TDD for Kotlin (and beginners)

A brief crash course

whoami

- Android engineer by day @ LichtBlick SE
- KMP believer by night
- Rust dilettante in my free time
- organizer for KUG & Rust & XTC Berlin

whoami

- Find me on Twitter/GitHub: @BitPogo
- Find me on all the Slacks: Matthias Geisler
- Join: Kotlin Slack; Software Crafters Slack

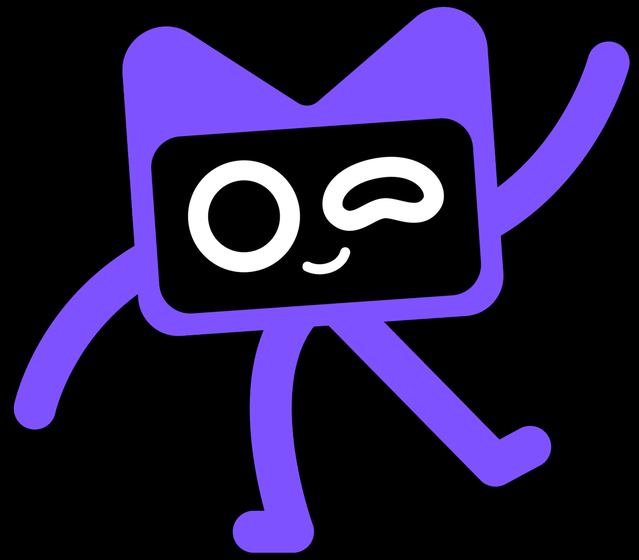
Let’s get to know each other...

1. Who are you and on which side of the force you are working?
2. Why you are here?
3. What are your goals?

Please pair, collect the information

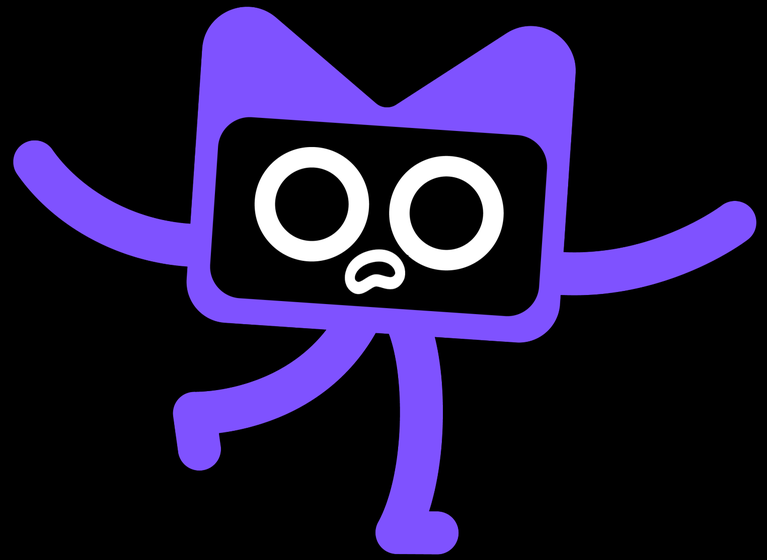
and your partner should share it with the group

you have 10 minutes



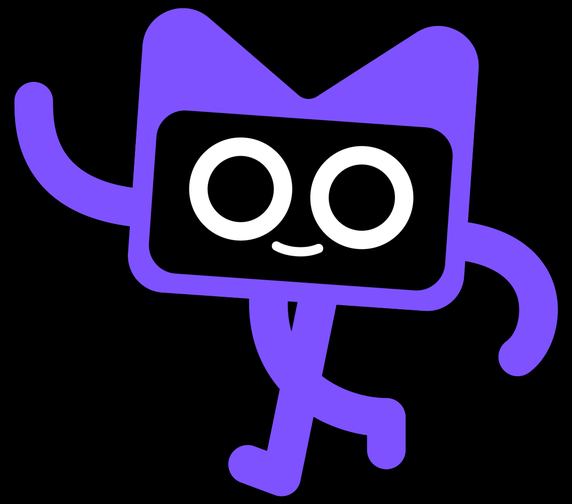
What we will not achieve today…

- TDD Mastery
- A understanding of the methodologies in depth
- You have a clear picture how to apply it
- …



