RUST (the good kind)

...

The introduction

Getting started

Follow along

Development Environment

I recommend WSL or Linux but windows is fine too (i guess)

NIX: curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

WIN: https://www.rust-lang.org/tools/install

Most important VSCode package
“rust-analyzer”

Don’t use the ‘rust’ extension, very confusing I know, but trust me.

Nothing is working and i just want to code

There are online compilers and editors that are a good last resort

https://www.tutorialspoint.com/compile_rust_online.php

TA's will be around during the practice times if you are really struggling to get this setup. THIS IS THE HARDEST PART OF THE LESSON!

Cargo, the center of rust

Cargo first and foremost is a package manager.

- Crates.io is the 'official' public rust repository
- Manages dependencies

Cargo also acts as a build system:

- "**cargo build**" will build a "crate" based on some metadata files.
- Wraps "rustc" the rust compiler, so an integrated build system.

Cargo even has integrated testing:

- "**cargo test**" will find and run all of your tests, talk more about how tests work natively in rust later

Use it to build and run "**cargo run**"

"**cargo new**" will rapidly create an empty project for you, they thought of it all.

Adding dependencies

What to do if we want to use a “crate”:

Cargo add rand

Cargo build

Cargo run

```rust
use rand::prelude::*;

▶ Run | Debug
fn main() {
    let mut r: ThreadRng = rand::thread_rng();
    let x: f64 = r.gen(); //random 0-1
    let y: u32 = r.gen_range(0..100);
    println!("X: {x}, and Y: {y}");
}
```

Cargo.toml

```toml
[package]
name = "ex1"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/man

[dependencies]
rand = "0.8.5"
```

Basic Syntax

...

(basically python)

The Basics

let a => declares const variables

let mut a => declares mutable variables

let a: i32 => declares a var of type i32 (32 bit signed integer)

let _a => unused variable hint (think python)

fn a() => defines a function

fn a(first: i32, second: &str) -> Option<i32> => typed function

Note about typing, do it! Treat it like python, its better for readability and removes ambiguity.

Examples

```rust
let a = 2; // const i32
let mut b: i32 = 3; // mutable signed 32 bit integer
let c: u32 = 4; //  const unsigned 32 bit integer
let d = "Hello there"; // const str ref
let e: &str = "Gerneral"; //const str ref
let mut f: f64 = 3.14; // mutable float 64 bit
let j: Option<i32> = None; // const Optional integer
let mut k: Option<i32> = Some(34); // mutable Optional integer
match k { // 34
    Some(v) => println!("{}", v),
    None => println!("No integer here"),
}
```

Functions

```
//takes nothing, returns nothing
fn example() {

}
```

```
//Takes 3 parameters
fn example1(_p1:u32, _p2:u32, _p3:u32) {

}
```

```
//returns an unsigned int
fn example2() -> u32 {
    1 //last line of function is ret (no ;)
}
```

Structs and methods... to be continued later

```
0 implementations
struct animal {
    age: u32,
    weight: u32,
    name: String
}
```

```
impl animal {
    fn who(&self) -> String{
        format!("I am {}, i am {} years old", self.name, self.age)
    }
}
```

```
let cow = animal{age: 3, weight: 133, name: String::from("stacy")};
println!("{}", cow.who())
```

Practice!

Practice_easy: Implement a variation of min(Vector) function

Practice_easy2: Fibonacci sequence generation, as a vector and linked list

Practce_medium: Fizz Buzz (string practice)

Practice_very_hard: Implement http server that serves local files (think python -m http.server)

All practice problems except the hard ones have test suites. To test your solution run "**cargo test**".

During development "cargo run" will execute the main function in main.rs.

Things we didn't talk about that exits

- List comprehensions (see bonus slide)
- Errors, !painc, etc
- Smart pointers, (see std::Box) how to get heap access.
- Threads and concurrency
- Traits (so much more than what we discussed)
- Macros (println! is a macro)