Rust 101

Introduction

Rust is a memory safe, concurrent system, compiled programming language .

Can be installed form the official website.

cargo is the package manager for rust.

Macros are build in functions like println!

Documentation

https://doc.rust-lang.org/stable/book

https://doc.rust-lang.org/rust-by-example/

comments

// /**/

/// is used for documentation document, which can be parsed using `rustdoc`

Primitive Data Types

Integers

Signed: i8 i16 `i32` i64 isize (Depending on the architecture) possitve and negative

unsigned: u5 u16 u32 u64 usize can't be negative.

Floats

we have f32 (7 decimal digit) `f64` (16 decimal digit)

```
println!("The Max size of float64 is {}", f64::MAX)
```

bool

true or false

char

4 bytes,

Variables

```
//And can be decleraded globally
const ARE_IN_UPPER_CASE:f32 = 5.2;


fn main () {
    //variables are imutable by defaut
    let x = 5;
    //We can make then mutable
    let mut y = 4;
    y += 1;

    println!("{} + {} = {}", x, y, x + y)


}
```

Scope and Shadowing

```
#![allow(unused)]

fn main () {
```

```
    let x = 1;

    {
        let x = 2;
        let y = 2;
        println!("{}", x + y);
        //Here x is equal to 2
        println!("{}", x);
    }

    //we can't see y outside of it's code block!
    // Here x is equal to 1
    println!("{}", x);
}
```

Suffix and _

```
#![allow(unused)]

fn main () {
  let x = 32u8; //To identify its type.
  let y = 1_000_000;
}
```

Tuples

```
#![allow(unused)]

fn main () {
    //Tuples are immuatble

  let tuple = ("Abdulrahman", 19, 3.78);
```

```
  //My name is Abdulrahman and I'm 19 years old, with an overall
  println!("My name is {} and I'm {} years old, with an overall
}
```

Arrays

```
#![allow(unused)]

fn main () {
   //Arrays are of the same data type

   let arr = [1,2,3]; //Auto populated  for us.

   let arr0 : [&str; 3] = ["Abdulrahman", "MOhamed", "mostafa"];


   println!("{} is student number {}", arr0[0], arr[0]);

Abdulrahman is student number 1
}
```

Slices

```
#![allow(unused)]

fn main () {
    //That means we can change the array.
    let mut arr = [1,2,3,4,5];


    //The slice will take elements from the array form index 0 t
    let slice = &mut arr[0..3];
```

```
    slice[0] = 0;
    slice[1] = 0;
    slice[2] = 0;

    println!("{:?}", arr);


}
```

Strings

```
fn main () {
//&str is borrowed string slice, which is Read only;
let greeting = "Hello";

//String is string slice which can be modified.
let mut x = String::new();
x.push_str("string");

let mut y = String::from("Hello");


}
```

Escaping

```
fn main () {
//Escaping using '\'

println!("he said \'Welcome \n to the world\'");
```

```
/*

\\: backslash
\": double quote
\': single quote
\n: newline
\r: carriage return
\t: tab
\0: null character
*/
}
```

Input

Standard Input Stream

```
fn main () {
        use std::io;


        let mut name = String::new();
        io::stdin().read_line(&mut name);

        println!("Welcome {} to the program", name.trim_end());

        //To take arguments from the terminal.

        use std::env;
        //We put all the arguments in a vector.
        let args: Vec<String> = env::args().collect();

        println!("you first argument is {} ....", args[1]);
```

```
    }
```

Terminal

```
use std::env;

fn main() {
  let args: Vec<String> = env::args().collect();
  println!("The first argument is {}", args[1]);
}
```

Dependencies

```
fn main () {
        use rand::Rng;

        let x = rand::thread_rng().gen_range(1..101);

        print!("{}", x);

        //https://crates.io/ to add dependancees.
    }
```

Challenge B

```
fn main () {
       /* Build a simple calculator that takes two user inputs
       then calculates the addition, subtraction, multiplicatio
       of those two inputs.
```

```
    */
    print!("Please Enter the first number\n");  print!("Please E
        use std::io;

        let mut num1 = String::new(); let mut num2 = String::new

        io::stdin().read_line(& mut num1);  io::stdin().read_lin

        let mut x:i32 = num1.trim().parse().expect("Enter a vali

        let sum = x + y;
        let sub = x - y;
        let multi = x * y;
        let div = x / y;









        print!("The sum of the two numbers is {}, the subtractio









    }
```

