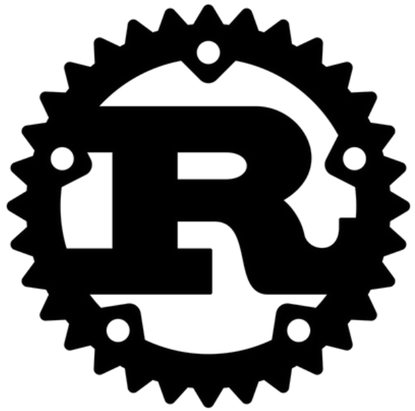
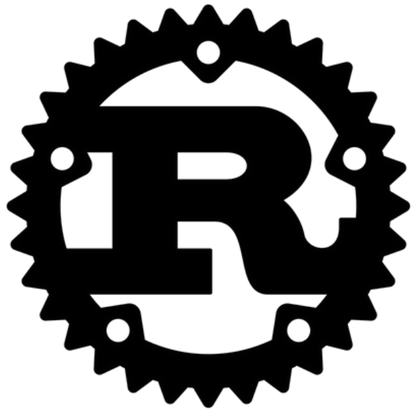
COMP6991 24T1

Rust Basics



Live Example

HELLO, WORLD!



Live Example

LINES

let statements

> BINDINGS

> MUTABILITY

> SHADOWING

> ASIDE: CONSTANTS

Basic types

> INTEGER TYPES (FIXED, ARCH)

> FLOATING POINT TYPES

> BOOLEAN

> CHARACTER

Compound types

> TUPLES, ARRAYS

> STRUCTS, ENUMS

> ASIDE: PERVASIVE MUTABILITY

> ASIDE: UNIT

Expression vs Statement

> VALUE PRODUCTION

> IMPLICATIONS ON NESTING

> ITEM DECLARATIONS

> LET STATEMENTS

> EXPRESSION STATEMENTS

Functions

> PARAMETERS

> RETURN VALUES

> EARLY-RETURN

> EXPRESSION-RETURN

If expressions

> BRANCHING

> WHAT IS A VALID CONDITION?

> AS AN EXPRESSION

> TERNARY IF?

Looping

> LOOP

> WHILE

> FOR

> EARLY TERMINATION

> LOOP BREAK VALUE

Ownership

e.g. String, Vec

> ONE OWNER

> TRANSFER OF OWNERSHIP (MOVE)

> DROP (VS GC? RC?)

> ESCAPE HATCH: CLONE

Copy

e.g. i32, bool, char

> DOES NOT FOLLOW OWNERSHIP

> VALUE SEMANTICS

> SIMPLE SCALAR TYPES

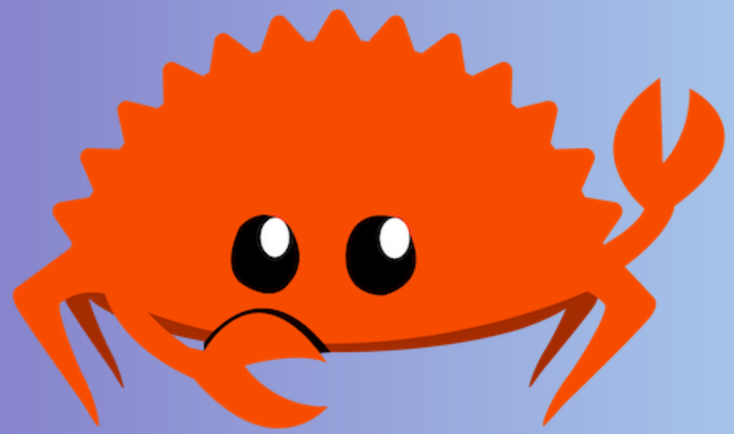
> TUPLES? ARRAYS?

> STRUCTS? ENUMS?

Ownership x fn

> PASSING OWNERSHIP INTO A FN

> OWNERSHIP OUT OF A FN



Welcome to COMP6991

See you tomorrow!

