UNIVERSITY OF CAMBRIDGE

Raspberry Pi Foundation

How should we teach debugging to secondary school students?

Laurie Gale

March 2024

Learning to program → dealing with errors

```
WARNING - pros.ga.analytics:send - Unable to send analytics. Do you have a stable internet connection?



Exception occurred: basic_string::_M_create
CURRENT TASK:
REGISTERS AT ABORT
 r0: 0x03960104  r1: 0x038b8064  r2: 0x0384df60  r3: 0xa5a5a5a5  r4: 0x038000dc  r5: 0x04a02694  r6: 0x04a026a4  r7: 0x00000007
 r8: 0x08080808  r9: 0x09090909 r10: 0x10101010 r11: 0x11111111 r12: 0x12121212  sp: 0x039600c0  lr: 0x0384edd4  pc: 0x03851200

BEGIN STACK TRACE
        0x3851200
        0x384edd4
END OF TRACE
HEAP USED: 6768 bytes
STACK REMAINING AT ABORT: 243043914 bytes
```

Why research K-12 debugging?

Computing programmes of study: key stages 3 and 4

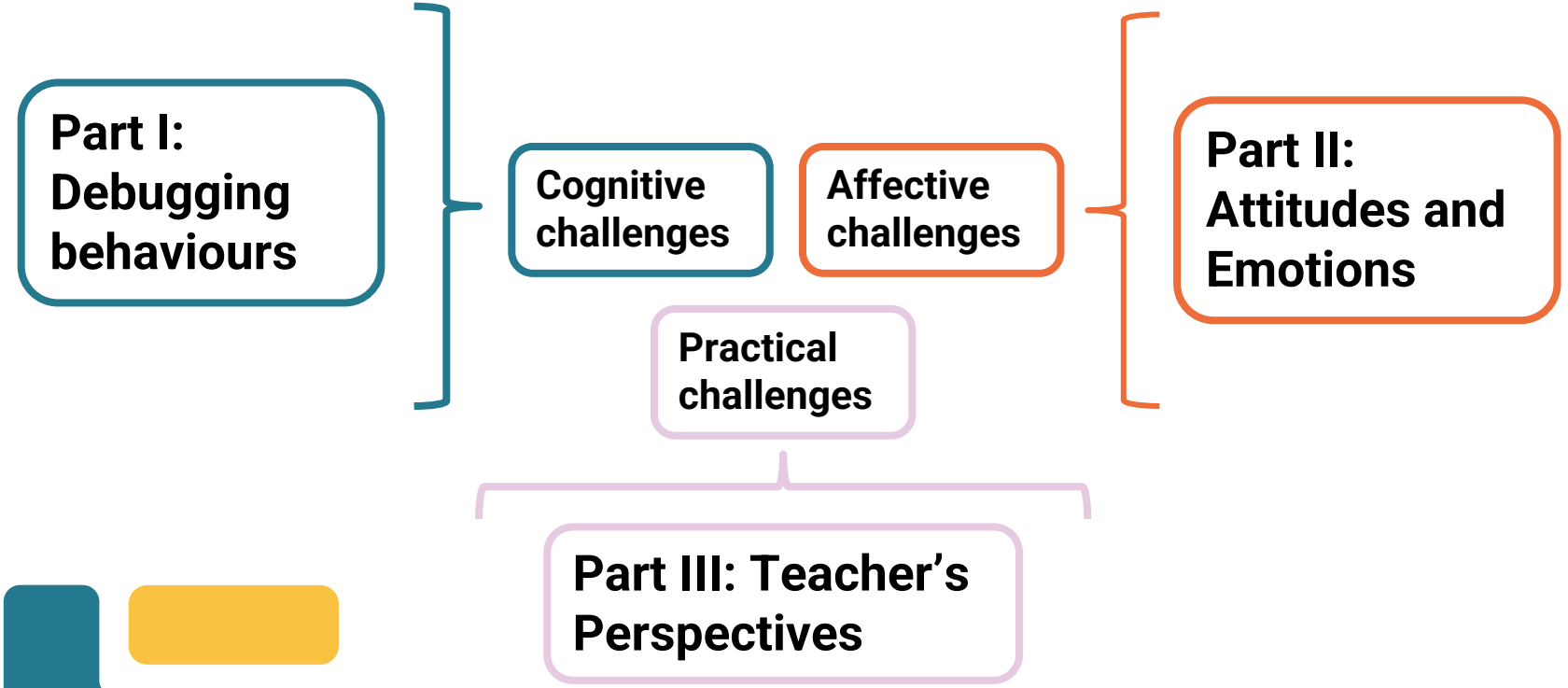
National curriculum in England

- use two or more programming languages, at least one of which is textual, to solve a variety of computational problems;

