Digital Career Institute

Python - Testing

Digital Career Institute

Goal of the Submodule

The goal of this submodule is to familiarize students with why we are using testing in software and how to do basic testing in Python.

Topics

- Errors in Software
- Historic failures of software
- Types of automated test
- How to write tests with unittest
- Overview of assert methods
- Practice TDD
- Why we need stubbing & mocking
- Introduce unittest.mock

Digital Career Institute

Errors in Software

Historic Failures of Software

Digital Career Institute

Software errors / Bug costs

Source: https://raygun.com/blog/cost-of-software-errors/

Failures of Software

Digital Career Institute

Is it possible to build software without errors?

- No software is ever 100% bug-free.
- Building software is an ongoing process.
- Software is affected by external factors:
  - Integration with existing systems
  - Server and hosting
  - Integration with hardware
  - Integration with third-party providers
  - Legacy data formats
  - Scalability
- Normal day for any developer is:
  - Write code for one hour.
  - Spend the rest of the day to fix it.

Some of the largest software bugs

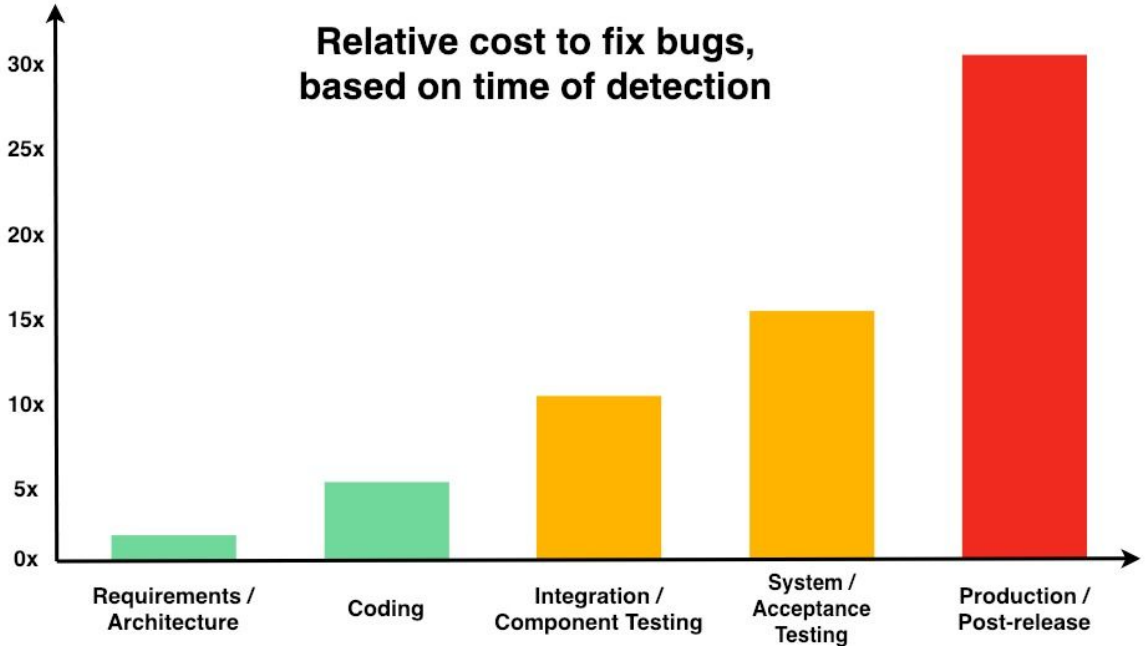
Some software errors have caused considerable material and human damage.

Watch this presentation of some of the largest bugs in history and try answering this question for each of them:

- Could the issue have been prevented?
- If yes, what should have been done to prevent the error from having the consequence it had?
- Could the issue have been prevented by letting a more senior software developer write the code?

Video: https://www.youtube.com/watch?v=AGI371ht1N8

Costs of fixing bugs at various stages



Source: NIST via [Deepsource]

Errors in Software

Digital Career Institute

Error Types in Software:

- **Syntax errors:**

  Can be detected easily because the program will not be able to run or compiled.

- **Run Time errors (bugs):**

  These kind of errors will only show after running the program.

Run Time Errors

1. **Functional errors:** This is a broad type of error that happens whenever software doesn't behave as intended (ex: the program doesn't show outputs).
2. **Logic errors:** represent a mistake in the software flow and causes the software to behave incorrectly (ex: incorrect output or crash).
3. **Calculation errors:** software returns an incorrect value for several reasons (wrong algorithm, data type mismatch, wrong encoding).
4. **Unit-level bugs:** most common and typically the easiest to fix.
5. **System-level integration bugs:** two or more pieces of software from separate subsystems interact erroneously.
6. **Out of bounds bugs:** the user sets a parameter outside the limits of intended use.

