UNIVERSITY OF CAMBRIDGE

Raspberry Pi Foundation

How should we teach debugging to secondary school students?

Laurie Gale

July 2024

Programming is not always plain sailing

```
1  first_name = "Laurie"
2  surname = "Gale"
3  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

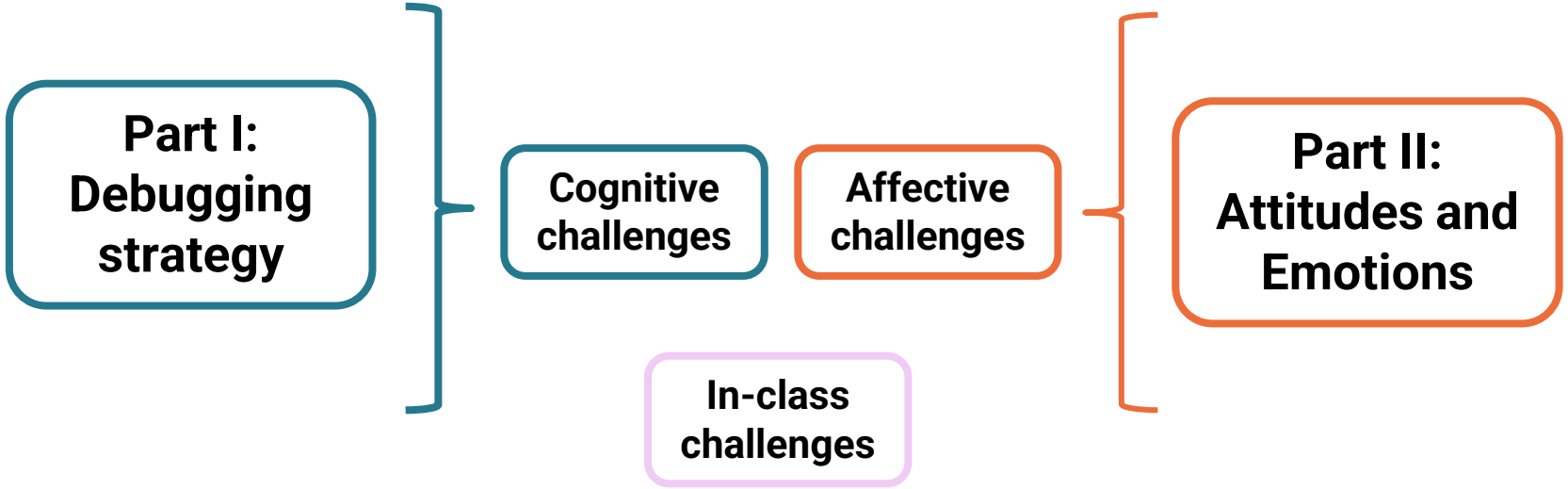
A useful practice of troubleshooting

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

Lots to think about when programming

Design of the program

Using built-in functions

Managing variables

Program syntax