Different context to university → different challenges

It's the curriculum → students don't opt in

The challenges of learning to debug in K-12



Part I: Debugging Behaviour

Learning to debug is *HARD*

Lots to store in working memory when learning to program[2]

Error messages are difficult for novices to read[3]

Errors may be based on misconceptions

Novices tend to use ineffective debugging strategies[4,5,6]

Cognitive challenges

What we know about debugging behaviour

- ✓ Undergraduate studies in block-based and text-based programming languages
- ✓ K-12 studies in block-based programming languages

- X K-12 studies in **text-based** programming languages

Study 1A

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

Study 1A

75 students (12–14-year-olds) from two schools

Debugging exercises

346 attempted exercises
≈7,000 snapshots

Qualitative content analysis

The debugging exercises

Programming Exercise 3

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

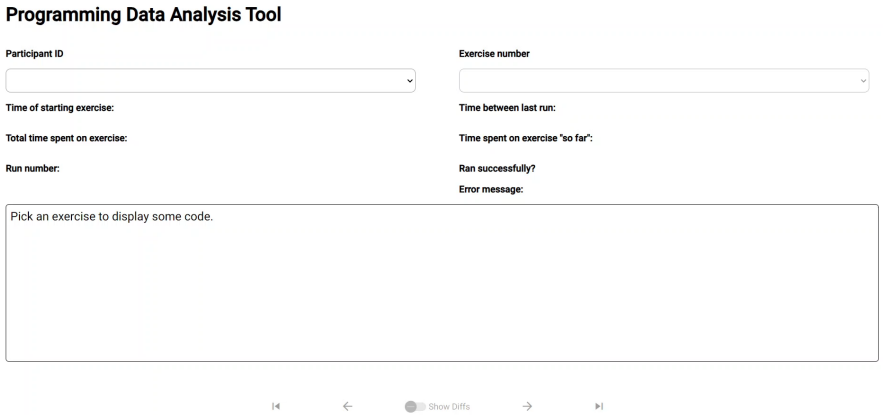
- Input the user's age.
- Input the user's response to the question "Do you have a passion for teaching computing? Enter 'yes' or 'no': "
- If the user is **21 or over** and does have a passion for teaching computing, the check should be a success. Otherwise, the check should be unsuccessful.
- Print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

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

```
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 and computing_degree == "yes":
7    allowed_to_apply = "Successful"
8  else:
9    allowed_to_apply = "Unsuccessful"
10 print("Result of check:",allowed_to_apply)
```

```
This program will check if you should apply to be a computing teacher
What is your age? 21
Do you have a passion for teaching computing? Enter 'yes' or 'no': yes
Result of check: Successful
```

Qualitative content analysis … on log data?



Manually inspected log data → Examined changes made between runs → Generated codes/ categories representing common behaviours

Some caveats

- Students were debugging “foreign code”
- They’d never used this code editor before
- The categorisation was binary and not ordered

Results

Six main themes:

1. First change
   - Ran code before making changes
2. Introduced additional errors
   - Made changes before running code
3. Resolved errors
4. Inconsequential changes
5. Positive debugging behaviours
6. Miscellaneous changes

Introducing some students

Alessia (struggled)
Perceived performance: 3/5
“I ran the code so I could see where and what line was wrong”

Gabriel (successful)
Perceived performance: 4/5
“Looked through line by line and used the error message”

Exercise 3

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.

```
# 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 or computing_degree = "yes":
  allowed_to_apply = "Successful"
else:
  allowed_to_apply = "Unsuccessful"
  print("Result of check:",allowed_to_apply)
```

Alessia

Perceived performance: 3/5
“I ran the code so I could see where and what line was wrong”

