Database Application Testing

IDATG2204 Data Modelling and Database Systems

Norwegian University of Science and Technology

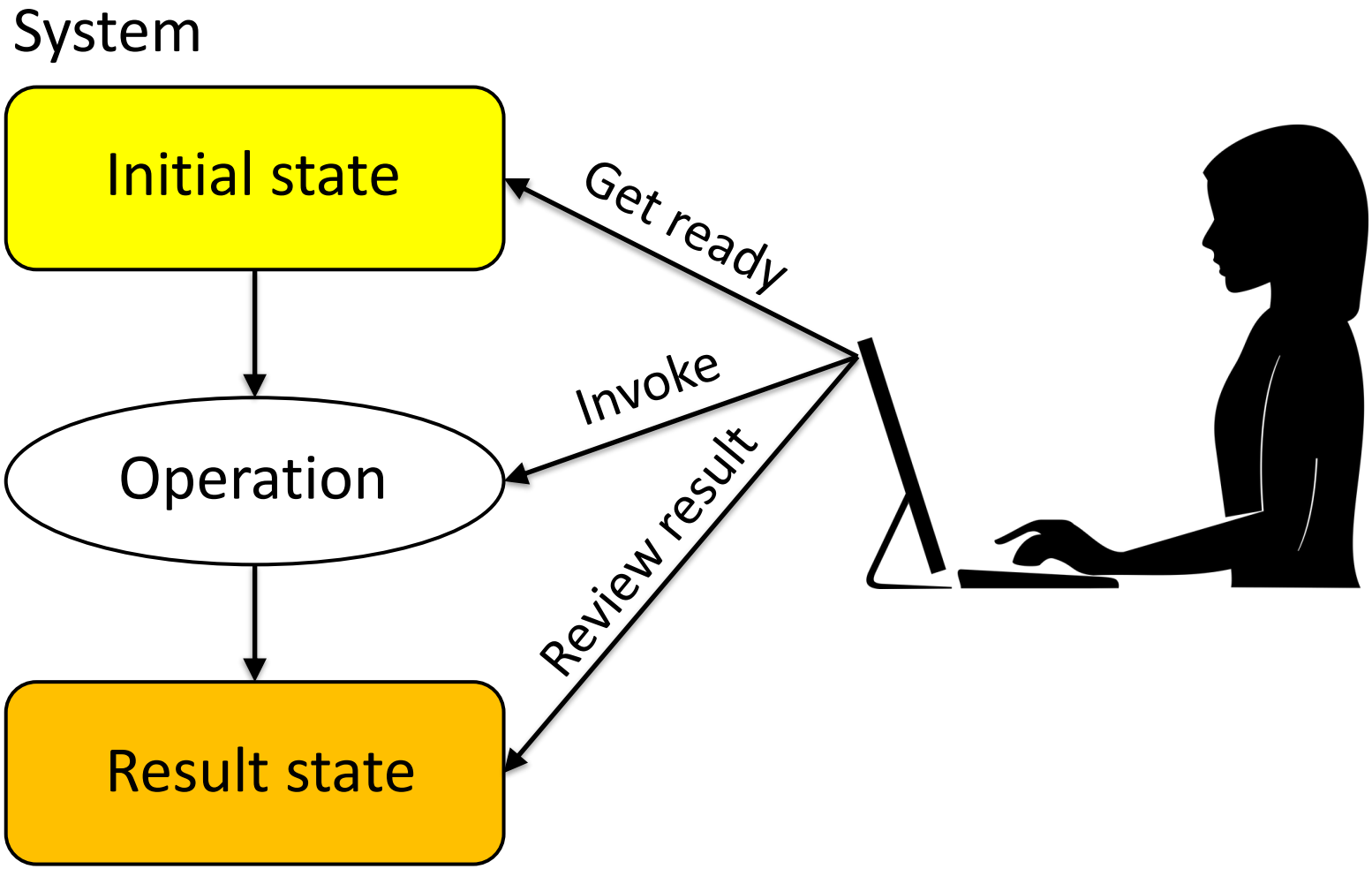
Where are We Now?

