CSCA08 Fall 2016

Week 11 - Testing

Brian Harrington & Thierry Sans

University of Toronto Scarborough

November 21 - 25, 2015

UNIVERSITY OF TORONTO SCARBOROUGH

Admin Stuff

- TT2
  - Back in tutorial this week
  - Unless you didn't check off your tutorial number

DocTest

- We've already been using this
- Good for quick sanity checks
- Tedious to do more than the basics
- Somewhat difficult for sets/dictionaries
- Doesn't work at all for i/o or OOP
- Really just testing "outside" the code

```
import unittest
import example_functions as func
class TestCommonChar(unittest.TestCase):
    def test_identical_single_char(self):
        self.assertEqual(
            func.common_chars('a', 'a'),
            (1,1),
            "identical single char"
        )
unittest.main(exit = False)
```

- One class per function to test.
- One method per test case (name must start with 'test')
- assertEqual method (fails if first two parameters aren't equal)
  - Example:
  - Return value of the function
  - Expected return value
  - Message for if/when error occurs

Break


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
© 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

Black Box Testing

- Imagine function as a “black box”
  - Can’t see in/out
  - Can only see what goes in/what comes out
  - Try to cover all major test areas + boundary cases
  - Testing that could be done by external user

White Box Testing

- Now we can “see inside” the box
  - Shouldn’t it be “clear box”?
- Can test for weaknesses specific to implementation details
- More focused testing
- When implementation details change, tests must change also
- Testing done by internal user

Coverage Testing

- Try to cover all possible scenarios
- Exhaustive testing too tedious/difficult
- Break up “test-space” into areas
- Pick representative examples from each area
- Pick examples from boundaries between areas

Adversarial Testing

- Try to break your (or preferably someone else's code)
- Usually white box
- Often monetary incentives
- Motivated to find bugs

Regression Testing

- When you make a change, check that you haven't introduced a bug to other code
- Built new test cases on top of old, run everything
- Costly, but effective
- Individual cases can be derived through other methods

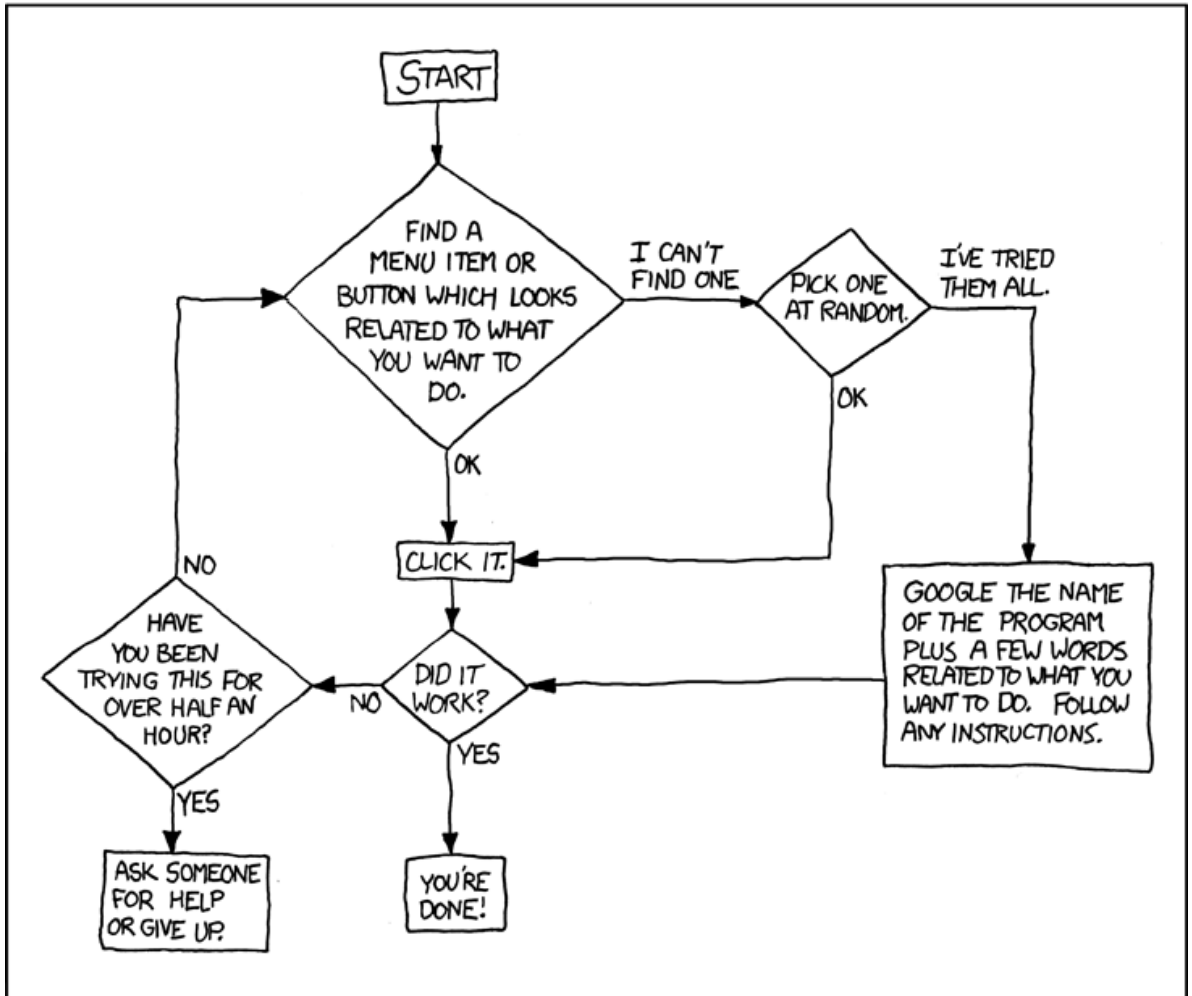
Testing Levels

- Unit - Test individual components
- Integration - Putting components together
- System - System as a whole
- Acceptance - Testing with users
- Release - Testing in the real world
  - alpha (select group, expecting buggy code
  - beta (larger group, expect mostly working code)
  - full-release (it better be working by this point)
- Rule of 10: As we move ahead 1 level, difficulty/cost of repairing a logic error multiplies by ~10x

Break

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



Debugging/Verification

- Debugging
  - Tool assisted
  - Print Debugging
  - Wolf-fence algorithms
  - Post-mortem
  - Rubber Duck debugging
- Verification
  - Formal verification
  - Verification vs Validation
  - Code Review

Some Programming Paradigms/Principles/Ideas

- Software development models
  - Waterfall
  - V-Models
  - Spiral Models
  - Prototyping
    - Vertical vs Horizontal
  - Meta-models
    - Agile
    - Lean
    - DevOps