A Crash Course in Rust 

Let's learn Rust together

What is Rust?

- Systems programming language
- Strongly typed
- No garbage collector
- Immutable by default
- Memory safety is checked at compile time
  - Prevents `undefined behavior`
    - Use after free (dereferencing a null pointer)
    - Data races
- Async/Await for high performance apps
  - Core IPC message broker
- Package management
- Workspace configuration

Installation

- Rustup

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
rustup default stable # Install and use the latest stable rust toolchain
```

Tooling

- `vscode`
- `rust-analyzer` extension

Creating a Project

```
cargo new learn-rust
cd learn-rust
cargo run -r
```

Data Structures

```rust
// main.rs
struct Dog;

fn main() {
    let _dog = Dog {};

    println!("Hello, world!");
}
```

Mutability, Functions, and Birthdays

```
struct Dog {
    age: u8,
}

impl Dog {
    pub fn celebrate_birthday(&mut self) {
        self.age = self.age + 1;
        println!("Fluffy is {} years old!", self.age);
    }
}

fn main() {
    let mut dog = Dog { age: 8 };
    dog.celebrate_birthday();
}
```

Constructors

```
struct Dog {
    age: u8,
}

impl Dog {
    pub fn new(age: u8) -> Self {
        Self { age }
    }

    pub fn celebrate_birthday(&mut self) {
        self.age = self.age + 1;
        println!("Wiggly butt is {} wags old!", self.age);
    }
}

fn main() {
    let mut dog = Dog::new(8);
    dog.celebrate_birthday();
}
```

Enumerations

```
enum BoneKind {
    Bacon,
    PeanutButter,
    Turkey,
}
```

Option

```
pub enum Option<T> {
  None,
  Some(T),
}
```

Optional Fields

```
struct Dog {
    age: u8,
    pub bone: Option<Bone>,
}

impl Dog {
    pub fn new(age: u8) -> Self {
        Self { age, bone: None }
    }

    // ...
}

fn main() {
    // ...
}
```

Wait a second...

- What if the dog already has a bone?
- What if the dog doesn't like the flavor?
- What if the dog refuses to take the bone?

Full Program

```
struct Dog {
    age: u8,
    pub bone: Option<Bone>,
}

impl Dog {
    pub fn new(age: u8) -> Self {
        Self { age, bone: None }
    }

    pub fn celebrate_birthday(&mut self) {
        self.age = self.age + 1;
        println!("Wiggly butt is {} wags old!", self.age);
    }
}

struct Bone {
    kind: BoneKind,
}

impl Bone {
    pub fn new(kind: BoneKind) -> Self {
        Self { kind }
    }
}

enum BoneKind {
    BaconFlavored,
    TurkeyAndStuffing,
    PeanutButter,
}

fn main() {
    let mut dog = Dog::new(8);
    dog.celebrate_birthday();
}
```