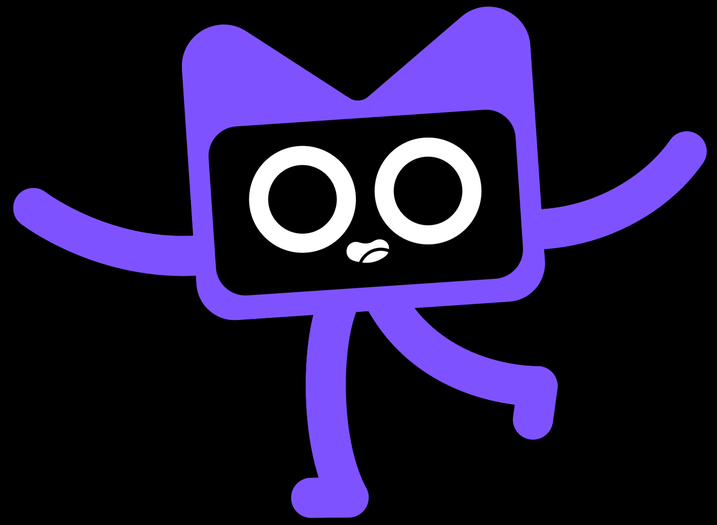
…but we'll manage

- Give you a glimpse what TDD is and how it works
- You know where to go on from here next
- You know whom to ask 😉
- …



What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ⬅️
3. Let’s make our hands „dirty"
4. Give Feedback

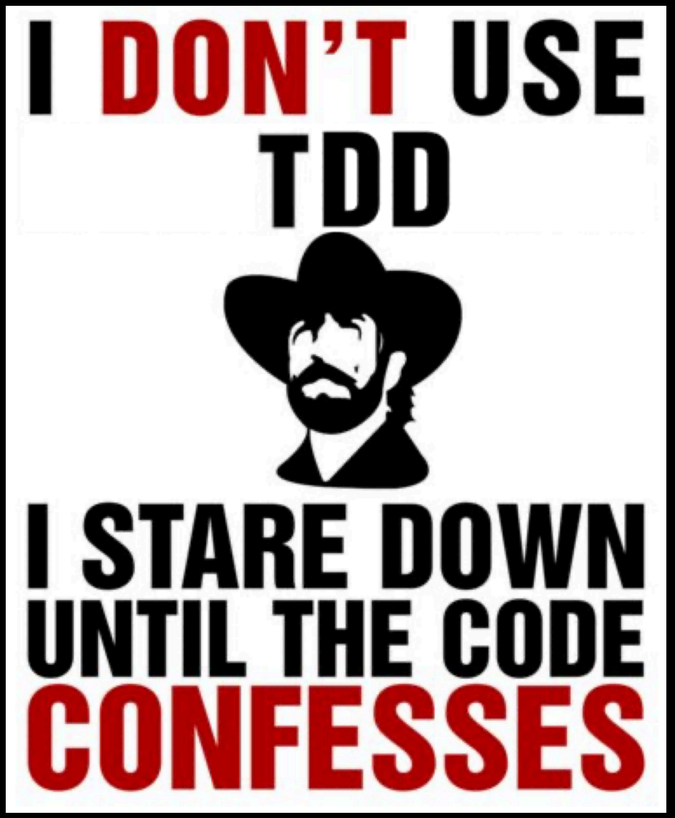


Why we are doing TDD?

THIS PAGE
INTENTIONALLY
LEFT BLANK

Why we are doing TDD?

- FEAR
- Feedback
- Protect us from ourself
- Lazyness
- Fun
- …



How we get there?

...conceptually...

Your daily mantra should be:

Write a failing test...

... make the test pass...

... refactor your code!

Writing a failing test...

- You are not allowed to touch any production before it
- Compiler errors count as a failing test

Why?

This way we make clear what we assume happens (behaviour), what are dependencies...

Make the test past...

