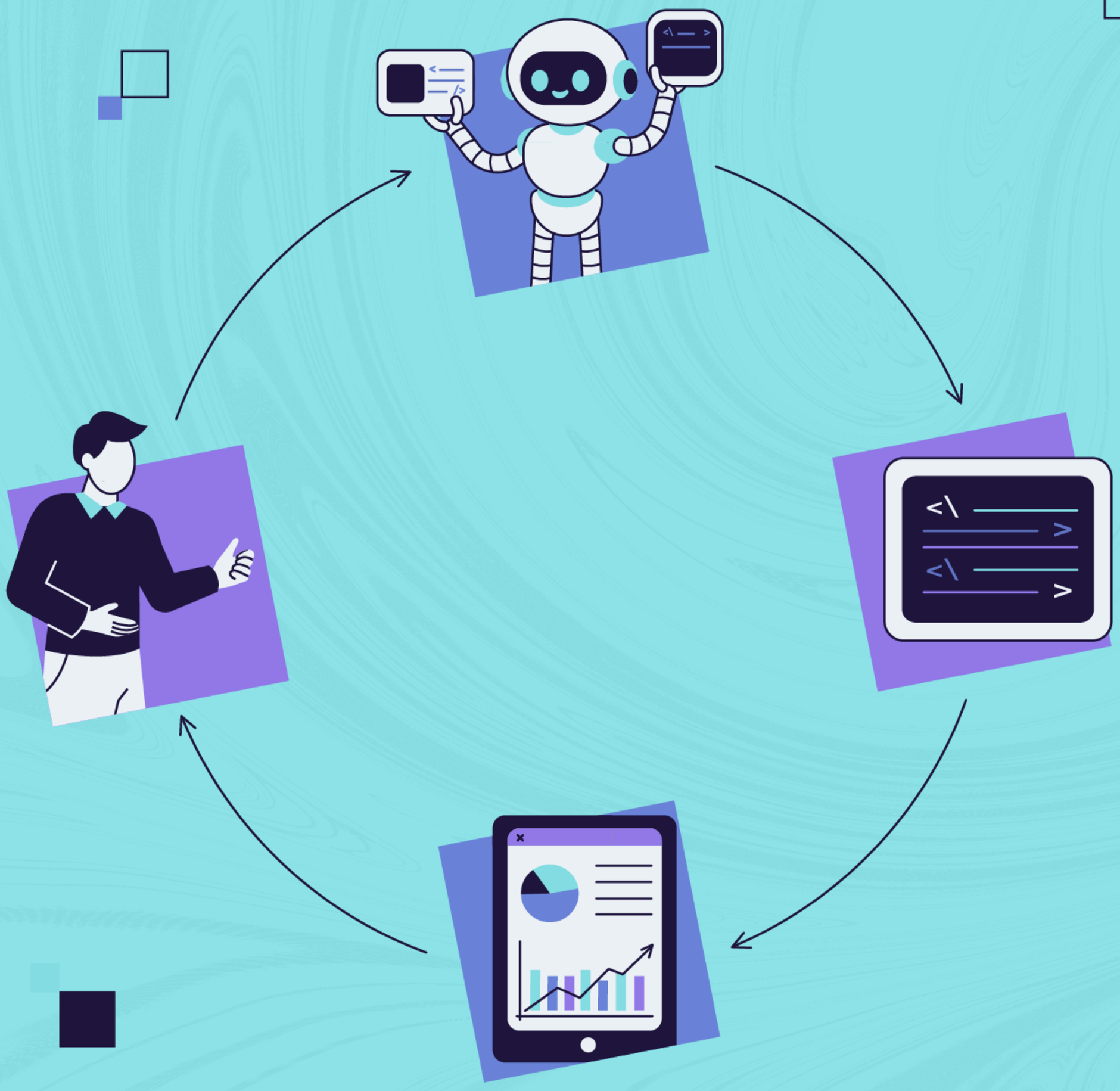
AI POWERED REWRITE FROM PYTHON TO RUST



Improve your code's performance by rewriting the computationally-intensive logic to Rust.

ENSURE EXISTING CODE IS WELL TESTED

Use AI agent to prepare a plan

Check the plan

Use auto-accept-edits

Check the tests

Does my plan include...?
Do my tests....

Cover All Logic

Logic without a test has no way to prove it survived the rewrite.

Cover All Variants

One format tested, three assumed.
Assumptions don't port to Rust.

Cover the Edge Cases

The rewrite will find these if you don't.

If it's not tested in Python, the Rust version can break it and you won't know it.

REWRITE EXISTING TESTS TO CONTRACT TESTS AND ADD A DUMMY RUST WRAPPER

Contract class

All existing tests

Abstract factory fixture.

Test class that inherits from the contract

Python Test Class
extends Contract Class

Rust Test Class
extends Contract Class

Dummy wrapper for Rust code

Tests for the PYTHON implementation must PASS

Tests for the RUST implementation must FAIL

Two objects, same set of tests

BUILD THE RUST IMPLEMENTATION

Install Rust compiler and Cargo

Write a prompt that is specific about the structure: what files to create, what function signature to expose, and how to wire things up.

The prompt should include requirement that all the tests should pass

The changes prompt causes

src/lib.rs

Rust implementation

pyproject.toml

tell Python how to build the Rust extension

updated wrapper

wrapper now ensures Rust and Python both expose same interface

Cargo.toml

Rust project configuration

USE PROPERTY-BASED TESTING TO ENSURE THE SAME OUTPUT FROM RUST AND PYTHON FOR ANY GIVEN INPUT

Generate Hypothesis strategies

Generate property tests

Each test runs n randomly generated examples you might not think of.

RECAP

1. **Expand test coverage**
   So you know what behaviour to preserve
2. **Restruscture tests into contracts**
   So both implementations share the same spec
3. **Implement the Rust version**
   Verify it against that spec
4. **Use property-based testing**
   So you find the bugs that intial test missed