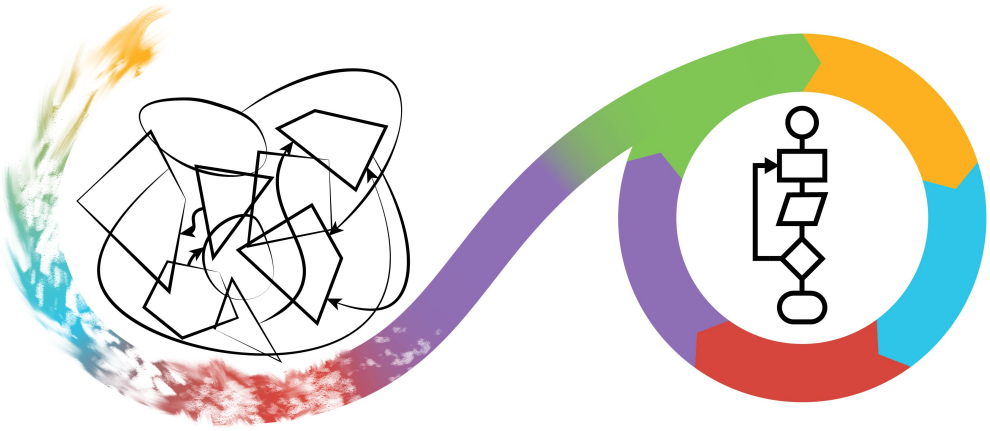
Test-Driven Development

Richèl Bilderbeek

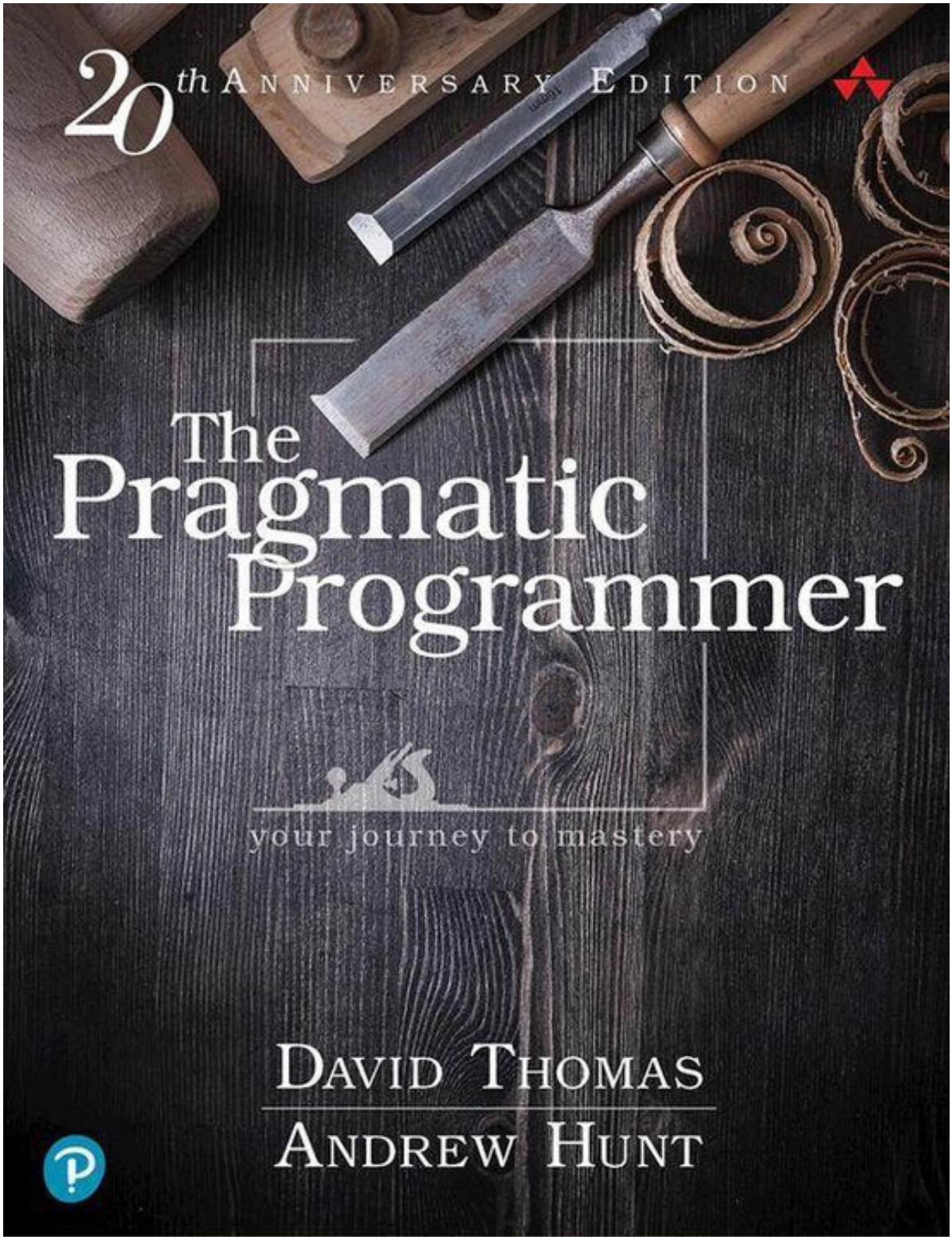
Test-Driven Development 

https://github.com/UPPMAX/programming_formalisms/blob/main/tdd/tdd_lecture/tdd_lecture.qmd



Problem

How do you grow/develop your code?


20th Anniversary Edition
The Pragmatic Programmer
your journey to mastery
David Thomas
Andrew Hunt

Newbie developers

