Eric Roberts
CSCI 121

Handout #26
October 1, 2018

Debugging

Debugging

Eric Roberts
CSCI 121
October 1, 2018

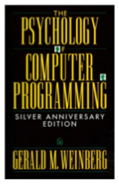
The Discovery of Debugging

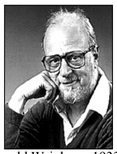
Maurice Wilkes, 1913–2010

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."

—Maurice Wilkes, lecture on "The Design and Use of the EDSAC," September 23, 1979

The Importance of Psychology





Gerald Weinberg, 1933-

For programming is not just human behavior; it is *complex* human behavior.

Although programming is a form—a complex form—of human behavior, few people have studied programming from this point of view. But perhaps there is a reason why programming has not been so viewed? Perhaps programming is too complex a behavior to be studied and must remain largely a mysterious process.

The Psychology of Debugging

First, there is only the *gestalt,* a general feeling that something is out of place without any particular localization. Then follows the ability to shake loose from an unyielding situation—the ability to change one's point of view. . . . Then, however, one must go from the general to the particular—"focusing" as it was called here. Although one does not find errors by a detailed search of each line, word, or character, the ability to get down to details is essential in the end. Thus, for debugging, an almost complementary set of mental powers is needed. No wonder good debuggers are so rare!

—Gerald Weinberg, *The Psychology of Computer Programming,* 1971

Roles in the Programming Process

Programming requires you to assume a variety of roles over the course of the development cycle:

Design	Architect
Coding	Engineer
Testing	Vandal
Debugging	Detective

Sherlock Holmes on Debugging

There is nothing like first-hand evidence.

—*A Study in Scarlet,* 1888

It is a capital mistake to theorise before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

—*A Scandal in Bohemia,* 1892

It is of the highest importance in the art of detection to be able to recognize out of a number of facts which are incidental and which vital. Otherwise your energy and attention must be dissipated instead of being concentrated.

—*The Adventure of the Reigate Squires,* 1892

Literary Sources of Debugging Wisdom

Regard with distrust all circumstances which seem to favor our secret desires.
—Émile Gaboriau, *Monsieur Lecoq,* 1868

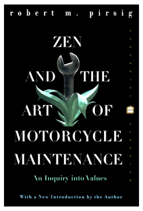
With method and logic one can accomplish anything.
—Agatha Christie, *Poirot Investigates,* 1924

A great detective must have infinite patience. That is the quality next to imagination that will serve him best. Indeed, without patience, his imagination will serve him but indifferently.
—Cleveland Moffett, *Through the Wall,* 1909

Detection requires a patient persistence which amounts to obstinacy.
—P. D. James, *An Unsuitable Job for a Woman,* 1972

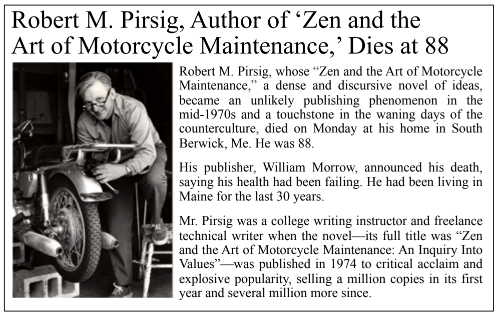
It was always more difficult than you thought it would be.
—Alexander McCall Smith, *The No. 1 Ladies' Detective Agency,* 1998

And Best of All . . .


robert m. pirsig
ZEN AND THE ART OF MOTORCYCLE MAINTENANCE
An Inquiry into Values
With a New Introduction by the Author

The best discussion I have ever encountered of the psychology of debugging occurs in *Zen and the Art of Motorcycle Maintenance* by Robert Pirsig. In Chapter 26, Pirsig introduces the idea of the ***gumption trap***, a psychological barrier that stands in the way of understanding the solution to a problem. Gumption traps occur frequently in computing, usually in the context of the debugging phase.

In Memoriam



Robert M. Pirsig, Author of 'Zen and the Art of Motorcycle Maintenance,' Dies at 88

Robert M. Pirsig, whose "Zen and the Art of Motorcycle Maintenance," a dense and discursive novel of ideas, became an unlikely publishing phenomenon in the mid-1970s and a touchstone in the waning days of the counterculture, died on Monday at his home in South Berwick, Me. He was 88.

His publisher, William Morrow, announced his death, saying his health had been failing. He had been living in Maine for the last 30 years.

Mr. Pirsig was a college writing instructor and freelance technical writer when the novel—its full title was "Zen and the Art of Motorcycle Maintenance: An Inquiry Into Values"—was published in 1974 to critical acclaim and explosive popularity, selling a million copies in its first year and several million more since.

