Unit 3: Core tools

Writing unit tests

Lesson 27

Derek Ruths

I DON'T ALWAYS EDIT MY CODE
BUT WHEN I DO, I JUST ASSUME IT WORKS
imgflip.com

Overview of unit

Objectives:
- Understanding how data science activities necessitate certain kinds of tools
- Fundamentals with core data science tools

1. Why core tools?
2. Project organization
3. Python
4. Best practice: write CLI tools
5. Best practice: write unit tests
6. Best practice: resource referencing
7. Github
8. Jupyter notebooks
9. Jupyter & statefulness
10. Bokeh
11. Advanced bokeh
12. HW 3

Lesson overview

Objectives

- Be convinced that unit tests are the best way of ensuring your code still works.

Outline

- Hope everything works, assume everything is broken
- Unit tests

Assume everything you write is broken

How can you prove that your code works? You have to run it!

Unit tests do this quickly, in a reproducible way.

Where is unit testing easy?

import unittest

self.assertTrue(func())

class checkWordlistTests(unittest.testcase)

Scripts, functions … stable code!

Not so much in Jupyter… exploratory code is mushy and hard to test!

eg

```
def check_word_list(word_list: list) -> bool:
    for w in word_list:
        if ' ' in w:
            return False
        elif any([c.isupper() for c in w]):
            return False

    return True
```

```
if __name__ == "__main__":
    unittest.main()
```

runs in unit test file to find all tests

Lesson wrap-up

Takeaways

- Unit tests will make you feel super confident.

from util.py

import unittest

make a class for every method you want to test in tests.py

Up next

- Resource referencing