TDD tips and notes:

Chapter 3:

Useful Commands and Concepts

Running the Django dev server

**python3 manage.py runserver**

Running the functional tests

**python3 functional\_tests.py**

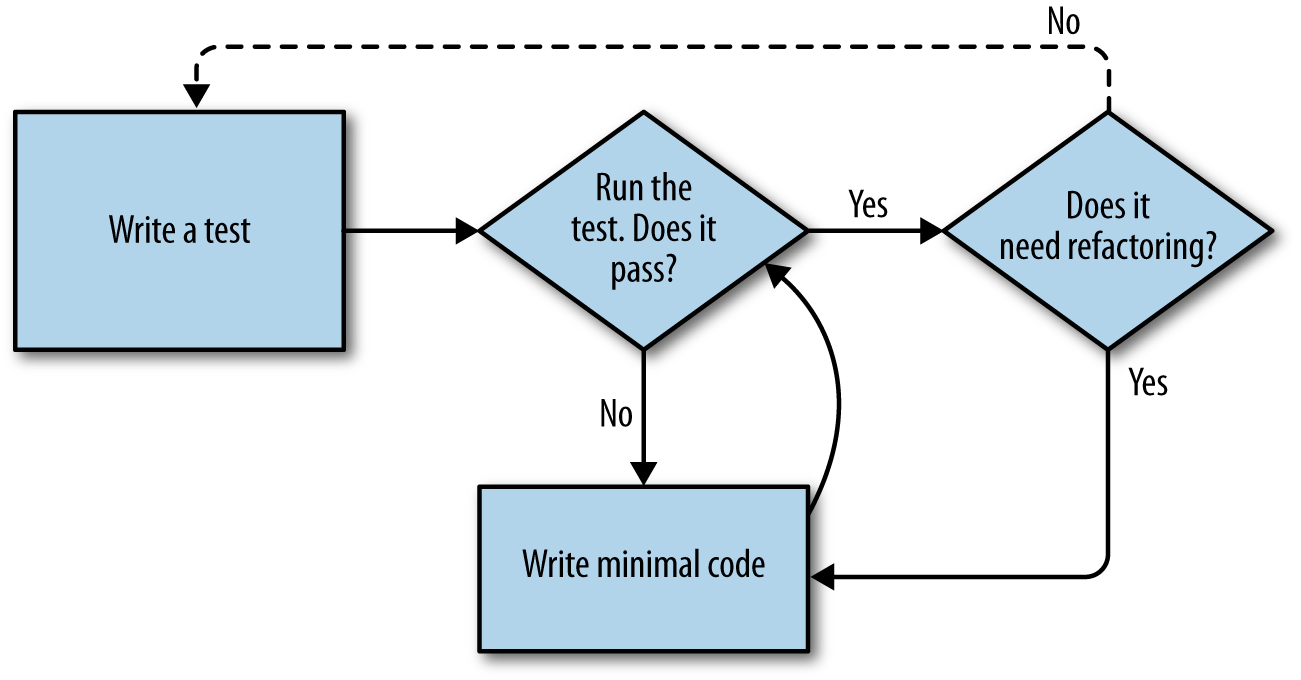
Running the unit tests

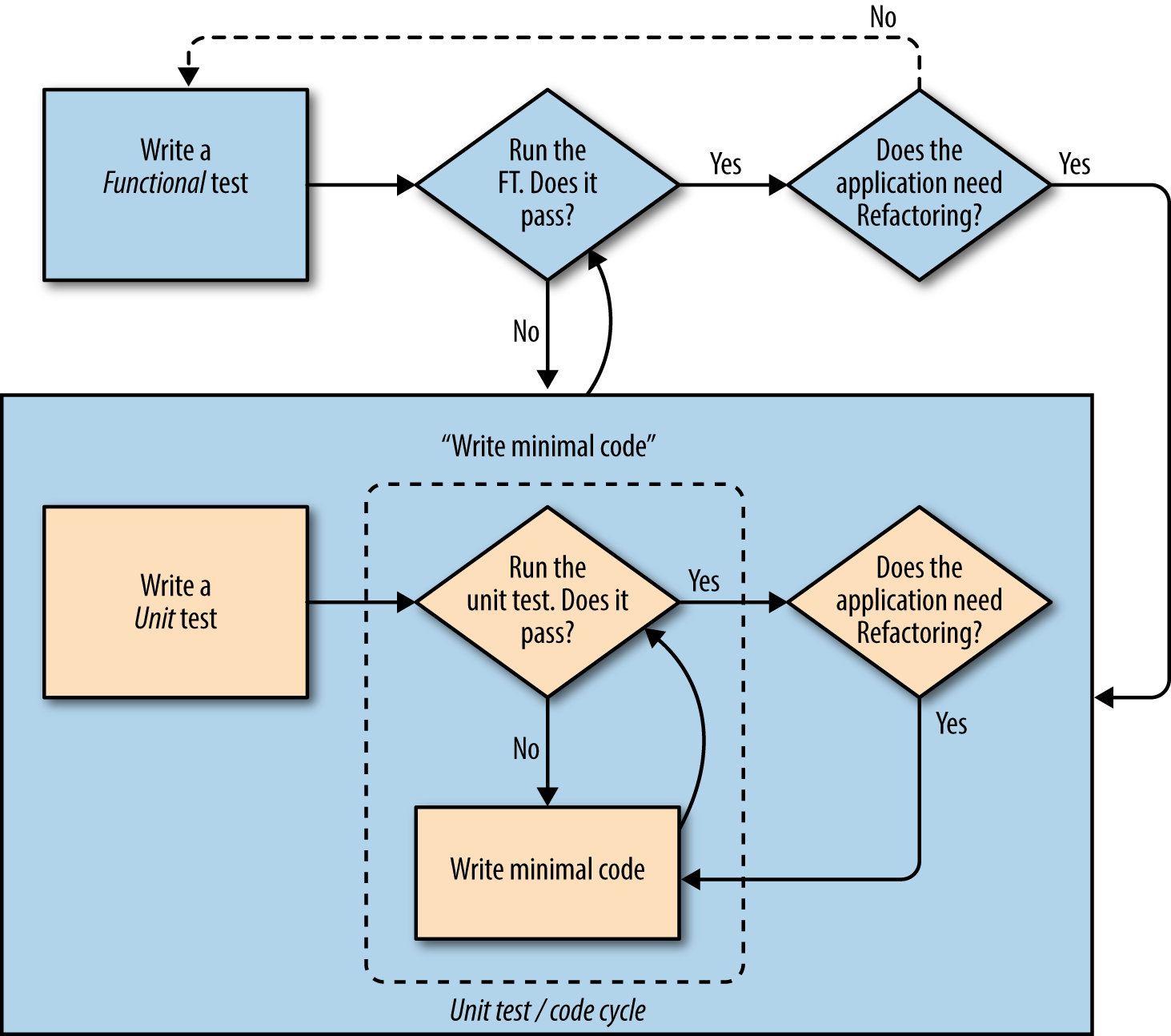
**python3 manage.py test**

The unit-test/code cycle

1. Run the unit tests in the terminal.
2. Make a minimal code change in the editor.
3. Repeat!

Refactor = improving code without changing functionality





{{ blahblah }} displays object as string

The red/green/refactor process: Write a test that fails (red), get it to pass, even in a trivial way (green), refactor until it works like you want it to (refactor)

Useful TDD Concepts

Regression

When new code breaks some aspect of the application which used to work.

Unexpected failure

When a test fails in a way we weren’t expecting. This either means that we’ve made a mistake in our tests, or that the tests have helped us find a regression, and we need to fix something in our code.

Red/Green/Refactor

Another way of describing the TDD process. Write a test and see it fail (Red), write some code to get it to pass (Green), then Refactor to improve the implementation.

Triangulation

Adding a test case with a new specific example for some existing code, to justify generalising the implementation (which may be a "cheat" until that point).

Three strikes and refactor

A rule of thumb for when to remove duplication from code. When two pieces of code look very similar, it often pays to wait until you see a third use case, so that you’re more sure about what part of the code really is the common, re-usable part to refactor out.

The scratchpad to-do list

A place to write down things that occur to us as we’re coding, so that we can finish up what we’re doing and come back to them later.

