COMP6991 23T1

Rust Basics

Live Example

HELLO, WORLD!

Live Example

GUESSING GAME

Live Example

STRUCTS

Live Example

ENUMS

Live Example

OPTIONAL VALUES

Rust
`Option<T>`

NON-MAGIC TYPE

NO IMPLICIT NULLABILITY

NO IMPLICIT UNWRAPPING

UNWRAP NONE => UNWIND

C
`<type> *`

COMPILER BUILT-IN (MAGIC)

IMPLICIT POINTER NULLABILITY

UNWRAP ON * AND ->

DEREFENCE NULL => UB

ROBUST PROGRAMS MUST BE DEFENSIVE!

C++17
`std::optional<T>`

SIMILAR TO RUST OPTION

NON-MAGIC TYPE

IMPLICIT POINTER NULLABILITY

UNWRAP NULLOPT => UB

Java
`Optional<T>`

SIMILAR TO RUST OPTION

NON-MAGIC TYPE

IMPLICIT OBJECT NULLABILITY

UNWRAP EMPTY => UNWIND

OPTIONAL<T> CAN BE NULL!

Golang
`* <type>`

SIMILAR TO C

COMPILER BUILT-IN (MAGIC)

IMPLICIT POINTER NULLABILITY

DEREFERENCE NULL => CRASH

USING POINTERS CAN BE PAINFUL

Python
None

ANY VALUE IN PYTHON CAN BE NONE

IMPLICIT UNWRAP EVERYWHERE

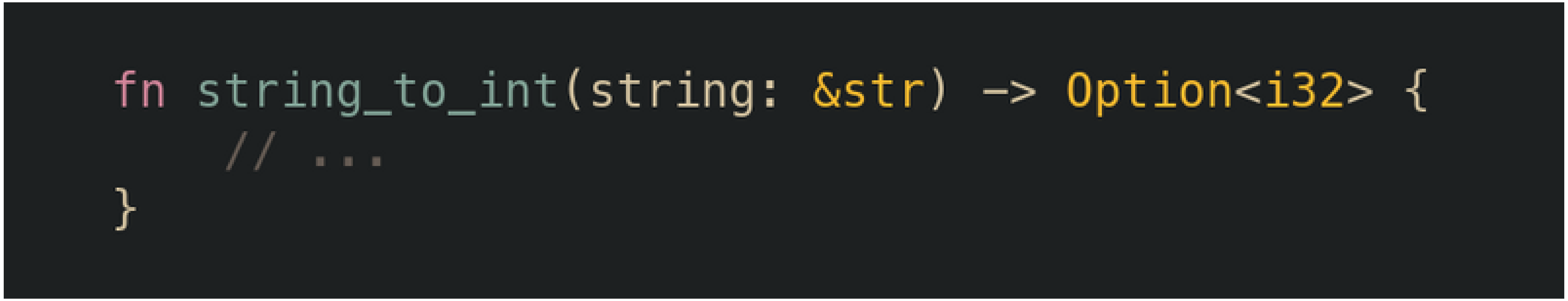
UNWRAP NONE => UNWIND

LIFE ON THE EDGE!

Question

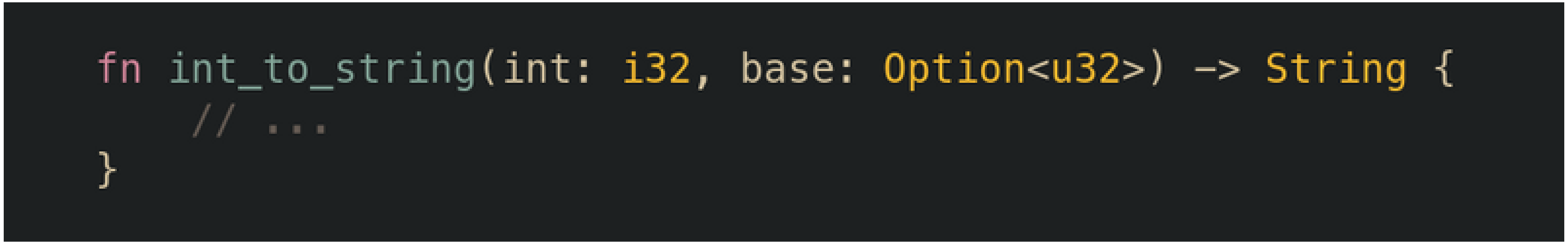
WHAT DOES AN "OPTIONAL VALUE" EVEN REPRESENT?

