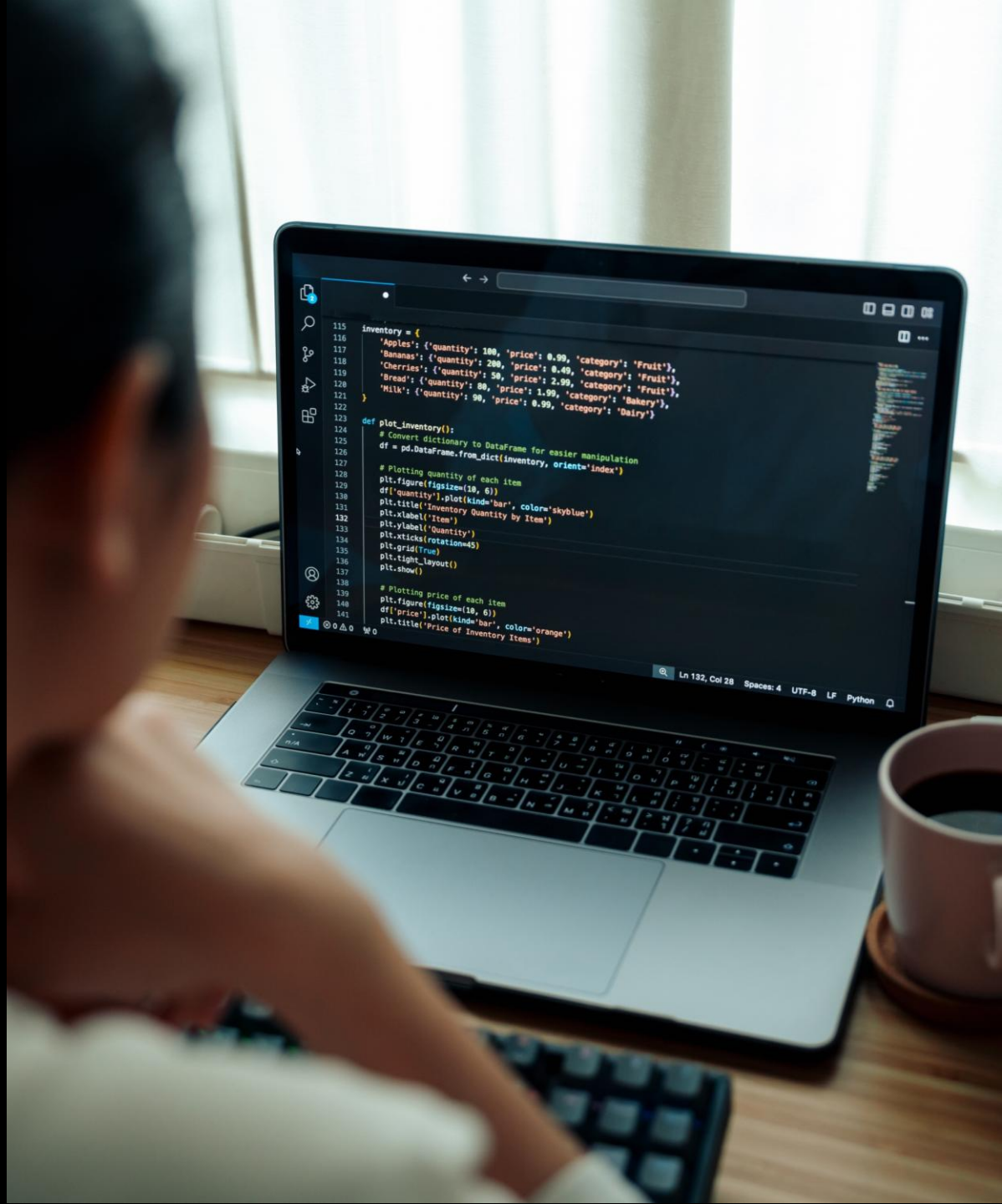
Test-Driven Development (TDD)



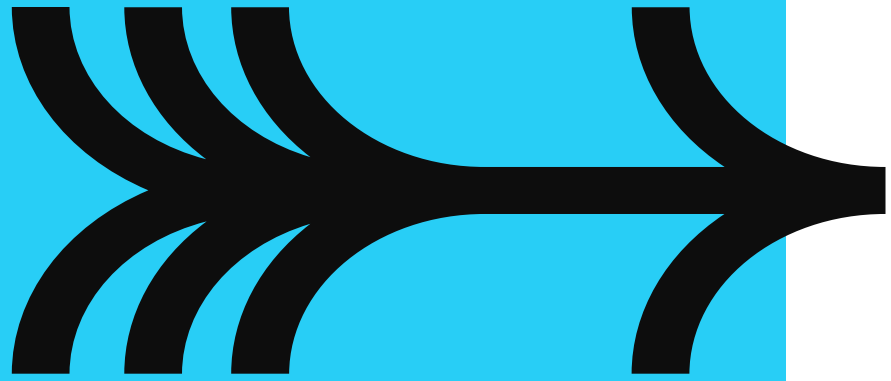
QA

Learn. To Change.

