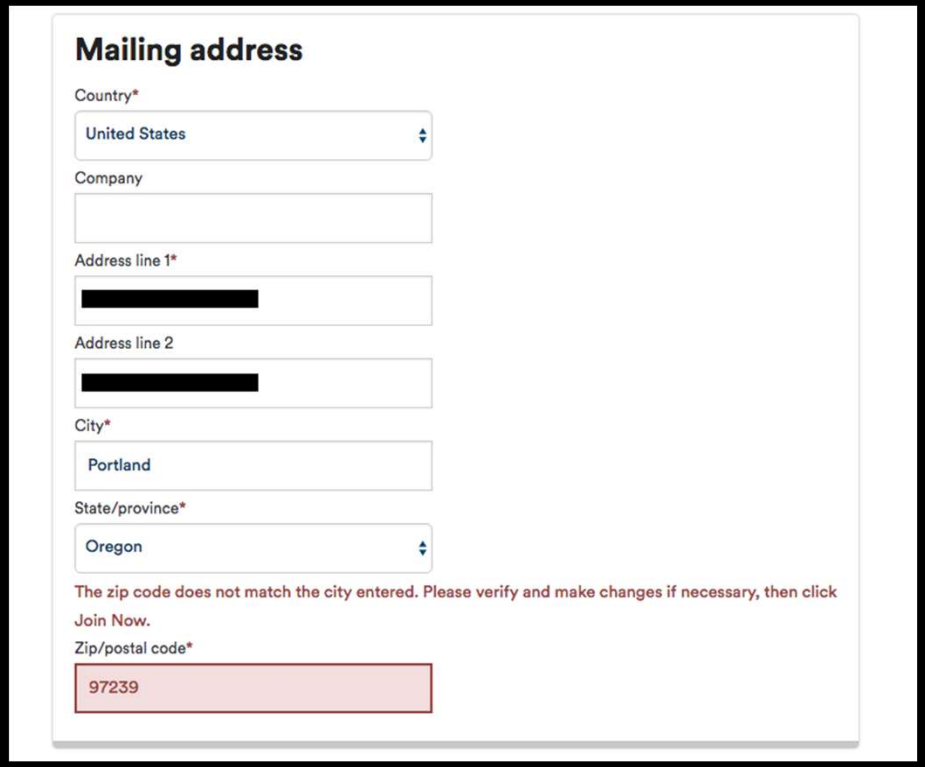
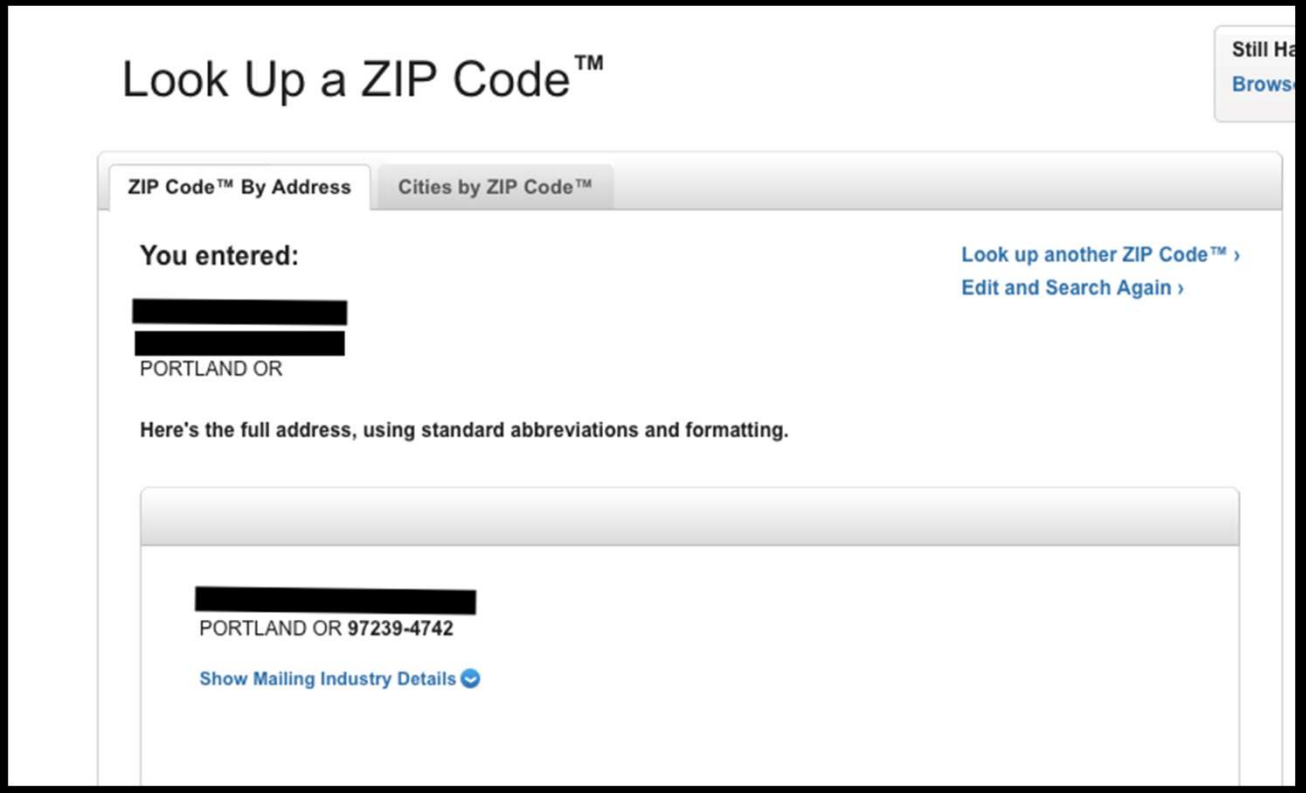
Building Automated Tests for Python Toolbox Tools

