



TDD y Python

javierj@us.es
/
@IWT2_javier



TDD no es probar

1. You are not allowed to write any production code unless it is to make a failing unit test pass.
2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
3. You are not allowed to write any more production code than is sufficient to pass the

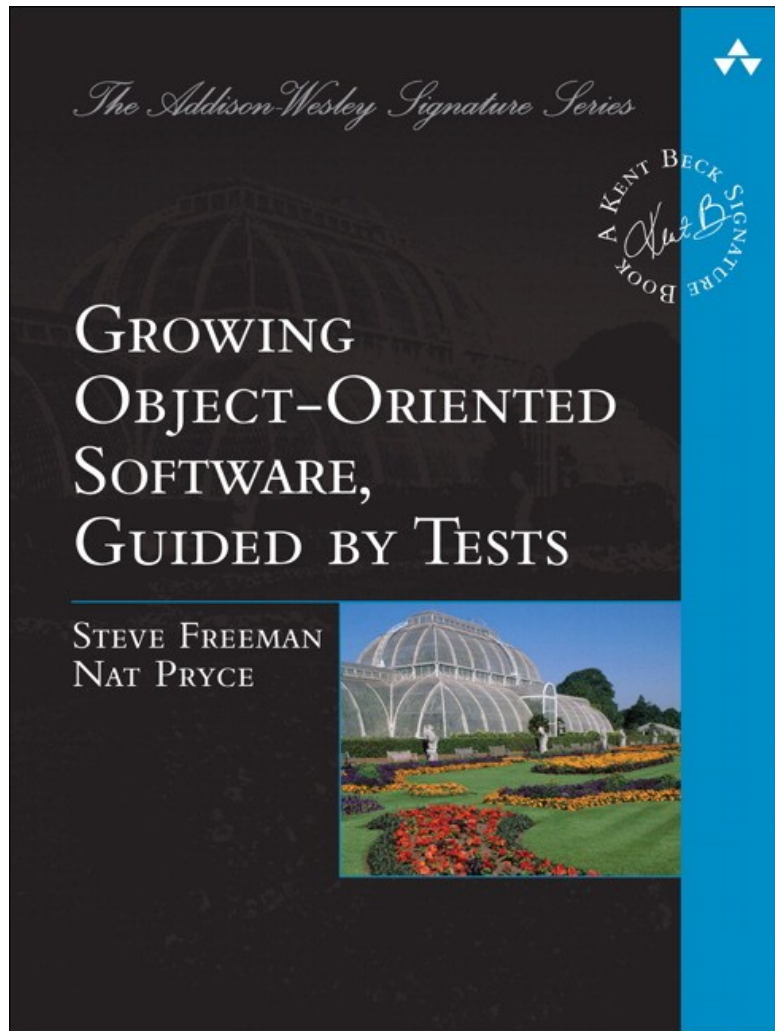


Esto es todo

A graphic of red curtains with a scalloped top edge, framing the text in the center.

Gracias por
su atención

La complejidad



“Test-Driven Development (TDD) is a deceptively simple idea”

“El Desarrollo Dirigido por Pruebas (TDD) es una idea engañosamente simple”

¿Qué buscamos?