- You are allowed to write only enough code to make the test pass

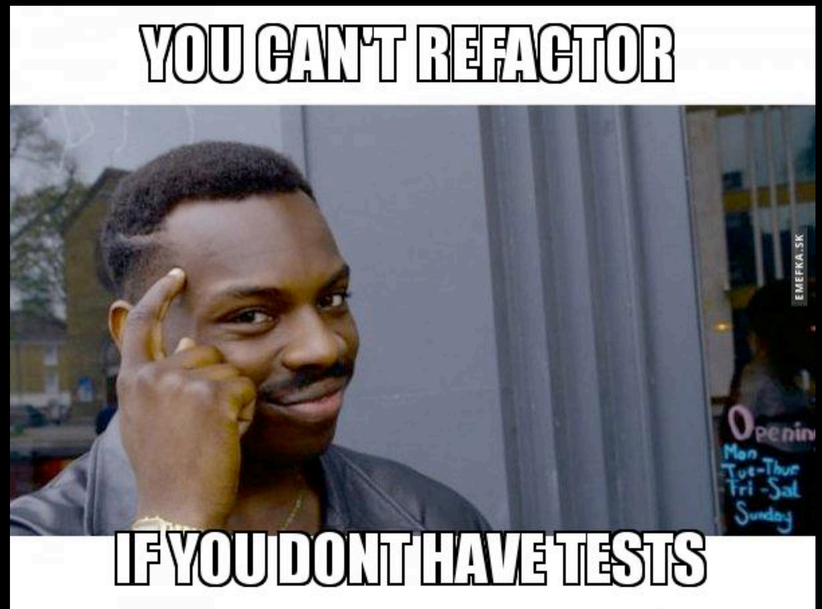
Why?

This way we make sure we do not cheat ourself and introduce behaviour what we had not in mind and can produce side effects.

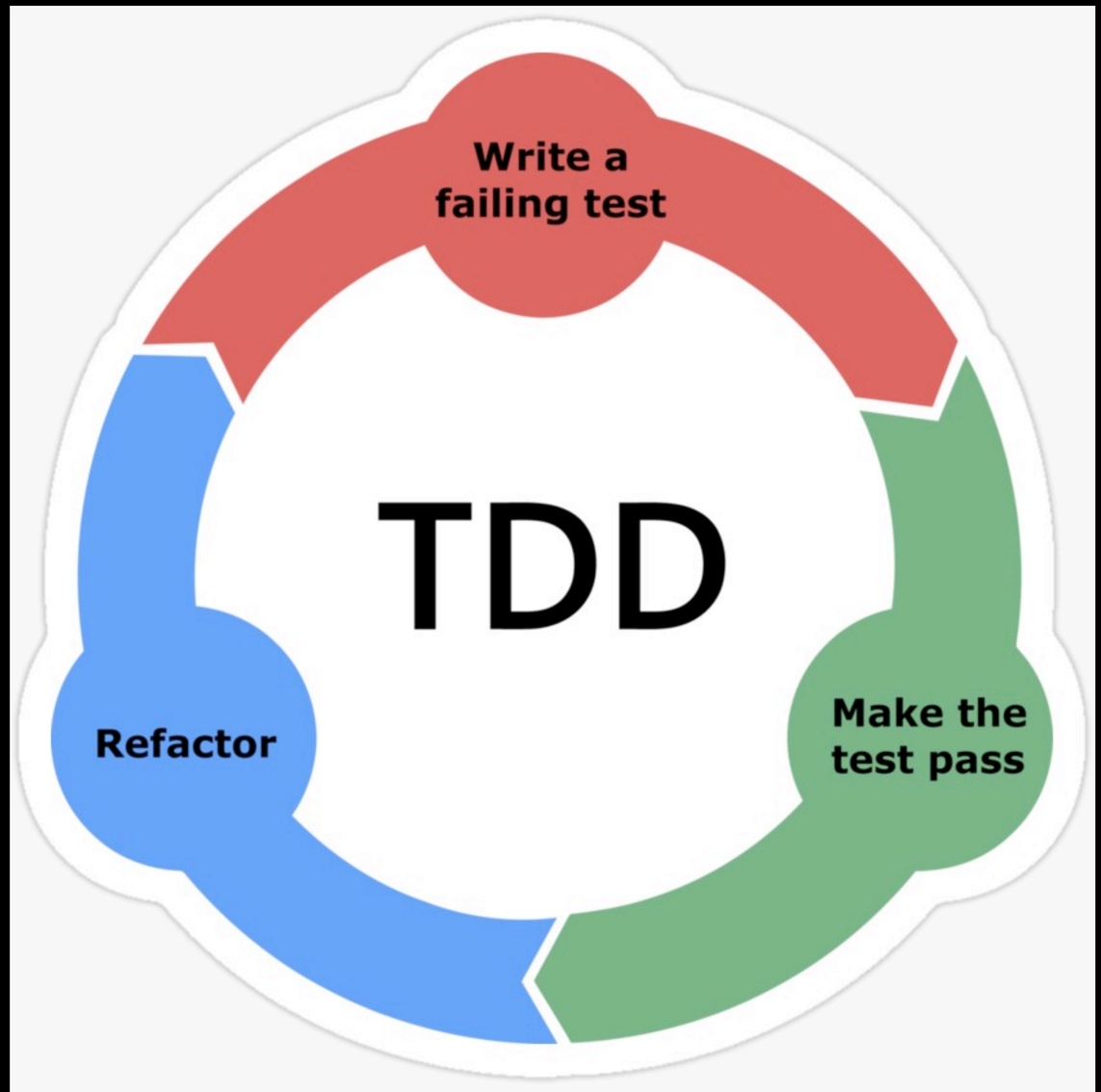
Refactor...