(Making My Life Easier)

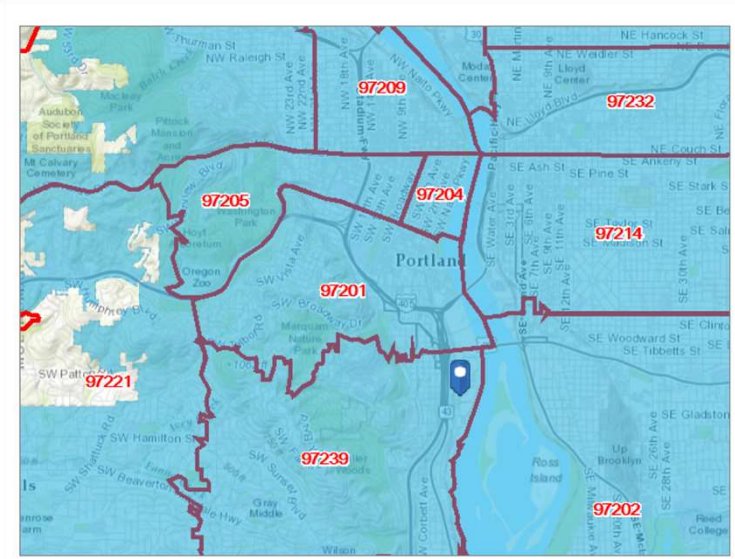
def a_story_about_testing():