Sequential execution

Control structures

Interaction of programming concepts

Learners have misconceptions

“Several lines of a (simple non-concurrent) program can be simultaneously active.”[1, p. 261]

Sequential execution

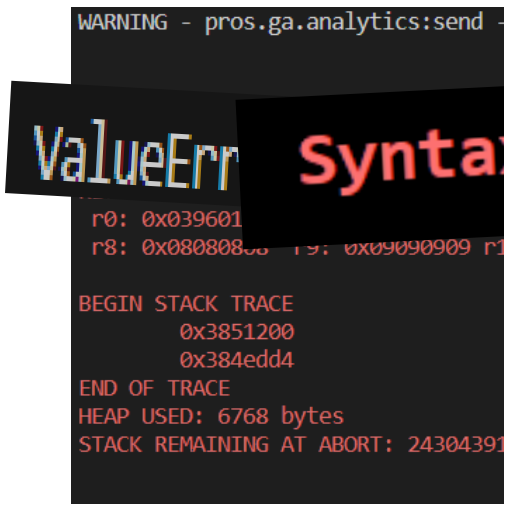
Learners have misconceptions

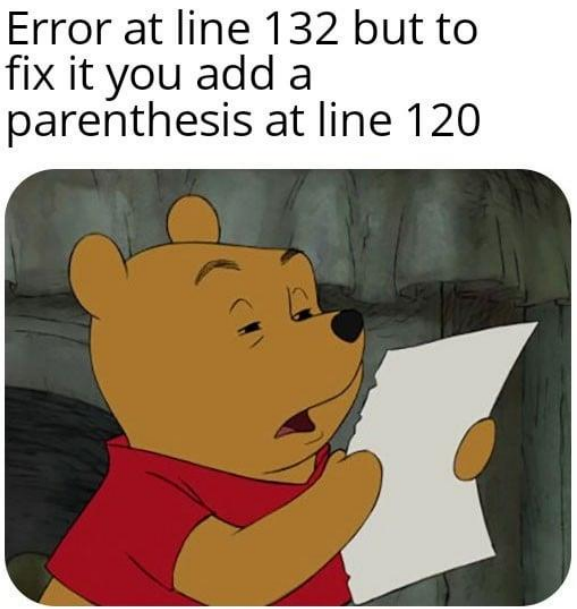
Managing variables

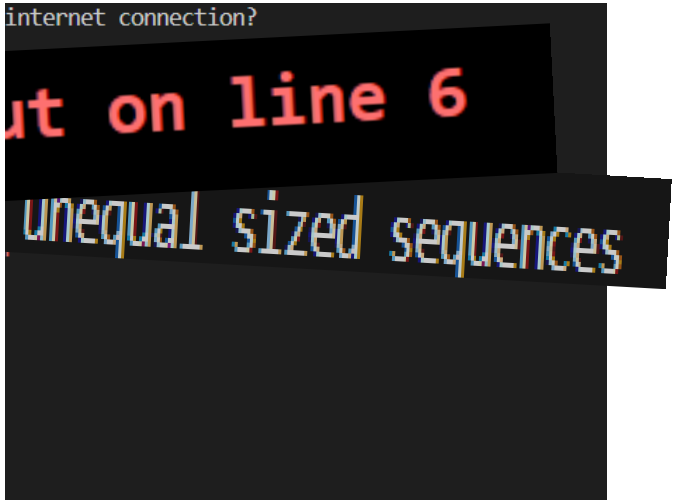
```
a = "Hello world"
"Hello world" = a
```

[1, p. 261]

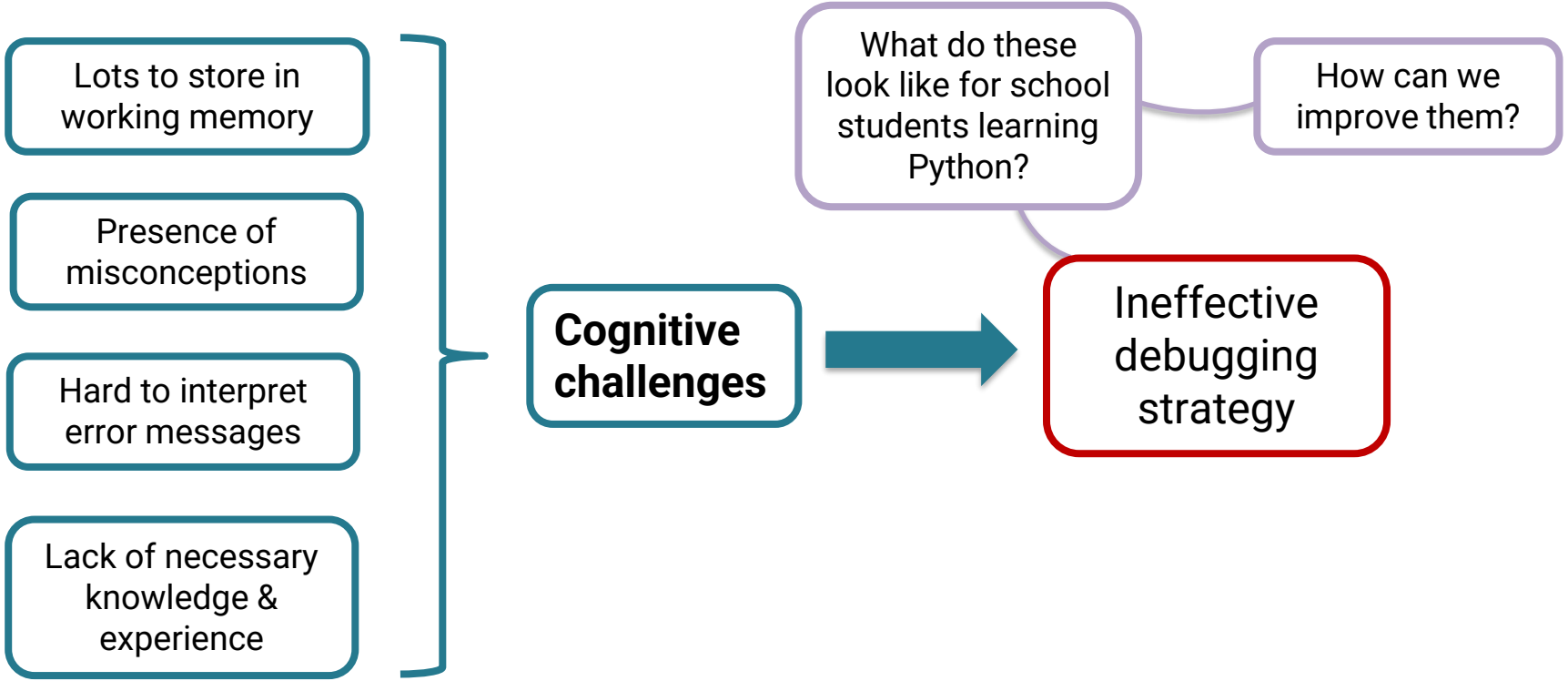
Error messages are confusing







