Testing

Let's talk about testing

Why testing in general?

- Ensure changes to your code work as expected
- Ensure reliability, functionality, and quality of software
- Documentation
- More...

Manual vs Automated testing

Let's focus on automated **unit tests** here:

- Low level, it tests individual methods (functions) of the classes, components, or modules.
- They are pretty cheap to automate and run.

How do we have good coverage without testing every single function?...

...and avoiding TDD. (: And remember: 100% coverage does not guarantee it will work forever.

Let's cover branches

Test all possible branches from the root call, and it will indirectly test the other child calls:

```
translate(word, language)
translator(word, language)
_get_portuguese(word)
```

When avoid relying on the root call?

- Complex functions/classes
  - Create tests for complex logic, functions, and classes even if tested more than once.
- Maybe when you have deeper function calls
  - Make the tests for the deeper functions
- Follow your instincts, but let's try to avoid redundant tests.

Covering branches depends on the tool used

- We are using **coverage** for this case, but other tools can have other ways to measure *decision branches*.
- I remember seeing in JavaScript, probably **Jest** to cover branches, and it may be different from **coverage**.
- `a = b or c` have only **one decision branch** for **coverage**.
- But It actually has two branches at least. Because `a` can be `True` or `False`. We might test both cases.
- Jest do something like it as I remember.

References

[1] Allainclair Flausino dos Santos: allainclair@gmail.com
[2] Atlassian