BLOCK NONCE #6: Snapshot Testing

What?

Snapshot tests compare a value against a hard-coded expected output, stored right there in the codebase. If the output changes — the test fails and shows you the diff, as if to say: "Here's what the output used to be. Here's what it is now. Are you cool with this?"

Why?

Imagine you've just finished writing a fancy function:

```
fn div_ceil(x, y) { x / y }
```

and now it's time to test it!

```
#[test]
fn test_div_ceil() {
  assert_eq!(div_ceil(7, 2), 7 / 2);
}
```

If the function is wrong, the test is wrong **in the exact same way** — and you'll never catch it.

```
#[test]
fn test_div_ceil() {
  assert_eq!(div_ceil(7, 2), 4);
}
```

This snapshot test checks the expected result without repeating the implementation. It's also easier to review the test this way.

How?

Just find an output you know is correct and compare it to the result of the function.
How **NOT** to do it ?

❌ Don't assume snapshot tests give you full coverage, they only check one specific case!

❌ Don't snapshot non-deterministic output, for example, hashmaps!

❌ Don't snapshot output that changes over time, for example, timestamps!

❌ Don't snapshot output that depends on the environment, for example, file paths!

TIP💡 Snapshot tests are also known as golden tests. Why "golden"? Because you're comparing the current output to a "golden" version — a known-good result saved from a previous run.