Control Flow

Conditions

```
fn main () {
    use std::io;
```

```
    println!("Enter your name sir "); let mut name = String::new


    io::stdin().read_line(&mut name);
 println!("Enter your age sir "); let mut age_str = String:: new
    io::stdin().read_line(&mut age_str);

    let mut age:i32 = age_str.trim().parse().expect("Enter a va

    if (age >= 18) && (name.trim_end()A == "Abdulrahman") {
        println!("Welcome to elderhood!");
    }
    else if (age >= 18) {
        println!("Welcome Sneeky panda");
    }
    else {
        println!("Hey kiddo!");
    }
}
```

Matching

```
fn main () {
    use std::cmp::Ordering;



    let age = 19;




    //All the Range should be defined in the arms!
    match age {
        0..=18 => println!("Sneeky Panda!"),
```

```
        19..=45 => println!("Elderhood!"),
        _ => println!("OGs")
    }

    let hook_up_age = 18;

    match hook_age.cmp(&age) {
        Ordering::Equal => println!("You barely made it"),
        Ordering::Greater => println!("yeah!"),
        Ordering::Less => println!("Small")
    }
}
```

Loops

```
fn main () {
    //for loops:

    for  i in 1..=10 {
        print!("{} ", i);
    }
    print!("\n");

    //while loop:
    let mut i = 0;

    while i <= 10 {
        print!("{} ", i);
        i += 1;
    }
    print!("\n");


    //loop, Runs till break statment;
    i = 0;
```

```
    loop {
        print!("{} ", i);
        i +=1;
        if (i > 10) {
            break;
        }
    }



}
```

Functions

```
fn main () {
    let (added, subed) = added_subed(3, 2);


    print!("Added is {}, subed is {}", added, subed);
}

fn added_subed(num1:i32, num2:i32) -> (i32, i32){
    (num1 + num2, num1 - num2) //No sime coloun for return, or w
}
```

Challenge C

```
use std::io::stdin;

fn main () {
    // Create functions for +, -, *, /
    // Use if/else or Match for operator
```

```
    // Might take a little research!
    use std::io;

    let mut num1_str = String::new();   let mut num2_str = Strin
    println!("Enter the first Number");     io::stdin().read_lin
    println!("Enter the second Number");     io::stdin().read_li
    println!("Enter the sign");     io::stdin().read_line(&mut s

    let mut num1:i32 = num1_str.trim().parse().expect("Enter a v
    let mut num2:i32 = num2_str.trim().parse().expect("Enter a v

    if (sign.trim() == "+") {
        println!("{} + {} = {}",num1, num2, add(num1, num2));
    } else if (sign.trim() == "-") {
        println!("{} - {} = {}", num1, num2, sub(num1, num2));
    } else if (sign.trim() == "*") {
        println!("{} * {} = {}", num1, num2, multi(num1, num2))
    }
    else if (sign.trim() == "/") {
        println!("{} / {} = {}", num1, num2, division(num1, num2
    }
}

fn add(num1:i32, num2:i32) -> i32 {
    (num1 + num2)
}

fn sub(num1:i32, num2:i32) -> i32 {
    (num1 - num2)
}
fn multi(num1:i32, num2:i32) -> i32 {
    (num1 * num2)
}
fn division(num1:i32, num2:i32) -> i32 {
```

```
    (num1 / num2)
}
```

Vector

```
fn main () {
    let mut vec = Vec::new();
    vec.push(1);    vec.push(2); vec.push(3);

    let mut vec0 = vec![3,2,1];

    for i in vec.iter() {
        println!("Element: {}", i);
    }

    println!("The lenght of vector is {}",vec0.len());
}
```

Struct

```
struct Person {
    name:String,
    age: i32
}

impl Person {
    fn talk(&self) {
        println!("My name is {}, and my age is {} ",self.name, s
    }
}

fn main () {
```

```
    let mut Abdo = Person{name:String::from("Abdulrahman"), age

    Abdo.talk();

}
```

Generics

```
use std::{ops::Add, process::Output};

fn main () {
    println!("{}", add(5.4,3.5));
}

fn add <T:std::ops::Add<Output = T>> (a: T, b: T) -> T{
    a + b
}
```

Ownership

```
fn main () {
    //Every value in Rust has an owner, when the owner gets out

    let name  = String::from("Abdo");

    let new_name = name;

    //println!("{}", name); -> we can't do that, because we tra


    //Stack is fast, fixed size, Ordered, and it cantain ptr tha
    //heap is Slow, Unorderd, var in size
```

```
}
```