'Just start somewhere'

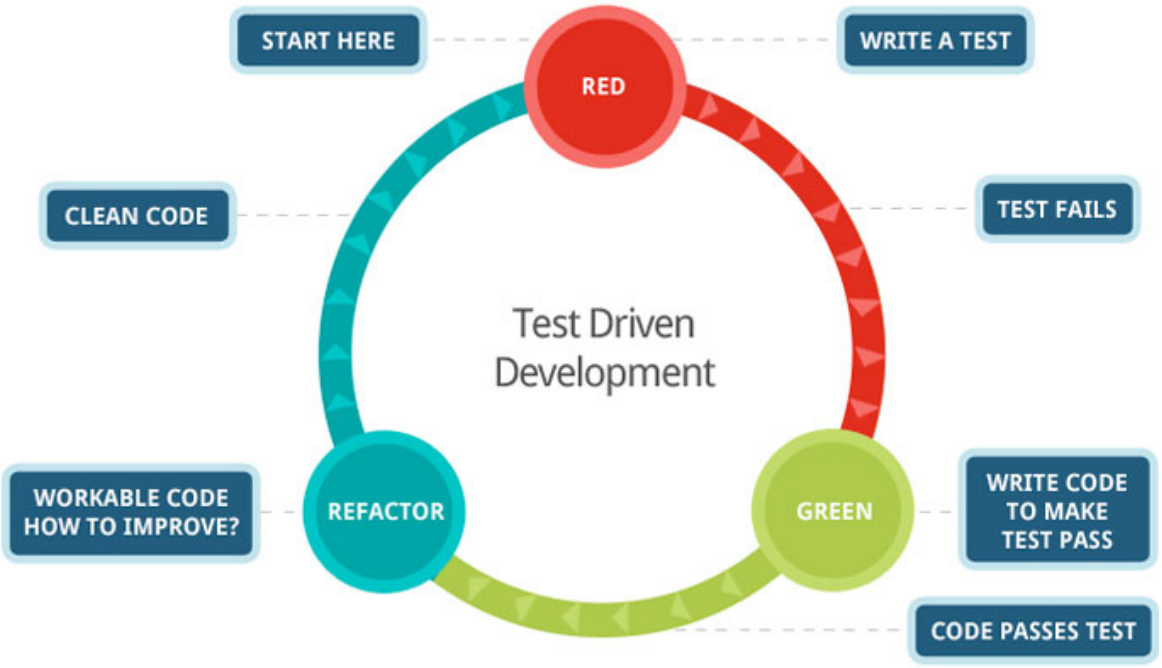
Experienced developers

Work systematically

TDD

Short for ‘Test-driven development’ A systematic way to grow code

TDD cycle



Example exercise: `is_zero`

- Function name: `is_zero`
- Output:
  - Returns `True` if the input is zero
  - Returns `False` if the input is not zero
  - Gives an error when the input is not a number