Why?

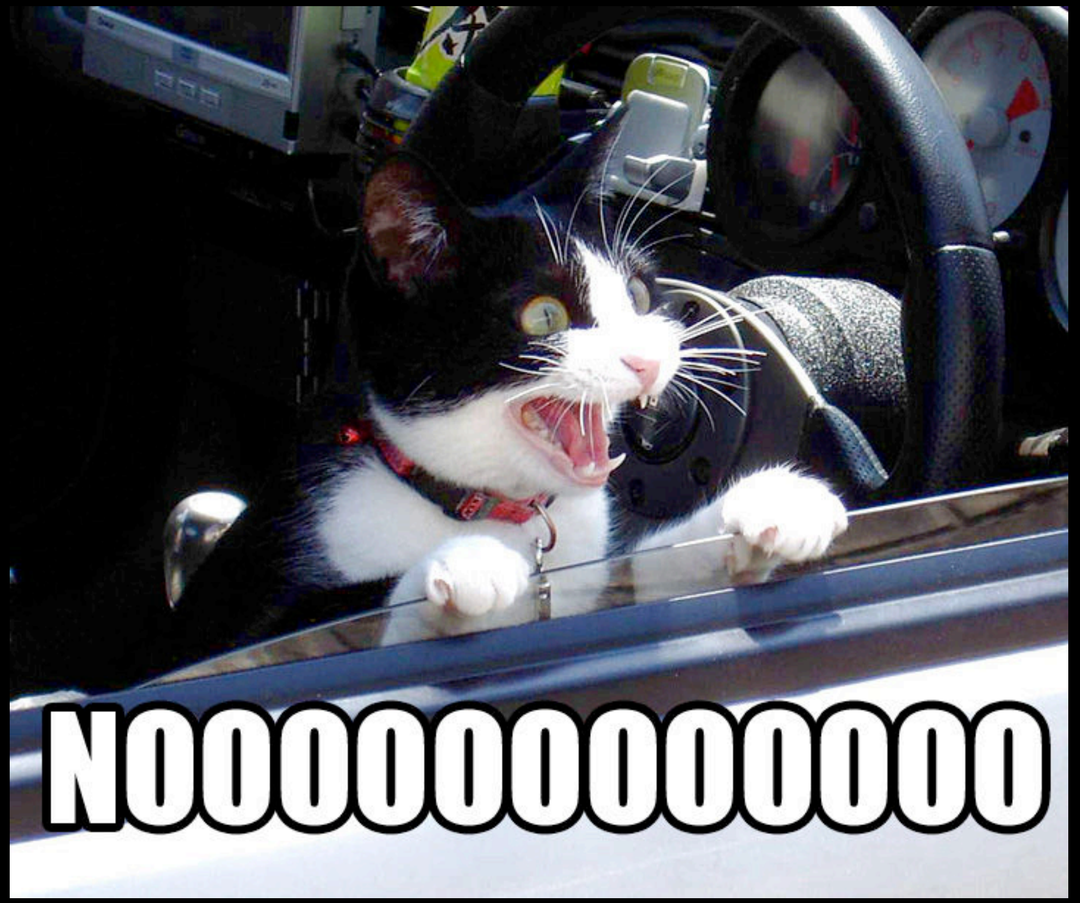
Clean up your mess, before your mess cleans you up. (Like duplication)



