spammed run”

More successful students...
• Made less changes with (seemingly) more intentionality
• Got their code running quicker
• Didn’t run their code as much
Introducing some students

Alessia
What would you rate your performance on the exercise? 3/5
What debugging techniques did you use? “I ran the code so I could see where and what line was wrong”

Gabriel
What would you rate your performance on the exercise? 4/5
What would you rate your performance on the exercise? “Looked through line by line and used the error message”
What Alessia did

1. Ran program before making changes (95% of students did this)...

... after 22 seconds (5 seconds longer than the average)
What Alessia did

2. Made changes on the line the error message was pointing to (97% of students did this)
3. Added multiple **syntax** errors (86% of students did this)
What Alessia did

4. “Spammed run” while program had errors (74% of students did this, with the median time between runs being 0.379 seconds)
5. Ended program with more errors than she started with

(program had errors in every run – same as min. 34% of exercises)
A comparison of their behaviours

Ran code after 26 seconds

Ran code after 22 seconds
A comparison of their behaviours

(Potentially) resolved logical errors through testing

(Potentially) used error message for guidance
A comparison of their behaviours

Made several corrective changes

Made no corrective changes
A comparison of their behaviours

Added one syntax error, which was resolved straightaway

Added several syntax errors
A comparison of their behaviours

Ended exercise in correct state

Ended with 3 syntax errors and 3 logical errors
What’s stopping more students debugging successfully?

1. Knowledge of Python syntax
2. Time taken to get program successfully executing
3. Affective factors – motivation, resilience, mindset towards errors?
How can this help teachers and students?

Suggestions

- More explainable error messages
- Syntax-related tooling
- Systematic debugging processes
- Discourage ineffective debugging?
- Teaching effective debugging
- Live coding
- Frame-based editing

With LLMs?
Frame-based editing

https://strype.org/
Test a program.

If it's not right...

Ask yourself, "What is the problem?"
And "What type of bug could cause the problem?"

Then ask yourself, "Does the program have subprograms?"
And "Where might the bug be?"
- in a subprogram
- in a REPEAT or IF
- after a certain command

Then use the information to find the bug.
Otherwise, read every command
And decide whether it's correct.

Once you've found the bug, ask yourself, "What should the fix be?"
Then make the fix, and
Re-test the program.
Debugging made easy

1. **Compile**
   - Is the program compiling successfully?
   - Adjust your program
   - Compile-time errors

2. **Run**
   - Does the program run without errors?
   - Adjust your program
   - Runtime errors
   - Read and understand the first error message
   - Determine the error and find the relevant lines of code
   - "What's the cause?"
   - Modify your assumption or make a new one

3. **Compare**
   - Do expected and actual behavior match?
   - Adjust your program
   - Logical errors
   - "Why is this the case?"
   - Modify your assumption or make a new one
   - Determine the error and find the relevant lines of code

4. **Done!**
The challenges of learning to debug in K-12

Part I: Debugging strategy

Cognitive challenges

Affective challenges

Part II: Attitudes and Emotions
Part II: Attitudes and Emotions Towards Debugging
Struggling when debugging

After “being hit” [encountering an error], students are dazed, with little sense of what to do next ... These experiences left students puzzled, confused, frustrated, overwhelmed, and annoyed.⁵, p. 81
Struggling when debugging

“The majority of emotional consequences of encountering a problem are negative – though this can range from confusion and puzzlement to much more anguished frustration, anger, and sense of “oh no, not again.””

5, p. 81
“It’s just a let down, as I said. It’s a confidence roller coaster. Hitting that compile button”6, p.113
The consequences of debugging struggle

- Cognitive challenges
- Emotional challenges

Formation of attitudes
Struggling when debugging

“Every time after I type code and I run it for the first time, I expect it to fail.” 6, p. 115
What attitudes and emotions do lower secondary students have towards debugging?

(and how can these be improved?)
The study outline (again)

- 73 lower secondary students
- Debugging exercises
- Survey
- Analysis
The many correlations between attitudes

Gender
Treated as a binary variable where:
Male = 1
Female = 2

Self-efficacy

Perceived performance

Anxiety

Usefulness

Frustration
Key finding

Attitudes and emotions towards debugging are interlinked – important to bear in mind
How can this help teachers and students?

Some suggestions from teachers:

- Normalise errors
- Encourage sabotage
- Positive error culture
- Celebrate the effort as well as the outcome
- Scaffolded debugging exercises
- Give chances to successfully debug early on

“Stump the teacher”
“Spectacular failures. I want those reported and celebrated as well. If something should have gone right and went badly wrong but somebody found something interesting on the way...you celebrate it. Take the fear out of it.”
Some Takeaways
Lessons learnt

Debugging is a powerful and challenging skill to learn
Some students resort to ineffective debugging strategies
These can easily have emotional consequences
But there’s lots of tools/pedagogies that can help!
Some questions to ponder

How do these findings compare to your experiences with novice debugging?

Have you found any effective approaches for teaching debugging? If so, what are they?

What role does GenAI have to play in all of this?
Thanks for listening!
References


Thanks to FLATICON for all the nifty icons used in the presentation, they look really cool (in my opinion)