Live demo (15 minutes)

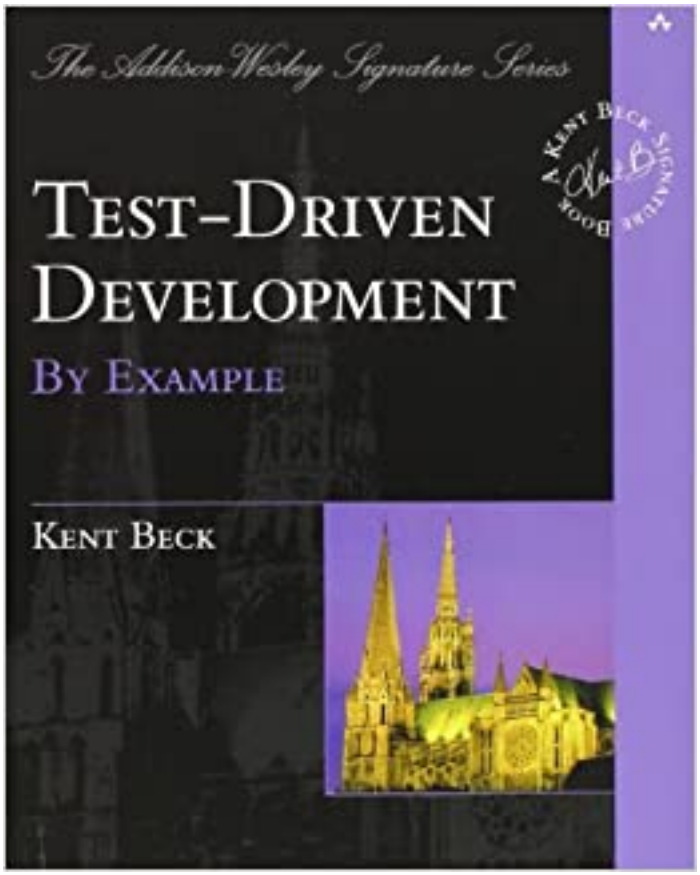
- **Python video for 'is_zero'**
- R video for 'is_one'
- Or see also slides beyond end

Reflection

Q: Do developers really do this?

. . .

A: Yes



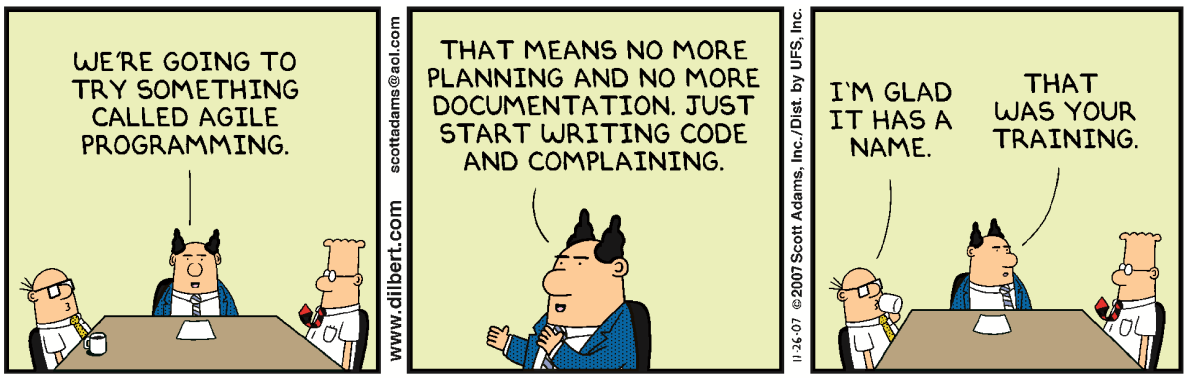
Exercise 1: `is_even`

- Create a GitHub repository called `is_even`
- Share URL of repo with teachers

- Develop a function called `is_even`
- Try to be **exemplary**
- We'll discuss a repo history after the break

Done? Write `is_odd`, then `is_probility`.