Files

```
use std::{fs, io};
use std::fs::{File,OpenOptions};
use std::io::prelude::*;
use std::io::Read;

fn main () {
  /*   let mut file = File::create("src/test.txt").expect("msg");
    file.write_all(b"welcome"); //Removes everything and types

    file = OpenOptions::new().append(true).open("src/test.txt").

    file.write_all(b"SUIII").expect("msg");
*/ //To WRITE

// To read

/*let mut file = File::open("src/test.txt").expect("msg");

let mut content = String::new();

file.read_to_string(&mut content).unwrap();

println!("{}", content);
  */

  // To Remove a file:
  fs::remove_file("src/test.txt").expect("msg");
}
```

Error Handling

```
Error Handling:

In Rust, error handling is typically done using the Result enum

Helper Methods:

Rust provides a few helpful methods on the Result type for handl

Example:

fn divide(x: i32, y: i32) -> Result<i32, String> {
  if y == 0 {
    return Err(String::from("Cannot divide by zero"));
  }
  Ok(x / y)
}

fn main() {
  let result = divide(10, 2);
  match result {
    Ok(value) => println!("Result: {}", value),
    Err(msg) => println!("Error: {}", msg),
  }
}

In this example, the divide() function returns a Result type tha

The unwrap() method can be used to get the value of the Ok varia

fn main() {
  let result = divide(10, 2).unwrap();
  println!("Result: {}", result);
}
```

```
This code will panic if the Err variant is returned from the div

The expect() method can be used to provide a custom error messag

fn main() {
  let result = divide(10, 0).expect("Division by zero");
}

This code will panic with the message "Division by zero" if the

The ? Operator:

The ? operator in Rust can be used as a shorthand for propagatin

Code before using ? Operator:

use std::fs::File;
use std::io::Read;

fn read_file(path: &str) -> Result<String, std::io::Error> {
  let mut file = match File::open(path) {
    Ok(file) => file,
    Err(e) => return Err(e),
  };
  let mut contents = String::new();
  match file.read_to_string(&mut contents) {
    Ok(_) => Ok(contents),
    Err(e) => Err(e),
  }
}

fn main() {
  let result = read_file("test.txt");
  match result {
    Ok(contents) => println!("File contents: {}", contents),
```

```
    Err(err) => println!("Error reading file: {}", err),
  }
}

In this version of the code, the read_file() function uses matcl

The main() function remains the same, using a match expression t

Example using ? Operator:

fn read_file(path: &str) -> Result<String, std::io::Error> {
  let mut file = File::open(path)?;
  let mut contents = String::new();
  file.read_to_string(&mut contents)?;
  Ok(contents)
}

fn main() {
  let result = read_file("test.txt");
  match result {
    Ok(contents) => println!("File contents: {}", contents),
    Err(err) => println!("Error reading file: {}", err),
  }
}

While the ? operator can be a more concise and idiomatic way to

Panic Macros:

fn divide(x: i32, y: i32) -> i32 {
  if y == 0 {
    panic!("Cannot divide by zero");
  }
  x / y
}
```

```
fn main() {
  let result = divide(10, 0);
  println!("Result: {}", result);
}

In this example, the divide() function panics with the message "

The panic!() macro can be useful for handling unexpected or unre
```

SHA 256 Cracker

```
#![allow(unused)]
use std::{env,fs,io::{self, BufRead, BufReader},process::{self,
use sha2:: {Sha256, Digest};

fn main () {


  let args: Vec<String> = env::args().collect();
  if args.len() != 2 {
    println!("cargo run <sha256 hash>"); //run is args[0]
    std::process::exit(1);
  }

  let wanted_hash = &args[1];

  let password_file = "src/rockyou.txt";
  let mut attemps = 1;

  println!("Starting to crack {}", wanted_hash);

  let pass_list = std::fs::File::open(password_file).unwrap();

  let reader = BufReader::new(pass_list);
```

```
  for line in reader.lines() {
    let line = line.unwrap();
    let password = line.trim().to_owned().into_bytes();
    let pass_hash = format!("{:x}", Sha256::digest(&password));

    println!("Attempt {} : {}", attemps, std::str::from_utf8(&pa

    if &pass_hash == wanted_hash {
      println!("FOund: {}", std::str::from_utf8(&password).unwra
      exit(0);
    }
    attemps += 1;
  }

  println!("Couldn't Crack!");
}
```