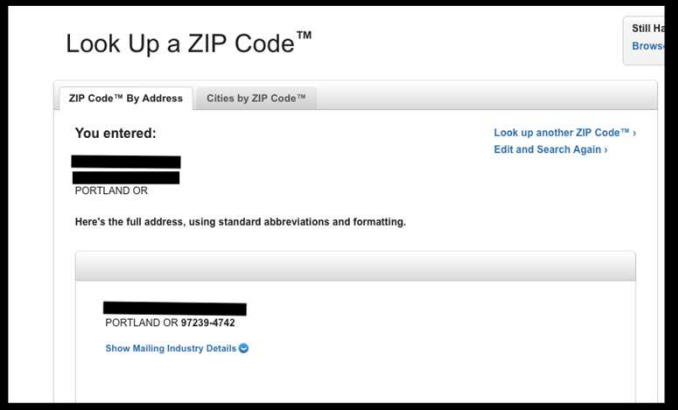


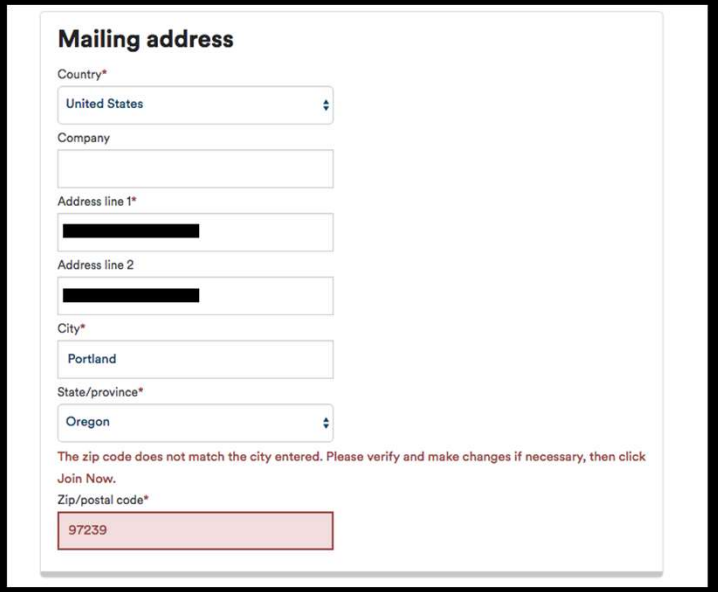
def a_story_about_testing():



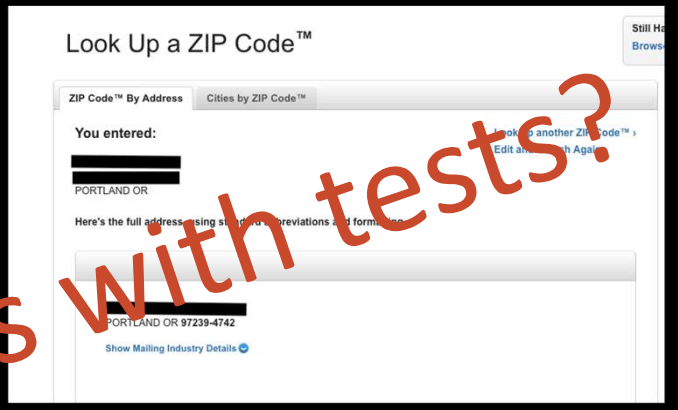
def a_story_about_testing():

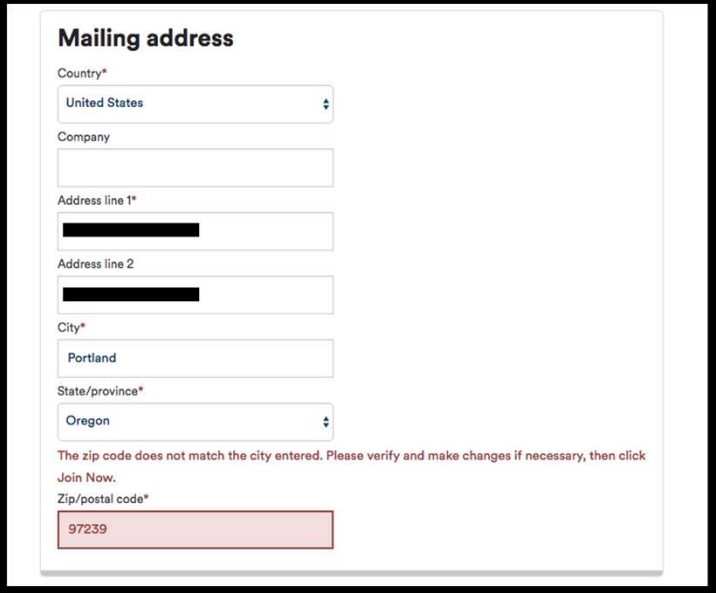






def a_story_about_testing():