- W02: Introduction, Relational Algebra
- W03: SQL
- W04: SQL, Conceptual Modelling
- W05: Conceptual Modelling
- W06: Normalisation, Logical Modelling
- W07: Logical Modelling, Physical DB Design, NOSQL
- W08: Physical DB Design, NOSQL
- W09: DB Application Testing, DB Security
- W10-W14: Project Kick-off, Project Work with Peer Review
- W15: Indexing, query processing, concurrency
- W16: Recovery
- W17: More SQL and NOSQL
- W18: Review and Wrap-up

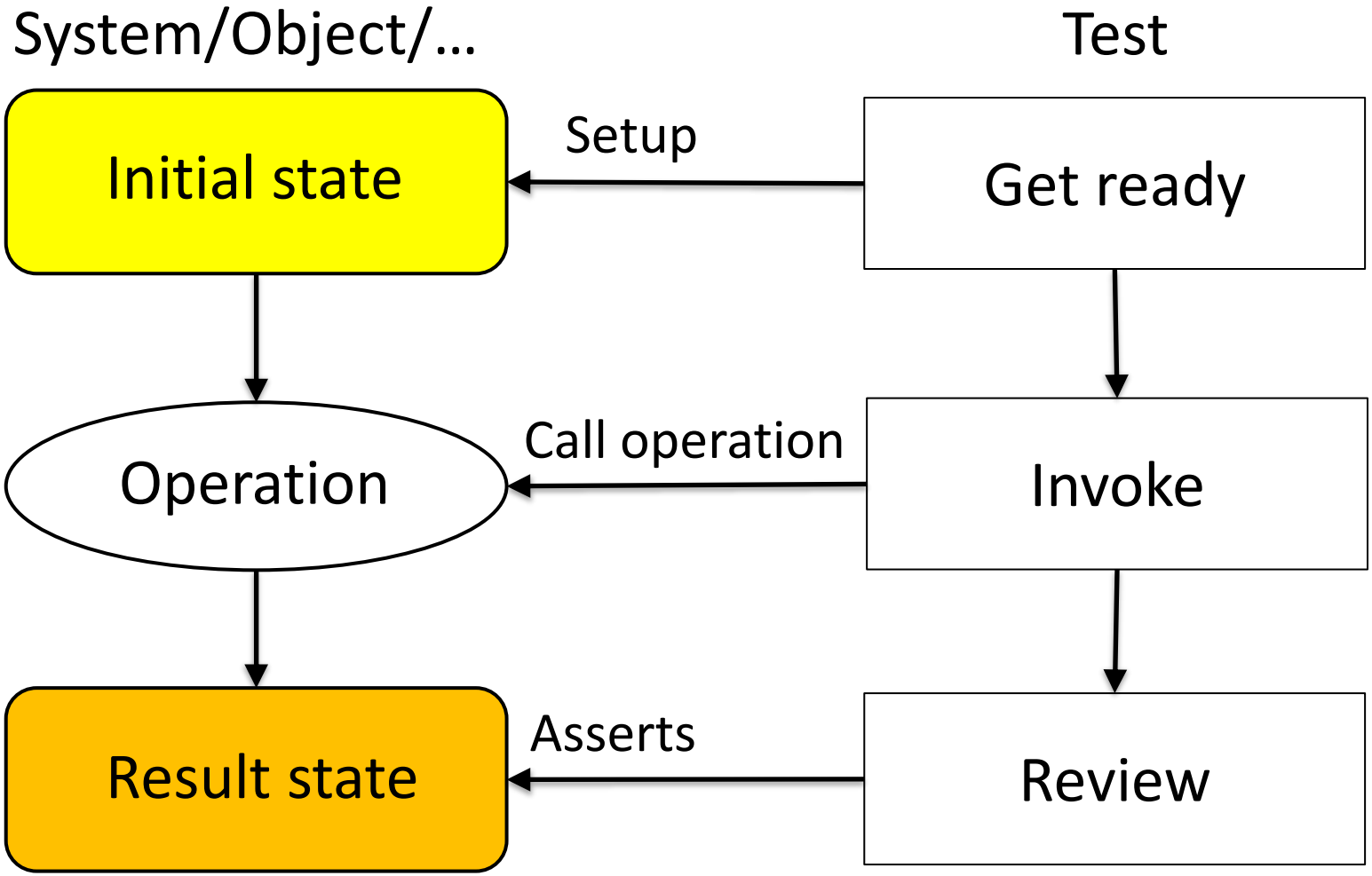
Outline

- Automated testing:
  - Introduction
  - Types of tests
- Database application testing:
  - Python example

Manual Test Process



Automatic Test Process



Test Cases/Values

- Usually, it is unnecessary and impractical to test all possible values
- We therefore need to choose representative values:
  - Equivalence partitioning:
    - Input data for a program unit usually falls into a number of partitions, e.g. all negative integers, zero, all positive numbers
    - By identifying and testing one member of each partition we gain a 'good' coverage with 'small' number of test cases.