Break 1



Reflection

Q: Does this really save time?

. . .

A: No, it takes longer

Study	Extra time	Effect
(1)	16%	18% more black-box tests pass
(2)	15%	2x higher code quality
(3)	15-35%	40%-90% less defects

Reflection

Q: Why do TDD?

. . .

A:

- TDD makes developers more productive (4)
- TDD increases quality of the code (4) (5) (6)

  - There are plenty of costly programming mistakes documented!

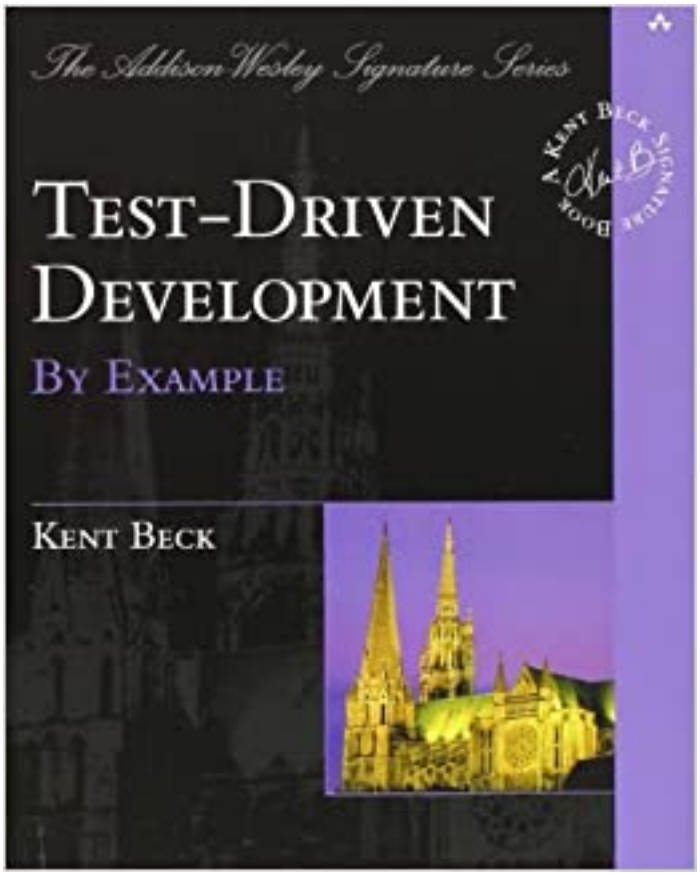
- TDD helps shape the project architecture (7)
- TDD helps better modularisation (8)
- TDD works great with Xtreme programming and CI

Reflection

Q: How many tests should I write?

. . .

A: Until you cannot break your function anymore.



Exercise 2: `is_odd`

- Create a GitHub repository called `is_odd`
- Share URL of repo with teachers

- Develop a function called `is_odd`
- Try to be **exemplary**
- We'll discuss a repo history after the break

Done? Try exercise 3: `is_probability`.

Break 2

Exercise 3: `is_probability`

- Create a GitHub repository called `is_probability`
- Share URL of repo with teachers
- Develop a function called `is_probability`
- Try to be **exemplary**
- We'll discuss a repo history after the break