—Paul Vitello, *The New York Times*, April 24, 2017

Lest the Title Put You Off

What follows is based on actual occurrences. Although much has been changed for rhetorical purposes, it must be regarded in its essence as fact. However, it should in no way be associated with that great body of factual information relating to orthodox Zen Buddhist practice. It's not very factual on motorcycles, either.

—Robert Pirsig, "Author's Note," *Zen and the Art of Motorcycle Maintenance*

Example of a Gumption Trap

Of the value traps, the most widespread and pernicious is value rigidity. This is an inability to revalue what one sees because of commitment to previous values. . . .

The typical situation is that the motorcycle doesn't work. The facts are there but you don't see them. . . .

This often shows up in premature diagnosis, when you're sure you know what the trouble is, and then when it isn't, you're stuck. Then you've got to find some new clues, but before you can find them you've got to clear your head of old opinions. If you're plagued with value rigidity you can fail to see the real answer even when it's staring you right in the face because you can't see the new answer's importance.

—Robert Pirsig, *Zen and the Art of Motorcycle Maintenance,* 1971

Gumption Traps in Debugging

- As an illustration of the kind of "gumption trap" that novices encounter when debugging, most students when faced with a bug set out to determine why their program ***isn't*** doing what they want it to do.
- In general, this strategy is less effective than trying to figure out why the program ***is*** doing what it in fact does. In all likelihood, understanding what the program is actually doing provides precisely the insight necessary to repair it.

Debugging Strategy #1

Concentrate on what your program *is* doing rather than on what it *isn't* doing.

- This strategy arises from one of the most important cognitive blocks that arises in debugging, particular for novices. When faced with a program that isn't working, most students set out to determine why their program ***isn't*** doing what it is supposed to be doing.
- In general, this strategy is less effective than trying to figure out why the program ***is*** doing what it in fact does. In all likelihood, understanding what the program is actually doing provides precisely the insight necessary to repair it.

Debugging Strategy #2

Let Python tell you what your program is doing. The **`print`** function is your friend.

- One of the reasons that it is better to focus on what your program is actually doing is that Python can help you with that process.
- Use the **`print`** function to display the contents of variables to see whether they have the values you expect.

Debugging Strategy #3

Pay attention to what Python tells you in its error messages.

- Although Python's error messages may seem cryptic, it is important to listen to what they are trying to tell you.
- Several students coming to office hours were stuck on errors that were clearly reported in the error messages. They simply didn't look at them.

Debugging Strategy #4

Divide your program into pieces that are small enough to test them as you go.

- Trying to write your entire program first and then get it working is a recipe for disaster. What you need to do instead is break it down into smaller pieces that you can then test independently.

Debugging Strategy #5

Try to minimize the need for debugging by checking your program carefully.

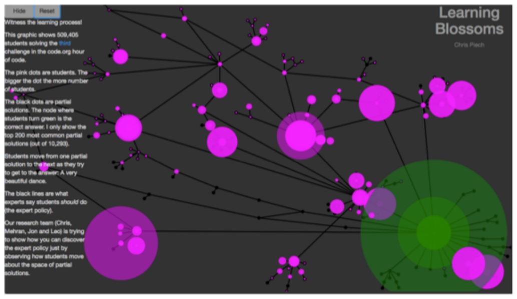
- Now that running a program is essentially instantaneous, many students get into the habit of trying to run a program before having convinced themselves that it works by checking the code as carefully as possible.
- Imagine that you were back in the days when running your program happened overnight. If making a silly mistake can cost you a day, you pay more attention to getting things right.

Debugging Strategy #6

Don't make random changes to your code hoping to get it working.

- One of the most dangerous habits that I see students fall into these days is making random changes to their programs hoping that the most recent change will fix things.
- The change-until-it-works strategy is at best totally inefficient and at worst completely unworkable.
- Whenever you make a change, you should understand exactly why you are making it. You should always have a plan.

Chris Piech's Learning Blossoms



Witness the learning process!

This graphic shows 509,405 students solving the third challenge in the code.org hour of code.

The pink dots are students. The bigger the dot the more number of students.

The black dots are partial solutions. The node where students turn green is the correct answer. I only show the top 200 most common partial solutions (out of 10,293).

Students move from one partial solution to the next as they try to get to the answer. A very beautiful dance.

The black lines are what experts say students should do (the expert policy).

Our research team (Chris, Mehran, Jon and Leo) is trying to show how you can discover the expert policy just by observing how students move about the space of partial solutions.