Bug-fixing as a course section

My own Response to Dijkstra

Largely accept Dijkstra; teach programming using pen and paper.

Use reading, discussing, writing, not experimenting to learn.

Teach "bug-fixing" as a separate section of the course.

Should teach it.

Should assess it.

Must decide which 25-33% of the course to drop to make room.

"Bug fixing" (2) : the basic reasons

- Show students the difference between RDW and "bug-fixing", and how they relate / contrast.
- The argument about RDW → and perhaps now RDW+B
- It is a graduate attribute – general life skill.
- [my garage job] Many things we can, and must, succeed without understanding why.

"Bug fixing" (3): an example

My desktop Mac was nagging me to install software updates.
I believed it was OS patches, but was an update to iTunes.
On reboot, my Safari browser said "this app cannot be opened"

<flailed around. Googled about how to reinstall Safari. Long time.
Didn't fix it.
Consulted my friend Joe. He googled around; but more successfully.
Online was a 3 line fix that took about 3 mins and was completely successful (and didn't need a reboot).

It was a bug in iTunes, which I don't use; but which disabled something I rely on.

"Bug fixing" (4): Features of the case

Much of the report on the case is the view with hindsight.
But when the bug manifests, you often have no idea what it is, where it is, and where to look for a solution.

Asking a friend AND consulting google/the web were THE key techniques.

Addressing "bugs" is full of estimates about trustworthiness.

And estimates of what approach to use in looking for a fix.

And estimates of whether an applied fix is good enough for you.

Where in CompSci are these, or even this issue, formally taught and assessed?