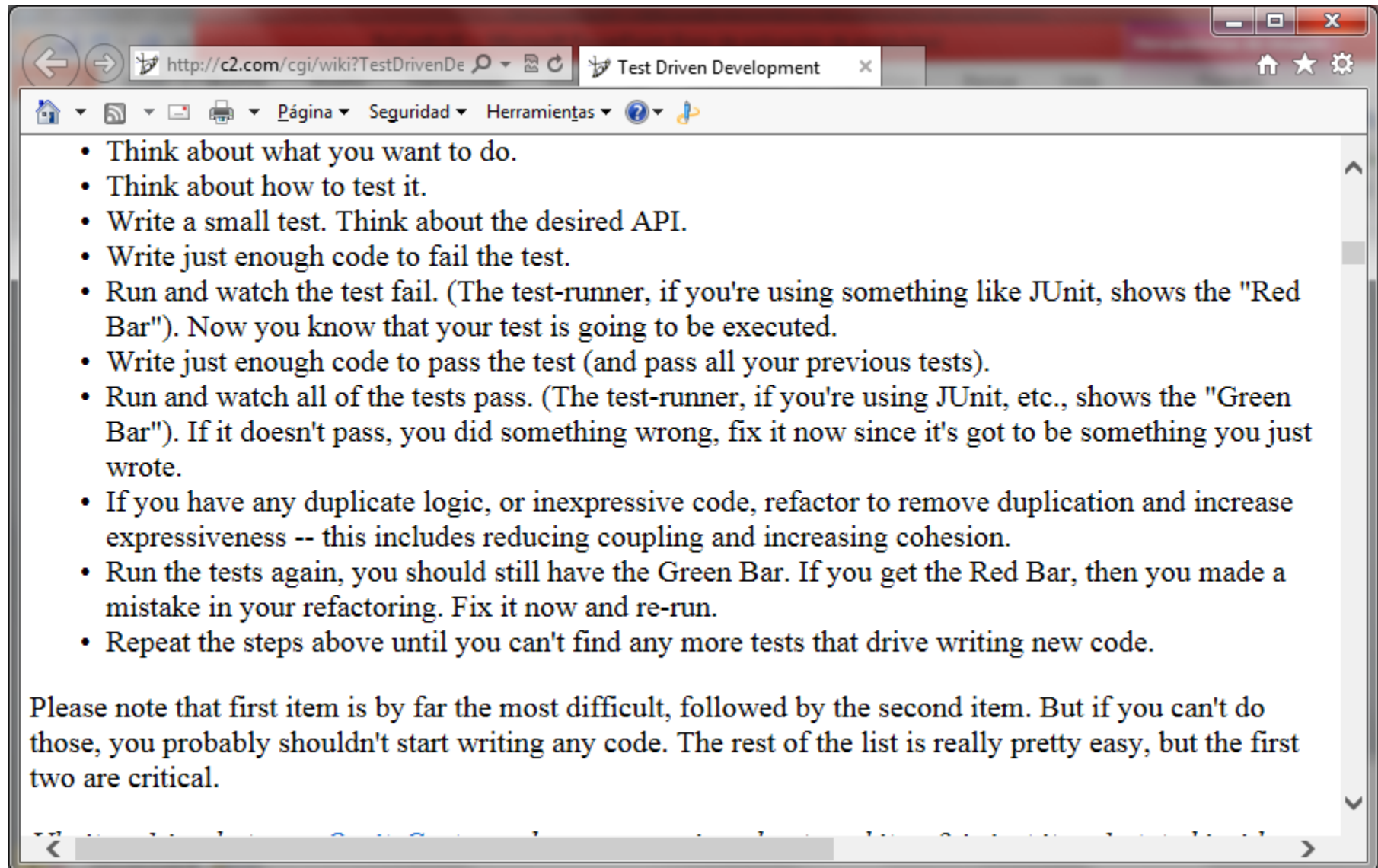
¿Por qué queríamos hacer TDD?



La complicación



Una buena manera de aplicar TDD

A screenshot of a web browser window. The address bar shows 'http://c2.com/cgi/wiki/TestDrivenDe'. The page title is 'Test Driven Development'. The main content is a list of 10 steps for TDD. The browser interface includes a back button, a search bar, and various menu items like 'Página', 'Seguridad', and 'Herramientas'.

- Think about what you want to do.
- Think about how to test it.
- Write a small test. Think about the desired API.
- Write just enough code to fail the test.
- Run and watch the test fail. (The test-runner, if you're using something like JUnit, shows the "Red Bar"). Now you know that your test is going to be executed.
- Write just enough code to pass the test (and pass all your previous tests).
- Run and watch all of the tests pass. (The test-runner, if you're using JUnit, etc., shows the "Green Bar"). If it doesn't pass, you did something wrong, fix it now since it's got to be something you just wrote.
- If you have any duplicate logic, or inexpressive code, refactor to remove duplication and increase expressiveness -- this includes reducing coupling and increasing cohesion.
- Run the tests again, you should still have the Green Bar. If you get the Red Bar, then you made a mistake in your refactoring. Fix it now and re-run.
- Repeat the steps above until you can't find any more tests that drive writing new code.

Please note that first item is by far the most difficult, followed by the second item. But if you can't do those, you probably shouldn't start writing any code. The rest of the list is really pretty easy, but the first two are critical.