This function may not produce an output



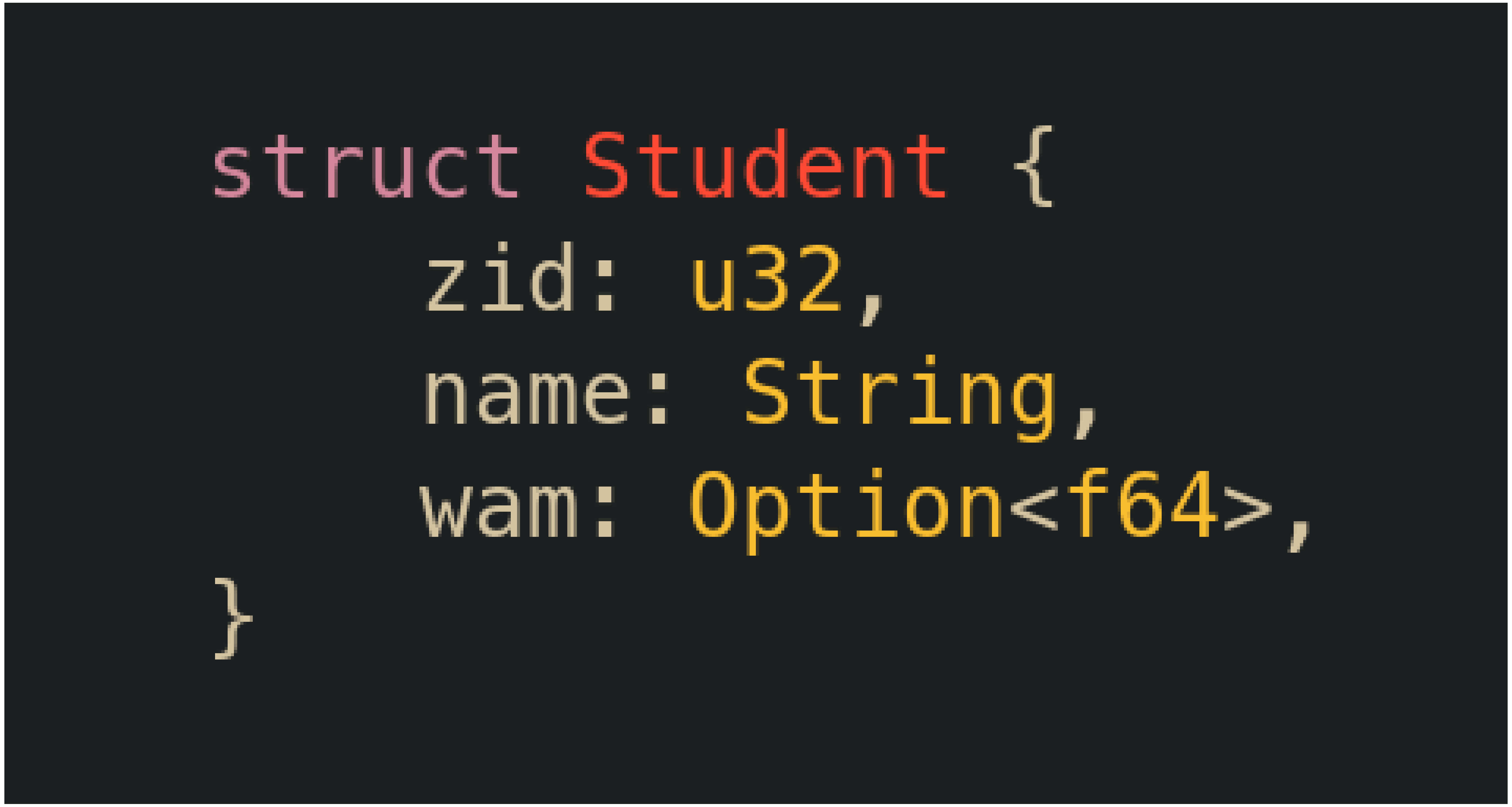
```rust
fn string_to_int(string: &str) -> Option<i32> {
    // ...
}
```

This function may not require an input



```rust
fn int_to_string(int: i32, base: Option<u32>) -> String {
    // ...
}
```