Time of starting exercise: Wed, 08 Mar 2023 09:26:03 GMT

Time between last run: 22.326 seconds

Total time spent on exercise: 12 minutes 40.477 seconds

Time spent on exercise "so far": 22.326 seconds

Run number: 1 out of 32 attempts on exercise

Ran successfully? No

Error message: `SyntaxError: bad input on line 6`

```
# 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 or computing_degree = "yes":
  allowed_to_apply = "Successful"
else:
  allowed_to_apply = "Unsuccessful"
  print("Result of check:",allowed_to_apply)
```

Show Diffs

Gabriel

Perceived performance: 4/5
“Looked through line by line and used the error message”

Time of starting exercise: Mon, 27 Feb 2023 10:22:15 GMT

Time between last run: 27.925 seconds

Total time spent on exercise: 2 minutes 26.856 seconds

Time spent on exercise "so far": 27.925 seconds

Run number: 1 out of 9 attempts on exercise

Ran successfully? Yes

Error message:

```
# 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 or computing_degree == "yes":
  allowed_to_apply = "Successful"
else:
  allowed_to_apply = "Unsuccessful"
  print("Result of check:",allowed_to_apply)
```

Show Diffs

A comparison of their behaviours

Ran code after 22 seconds

(Potentially) used error message for guidance

Made no corrective changes

Added several syntax errors

Ended with 3 syntax errors and 3 logical errors

Ran code after 26 seconds

(Potentially) resolved logical errors through testing

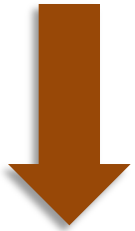
Made several corrective changes

Added one syntax error, which they resolved straightaway

Ended exercise in correct state

What's stopping more students debugging successfully?

1. Knowledge of Python syntax
2. Time taken to get program successfully executing
3. Affective factors



Fragile knowledge

"Knowledge that is partial, hard to access, and often misused"[7, p.4]

How can this inform practice?

Some suggestions:

- More explainable error messages[3,8]
- Tooling to help with syntax errors[9]
- Teaching effective debugging strategies[10,11]
- Discouraging ineffective ones

Part II: Attitudes and Emotions Towards Debugging

Struggling when debugging

“Lacking a ready answer to the difficulty, the **student not only feels at a complete loss**, but is unwilling to explore the problem any further”[4, p.42]

Struggling when debugging

"The consequences that resulted after encountering the error seems to reflect the **struck by lightning experience** as well. … These experiences left students **puzzled, confused, frustrated, overwhelmed, and annoyed**"[12, p.81]

Struggling when debugging

"The majority of students **look horrified, put their hands up** and say 'that's red, there's a mistake', and expect the teacher to present the solution to them" [13, p. 1034]

Struggling when debugging

“Every time after I type code and I run it for the first time, I expect it to fail. So that’s why ... it didn’t affect me that much either way” [14, p. 115]

Learning to debug is ***EMOTIONAL***

- Feelings of frustration, anguish, and denial[12,15]
- Evidence of physiological reactions to error struggles[14]
- Negative emotional experiences contribute to negative attitudes[16]

→ **Affective challenges**

An interaction with a student

*“I **hate** computing.”*

“Why’s that?”

*“Because **I can’t do any of it**.”*

Study 1B

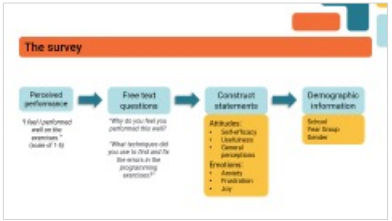
To measure lower secondary students' **thoughts** and **feelings** towards debugging

Study 1B

75 lower secondary students from two schools

Debugging exercises

346 attempted exercises
≈7,000 snapshots



Survey

Survey response data
73 responses

Correlational analysis

The survey

Perceived performance

"I feel I performed well on the exercises."
(scale of 1-5)

Free text questions

"Why do you feel you performed this well?

"What techniques did you use to find and fix the errors in the programming exercises?"