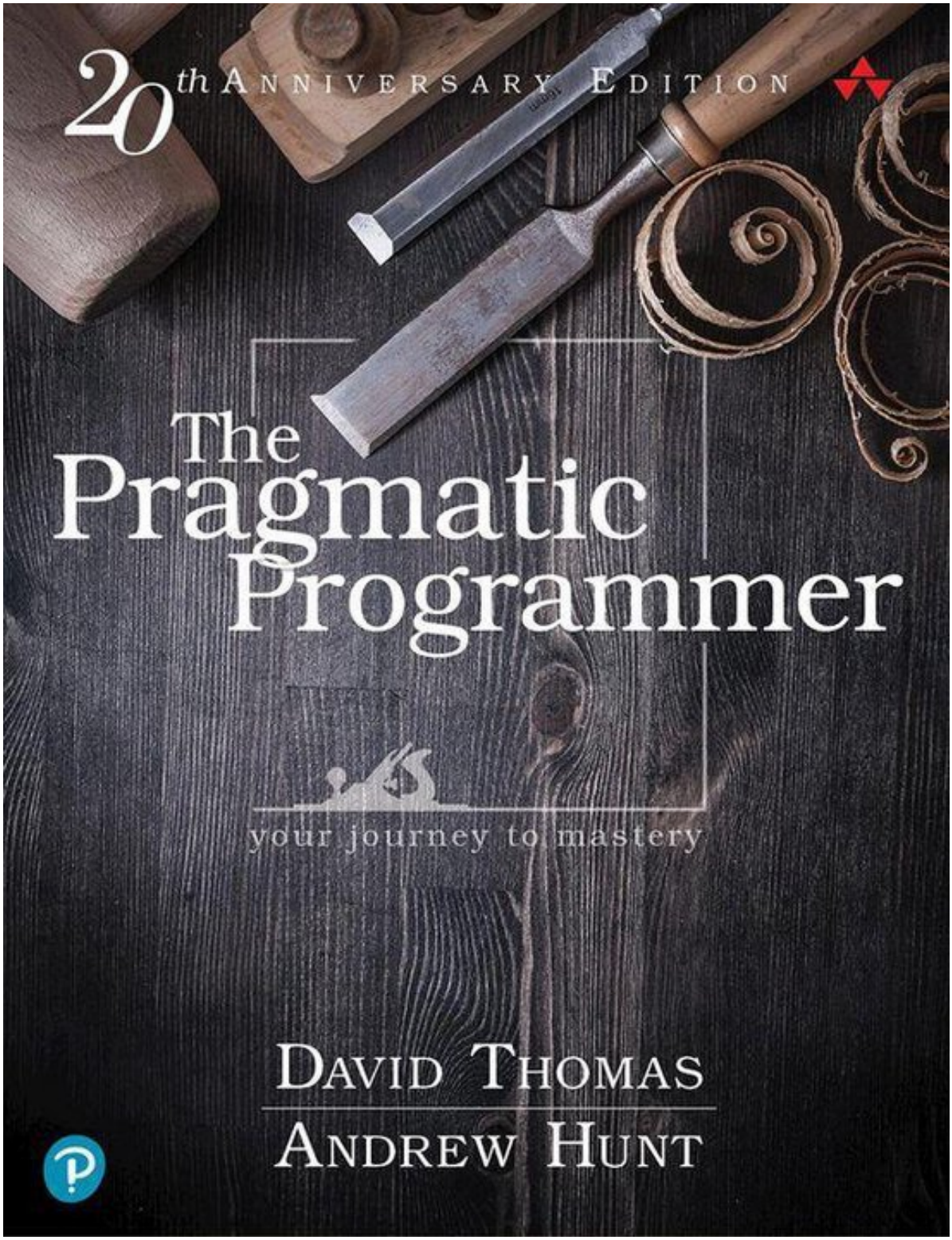
Extra exercises

Done?

Exercise	Function name	Function purpose
4	`is_number`	Determines if an object is a number
5	`are_numbers`	Determines if an object is a list of numbers
S1	`is_roman_number`	Determine if a string is a roman number
S2	`is_prime`	Determine if a number is a prime number

Bottom line

- Today we wrote **unit tests**
- It is only those your boss may read
- The literature assumes a responsible programmer writes tests, in C++ (9), R (10) and Python (11)


20th Anniversary Edition
The Pragmatic Programmer
your journey to mastery
David Thomas
Andrew Hunt

