Real Pitfall: Unlikely Scenarios

Code can also "hide":
- possibly unnecessary and/or broken, but
- **difficult to execute**, so no tests cover it.

Methodologies exist for this type of problem:
- **make scenarios likely** in a debugger
- **inject failures** externally (a tool causes the unlikely scenarios), and
- concolic testing tools **try to find inputs that cover all paths**.

These techniques are beyond our class' scope.*

*The automated feedback tool uses concolic testing.

9

Performing Tests is Easier if One Thinks Ahead

How does one perform tests?

Best answer:
design the code to be easily testable!

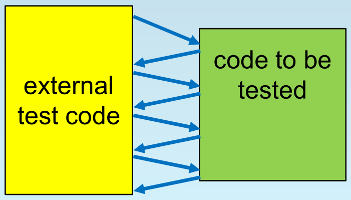
One can also
- use scripts to transform code and expose details, but
- it's better to make the tester's life easy.
- Good testing is challenging enough.

If you write some tests first,
you'll be forced to write more testable code.

10

Best Practice: Write Code that Uses the Code Under Test

Option 1:
- external code calls code to be tested and
- inspects state/results.

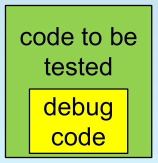


Best practice, but sometimes hard to test thoroughly.

11

Also Common: Add Debug Code to Code Under Test

Option 2:
- **extend** code to be tested **with debug code**,
- then, during development and testing,
- use debug version (with extra code).



Common approach, but code usually shipped with debug code left in place—removing it is questionable.

12