

# Test Driven Development

Desarrollo Dirigido por Test

Víctor del Río

¿Qué es el TDD?

# ¿Qué es el TDD?

Es una forma de enfocar el desarrollo de software en que los test se escriben **antes** que la implementación

Pero...  
¿de dónde viene todo esto?

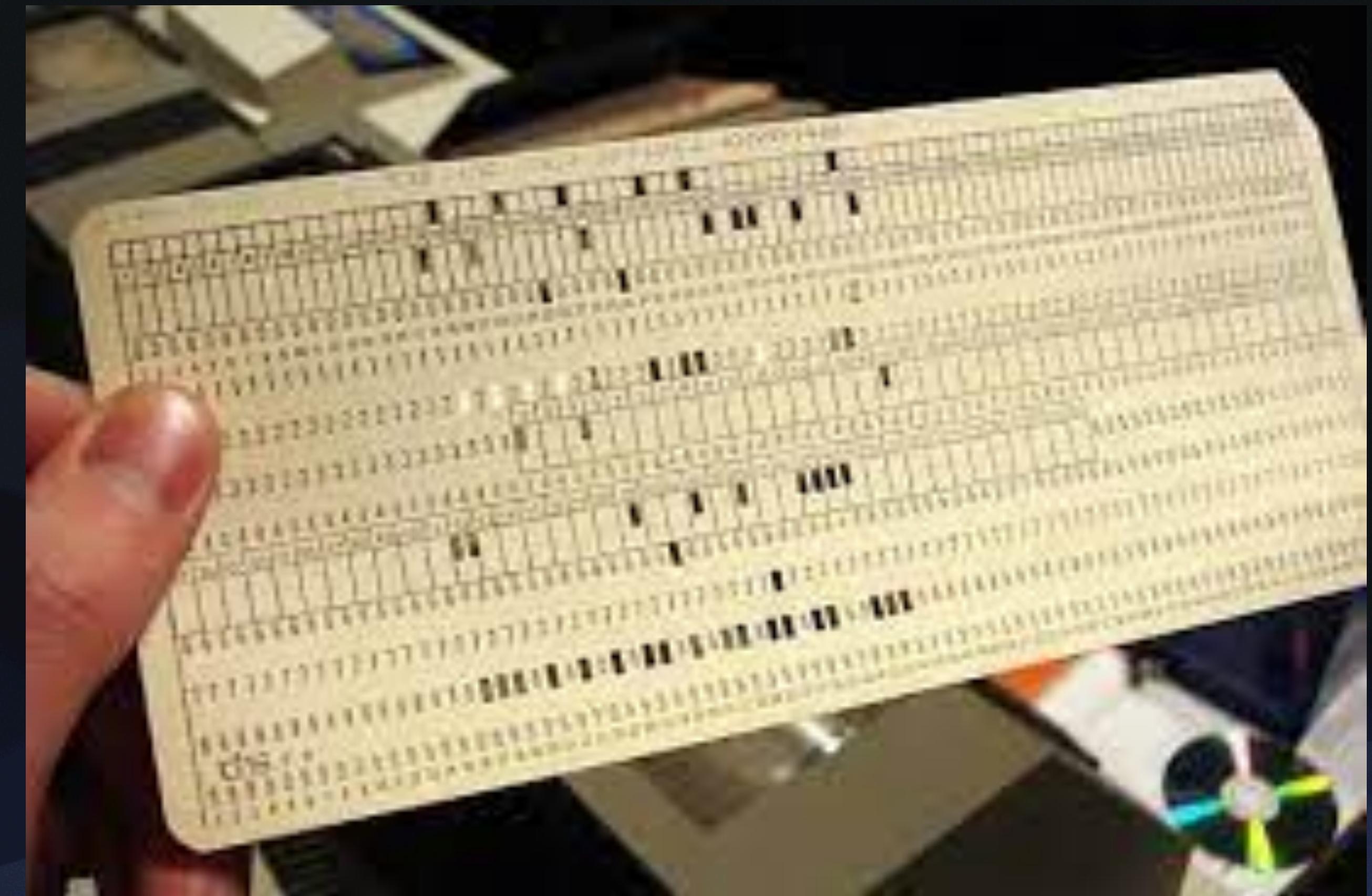
Kent Beck  
Creador de Agile  
Development, Extreme  
Programming y Test  
Driven Development



