BLOCK NONCE #6: Snapshot Testing

What?

Snapshot tests are tests that compare a value against a hard-coded expected output, stored right there in your code (or in a separate file). 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 we still cool with this?"

Why?

Imagine ou have just finished to write a fancy function:

```
fn square(x: i32) -> i32 {
  x * x
}
```

and now it time to test it!

```
#[test]
fn test_square() {
  assert_eq!(square(3), 3 * 3);
}
```

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

```
#[test]
fn test_square() {
  assert_eq!(square(3), 9);
}
```

This snapshot test checks what the result should be, not just repeats how it's computed.

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 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/user, 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.