Ahora a por un problema

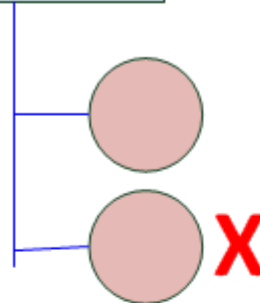


Hazme un programa que me
busque ficheros del mismo tamaño
y me los borre

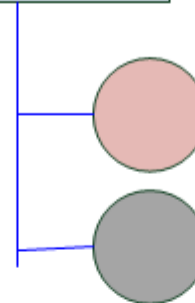
Piensa



/Directorio

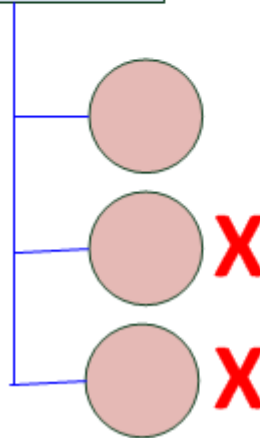


/Directorio

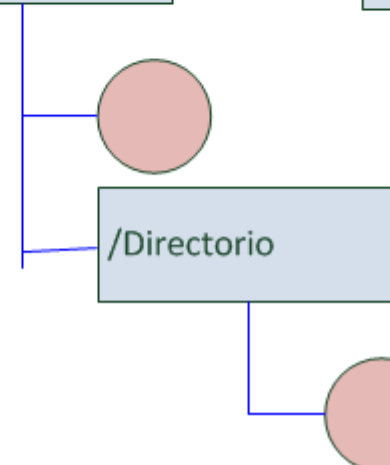


/Directorio

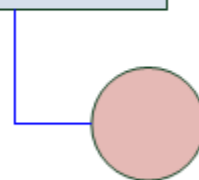
/Directorio



/Directorio



/Directorio



Y tú, ¿qué harías?



Con TDD

Escribe una prueba

Mi primera prueba:

Arrange:

- Un directorio
- Dos ficheros iguales dentro de ese directorio

Act:

- Llamar al algoritmo de borrado

Assert:

- Uno de los ficheros ya no está y el otro sí



Escribe una prueba

Mi primera prueba:

Arrange:

Un directorio
Dos ficheros iguales dentro
de
ese directorio

Act:

Llamar al algoritmo de
borrado

Assert:

Uno de los ficheros ya no
está
y el otro sí

¿Creo un directorio y dos
ficheros a mano?
Tengo que crearlo cada vez
que ejecute la prueba

¿Creo un directorio y fichero
con código?
Tengo que aprender a
hacerlo necesito código
para restaurarlo

¿Obtengo un listado de los
ficheros?

No lo necesito
(aún)