Testing Software before Release

Pros	Cons
Bugs will often be found before the software is released.	It takes extra time to test the software.

Software testing: Running the software and checking that it is producing the results that are expected.

Manual testing: Someone other than the developers is running the software prior to launch to see if they can find bugs in them.

Automatic testing: A script is written by a developer and this script runs the software and ensures that it produces the desired results

Testing for the largest software bugs

Thinking of the ten software issues mentioned in the video:

- Which type of error were each of them (functional, logic, calculation, unit-level, system level-integration, out of bounds)?
- Which ones could have been found by testing?
- Which ones would likely have been found by manual testing and which ones by automatic testing? Why?

Pros and Cons of Automated Tests

Pros	Cons
Run quickly by a computer.	The testing scripts have to be written.
Fantastic return on investment. Write them once, then run them many, many times.	The test code has to be maintained.
Help you to maintain your application because the checks become part of the code.	Computers can't creatively find bugs that aren't checked for by the test.
Test can help other developers to understand the program, thus can act as a kind of documentation.	You have to be precise when you write the test scripts.

Pros and Cons of Human / Manual Tests

Pros	Cons
Uses human creativity.	Humans are slow.
Does not require an exact, painfully precise script to be written.	Humans can only do one thing at a time.
Humans can find bugs creatively – exploratory testing.	Humans make mistakes.
	Human time is valuable and you have to worry about whether this is the best use of their time.
	Every time you run the test a human has to be there to run it.

Pro & Cons, Challenges

Pros	Cons
Run quickly by a computer.	Tools still take time.
Reusable as automation process can be recorded.	It is expensive in the initial stage.
More interesting and everyone can see results.	It requires experienced developers to create good tests.
Test scripts can be run unattended.	Not every tool supports all types of testing.

At the Core of the Lesson

Digital Career Institute

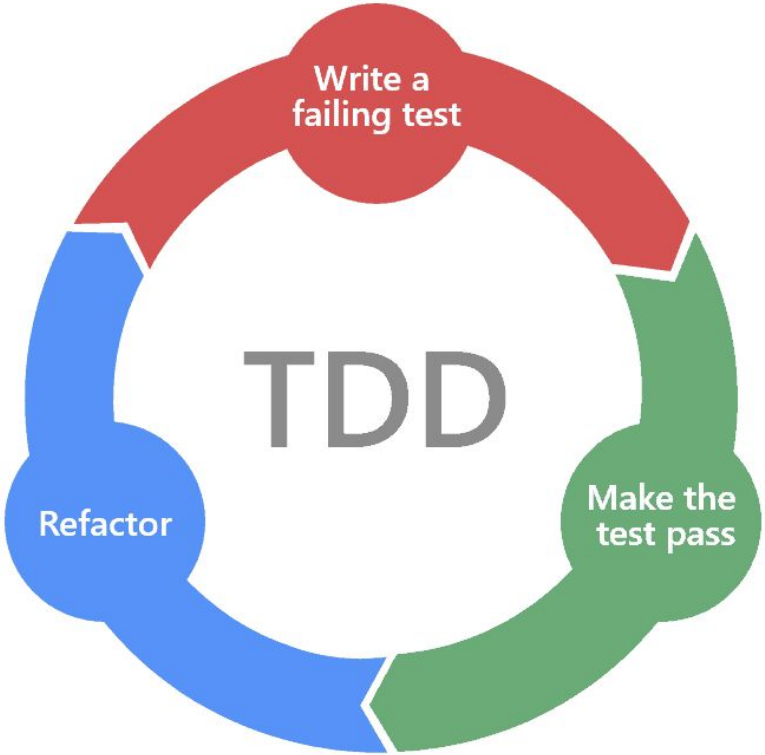
Lessons Learned:

- Error costs
- Historic failures of software
- Errors Types in Software
  - Syntax errors
  - Run Time errors
    - Functional errors
    - Logic errors
    - Calculation errors
    - Unit-level bugs
    - System-level integration bugs
    - Out of bounds bugs
- Automatic and manual testing

Test Driven Development and Types of Automated Tests

Digital Career Institute

Test Driven Development (TDD)



Test Driven Development (TDD)

1. Write tests before you code.
2. Then adjust the code until the tests pass.
3. Refactor code so it's easier to read and doesn't contain unnecessary duplications

Source:
https://medium.com/swlh/should-you-use-tdd-or-not-why-not-both-2e6f58767065

Pro & Cons of TDD

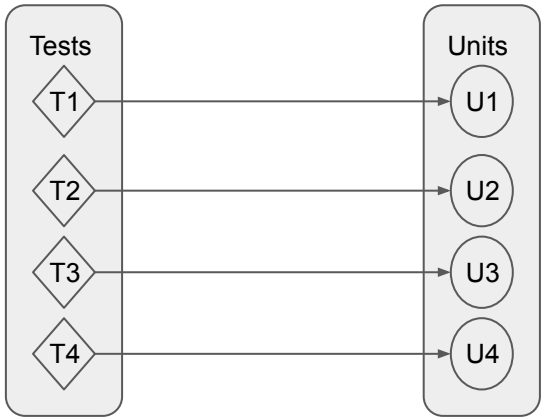
Pros	Cons
Separation of test and code writing means you will not forget testing parts that were hard to code.	Requirements and outside APIs may not be known at the time when code is initially written.
High test coverage: Creates tests for every feature and thereby makes code stable long-term.	It will slow-down the coding process.
	It is effective mostly only if all team members use it.