Learning to debug is *HARD*



Investigating debugging behaviours

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

Study outline

73 lower secondary students

Debugging exercises

Survey

Analysis

The debugging exercises

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.

Exercise description

```
1  # Question 3
2  print("This program will check if you should apply to be a computing teacher")
3  age = int(input("What is your age? "))
4  computing_degree = input("Do you have a passion for teaching computing? Enter 'yes' or 'no': ")
5
6  if age > 21 or computing_degree = "yes":
7    allowed_to_apply = "Successful"
8  else:
9    allowed_to_apply = "Unsuccessful"
10   print("Result of check:",allowed_to_apply)
```

Buggy program

Code editor

Results

The results in numbers

73
students

346
exercise attempts

- 40% were successfully completed
- 36% of students didn't successfully complete any

7,247
runs

WIDE RANGE of debugging behaviours

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)

What Alessia did

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**)

What Alessia did

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

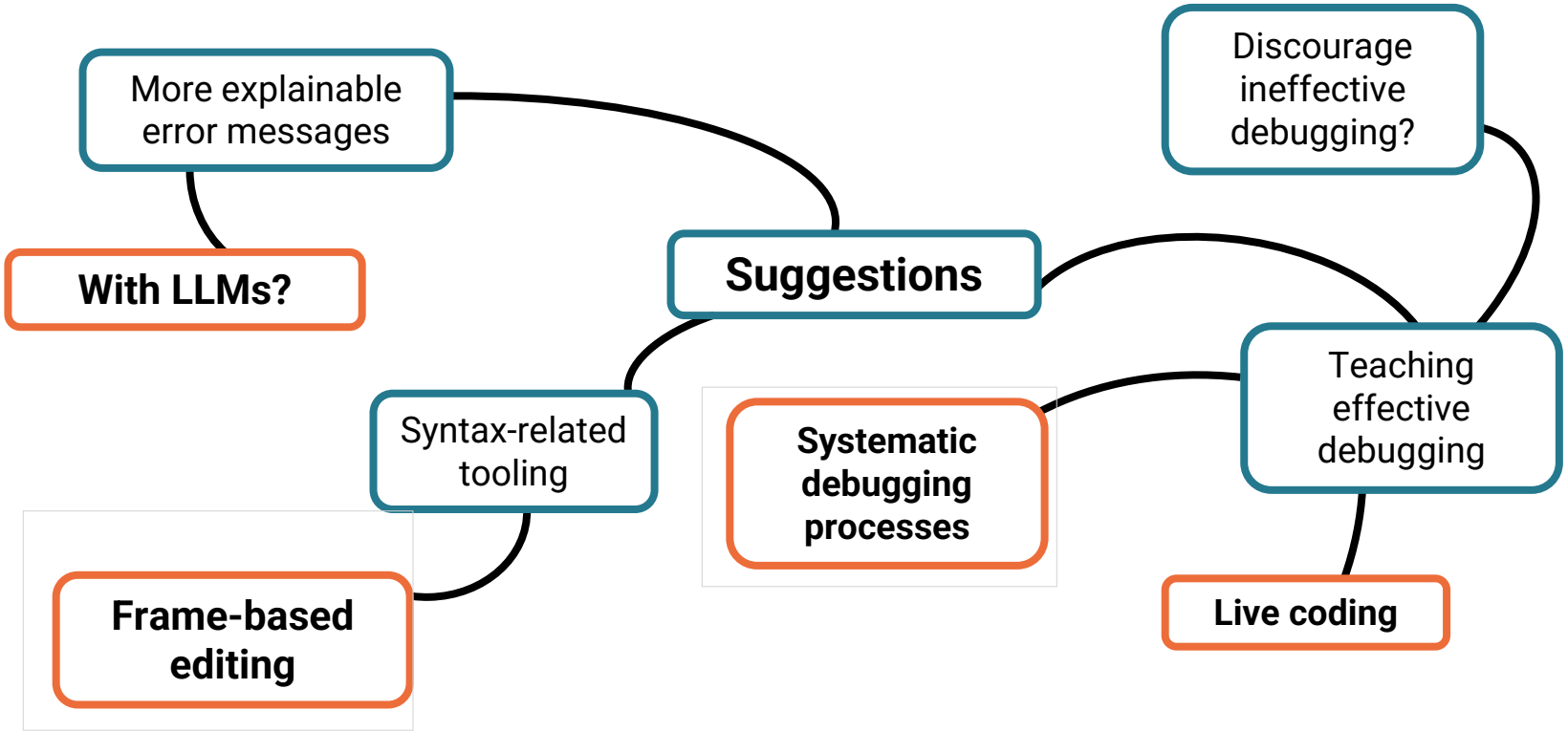
Ended exercise in correct state

Ended with 3 syntax errors and 3 logical errors

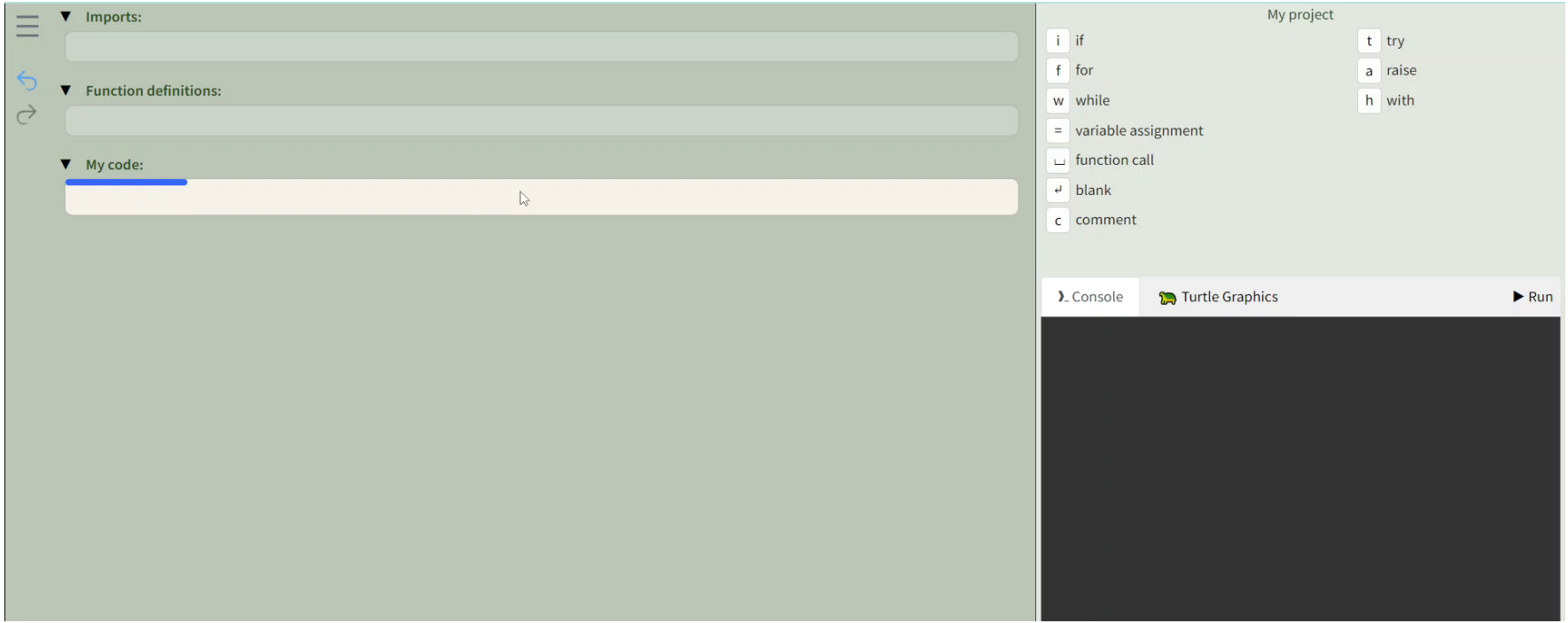
What's stopping more students debugging successfully?

1. Knowledge of Python syntax
2. Time taken to get program successfully executing
3. Affective factors – motivation, resilience, mindset towards errors?

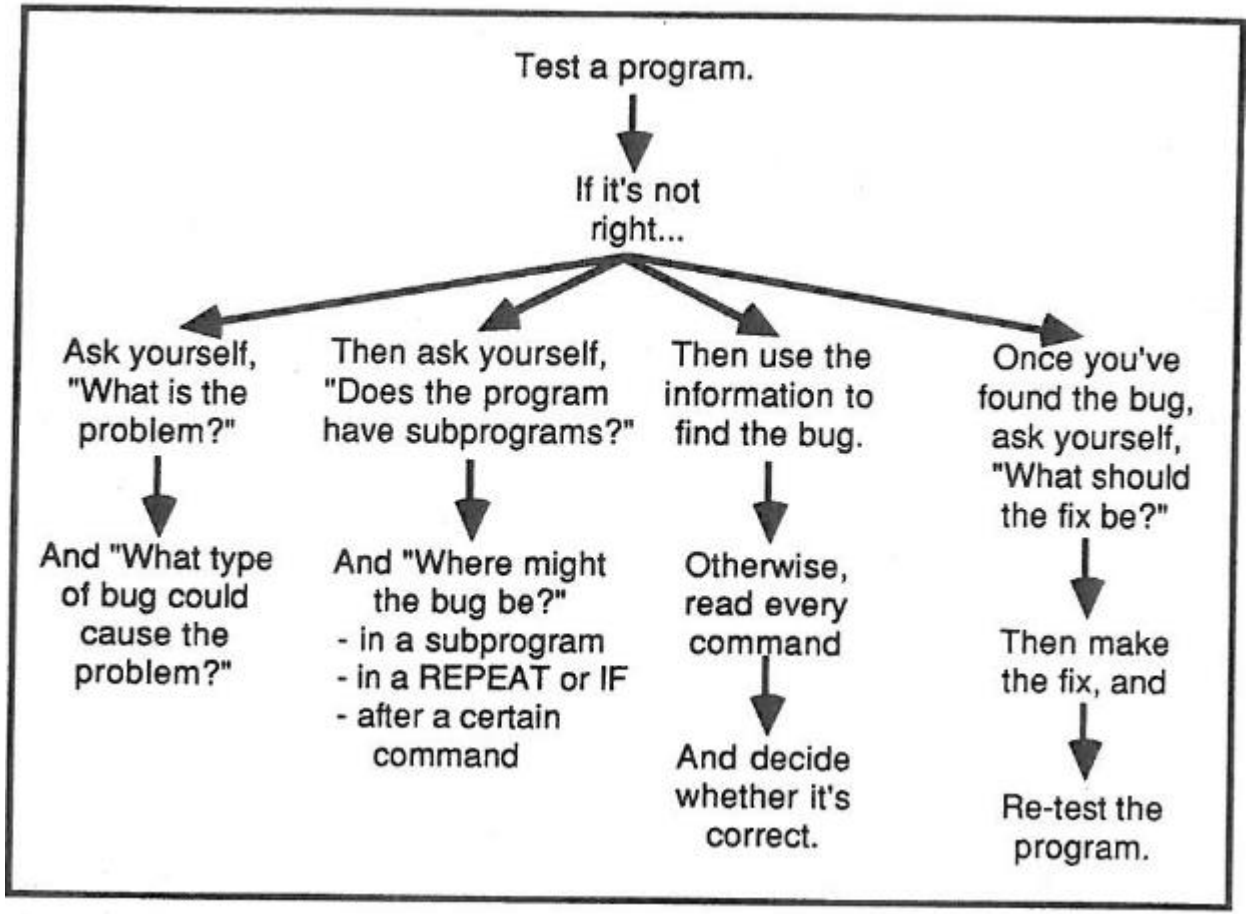
How can this help teachers and students?

