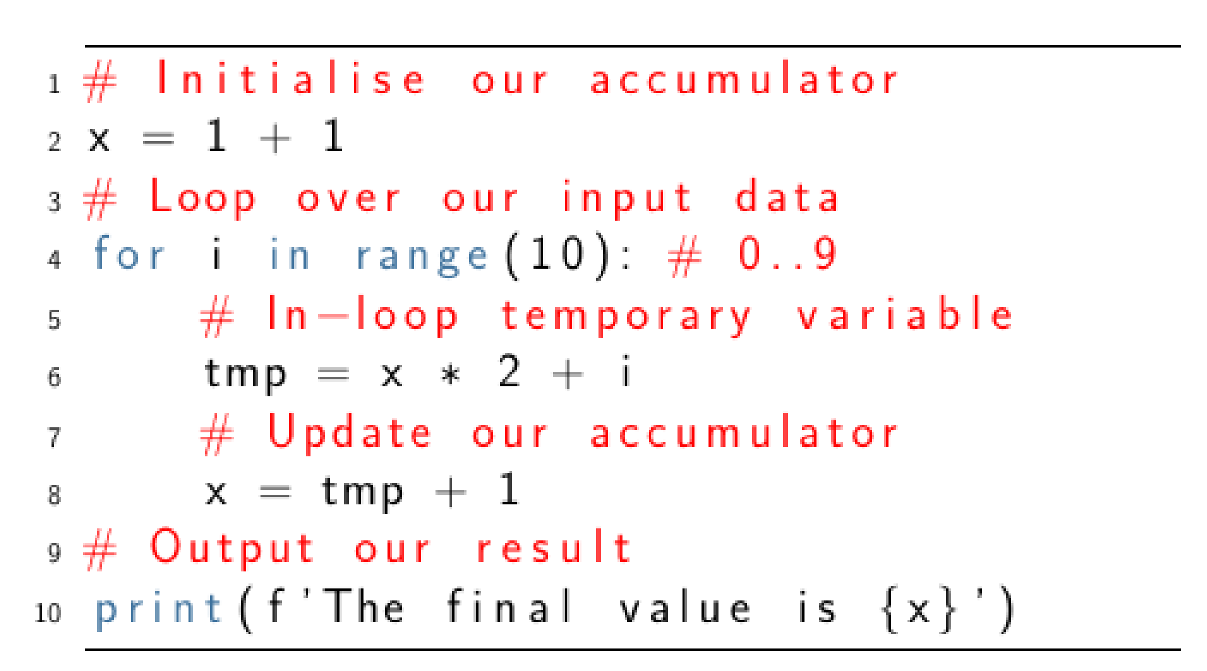
DEBUGGING, TRACING, AND FADING: IMPROVING STUDENT'S MENTAL MODEL OF CODE

Problem

Students have a poor working model of code which leads to numerous issues: they struggle to debug, to adjust copy-pasted code, and to predict runtime behaviour.



```
# Initialise our accumulator
x = 1 + 1
# Loop over our input data
for i in range(10): # 0..9
    # In-loop temporary variable
    tmp = x * 2 + i
    # Update our accumulator
    x = tmp + 1
# Output our result
print(f'The final value is {x}')
```