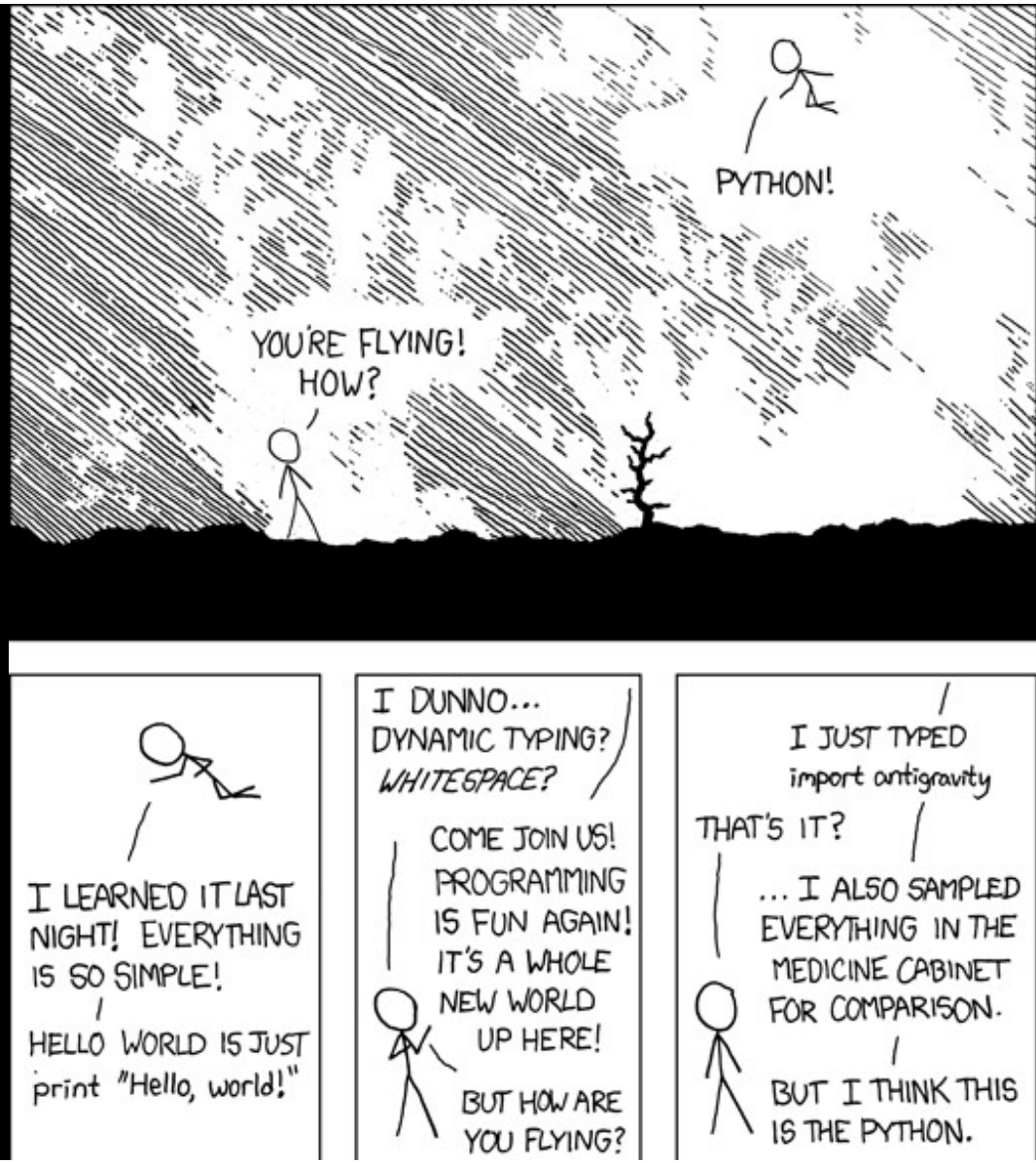
How would we avoid this with tests?

class SimpleTool

- Example ToolBox tool (PYT) for testing
- Inputs:
  - Point Layer with a “Type” column
  - Polygon Layer
- Output:
  - Copy of Polygon Layer with a Count of each Type (only two: RED, BLUE)
- Process:
  - Select Points for a type
  - Do a spatial join
  - Repeat
  - Make output