Vamos a escribir código

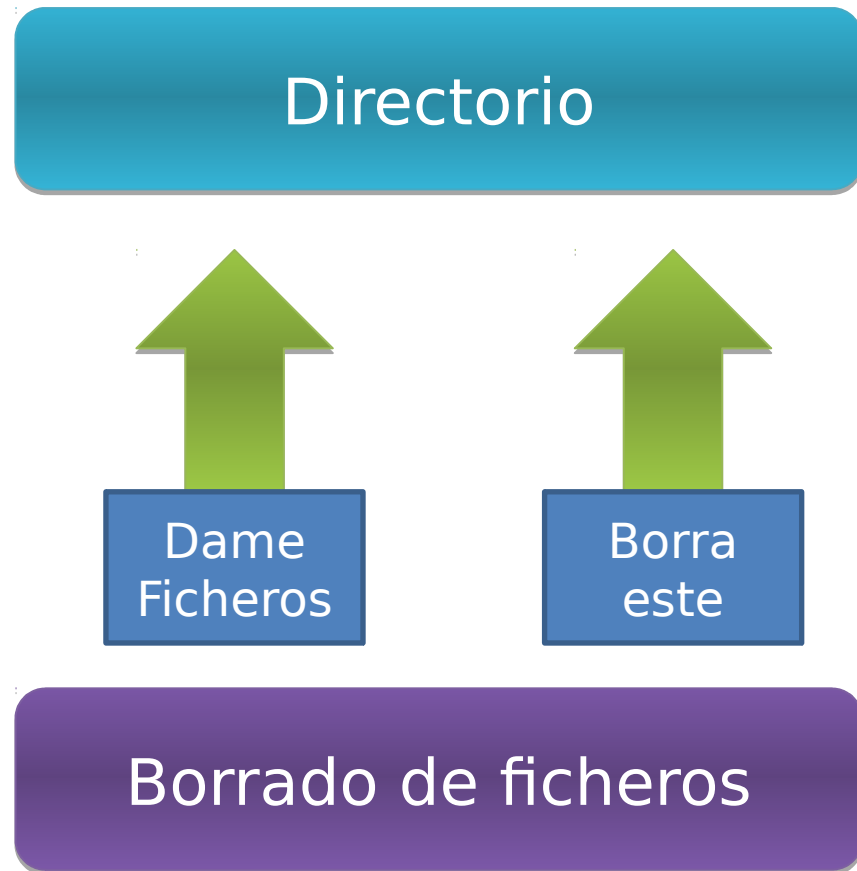


LiClipse

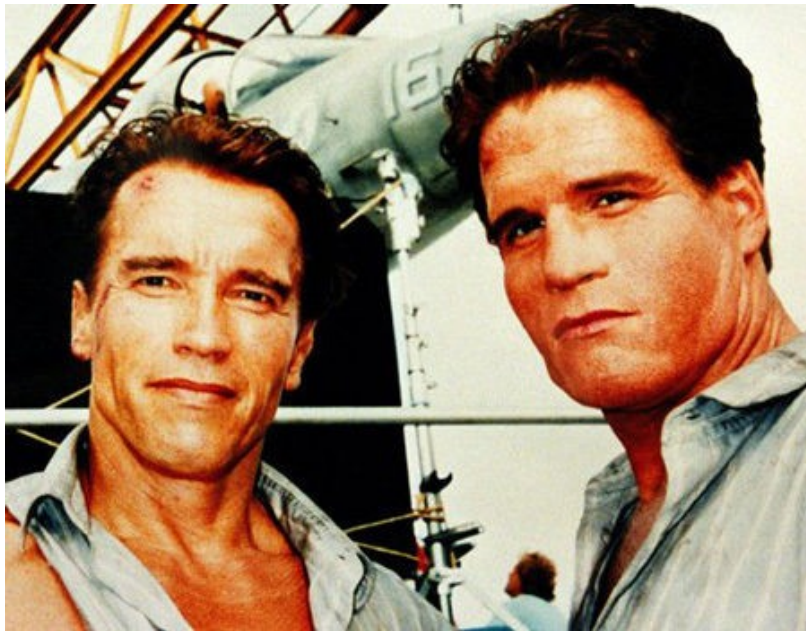
by Brainwy

Snippet 01

TDD = ¡¡¡ Piensa !!!!



Utilizando Dobles de Prueba



- Diseñamos el API que queremos.
- No nos preocupamos por la implementación
- Verificamos qué se está invocando
- Devolvemos los valores adecuados para la prueba

Single Responsibility Principle



A CLASS SHOULD HAVE ONLY ONE REASON TO CHANGE.

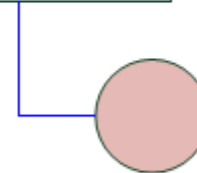
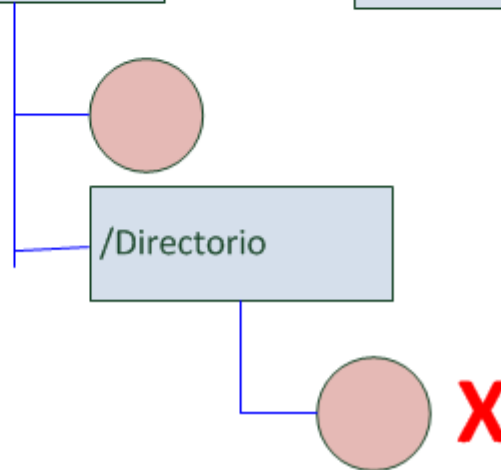
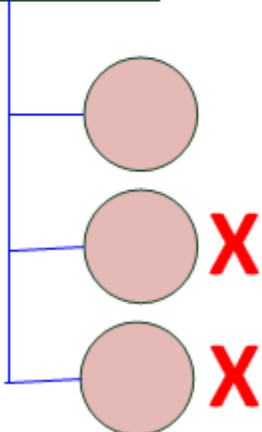
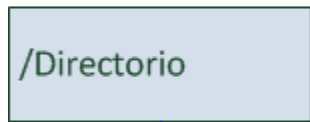
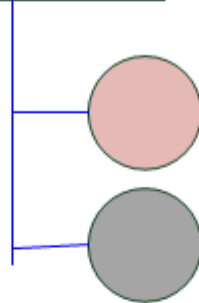
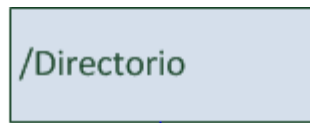
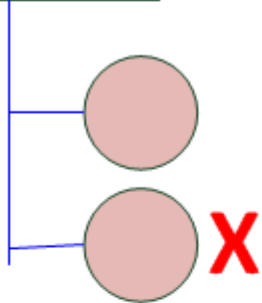
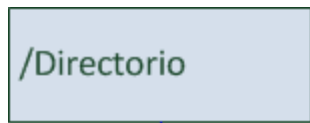
¿Cuál es el mínimo código?



LiClipse
by Brainwy

Snippet 02

Vuelta a empezar



Snippet 03

¿Y qué hacemos con los mocks?



¿Qué hacemos con los mocks?

Cuando el directorio está vacío entonces no tengo ficheros.

Cuando borro un fichero el directorio tiene un fichero menos.

