Python course 2019

book: Obey the Testing Goat!

TDD <=> unit tests

Define the project

- Requirements
- all minimal
- YAGNI (you aren't gonna need it): Think twice before adding features. In doubt leave it out.
- what are the minimal natural features the program should have
- first: command line program

Preparation

- isolation: virtual environments -> you should use them!
  - `python3 -m venv pr-py36`
  - `source pr-py36/bin/activate`
  - deactivate with `deactivate`
- install project `pip install -e .`
- version control systems (VCS)

Implementation

- Make it work: Tests
- Make it right: Clean code. Code is read more often than it is written!
- Make it fast (if you need to)

TDD

- London school double loop TDD approach
- functional tests
- unit tests
- similar to behavior driven development
- we write a story in comments in the tests
- 3 laws of TDD
  - don't write production code until you have a failing unit test
  - don't write more unit tests than is sufficient to fail
  - don't write more production code than is sufficient to pass the failing test
- TDD requires discipline
- unofficial mascot: Testing goat
  - single minded
  - one step at a time goat

Functional Tests

- test application from pov of user. From outside
- failing functional test -> how do I fix this problem -> write unit tests

Unit Tests

- test application from inside
- pov of programmer
- low level
- help you write good, clean, bug free code
- write simple tests
- `unittest.main()` just calls every function in every class(unittest.TestCase)

Workflow

- Write failing functional test
- how do we get it to pass
- start uit tests cycle
  - write one failing unit test
  - write smallest amount of application code to get unit test to pass
  - loop
- check if functional test works now

Spiking

- proof of concept pieces of code

refactoring

- restructuring the code without changing functionality
- make it more readable
- reduce complexity
- you cannot refactor without tests
- general rule: Three strikes and refactor # Generally
- avoid complexity
- setuptools
- entry_points = applications
- don't skip steps! Don't be the refactoring cat

Mocking and Patching

- monkeypatching: In test replace imported function with fake function
- from unittest.mock import Mock -> class where the objects have every attribute and function!

- Mocks are dangerous, avoid using unless needed