Construct statements

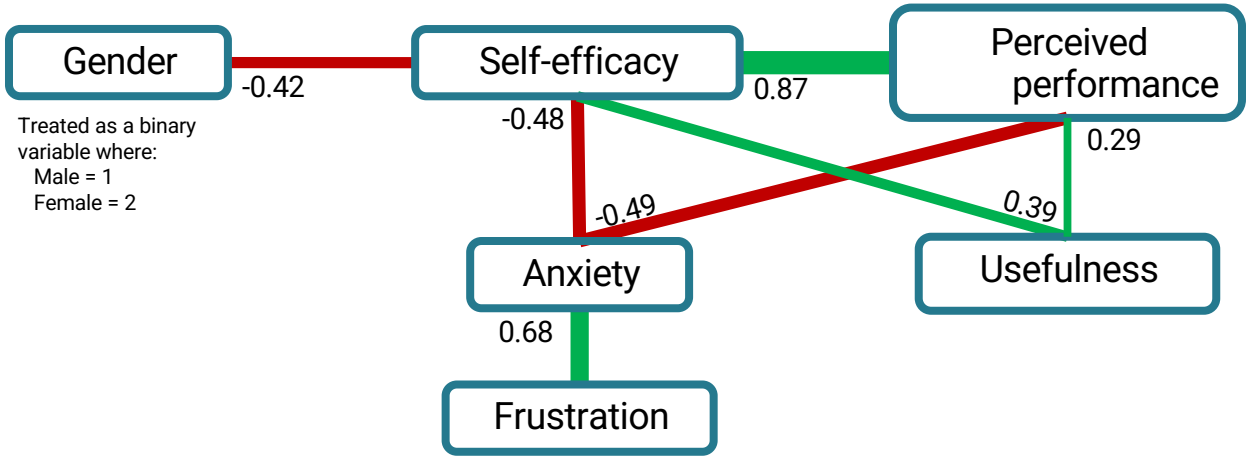
Attitudes:
- Self-efficacy
- Usefulness
- General perceptions

Emotions:
- Anxiety
- Frustration
- Joy

Demographic information

School
Year Group
Gender

Results – Correlation between attitudes



Reported Debugging Behaviours

“What techniques did you use to find and fix the errors in the programming exercises?

e.g. I searched the internet for a solution, I ran the code to see what errors it had, etc.”

Themes gathered using **qualitative content analysis**

Reported Debugging Behaviours

1) Running of code – 57 mentions
Initial running of code (for purpose of reading error messages) – *"I ran the code to see what and where the errors were"*
Repeated running of code – *"I ran the code many times and made slight adjustments"*

2) Inspection of code – 25 mentions
General inspection of code – *"[I] looked for obvious errors e.g. incorrect indent"*
Inspecting particular lines of code - *"check the lines of code where a bug is more likely first"*

3) Use of external resources– 12 mentions
Use of cheat sheet
Searching the internet

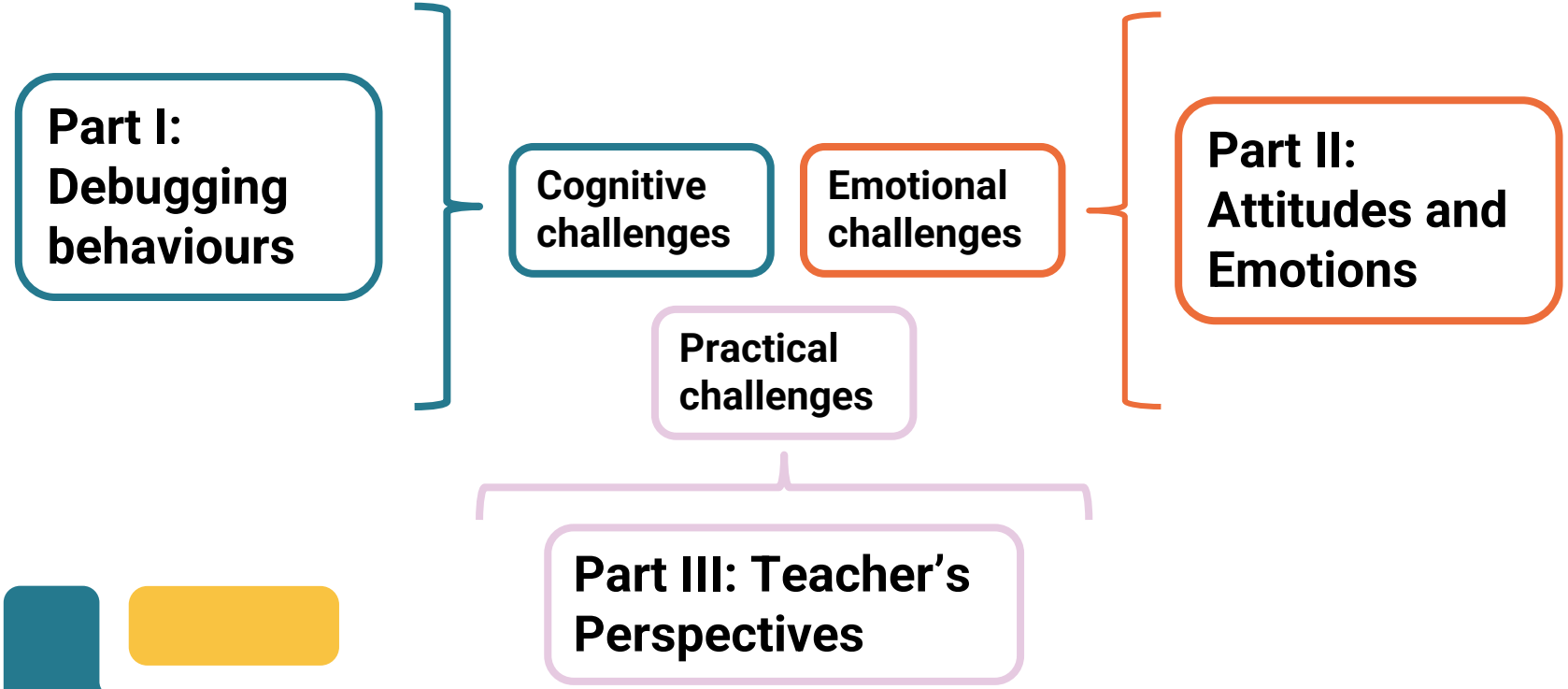
4) Trial and error – 10 mentions
"[I] just kept going till the program worked"