What is TDD?

- Repeating cycle of turning **requirements** into **test cases** and improving code to pass the tests
- **Test written before implementation**
- Core practice of XP. Can be adopted within other methodologies

- **Tests drive design** of API
  - Focus on behaviour, not implementation.
  - Tests describe *what* the code does, not *how* it does it.



TDD Benefits

- **Build up library of small tests** that protect against regression bugs*
- **Extensive code coverage**
  - No code without a test
  - No code that isn't required
- **Greatly helps with debugging**, which makes up for time spent developing tests
- **Tests as developer documentation**
  - **Confidence, not fear**
    Confidence in quality of the code

* A ***regression bug***
  A defect that breaks functionality after a release or refactoring.

Cons of TDD

- **Slower at first**
  It takes time to write tests before code
  Hard when requirements change rapidly.

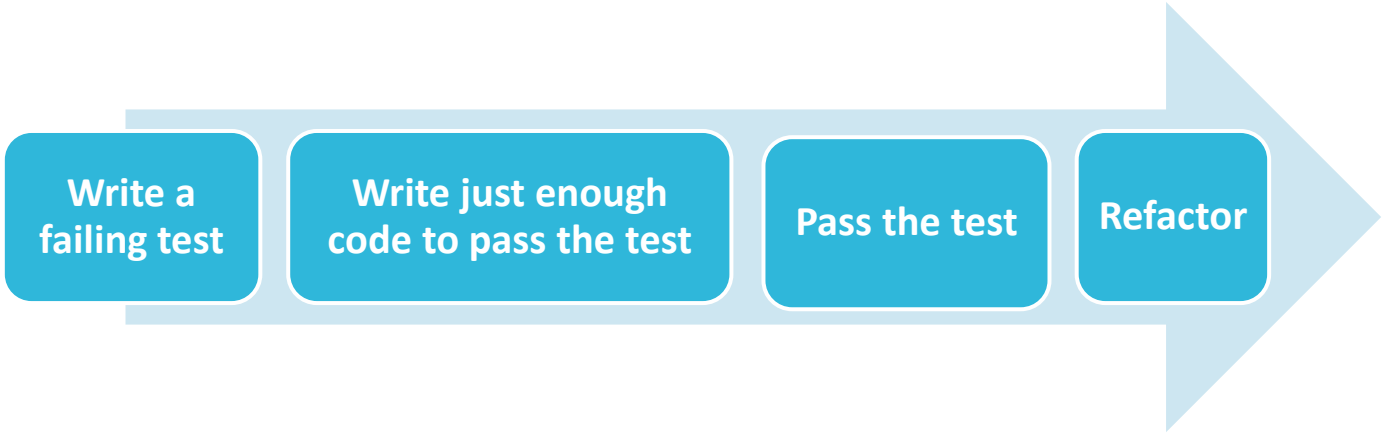
- **Temptation to write tests for trivial logic**
  Use your judgement - It is just a development method!

- **Maintenance overhead**
  Tests need updating when requirements change

- **Not so great for legacy or spaghetti code**
  Refactoring existing systems can be painful!

Use TDD for your core programs logic and high-risk areas
Not for trivial cases like getters, setters …