Types of Automated Tests:

- Unit Test
- Integration Test
- System Test
- Functional Test
- Black Box Test

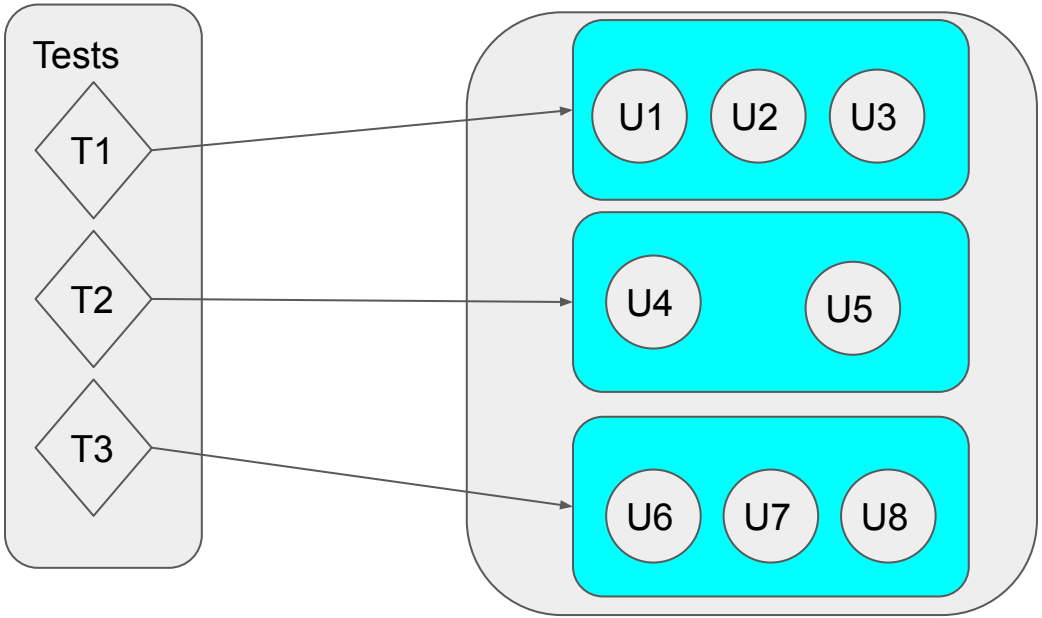
Unit Test

- Test for single unit (function, method)
- Describe what will happen with wrong input
- Describe what the code should or should not do
- Describe what the output should be
- Describe how much time the code should take to run