Unit/Integration vs. API Testing

- Unit/integration tests:
  - Easy to write
  - Easy to find cause of failure
  - Cheap to run
  - Encourages loose coupling
  - Usually written by the developers
- API tests:
  - Creates confidence in the application
  - Works for legacy code
  - No need for breaking encapsulation
  - Often be written by non-developers – possibly jointly with the customer

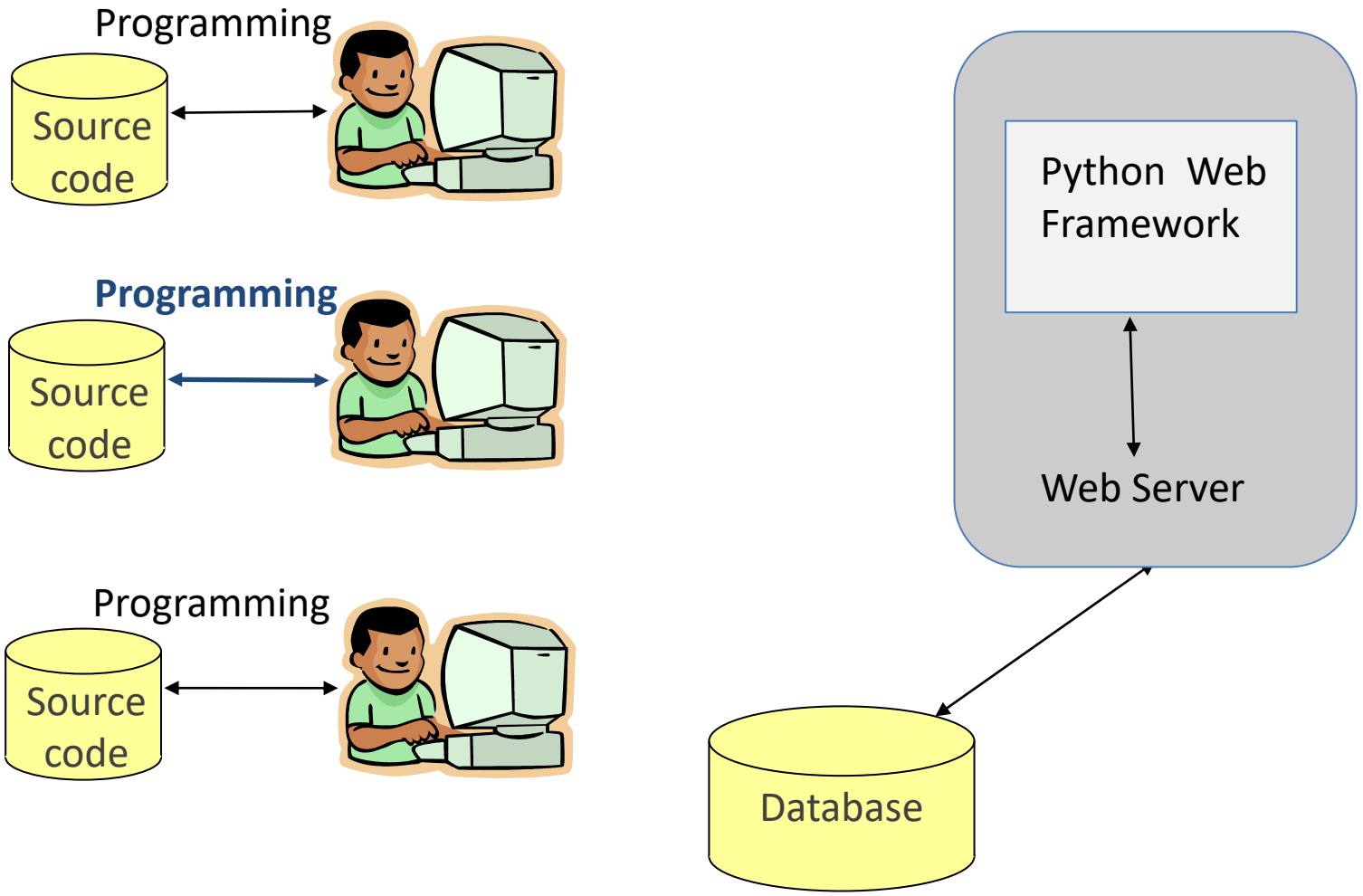
Test Strategy

- Agile strategy: Test early; test often
  - Test-driven development?
  - Regression testing:
    - Making sure that the quality does not deteriorate when code grows and gets refactored
- Combining automated tests and manual tests
- Combining unit/integration tests and api/acceptance tests
- Striving for a reasonable test coverage:
  - Seeking good coverage in complex/critical parts of the code
  - Accepting sparser coverage in trivial parts of the code

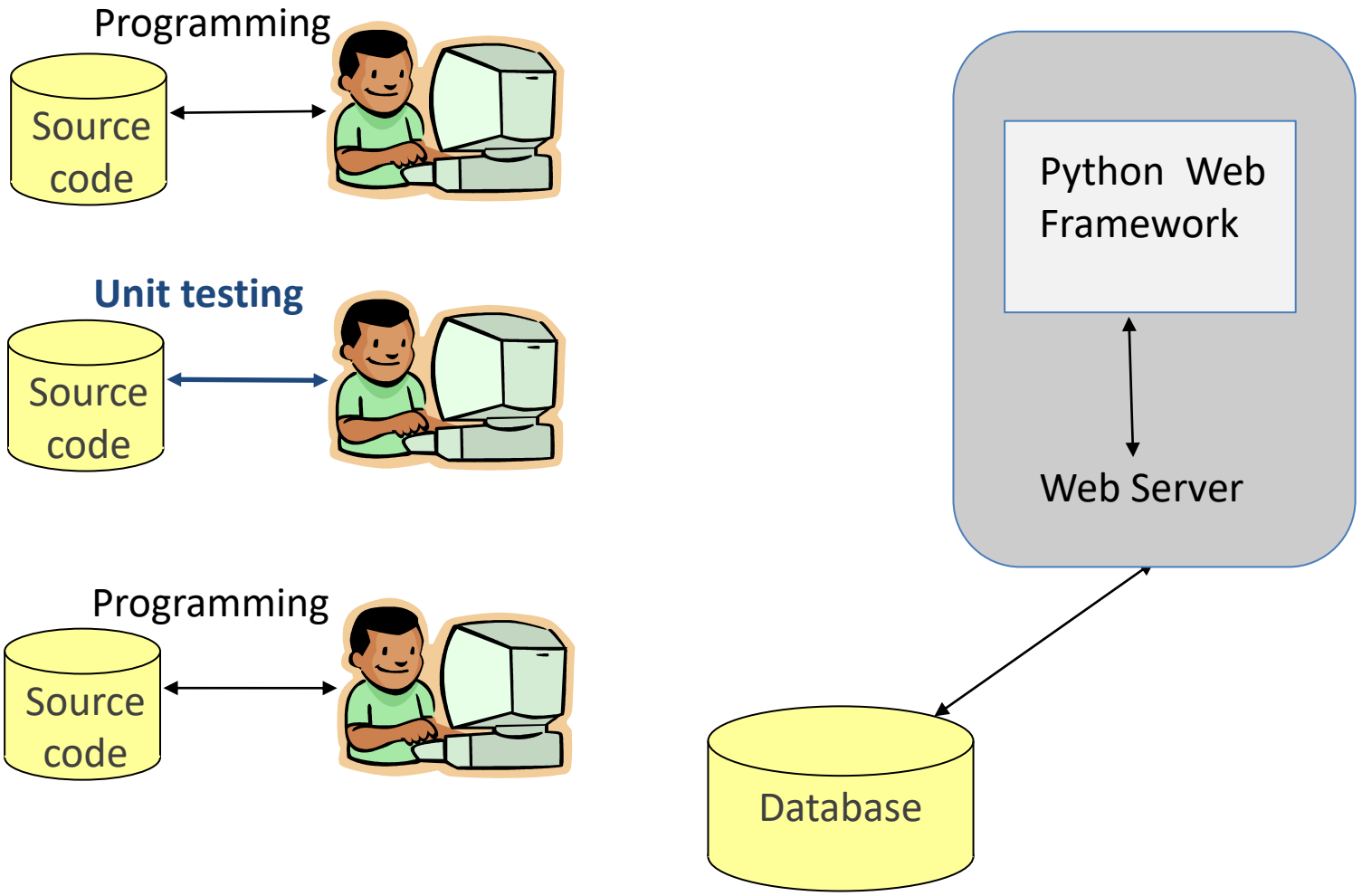
Outline

- Automated testing:
  - Introduction
  - Types of tests
- Database application testing:
  - Development workflow
  - Python example