Show Code

https://xkcd.com/353/



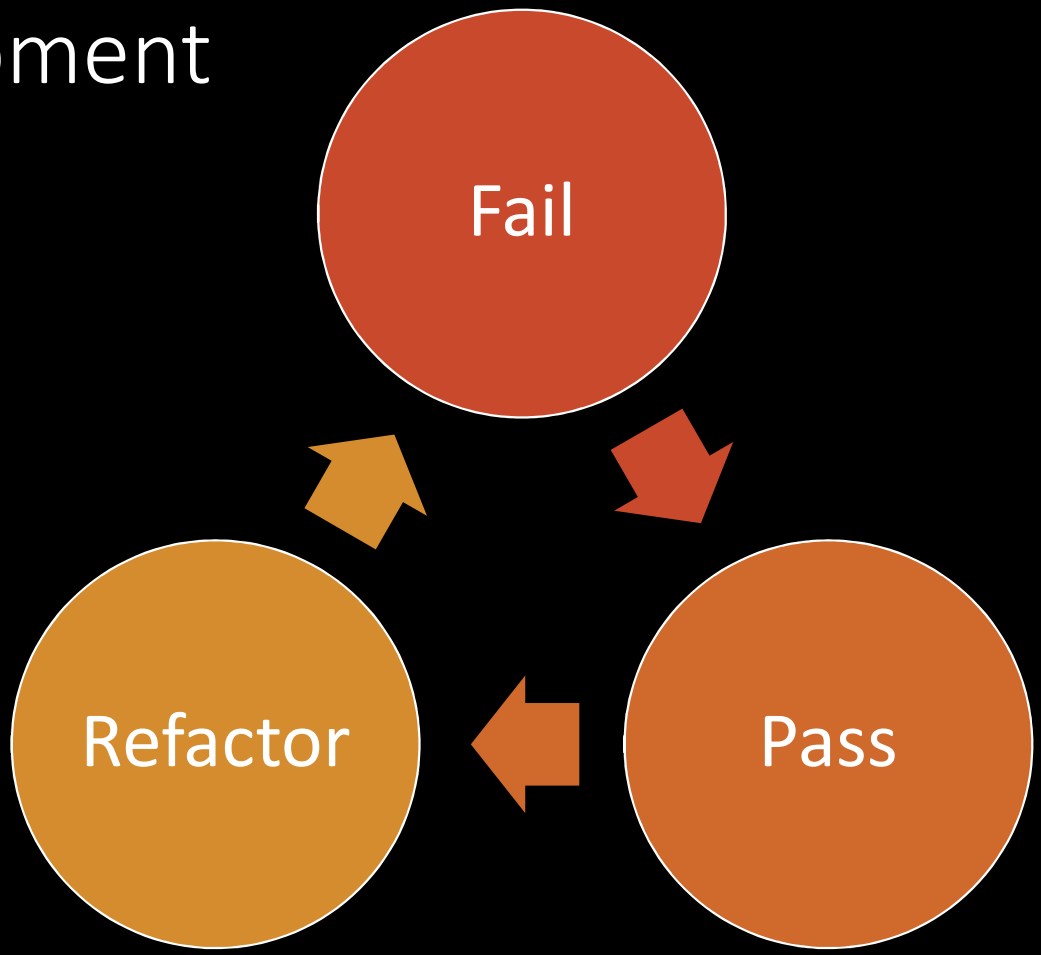
What are “Tests?”

- The process of verifying that software functions as intended. The software should meet the design criteria of functionality, performance and interactivity.
- Types of Tests:
  - Conformance – Does it work the way it is supposed to?
  - Interoperability – Does it work with other things?
  - Performance – Does it work fast enough? (or with less memory, etc.)

https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf

Test Driven Development

- Building tests before you build software
- Build Failing Tests before you fix an issue



Test Coverage vs Code Coverage

Test Coverage

- Amount of testing done by a test
- Do the tests cover all of the "paths" in our code?
- Functions, Methods, Classes, Branches, Loops, Exceptions

Code Coverage

- How much of your code is exercised by a test
- Do the tests cover the use cases in our code?
- Documented test cases

100% Coverage – A Myth?

- No
- Very hard to do, especially in complex software
- Point of diminishing return – eventually the cost (either in dollars or just plain boredom) is no longer worth the return
- “Tested By Relation”
  - One input, which is supposed to test one thing tends to test multiple things
  - Increases “Coverage”
  - Not measurable, except by “gut feel”
  - This can haunt you if it actually hides an issue…

Why bother?