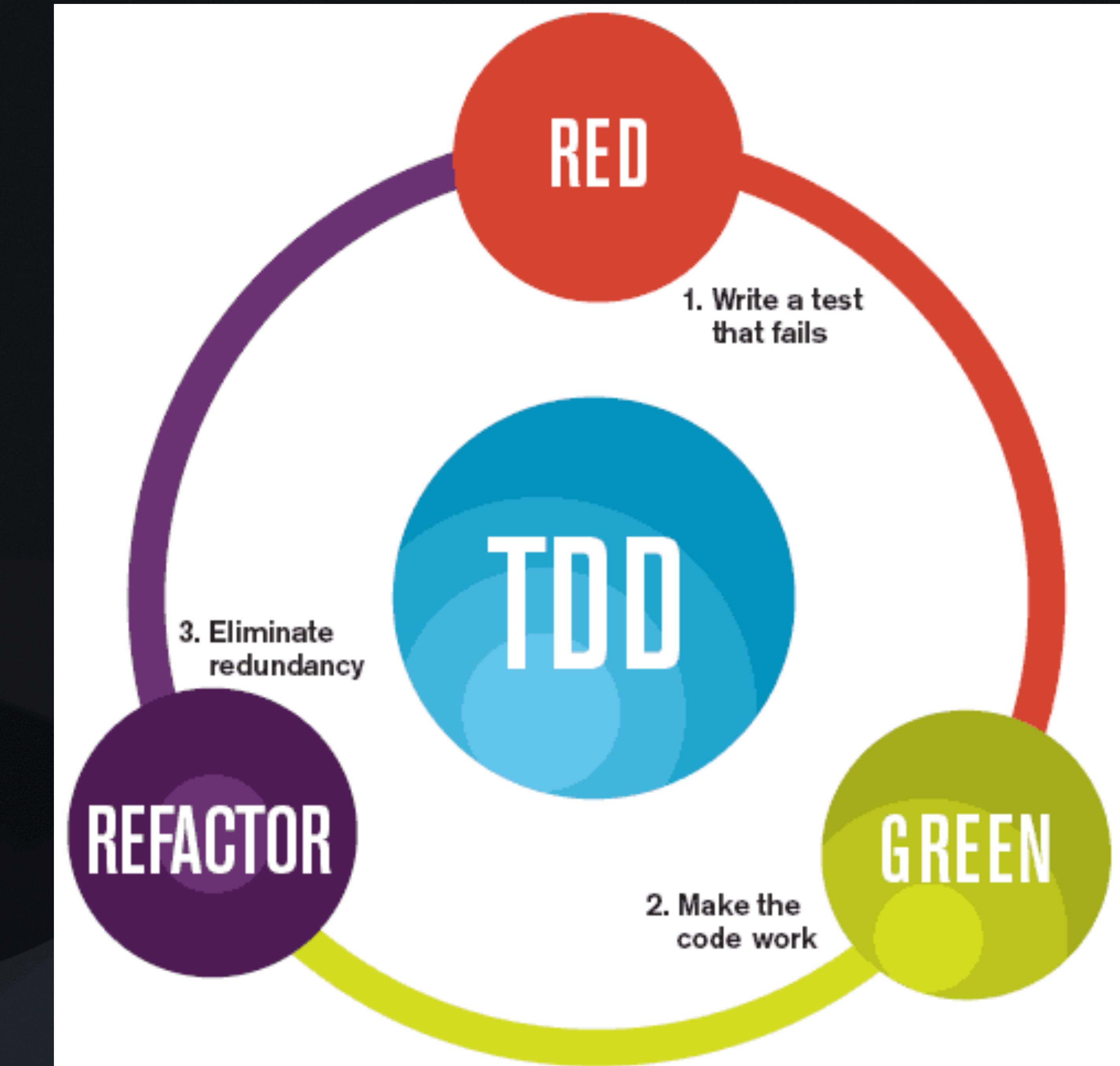
# ¿Kent Beck inventó el TDD?