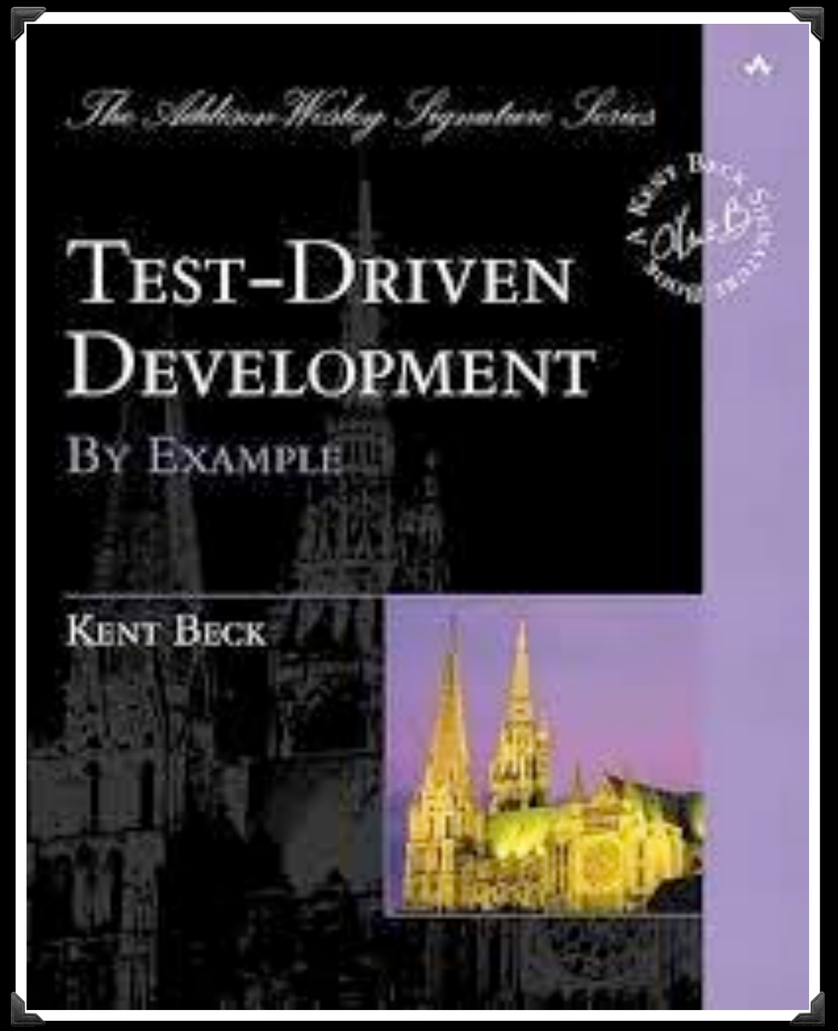
The cycle of life



Is that all?