- Pride in my own work
- My sanity
- Preventing Bugs (or at least knowing them)
- Figuring stuff out you haven't touched in years
- Examples
- Being Lazy

https://xkcd.com/303/

My Common Sense Approach

- Test “Core Code” as much as you can stand – strive for 100% coverage
- Test *all* common workflows
- Test edge cases
  - You may be able to group edge cases together
  - Consider alternate paths through code when testing edge cases
- Don’t test (at least not too much) what should already be tested
- Make better error messages

Testing Toolboxes

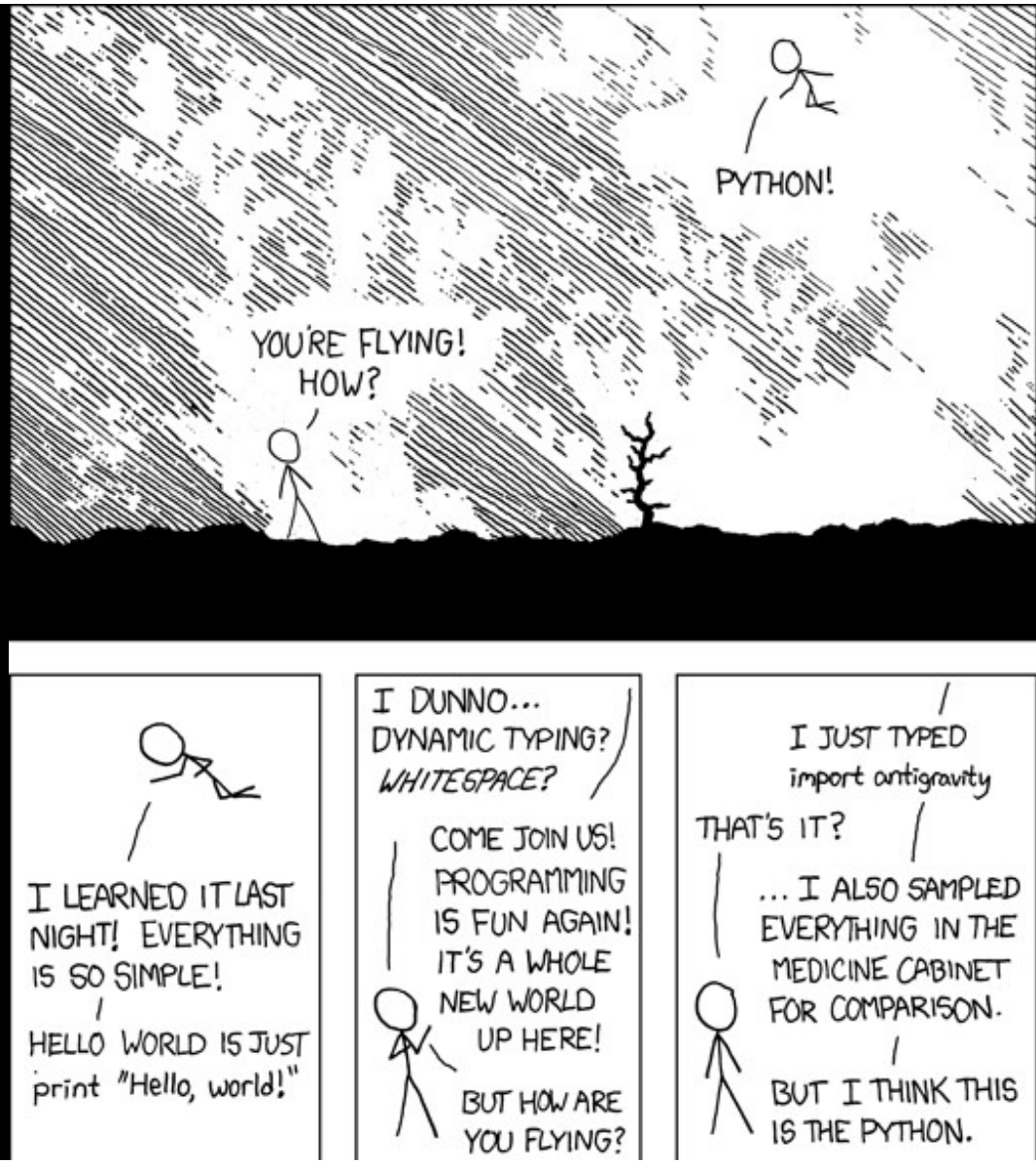
- Leverage Python unittest library
- Keep tests simple
  - Be able to read and understand the test
  - Test only one thing
- Toolboxes take in data and output data
  - Compare Table => assertTablesEqual
  - Compare Feature Class => assertFeatureClassesEqual
- Organize Test Data
- Check Test Data into repository?
- Testing to verify Errors
- Test on real data
- Reuse data where you can, but don’t if it doesn’t work, close enough isn’t good enough
- Record Order!