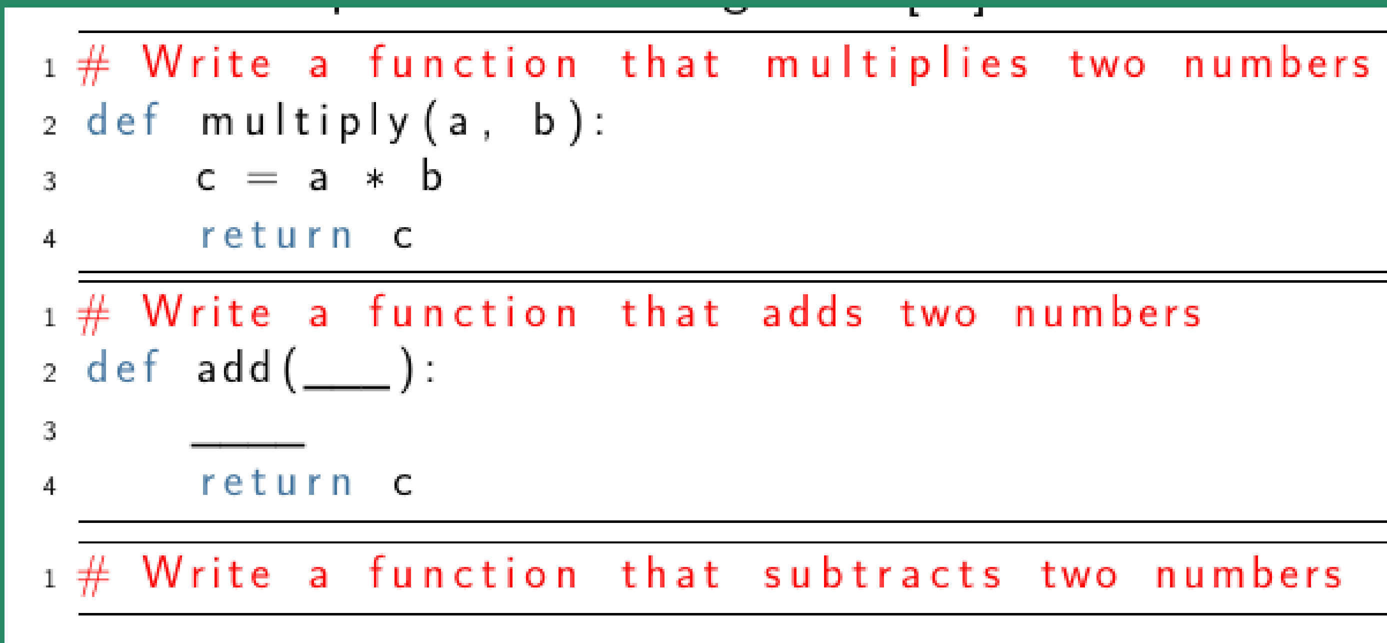
Tracing

"Tracing" is a valuable and easy to complete exercise, and the results can even be checked automatically. It improves student's **mental model**

Faded Examples

As students become more confident, we remove information.

A continuum of problem types!



```
# Write a function that multiplies two numbers
def multiply(a, b):
    c = a * b
    return c
```

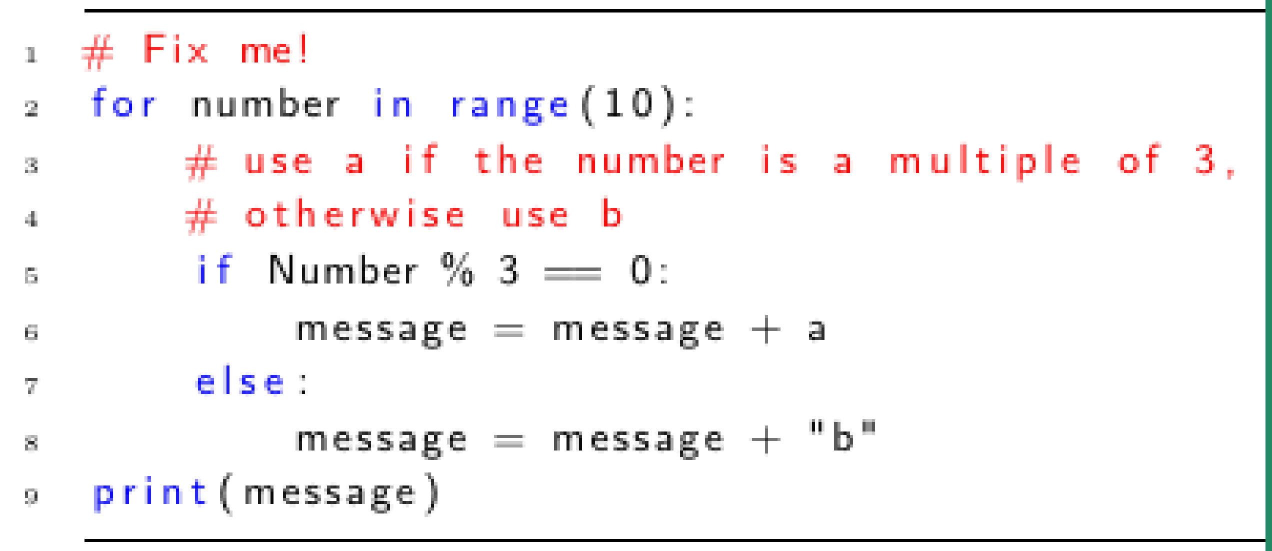
```
# Write a function that adds two numbers
def add(____):
    _____
    return c
```

```
# Write a function that subtracts two numbers
```