# ¿Kent Beck inventó el TDD?



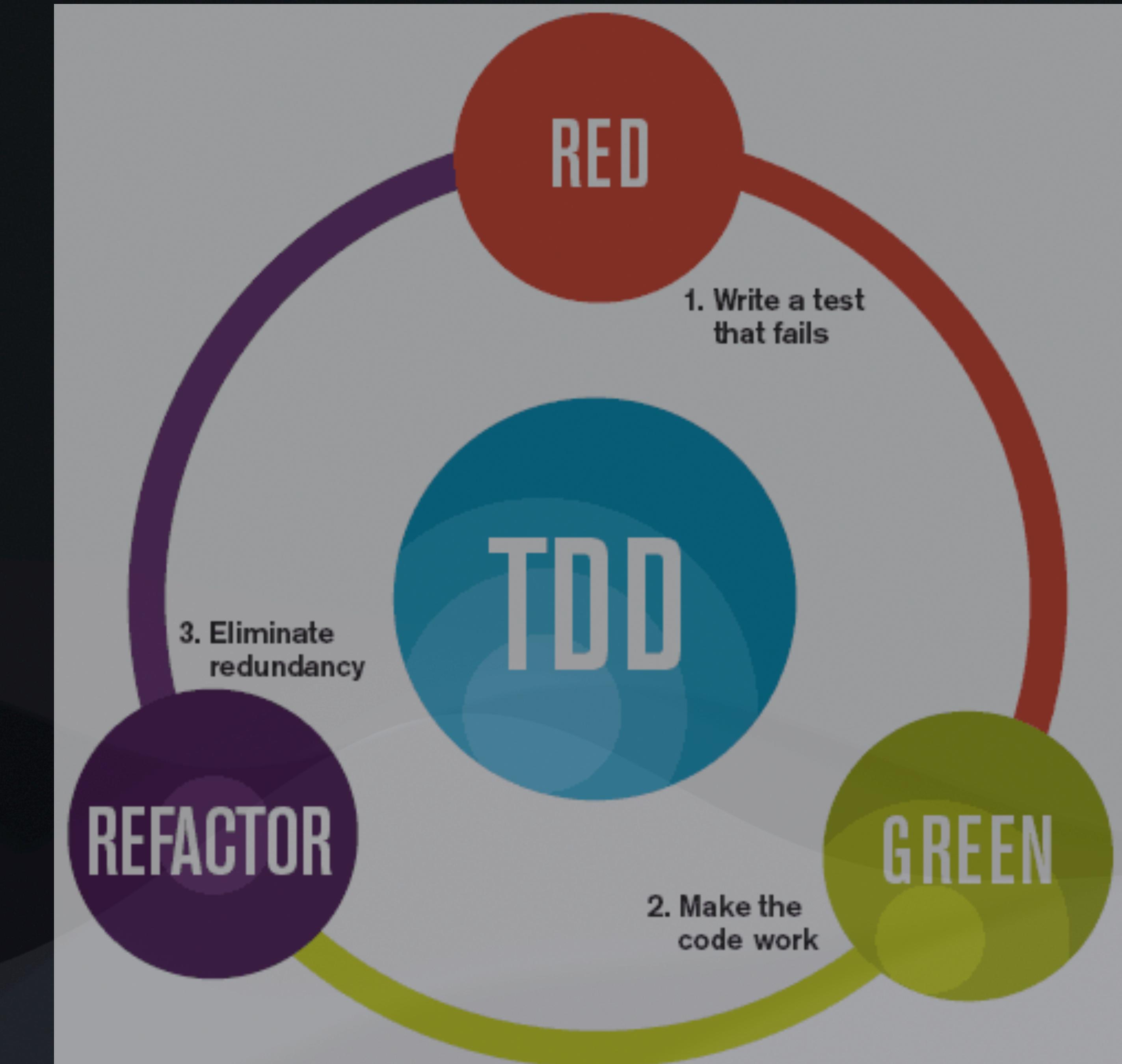
# TDD



The mantra of Test-Driven Development (TDD) is “red, green, refactor.”

# TDD

- **A**rrange
- **A**ction
- **A**ssert



The mantra of Test-Driven Development (TDD) is “red, green, refactor.”

# ¿Por qué TDD?

# ¿Por qué TDD?

Te permite crear un software

Más fiable

Más fácil de mantener

Diseñado con claridad

# ¿Por qué TDD?

Porque obliga a escribir solo el código necesario

Si escribes primero el test y luego sólo escribes el código que lo haga pasar:

- Evitas sobre-ingeniería
- No añades features que nadie pidió
- El código es más sencillo y limpio

El test define la necesidad, el código solo la satisface.

# ¿Por qué TDD?

Porque obliga a pensar antes de programar

Antes de escribir código te debes preguntar:

- ¿Qué debería hacer exactamente esta función?
- ¿Qué inputs tendrá?
- ¿Qué output espero?

Lo que evita improvisar y reduce errores.

# ¿Por qué TDD?

Porque da feedback inmediato

Cada vez que ejecutas los test:

- Te dicen si rompiste algo
- Te dicen qué parte falló
- Te permite avanzar con seguridad

Un código sin test es una bomba y en ningún sitio se trabaja sin al menos test de cobertura

Un código con TDD es un sistema monitorizado desde el primer día.

# ¿Por qué TDD?

Porque hace que el diseño sea mejor

Escribir test primero hace que tu código:

- Sea más modular
- Tenga funciones más pequeñas
- Tenga interfaces claras

Es casi imposible hacer un buen diseño sin testarlo

# ¿Por qué TDD?

Porque reduce drásticamente los bugs

En un proyecto real:

- Con TDD los bugs se detectan antes de integrar el código
- Documentas el proyecto con el comportamiento correcto
- Tenga interfaces claras

Menos bugs = menos tiempo perdido

# ¿Por qué TDD?

Porque te da seguridad para ponerte a refactorizar

Con TDD:

- Te atreves a limpiar código sin miedo
- Puedes reorganizar, renombrar y optimizar
- Si rompes algo, te avisará y sabrás dónde falla

Tu suite de test es tu red de seguridad

# ¿Por qué TDD?

Porque documenta tu proyecto mejor que cualquier documentación

Los test son documentación viva:

- No se quedan obsoletos
- Muestran ejemplos reales de uso
- Explican cómo se debe comportar en cada caso

Leer un test es muchas veces mejor que leer una especificación

# Nuevo entorno de test en python

- Tengo que tener python: `python3 -v`
- Creo un entorno para mi proyecto: `python3 -m venv file`
- Activo mi entorno: `source vent file`
- En el entorno installar el framework de test: `pip3 install pytest` (requiere pip3)
- Disfrutar!!!

# Listado de katas

<https://zerolivedev.gitlab.io/katayuno-jekyll/>

ScubaDivers