TEST-DRIVEN DEVELOPMENT

Max Brauer

ABOUT ME

studied computer science

Python developer

working at Delivery Hero

@mamachanko

TWO AXIOMS

We want to write software

We want to assert that it does what it's supposed to

SOFTWARE TESTING

manual testing

automated testing

HUMANS ARE...

extremely powerful testers

very slow testers

very unprecise

easily bored

very expensive

MACHINES ARE...

very fast testers

very precise testers

hard to set up & maintain

never bored...

PYTHON TOOLBOX

to the Python shell!

the possibilities are endless...

```
<=
>=
!=
%
assert
...
```

... but we don't want/need to reinvent the wheel!

UNITTEST

the classic testing framework

everything one needs

quasi standard

built-in

"code or it didn't happen!"

TEST-DRIVEN DEVELOPMENT

a special school of testing/programming

what's the difference?

TESTS COME FIRST. NO MATTER WHAT.

WHY'S IS THIS A BIG THING?

tests after are justification

tests first are challenge

stress requirements

how to test

good design emerges

minimalism

THE THREE LAWS OF TDD

as by Robert C. Martin

1. you may not write any production code without a failing test
2. you may not write more of a test than is sufficient to fail
3. you may not write more production code than is sufficient to pass

test
code
refactor
repeat

BUT WHY??

it emphasizes the process

helps you understand requirements

you will think about fringe cases early

breaks problems into manageable units

tests serve as documentation

good design emerges

but most of all:

IT GIVES YOU SAFETY.

WHY NOT?

not easy to understand

writing good tests is hard

you need to be very disciplined

you need to excercise a lot with Katas

Thanks!

Break first!
Katas after!

LINKS

these slides

The three rules of TDD

Python's unittest documentation

TDD katas