Hypothesis

We should augment lessons with:

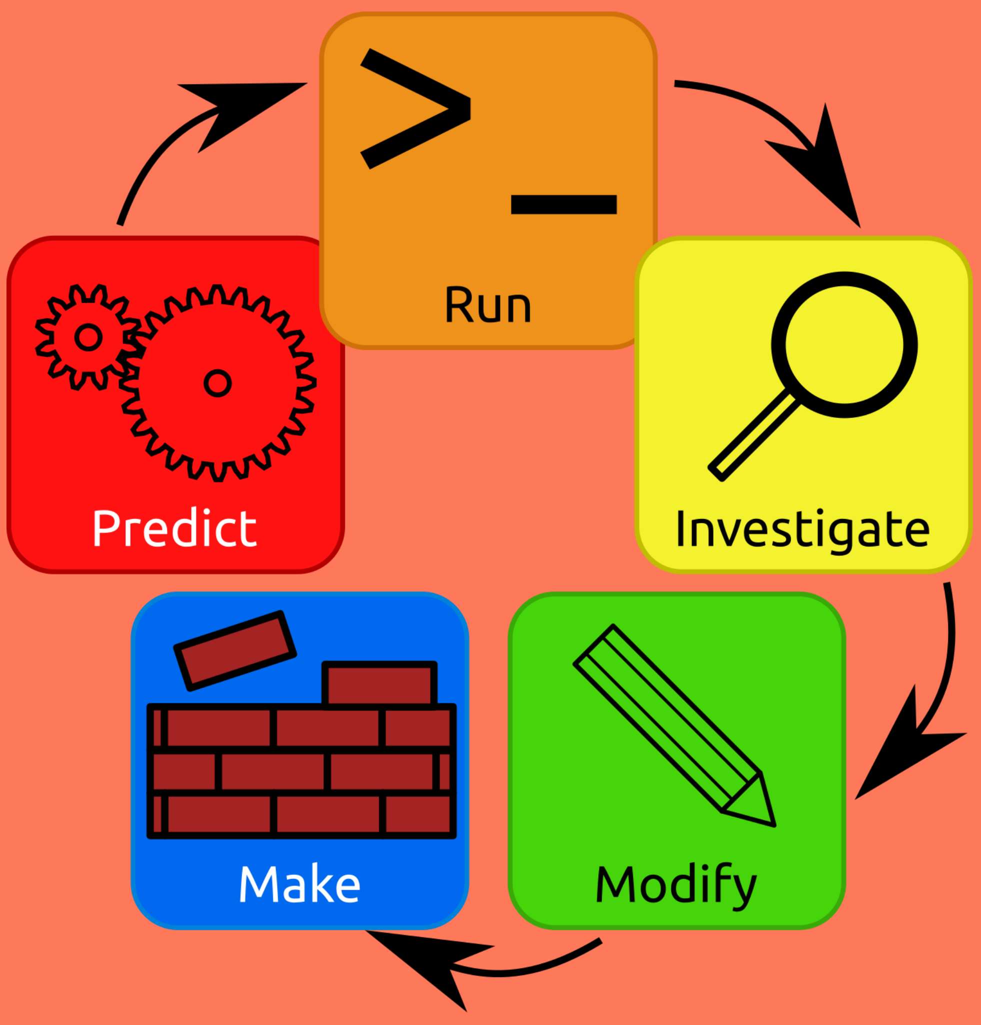
- Tracing - Stepping through the internal state
- Faded examples
- Debugging intentionally broken examples



```
# Fix me!
for number in range(10):
    # use a if the number is a multiple of 3,
    # otherwise use b
    if Number % 3 == 0:
        message = message + a
    else:
        message = message + "b"
print(message)
```