This type may not hold some data

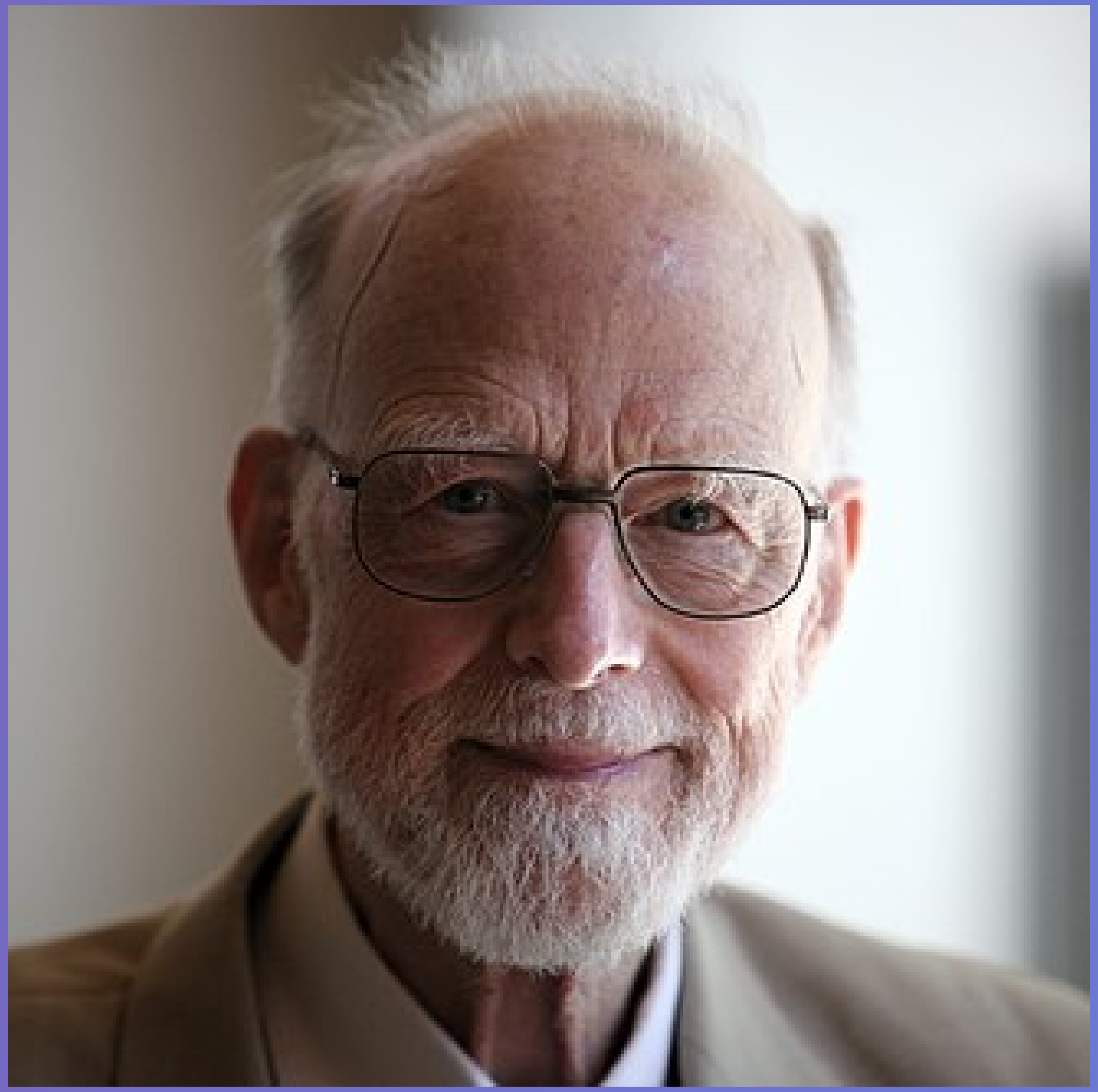


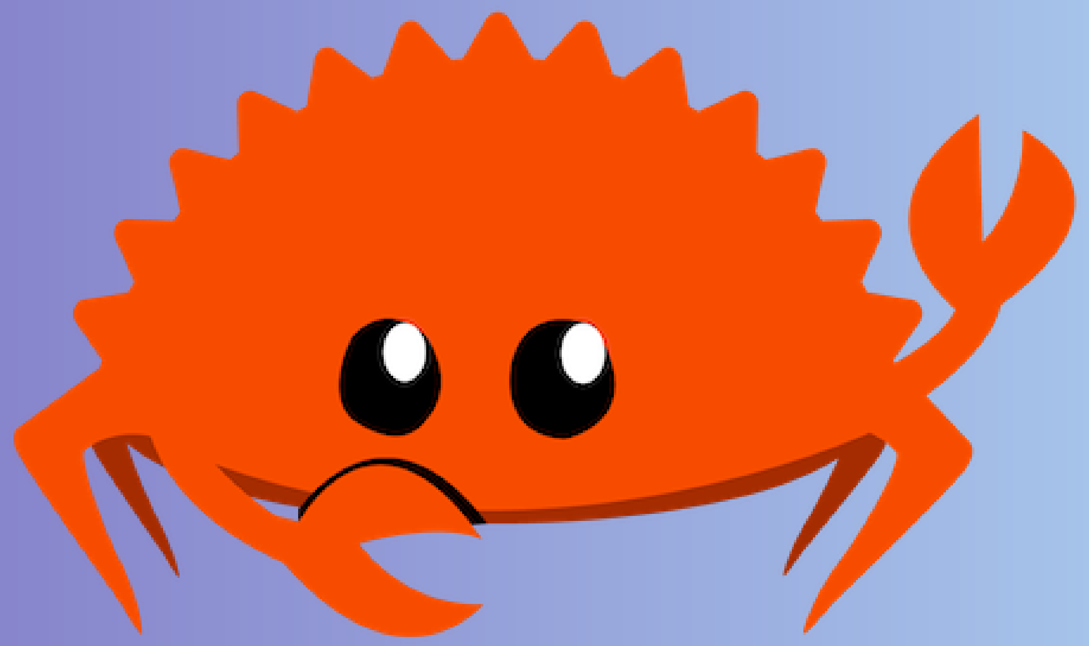
```
struct Student {
    zid: u32,
    name: String,
    wam: Option<f64>,
}
```

Question

HOW DOES LANGUAGE DESIGN MAKE WRITING ROBUST PROGRAMS EASIER?

Tony Hoare's billion dollar mistake





Welcome to COMP6991

See you tomorrow!