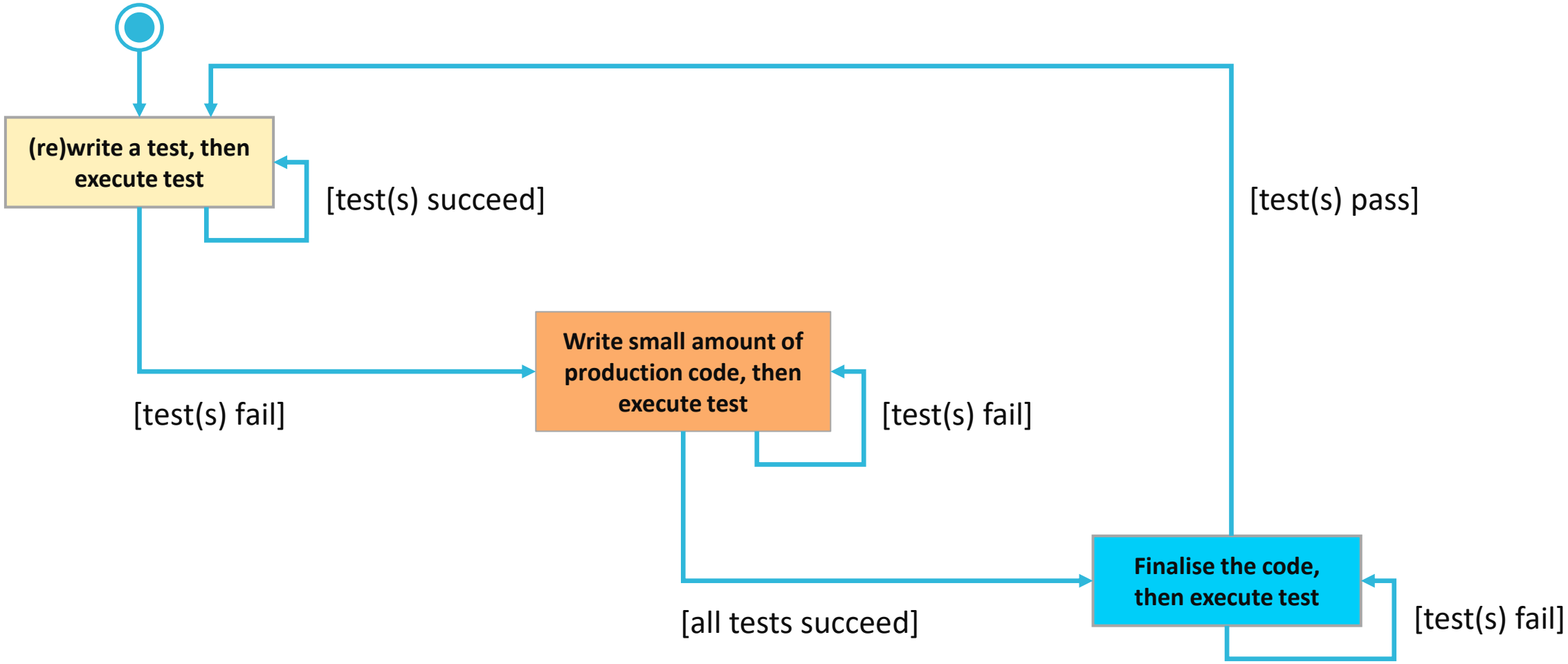
TDD Steps



Different to traditional development…

- Tests as an afterthought
- High level of defects leading to costly failures
- Lengthy testing phase after most of the code is written
- Poor maintainability

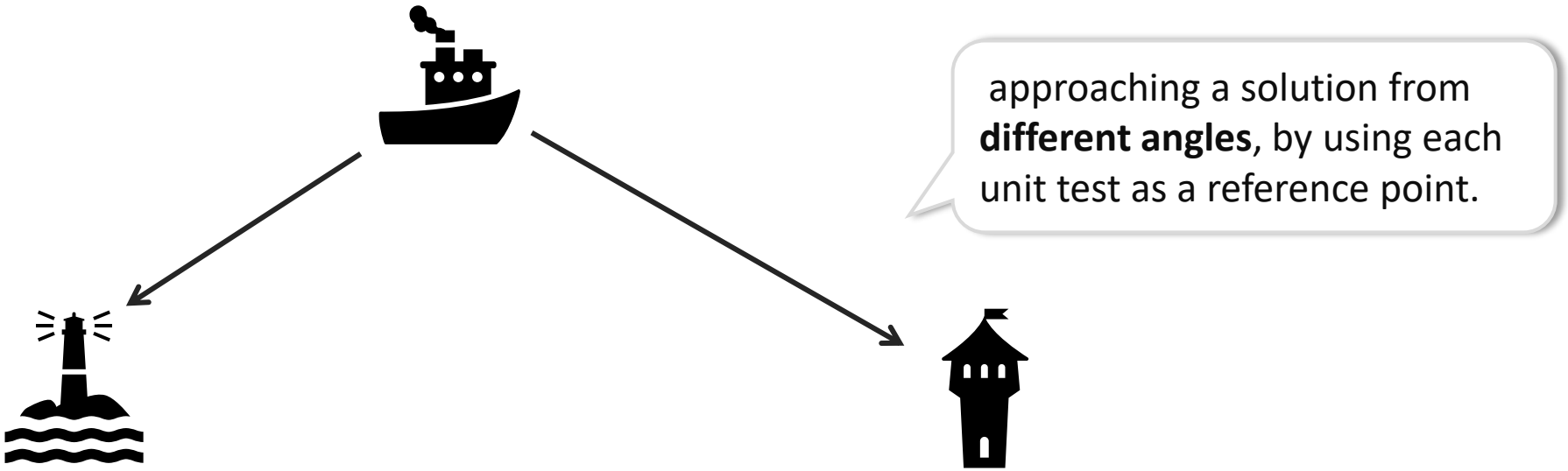
The Red - Green - Refactor Workflow



Beck's TDD Strategies

1. **Fake it.** Return a constant and gradually replace constants with variables.
2. **Obvious implementation.** If a quick, clean solution is obvious, type it in!
3. **Triangulation.** Only generalise your code when you have two or more different tests
4. **Work in very small steps,** continually re-running the tests**.**

Triangulate example: Locate position from two or more stations.



TDD Lab

Afterwards, we'll discuss what you thought

QA

Learn. To Change.