Debugging

Armed with techniques like the "Wolf Fence", students find bugs on their own



PRIMM: A similar model in K-12 education

- Cliburn, D. C. (2003). Experiences with pair programming at a small college. Journal of Computing Sciences in Colleges, 19(1), 20–29.
- Gauss, E. J. (1982). The Wolf Fence algorithm for debugging. Communications of the ACM, 25(11), 780. https://doi.org/10.1145/358690.358695
- Hannay, J. E., Sjoberg, D. I. K., & Dyba, T. (2007). A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Transactions on Software Engineering, 33(2), 87–107. https://doi.org/10.1109/tse.2007.12
- Hertz, M., & Jump, M. (2013). Trace-based teaching in early programming courses. Proceeding of the 44th ACM Technical Symposium on Computer Science Education - SIGCSE '13. https://doi.org/10.1145/2445196.2445364
- McDowell, C., Hanks, B., & Werner, L. (2003). Experimenting with pair programming in the classroom. Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, 60–64.
- Mendes, E., Al-Fakhri, L. B., & Luxton-Reilly, A. (2005). Investigating pair-programming in a 2nd-year software development and design computer science course. Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, 296–300.
- Mendes, E., Al-Fakhri, L., & Luxton-Reilly, A. (2006). A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. ACM SIGCSE Bulletin, 38(3), 108–112.
- Mentz, E., van der Walt, J. L., & Goosen, L. (2008). The effect of incorporating cooperative learning principles in pair programming for student teachers. Computer Science Education, 18(4), 247–260. https://doi.org/10.1080/08993400802461396
- Michaeli, T., & Romeike, R. (2019, October). Improving Debugging Skills in the Classroom. Proceedings of the 14th Workshop in Primary and Secondary Computing Education. https://doi.org/10.1145/3361721.3361724
- Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008). Debugging. ACM SIGCSE Bulletin, 40(1), 163–167. https://doi.org/10.1145/1352322.1352191
- Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education - ITiCSE '04. https://doi.org/10.1145/1007996.1008042
- Renkl, A., Atkinson, R. K., & Große, C. S. (2004). How Fading Worked Solution Steps Works – A Cognitive Load Perspective. Instructional Science, 32(1/2), 59–82. https://doi.org/10.1023/b:truc.0000021815.74806.f6
- Renkl, A., Atkinson, R. K., Maier, U. H., & Staley, R. (2002). From Example Study to Problem Solving: Smooth Transitions Help Learning. The Journal of Experimental Education, 70(4), 293–315. https://doi.org/10.1080/00220970209599510
- Retnowati, E. (2017). Faded-example as a Tool to Acquire and Automate Mathematics Knowledge. Journal of Physics: Conference Series, 824, 012054. https://doi.org/10.1088/1742-6596/824/1/012054
- Sentance, S., & Waite, J. (2017, November). PRIMM. Proceedings of the 12th Workshop on Primary and Secondary Computing Education. https://doi.org/10.1145/3137065.3137084
- Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. Journal on Educational Resources in Computing (JERIC), 4(1), 4-es.
- Wikipedia contributors. (2022). Debugging — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Debugging&oldid=1069955193
- Williams, L. (n.d.). Integrating pair programming into a software development process. Proceedings 14th Conference on Software Engineering Education and Training. "In Search of a Software Engineering Profession" (Cat. No.PR01059). https://doi.org/10.1109/csee.2001.913816
- Williams, Laurie, & Upchurch, R. L. (2001). In support of student pair-programming. ACM SIGCSE Bulletin, 33(1), 327–331. https://doi.org/10.1145/366413.364614
- Williams, Laurie, Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. Computer Science Education, 12(3), 197–212. https://doi.org/10.1076/csed.12.3.197.8618
- Zamary, A., & Rawson, K. A. (2018). Are Provided Examples or Faded Examples More Effective for Declarative Concept Learning? Educational Psychology Review, 30(3), 1167–1197. https://doi.org/10.1007/s10648-018-9433-y