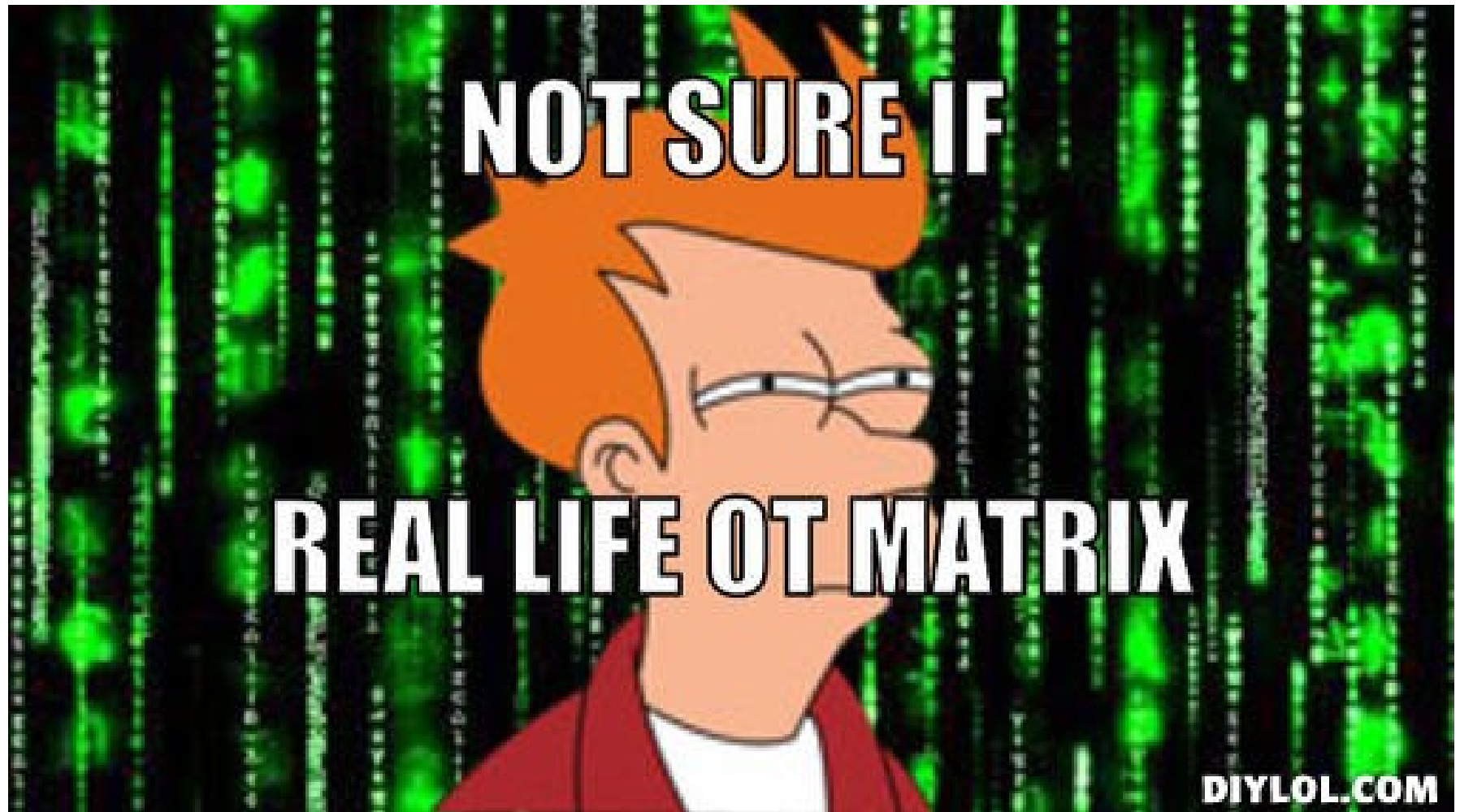
Cuando recupero un fichero de un directorio entonces puedo conocer su nombre.

Cuando recupero un fichero de un directorio entonces puedo conocer su tamaño.