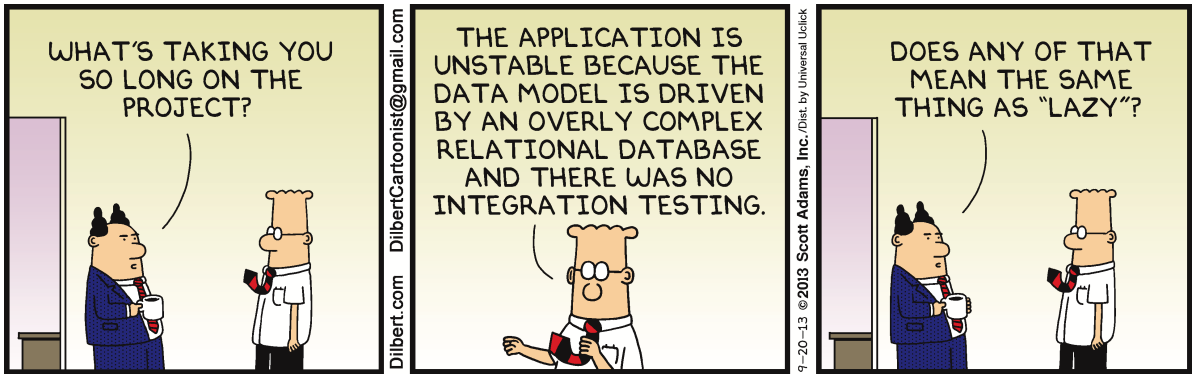
Problems

- We only test manually
- We only test om our computer
- We are not sure if our functions are tested completely
- We do not test the code for style
- We should consider using a testing framework

Finally

Time for a Reflection!

Afterwards, you can rest or ask your final questions.



The End

Links

- Mentimeter presentation of previous year

TDD cycles in text

- In both Python and R

First example: `is_zero`

- Function name: `is_zero`
- Output:
  - Returns `True`/`TRUE` if the input is zero
  - Returns `False`/`FALSE` if the input is not zero
  - Gives an error when the input is not a number

Cycle 1, red: write a test that breaks

```
assert is_zero(0)
```

```
library(testthat)
expect_true(is_zero(0))
```

code that is not run, uses , as a worm cannot run.

Cycle 1, green: make the test pass

```
def is_zero(number):
  return True

assert is_zero(0)
```

```
library(testthat)
```

```
is_zero <- function(number) {
  TRUE
}

expect_true(is_zero(0))
```

Cycle 1, blue: refactor and commit

```
git add .
git commit -m "Add stub of 'is_zero'"
git push
```

Cycle 2, red: write a test that breaks

```
assert is_zero(0)
assert not is_zero(42)
```

```
expect_true(is_zero(0))
expect_false(is_zero(42))
```

Cycle 2, green: make the test pass

```
def is_zero(x):
  return x == 0

assert is_zero(0)
assert not is_zero(42)
```

indent of 2 is non-standard, see PEP 8

```
library(testthat)

is_zero <- function(number) {
  number == 0
}

expect_true(is_zero(0))
expect_false(is_zero(42))
```

Cycle 2, blue: refactor and commit

```
git add .
git commit -m "'is_zero' responds correctly to numbers"
git push
```

1. George B, Williams L. A structured experiment of test-driven development. Information and software Technology. 2004;46(5):337–42.
2. Bhat T, Nagappan N. Evaluating the efficacy of test-driven development: Industrial case studies. In: Proceedings of the 2006 ACM/IEEE international symposium on empirical software engineering. 2006. p. 356–63.
3. Nagappan N, Maximilien EM, Bhat T, Williams L. Realizing quality improvement through test driven development: Results and experiences of four industrial teams. Empirical Software Engineering. 2008;13:289–302.

4. Erdogmus H, Morisio M, Torchiano M. On the effectiveness of the test-first approach to programming. IEEE Transactions on software Engineering. 2005;31(3):226–37.

5. Alkaoud H, Walcott KR. Quality metrics of test suites in test-driven designed applications. International Journal of Software Engineering Applications (IJSEA). 2018;2018.

6. Janzen DS, Saiedian H. Test-driven learning: Intrinsic integration of testing into the CS/SE curriculum. Acm Sigcse Bulletin. 2006;38(1):254–8.

7. Mayr H. Projekt engineering: Ingenieurmäßige softwareentwicklung in projektgruppen. Hanser Verlag; 2005.

8. Madeyski L, información G de sistemas de. Test-driven development: An empirical evaluation of agile practice. Springer; 2010.

9. Stroustrup B, Sutter H, et al. C++ core guidelines. Web Last accessed February. 2018;

10. Wickham H. Advanced R. CRC press; 2019.

11. Van Rossum G, Warsaw B, Coghlan N. PEP 8–style guide for Python code. Python org. 2001;1565.