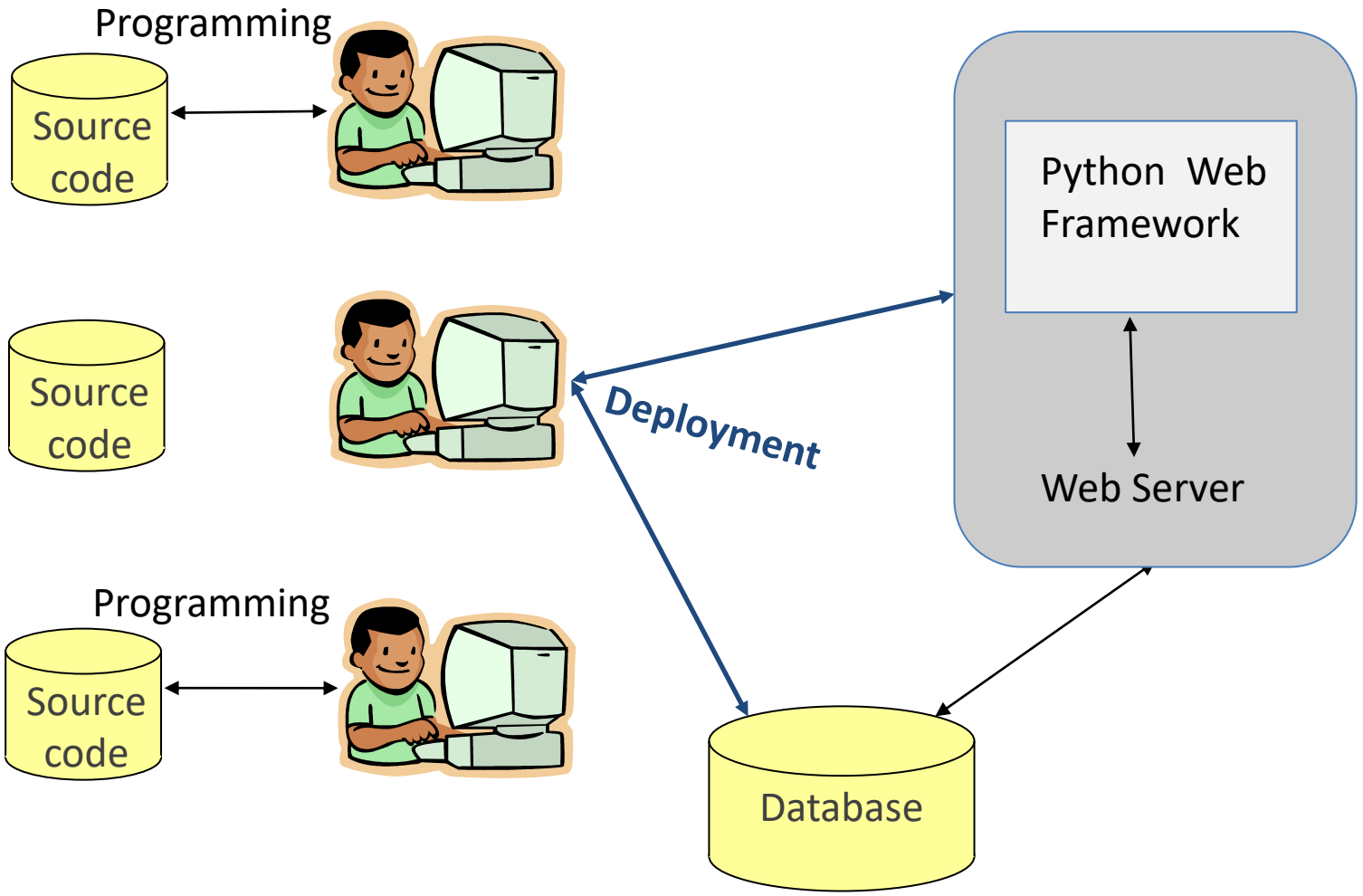
The Development Workflow



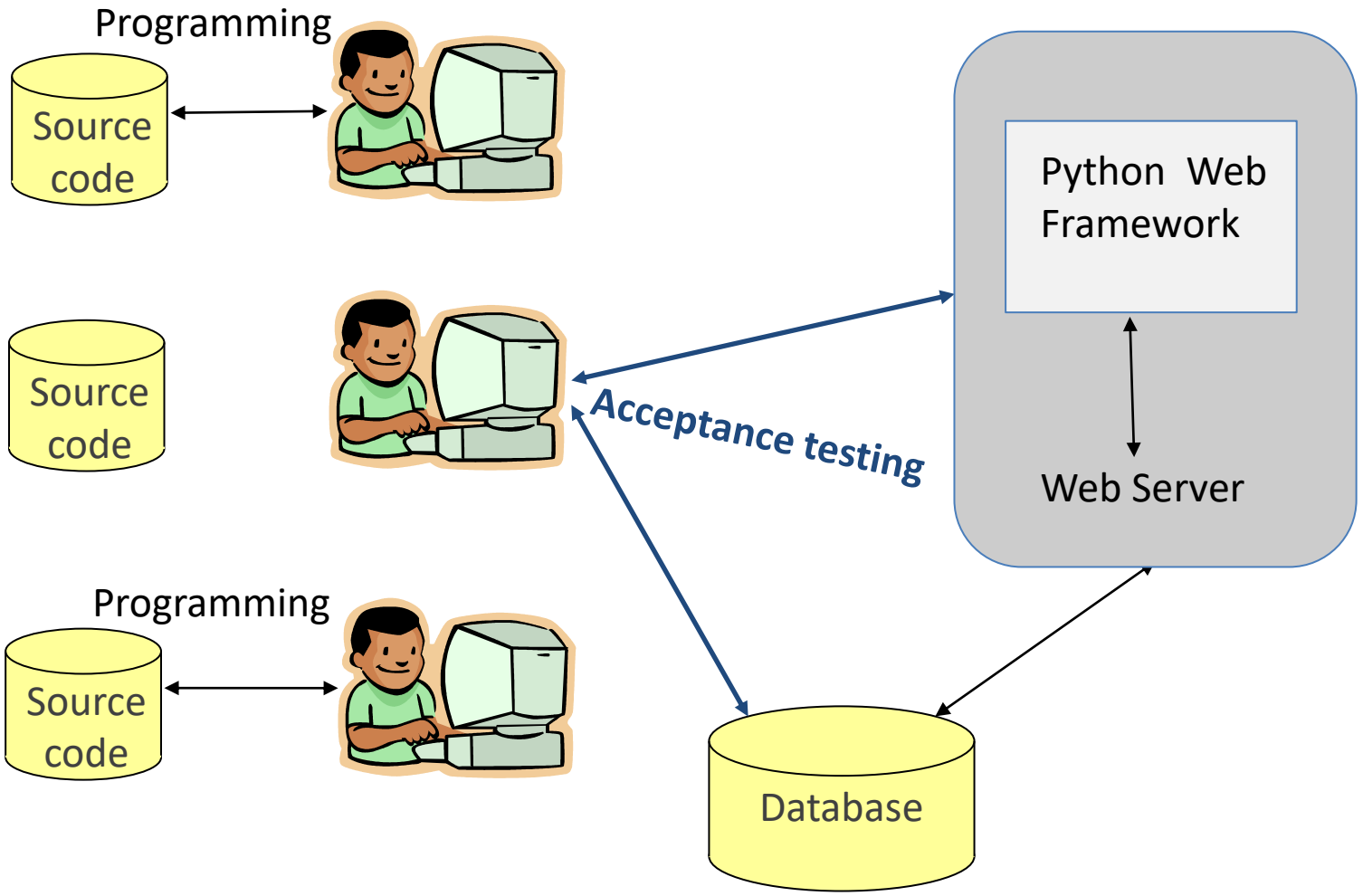
The Development Workflow



The Development Workflow



The Development Workflow



Combining Unit & API testing in Python

- We will be using python *unittest* Unit testing framework
- Setting up the test environment in VS Code
- Things to test in an API test
  - Check HTTP header
  - Check response code (both for correct and incorrect inputs)
  - Check response is JSON
  - Check response contains correct data
  - Check changes have been updated in database (after POST,PUT & UPDATE requests)