USE DOUBLE ## to indicate “meta comments” – comments about how the test is working, to distinguish from user story (that uses single hash)

WHEN MODIFYING COLS IN Db – run python manage.py makemigrations

Use ‘pass’ in an empty model class

When deleting migrations (/project/app/migrations/000x\_blahblah.py), NEVER delete if migration has been committed to your VCS

.item\_set is called a "reverse lookup"—it’s one of Django’s incredibly useful bits of ORM that lets you look up an object’s related items from a different table…

**ALWAYS ALWAYS ALWAYS DO A COMMIT BEFORE AND AFTER REFACTORING**

Note- in refactoring section at end of chapter 6, need to add from lists import views to

from django.conf.urls import url

urlpatterns = [

url(r'^new$', views.new\_list, name='new\_list'),

url(r'^(\d+)/$', views.view\_list, name='view\_list'),

url(r'^(\d+)/add\_item$', views.add\_item, name='add\_item'),

]

YAGNI – You ain’t gonna need it. Concept in feature design – only design features as they’re needed

SEND THIS GUY A NOTE WITH YOUR URL WHEN YOUR SITE IS UP

Why not ping me a note once your site is live on the web, and send me the URL? It always gives me a warm and fuzzy feeling … <[obeythetestinggoat@gmail.com](mailto:obeythetestinggoat@gmail.com)>.

**Automating**

Let’s recap our provisioning and deployment procedures:

Provisioning

1. Assume we have a user account and home folder
2. apt-get install nginx git python-pip
3. pip install virtualenv
4. Add Nginx config for virtual host
5. Add Upstart job for Gunicorn

Deployment

1. Create directory structure in *~/sites*
2. Pull down source code into folder named *source*
3. Start virtualenv in *../virtualenv*
4. pip install -r requirements.txt
5. manage.py migrate for database
6. collectstatic for static files
7. Set DEBUG = False and ALLOWED\_HOSTS in *settings.py*
8. Restart Gunicorn job
9. Run FTs to check everything works