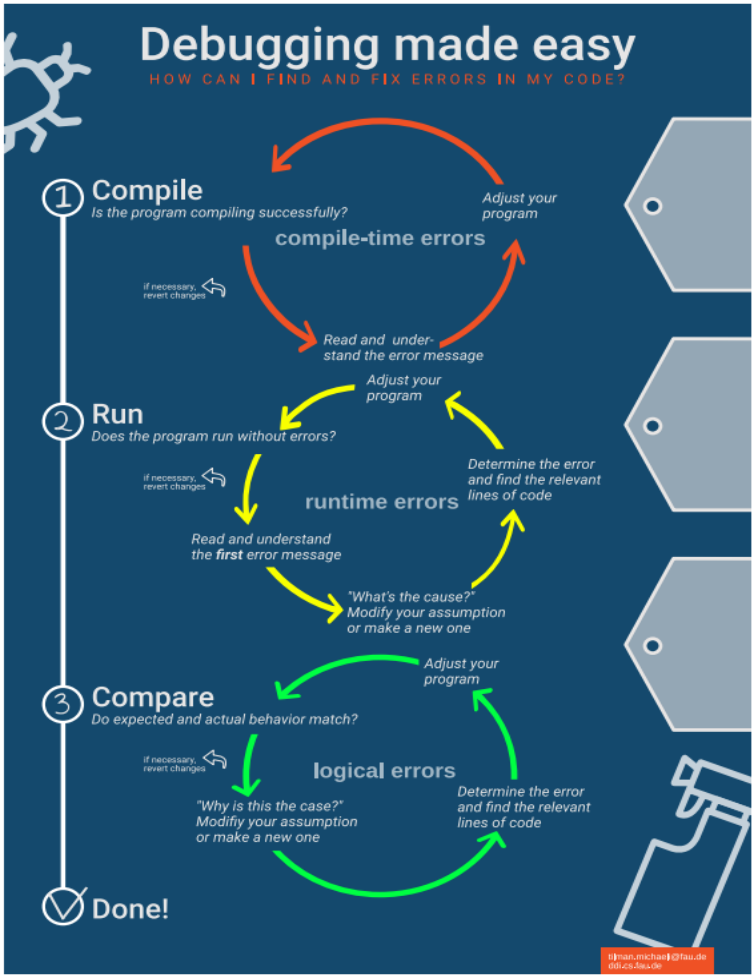


Frame-based editing



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
HOW CAN I FIND AND FIX ERRORS IN MY CODE?
1 Compile
Is the program compiling successfully?
Adjust your program
compile-time errors
if necessary, revert changes
Read and understand the error message
2 Run
Does the program run without errors?
Adjust your program
if necessary, revert changes
runtime errors
Determine the error and find the relevant lines of code
Read and understand the first error message
"What's the cause?" Modify your assumption or make a new one
3 Compare
Do expected and actual behavior match?
Adjust your program
if necessary, revert changes
logical errors
Determine the error and find the relevant lines of code
"Why is this the case?" Modifiy your assumption or make a new one
Done!