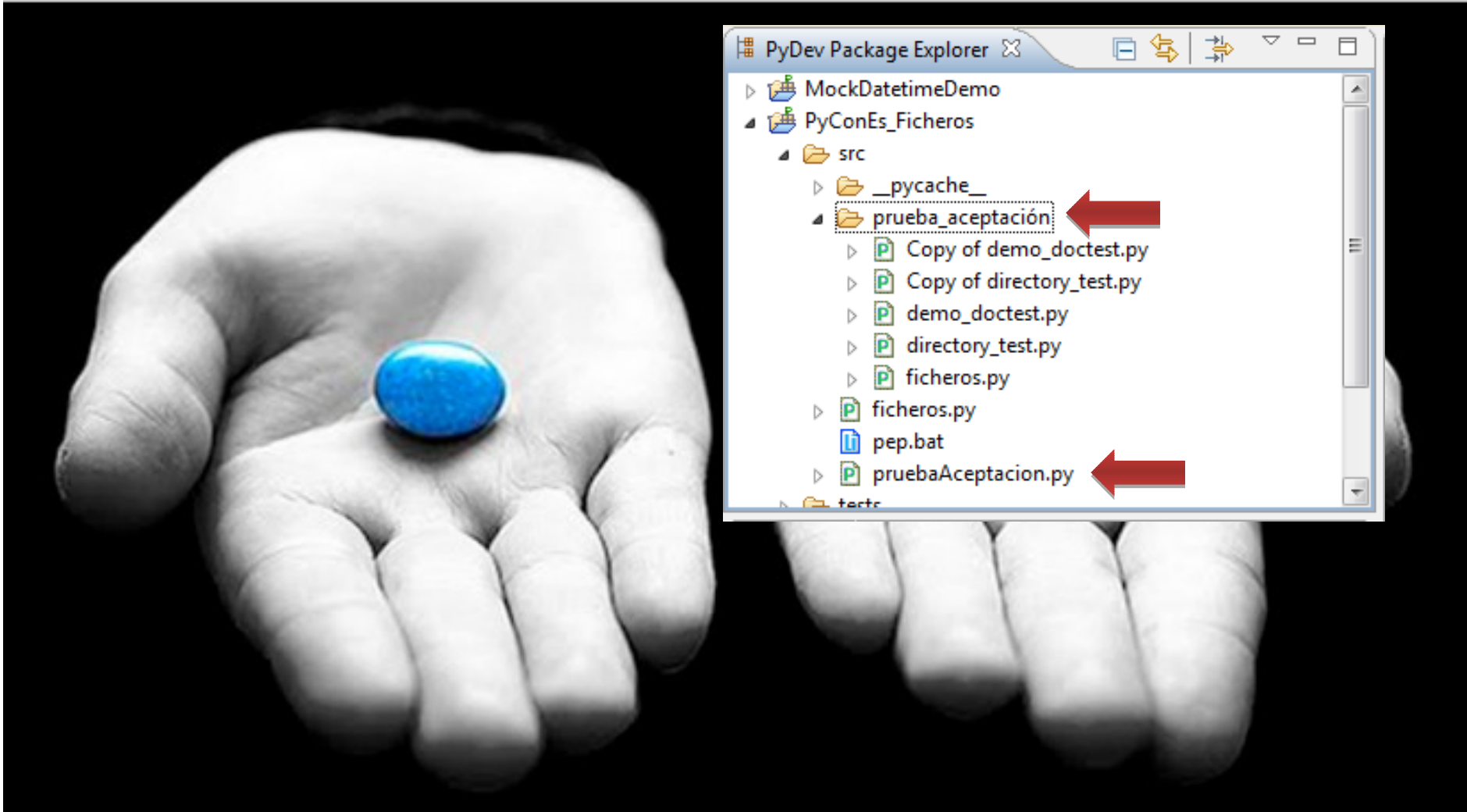
Etc.

Directorios temporales en Python (módulo *tempfile*)

¿Y cuando lo ponemos todo junto?