Test Structure

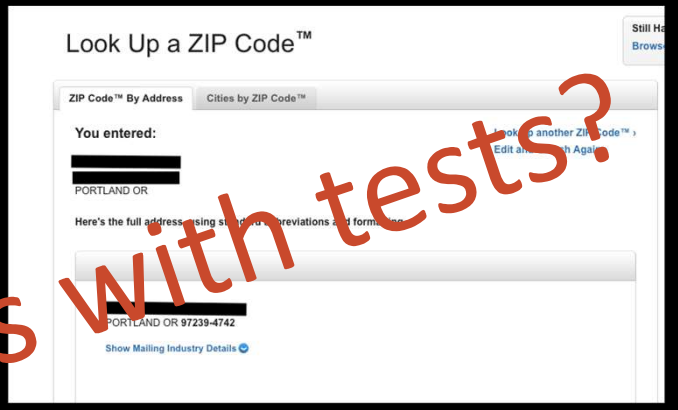
- Gather Data
  - Input
  - A place to put the output (scratch GDB)
  - The answer
- Execute the Tool
- Assert that the answer is equal to the output
- Clean up

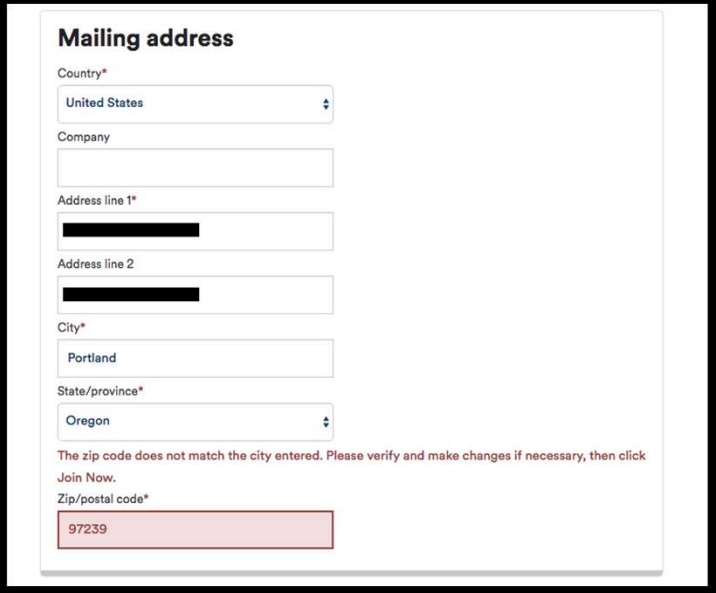
Show Code

https://xkcd.com/353/



def a_story_about_testing():





How would we avoid this with tests?

Going Forward

- Python can run in Pro, Desktop and Server
- Use tests to verify functionality in Pro, Desktop and Server
- Run tests often – Continuous Integration (You can do this with a bat file too)
- Use tests to verify or QC data (i.e. making sure cities and zip-codes are spatially accurate)
- NASA's 10 rules for developing safety-critical code - http://sdtimes.com/nasas-10-rules-developing-safety-critical-code/

Christopher Moravec

Moravec Labs, LLC

Source: https://github.com/MoravecLabs/ArcpyTestExample

Resources: https://goo.gl/BHSmNz

christopher@moraveclabs.com

www.MoravecLabs.com

@mor3havoc