Quality Assurance

- What it is
- Why it matters to you

Tiziano Zito

Off Topic: The UI

Off Topic: The UI

If programming is symbol manipulation, then you should remember that the user interface is also symbol manipulation, and it is a <u>much</u> harder problem than databases, sorting, searching, and all the other problems you learn about in academia.

Off Topic: The UI

If programming is symbol manipulation, then you should remember that the user interface is also symbol manipulation, and it is a much harder problem than databases, sorting, searching, and all the other problems you learn about in academia.

The user interface has to communicate over a rich but noisy channel using multiple under-specified protocols to a couple of pounds of meat which processes information using buggy heuristics evolved over millions of years to find the ripe fruit, avoid being eaten, and have sex. If you think getting XML was hard, that's nothing compared to user interfaces.

Off Topic: The UI

If programming is symbol manipulation, then you should remember that the user interface is also symbol manipulation, and it is a <u>much</u> harder problem than databases, sorting, searching, and all the other problems you learn about in academia.
The user interface has to communicate over a rich
but noisy channel using multiple under-specified protocols to a couple of pounds of meat which processes information using buggy heuristics evolved over millions of years to find the ripe fruit, avoid being eaten, and have sex. If you think getting XML was hard, that's <u>nothing</u> compared to user interfaces.

The fact that even bad UIs work at all is a credit to the heuristics, bugs and all, in the meat.

Steven D'Aprano

What it is

- QA are planned and systematic programming techniques that provide confidence in a software's suitability for its intended purpose and its reliability
- Key principles:
  - fit for purpose
  - right first time

What it is *not*

- QA cannot absolutely guarantee the production of *quality* software

but

What it is *not*

- QA cannot absolutely guarantee the production of *quality* software

  but

  makes this more likely!

Quality is *not* just testing

- ***Trying to improve the quality of software by doing more testing is like trying to lose weight by weighing yourself more often***
- **Quality is designed in**
- **Quality is monitored and maintained through the whole software lifecycle**

Basic Techniques

- Catching errors:
  try/except → exception handler

Basic Techniques

- Catching errors:
  try/except → exception handler
- Exception hierarchies:

```
Exception
          ArithmeticError
                         FloatingPointError
                         OverflowError
                         ZeroDivisionError
          IndexError
          TypeError
          ValueError
          EnvironmentError
                          IOError
                          OSError
```

Basic Techniques

- Catching errors:
  try/except → exception handler
- Exception hierarchies:

```
Exception
          ArithmeticError
                         FloatingPointError
                         OverflowError
                         ZeroDivisionError
          IndexError
          TypeError
          ValueError
          EnvironmentError
                          IOError
                          OSError
```

- Use exceptions to report errors, resist the temptation of returning special values: -1, False, None, ...

Advanced Techniques

- Test-Driven Design/Development → Day1
- Design by contract

Advanced Techniques

- Test-Driven Design/Development → Day1
- Design by contract
  - functions carry their specifications around with them:
    * keeping specification and implementation together makes both easier to understand
    * and improves the odds that programmers will keep them in sync

Advanced Techniques

- Test-Driven Design/Development → Day1
- Design by contract
  - functions carry their specifications around with them:
    - keeping specification and implementation together makes both easier to understand
    - and improves the odds that programmers will keep them in sync
  - a function is defined by:
    - *pre-conditions*: what must be true in order for it to work correctly
    - *post-conditions*: what it guarantees will be true if its pre-conditions are met

Advanced Techniques

- Test-Driven Design/Development → Day1
- Design by contract
  - functions carry their specifications around with them:
    - keeping specification and implementation together makes both easier to understand
    - and improves the odds that programmers will keep them in sync
  - a function is defined by:
    - *pre-conditions*: what must be true in order for it to work correctly
    - *post-conditions*: what it guarantees will be true if its pre-conditions are met
  - pre- and post-conditions constrain how the function can evolve:
    - can only ever relax pre-conditions (i.e., take a wider range of input)...
    - ...or tighten post-conditions (i.e., produce a narrower range of output)
    - tightening pre-conditions, or relaxing post-conditions, would violate the function's contract with its callers

Advanced Techniques

- Defensive programming

Advanced Techniques

- Defensive programming
  - specify pre- and post-conditions using assertion:
    * assert(len(input) > 0)
    * raise an AssertionError
  - use assertions liberally

Advanced Techniques

- Defensive programming
  - specify pre- and post-conditions using assertion:
    * assert(len(input) > 0)
    * raise an AssertionError
  - use assertions liberally
  - program as if the rest of the world is out to get you!
  - *fail early, fail often*
  - *the* less distance there is between the error and you detecting it, the easier it will be to find and fix

Advanced Techniques

- **Defensive programming**
  - **specify pre- and post-conditions using assertion:**
    - **assert(len(input) > 0)**
    - **raise an AssertionError**
  - **use assertions liberally**
  - **program as if the rest of the world is out to get you!**
  - ***fail early, fail often***
  - ***the* less distance there is between the error and you detecting it, the easier it will be to find and fix**
  - **it's never too late to do it right**
    - **every time you fix a bug, put in an assertion and a comment**
    - **if you made the error, the right code can't be obvious**
    - **you should protect yourself against someone "simplifying" the bug back in**

Take-home Messages

- The real goal of quality assurance isn't to find bugs: it's to figure out where they're coming from, so that they can be prevented

Take-home Messages

- The real goal of quality assurance isn't to find bugs: it's to figure out where they're coming from, so that they can be prevented
- But without testing, no one (including you) has any right to rely on the program's output

Take-home Messages

- The real goal of quality assurance isn't to find bugs: it's to figure out where they're coming from, so that they can be prevented
- But without testing, no one (including you) has any right to rely on the program's output
- Only way to ensure quality is to design it in