The challenges of learning to debug in K-12

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]

Struggling when debugging

“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

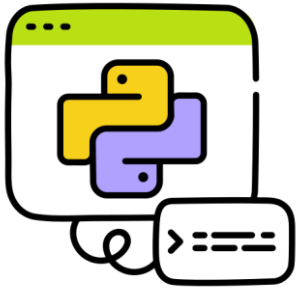
“Every time after I type code and I run it for the first time, **I expect it to fail**.” [6, p. 115]

*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

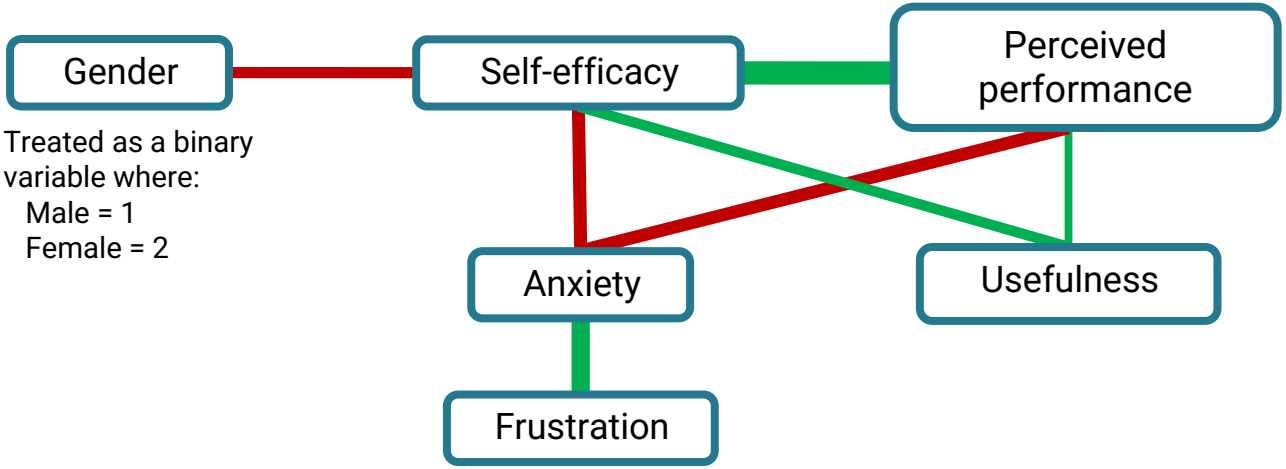


Debugging exercises

Survey

Analysis

The many correlations between attitudes

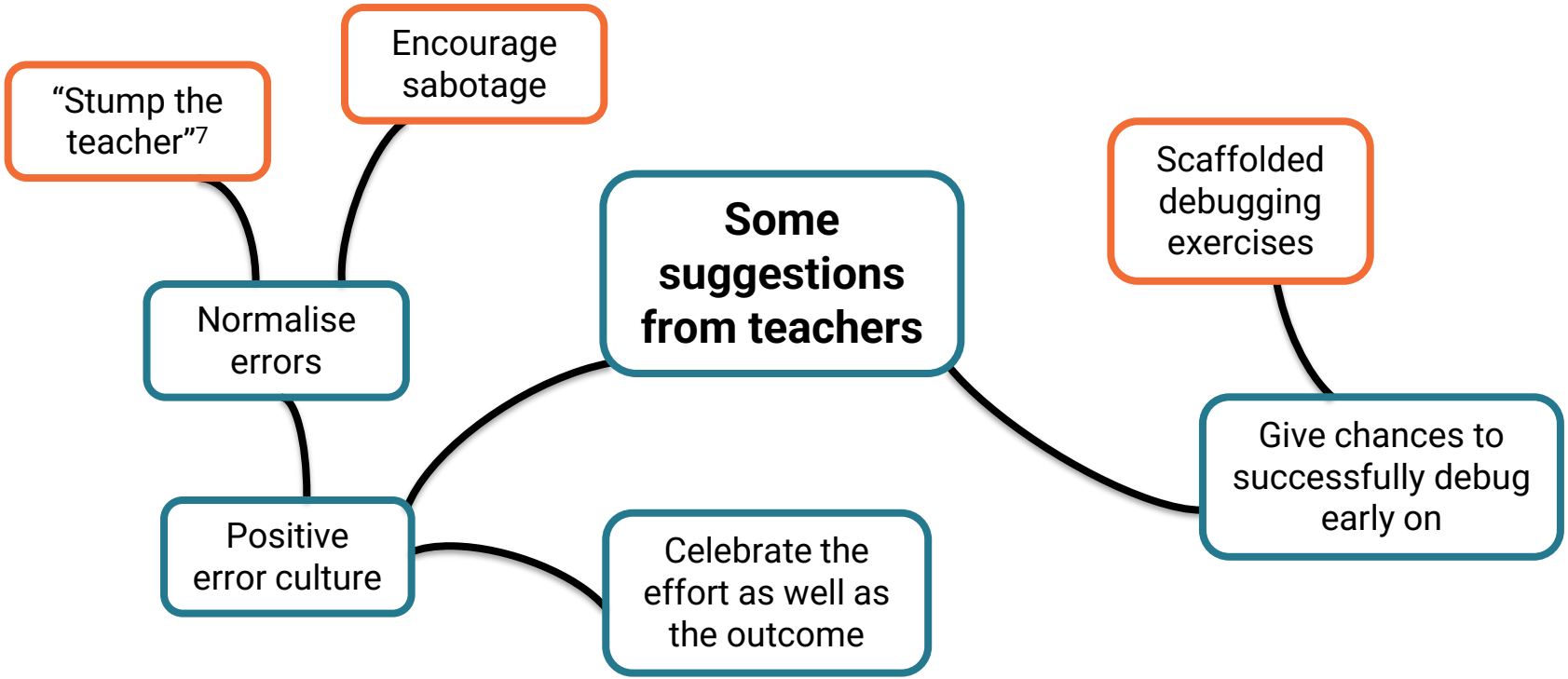


Treated as a binary variable where:
- Male = 1
- Female = 2

Key finding

Attitudes and emotions towards debugging are interlinked – important to bear in mind

How can this help teachers and students?



A quote from a brilliant 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?

UNIVERSITY OF CAMBRIDGE

Raspberry Pi Foundation

Thanks for listening!

References

1. Sorva, J. (2018). Misconceptions and the Beginner Programmer. In Computer Science Education: Perspectives on Teaching and Learning in School Bloomsbury Academic.
2. David Perkins and Fay Martin. 1985. Fragile Knowledge and Neglected Strategies in Novice Programmers. Technical Report. Educational Center for Technology, Cambridge, MA. 1–35 pages. https://eric.ed.gov/?id=ED295618
3. Sharon M Carver and Sally C Risinger. 1987. Improving children's debugging skills. In Empirical studies of programmers: second workshop. AblexPublishing Corp., Norwood, New Jersey, 147–171. https://www.researchgate.net/publication/262311647
4. Tilman Michaeli and Ralf Romeike. 2019. Improving debugging skills in the classroom - The effects of teaching a systematic debugging process. In WiPSCE'19: Proceedings of the 14th Workshop in Primary and Secondary Computing Education. Association for Computing Machinery, New York NY, USA, 1–7. https://doi.org/10.1145/3361721.3361724
5. Paivi Kinnunen and Beth Simon. 2010. Experiencing programming assignments in CS1: the emotional toll. In Proceedings of the Sixth international workshop on Computing education research (ICER '10). Association for Computing Machinery, New York, NY, USA, 77–86. https://doi.org/10.1145/1839594.1839609
6. Jamie Gorson, Kathryn Cunningham, and Marcelo Worsley. 2022. Using Electrodermal Activity Measurements to Understand Student Emotions While Programming. In ICER '22: Proceedings of the 2022 ACM Conference on International Computing Education Research. Association for Computing Machinery, New York, 105–119. https://doi.org/10.1145/3501385.3543981
7. Chris Kerslake. 2024. Stump-the-Teacher: Using Student-generated Examples during Explicit Debugging Instruction. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024). Association for Computing Machinery, New York, NY, USA, 653–658. https://doi.org/10.1145/3626252.3630934

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