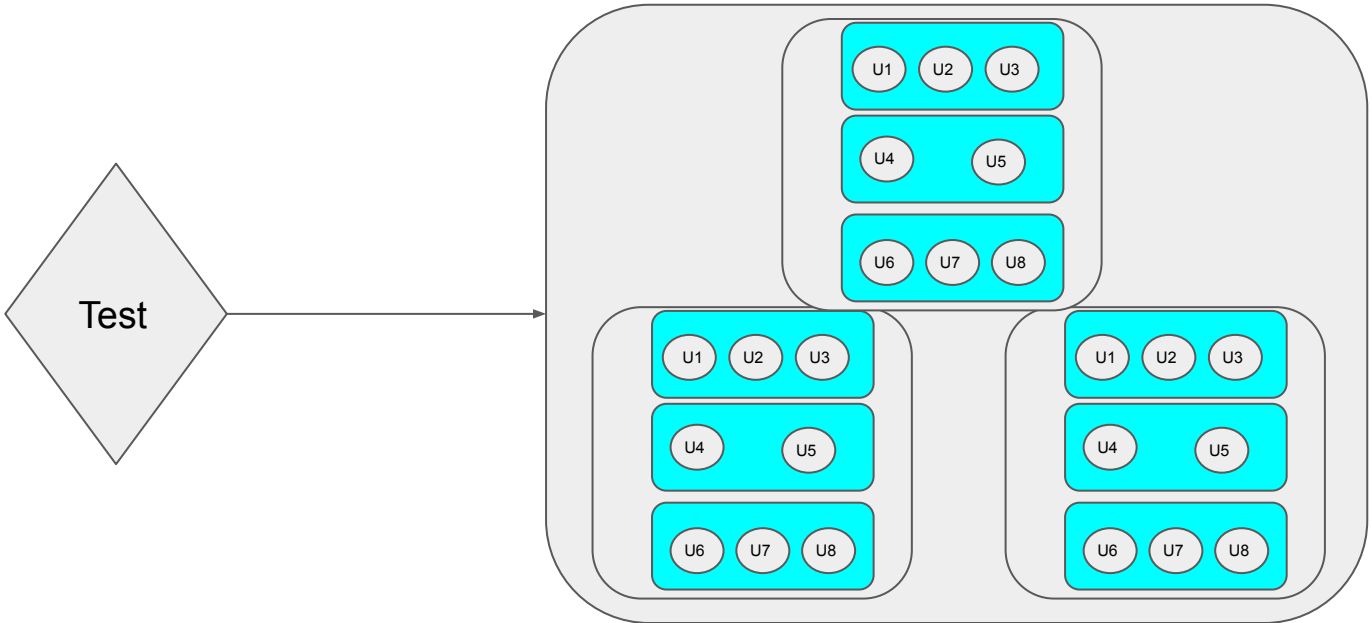
Integration Test

- After unit test
- Determine if independently developed units of software work correctly when they are connected to each other



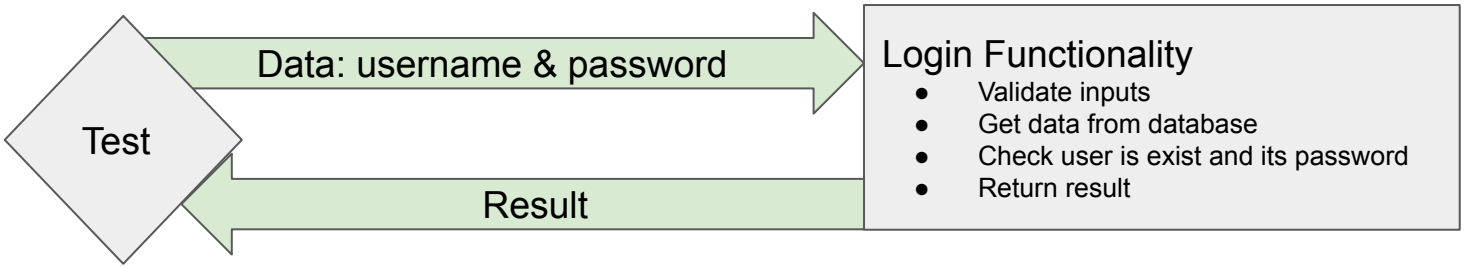
System Test

- After integration test
- Determine if all the component of software work correctly when they are connected to each other and it's called **end to end** test



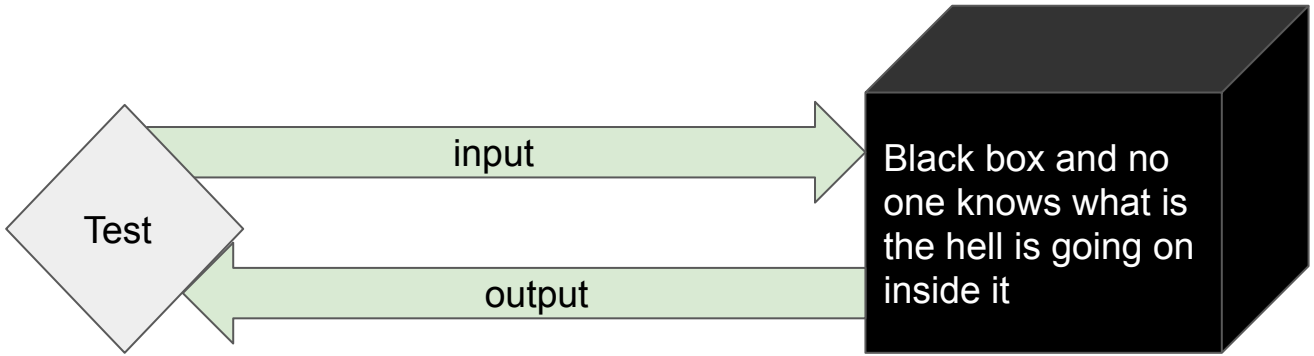
Functional Test

- Validates the software system against the functional requirements/specifications.
- Test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.



Black Box Test

- Test software without having knowledge of internal code structure, implementation details and internal paths.
- Focus on input and output of software applications and it is entirely based on software requirements and specifications.
- It can be an external library or a binary executable file.



At the Core of the Lesson

Digital Career Institute

Lessons Learned:

- Introduction of Test Driven Development (TDD)
- Types of Automated Tests
  - Unit Test
  - Integration Test
  - System Test
  - Functional Test
  - Black Box Test
- Coverage usage