Match

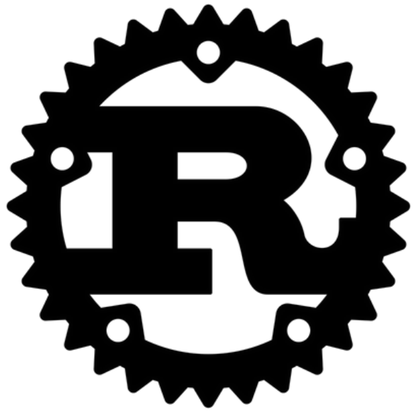
> PATTERN MATCHING

> EXHAUSTIVENESS

> CATCH-ALL

> EXAMPLE: MATCH OPTION<T>

> EXAMPLE: MATCH RESULT<T, E>



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

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

This function may not require an input

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

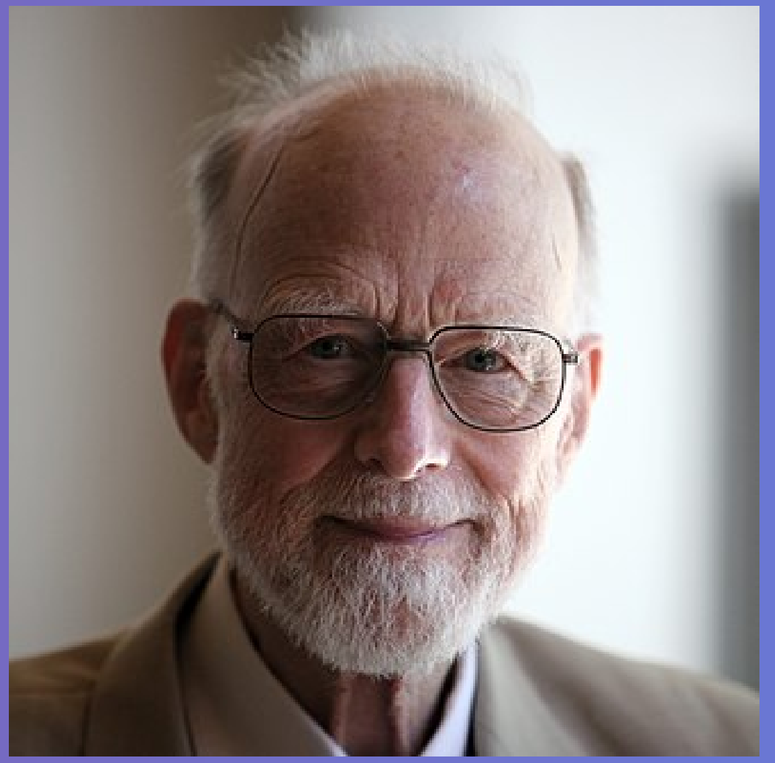
This type may not hold some data

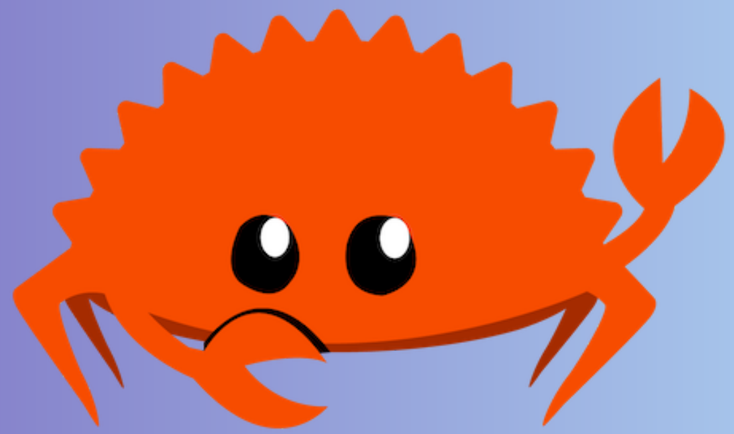
```
struct Student {
        zid: u32,
        name: String,
        wam: Option<f64>,
}
```

Question

HOW DOES LANGUAGE DESIGN MAKE WRITING ROBUST PROGRAMS EASIER?

Tony Hoare's billion dollar mistake





Hope you enjoy workshop 1!

See you next week!