What does this mean for lower secondary learners?

1. Attitudes and emotions must be considered when teaching debugging
2. Important for students' debugging self-efficacy to be strong
3. **Effective debugging behaviours should be taught to students**

Part III: The Teacher Perspective

The challenges of learning to debug in K-12



Learning to debug is *TOUGH FOR TEACHERS*

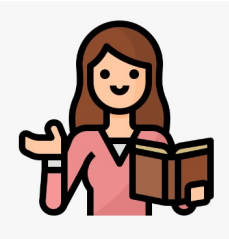
- Teachers aren't always confident programmers[17]
- Teachers rushing around the classroom[13]
- Limited classroom time for teaching programming

Practical challenges

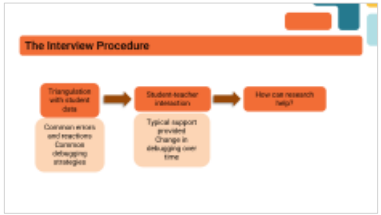
Study 2

*What are lower secondary computing teachers' **experiences** and **perspectives** relating to teaching debugging?*

Study 2



10 lower secondary computing teachers

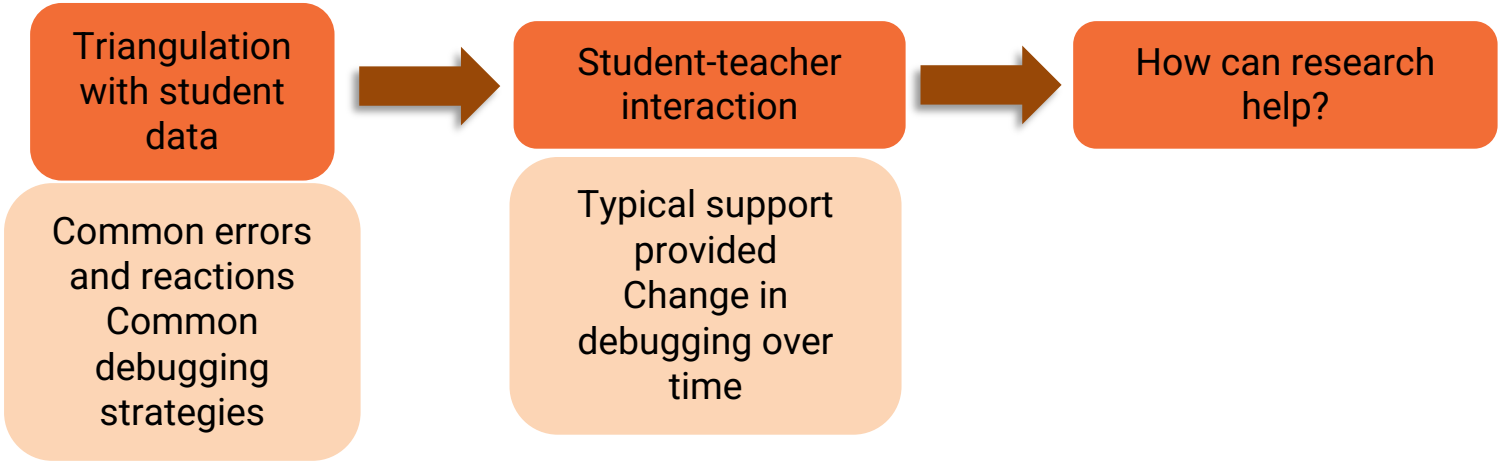


Teacher interviews

35-65 minutes in length

Reflexive thematic analysis

The Interview Procedure



Results – under construction!

Some contradictory findings:

On the barriers to syntax errors:

- *“So we don't get that many problems in terms of syntax errors. They're relatively easy for them to, to find.”*
- *“we didn't notice any particular frustration or any emotional reaction at the beginning stages with syntax errors.”*