En el mundo real

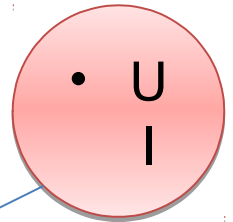


Conclusiones de este ejemplo

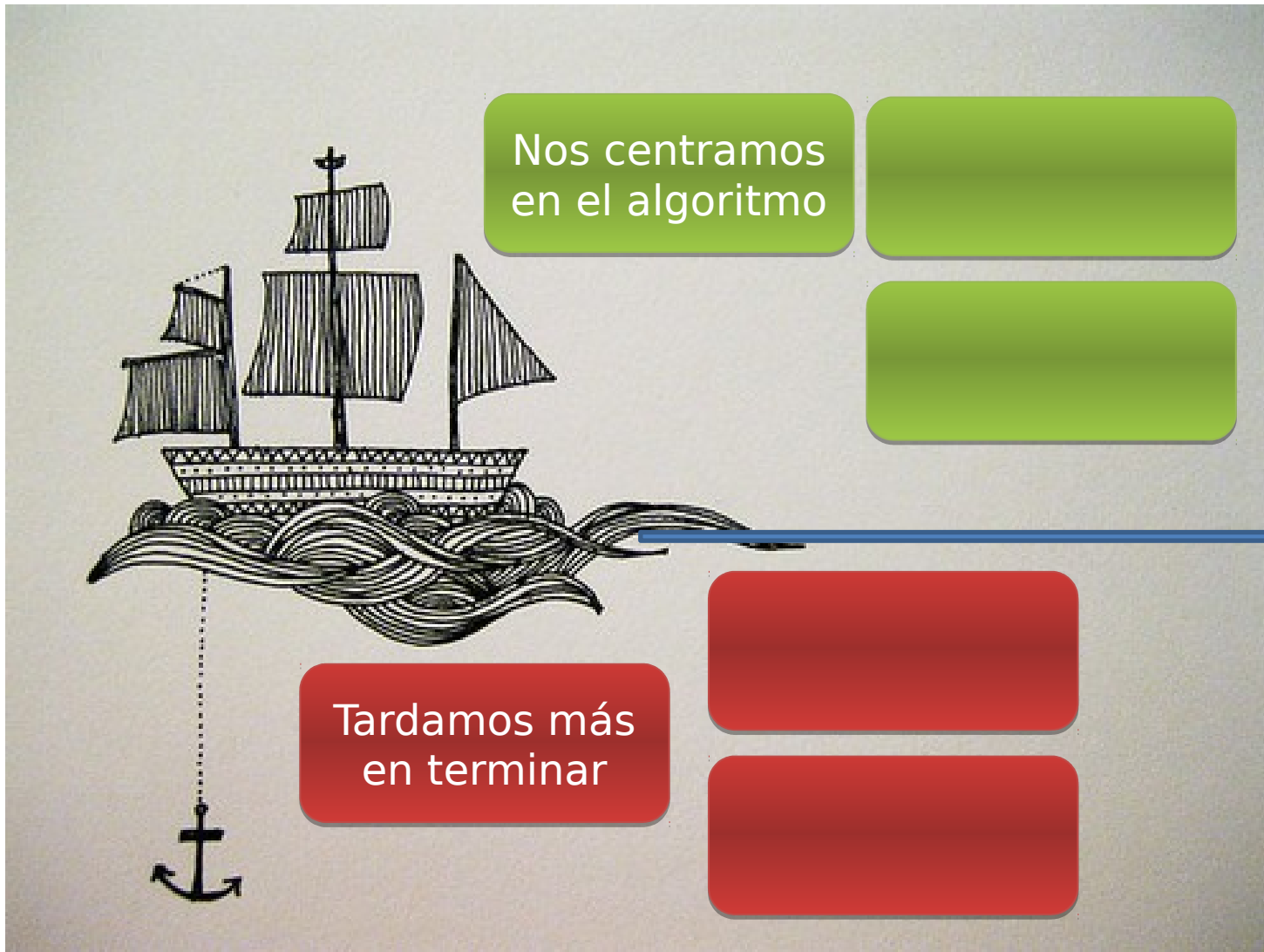


- ¿Cómo leemos los ficheros de un directorio?
- ¿Cómo borramos ficheros?
- ¿Cómo leemos el tamaño de un fichero?

- Cómo podemos leer una única vez cada directorio?
- ¿Cómo sabemos si hay más de un fichero con el mismo tamaño?



