**Introductory Rust Observations**

**Memory safety doesn't mean free from memory leaks**

In Rust, memory safety means:

* No use-after-free: you cannot read/write freed memory.
* No double-free: ownership rules ensure each value is dropped at most once.
* No dangling pointers or buffer overflows in safe code.

But memory leaks are still possible, because a leak is just memory that is never freed (still reachable, or intentionally forgotten). Rust considers leaks safe because the memory is never reused in a way that could corrupt other data.

Common causes of memory leaks:

* Rc/Arc are shared ownership. If two objects own each other with strong references, their counts never hit zero → destructors never run → memory leak.
* A detached thread/task
* A callback registry you never clear
* A global/static that always stays alive