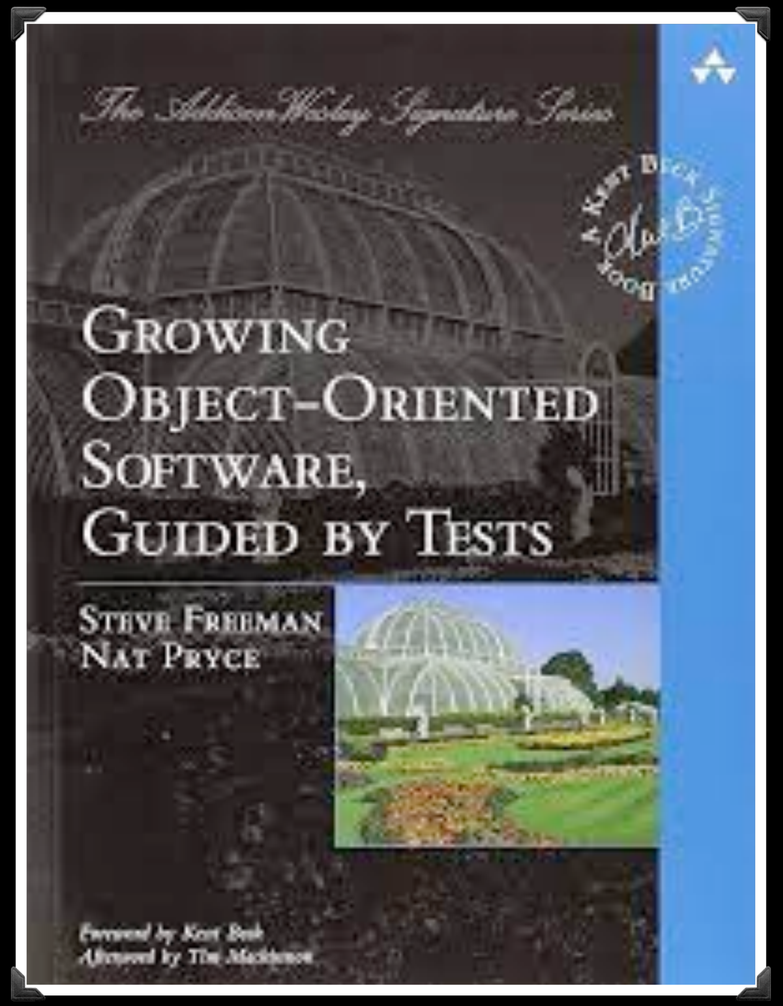
Chicago Style/Classic Style



„TDD is a good for geeks who form emotional attachments to code“

- Mini “Integration Tests“
- Fundamentals
- Fake it until you make it
- …

London Style/Mockist Style



All you want is green

- Acceptance Tests!
- Isolation is Queen
- Mocks (only on interfaces!)
- …

...and?

Both approaches are not mutual exclusive... but you will end up preferring one of them and even mix them up!

Use the approach which gives you/your team the highest confidence

Strategies

- Inside-out

  => we work us from single components to the system

- Outside-in

  => we work us from the system to components

What else?

- You need to learn to listen to the tests, since they constantly telling you something
- You need to be patient with yourself… learning TDD needs time… and practice!
- You need to be disciplined since changing you coding behaviour is a hard!

To think about...

What does that mean for things like Code Coverage?

What does that mean for coding conventions like Yoda style?

What does that mean for a team?

Please pair and discuss this in 15 minutes.

Little helpers

To practice:

- https://cyber-dojo.org

To read even more:

- Design Patter Book
- Mocks aren't Stubs
- Refactoring Book

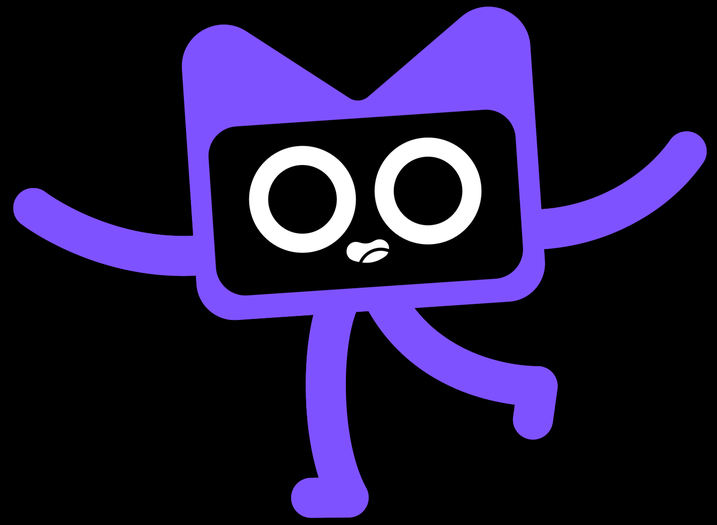
Break

"You can't program well if your back hurts.“

– Kent Beck

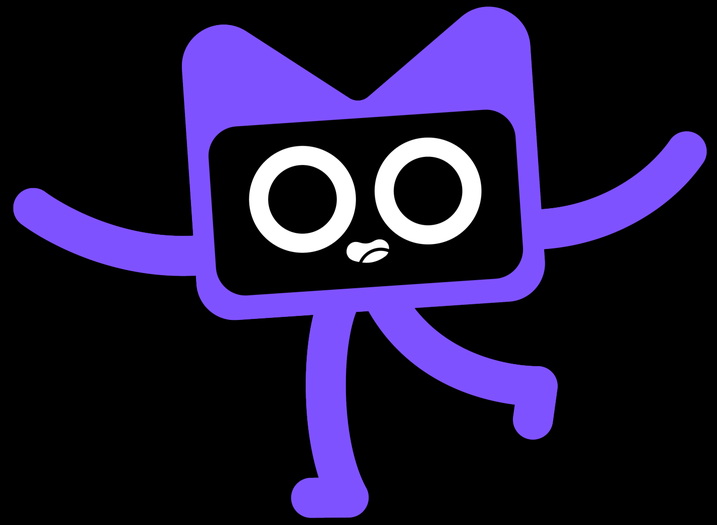
What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ✅
3. Let’s make our hands „dirty"⬅️
4. Give Feedback



What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ✅
3. Let’s make our hands „dirty" ✅
4. Give Feedback⬅️



Please give Feedback

https://tinyurl.com/yc27v5am