Retrospectiva de lo que Hemos Hecho



Testing en Python



Unittest,
Doctest, Nose

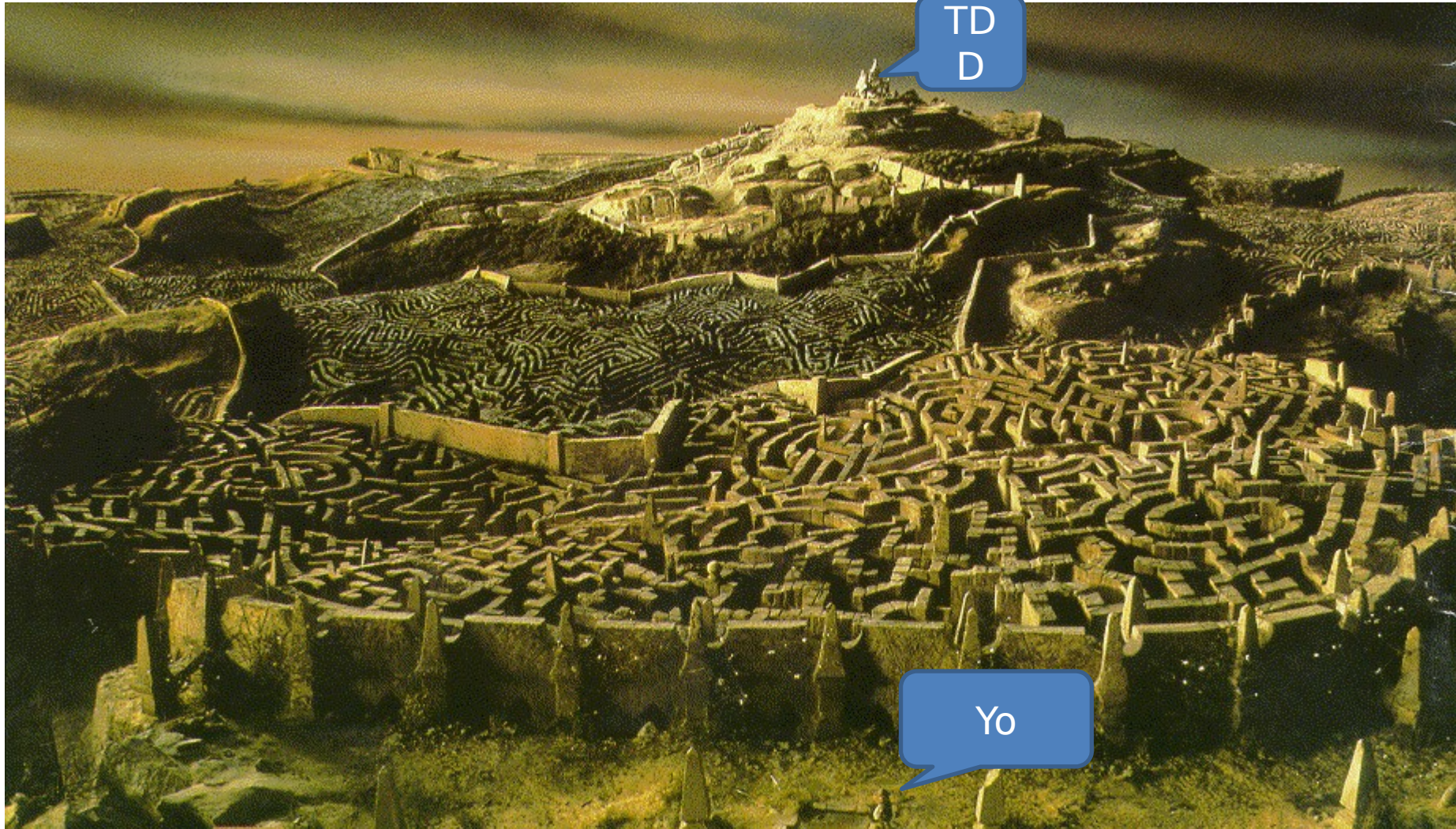
Mockito for
Python, Mocker,
Pydoubles

Selenium,
Django Testing

Behave, PyFIT,
Lettuce,
Mamba

<http://wiki.python.org/moin/PythonTestingToolsTaxonomy>

Advertencia



TD
D

Yo

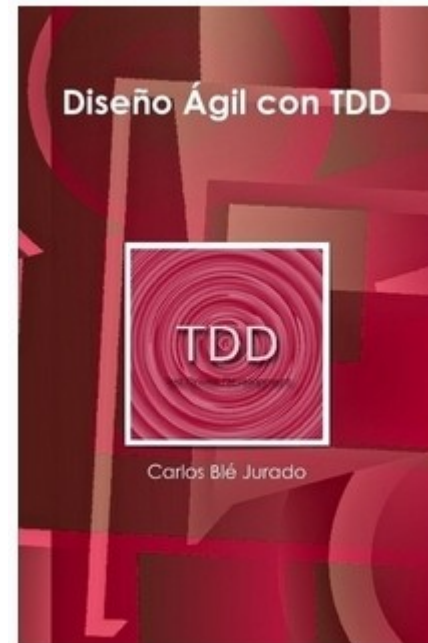
Conclusiones



Pruébalo y elige



Para saber más de
TDD.



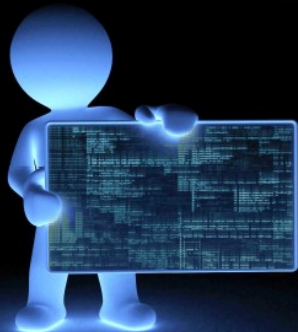
FAIL  PASS
HOLA TDD

 **meses**  **katas**

¿Quién soy yo?



Desarrollo Dirigido por
Pruebas **Práctico**



Aprendiendo TDD

TDD /



Pybonacci



libGDX



Extra

10,000 lines of C# code...Check.
124 .NET assemblies generated...Check.
52 Build Scripts written...Check.



10.000 líneas de código C#...
Comprobado.... 124
assemblies .NET generados....
Comprobado.... 52 scripts de
construcción... comprobado

Now that my unit tests are written,
I can start building my component!



Ahora que mis pruebas
unitarias están escritas
puedo empezar a construir
mis componentes.

Extra



TESTING

I FIND YOUR LACK OF TESTS DISTURBING.

DIYDESPAIR.COM

Enlaces

- Blog: <http://iwt2-javierj.tumblr.com/>
- Libro: <http://www.iwt2.org/web/opencms/IWT2/comunidad/LibroTDD/?locale=es>
- GameUS: <http://www.gameus.es/>
- IWT2 DojoUS: <http://www.iwt2.org/web/opencms/IWT2/comunidad/dojous/?locale=es>
- Pybonacci: <http://pybonacci.wordpress.com/2013/01/07/desarrollo-dirigido-por-pruebas-en-python-i-una-historia-que-pasa-todos-los-dias/#more-1352>
<http://pybonacci.wordpress.com/2013/06/19/desarrollo-dirigido-por-pruebas-en-python-ii-un-caso-practico-i/>
- Twitter: @IWT2_Javier/ @GameUsSev / @TDDPractico

Fin

Ahora sí