On debugging strategies:

- *“I think they fairly quickly learned that your error might not be in the line the errors directed at, so that is definitely one thing”*
- *“they are trained to use the debugging tools in Thonny, which are really, really good”*

Results – under construction!

And some additional findings:

On emotional reactions:

- *“you will hear often a loud exclamation of where you know something isn't working.”*
- *“The kind of raised fists in the air, it genuinely happens, and it's really exciting and we celebrate that”*

How can this inform practice?

Main problems with K-12 debugging:

- Catering for diversity in ability
- Not helping every student
- Helping students stay motivated

How research could help:

- Concrete debugging teaching strategy
- Set of debugging exercises for common errors

Some Takeaways

Lessons Learnt

Study 1A

- Students often exhibit ineffective debugging behaviours
- E.g. repeated runs, tinkering
- Hard to resolve errors once these have begun

Lessons Learnt

Study 1B

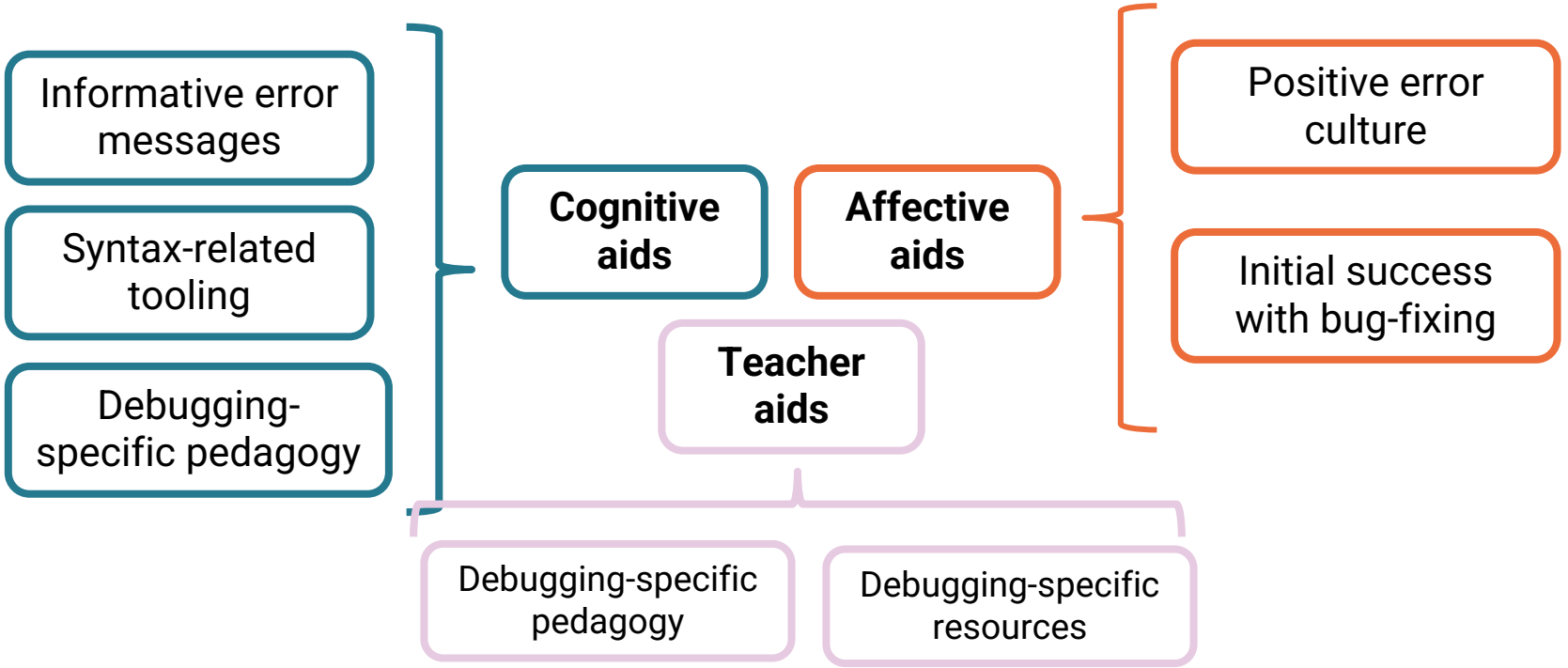
- Attitudes and emotions to debugging are interlinked
- Self-efficacy is particularly interlinked

Lessons Learnt

Study 2

- Teachers can’t help every student to debug
- *“A more scalable model is needed”*
- Teachers with debugging scaffolding in place fare better

How can research help?



Some questions to ponder

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

How do we best teach debugging within introductory programming? Separate lessons? Tooling?

What role does GenAI have to play in all this?

UNIVERSITY OF CAMBRIDGE

Raspberry Pi Foundation

Thanks for listening!

Any questions?

References

1. Computing programmes of study: key stages 3 and 4 National curriculum in England Purpose of study [2013], Technical report, Department for Education. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239067/SECONDARY_national_curriculum_-_Computing.pdf
2. Sweller, J. [1988], 'Cognitive load during problem solving: Effects on learning', Cognitive Science 12(2), 257–285.
3. Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). Association for Computing Machinery, New York, NY, USA, 480–486. https://doi.org/10.1145/3341525.3387384
4. D. N. Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. 1986. Conditions of Learning in Novice Programmers. Journal of Educational Computing Research 2, 1 (2 1986), 37–55
5. Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An analysis of patterns of debugging among novice computer science students. In ACM SIGCSE Bulletin, Vol. 37. Association for Computing Machinery, New York, NY, USA, 84–88.
6. Zhongxiu Liu, Rui Zhi, Andrew Hicks, and Tiffany Barnes. 2017. Understanding problem solving behavior of 6–8 graders in a debugging game. Computer Science Education 27, 1 (1 2017), 1–29.
7. 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
8. Veronica Cuicat and Jane Waite, 2024. A feedback literacy perspective of secondary educators' views of LLM explanations of program error messages for classroom use (pilot study) (in press)
9. Michael Kölling, Neil C. C. Brown, and Amjad Altadmri. 2015. Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15). Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/2818314.2818331
10. Carver, S. M. and Risinger, S. C. [1987], Improving children's debugging skills, in 'Empirical studies of programmers: second workshop', pp. 147–171.
11. URL: https://www.researchgate.net/publication/262311647Michaeli, T. and Romeike, R. [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, pp. 1–7. URL: https://doi.org/10.1145/3361721.3361724
12. 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
13. T. Michaeli and R. Romeike, "Current Status and Perspectives of Debugging in the K12 Classroom: A Qualitative Study," 2019 IEEE Global Engineering Education Conference (EDUCON), Dubai, United Arab Emirates, 2019, pp. 1030-1038, doi: 10.1109/EDUCON.2019.8725282.
14. 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
15. Alex Lishinski, Aman Yadav, and Richard Enbody. 2017. Students' emotional reactions to programming projects in introduction to programming: Measurement approach and influence on learning outcomes. In ICER 2017 - Proceedings of the 2017 ACM Conference on International Computing Education Research. Association for Computing Machinery, Inc, New York, NY, USA, 30–38. https://doi.org/10.1145/3105726.3106187
16. Alex Lishinski and Aman Yadav. 2019. Motivation, Attitudes, and Dispositions. In The Cambridge Handbook of Computing Education Research. Cambridge University Press, Cambridge, UK, 801–826
17. "After the reboot: computing education in uk schools," Tech. Rep. 978-1-78252-297-3, The Royal Society, London, November 2017.

A thanks to FLATICON for the icons used in the presentation