Rust Programming - Introduction

Week 2

SCHOOL OF
SCIENCE AND
TECHNOLOGY
PAN-ATLANTIC UNIVERSITY

Rust Programming Language

Rust is a systems level programming language, developed by **Graydon Hoare.**

it is a high-performance, statically-typed multi-paradigm programming language. With the main focus on safety and performance.

This language helps developers create robust and secure applications.

Hundreds of companies around the world choose Rust since it has numerous benefits. It's fast and memory-efficient.

Using no runtime or garbage collector, the language can conduct different performance services, integrate with other languages, and run on embedded devices.

Application v/s Systems Programming Languages

Application programming languages like Java/C# are used to build software, which provide services to the user directly. They help us build business applications like spreadsheets, word processors, web applications or mobile applications.

Systems programming languages like C/C++ are used to build software and software platforms. They can be used to build operating systems, game engines, compilers, etc. These programming languages require a great degree of hardware interaction.

Why Rust?

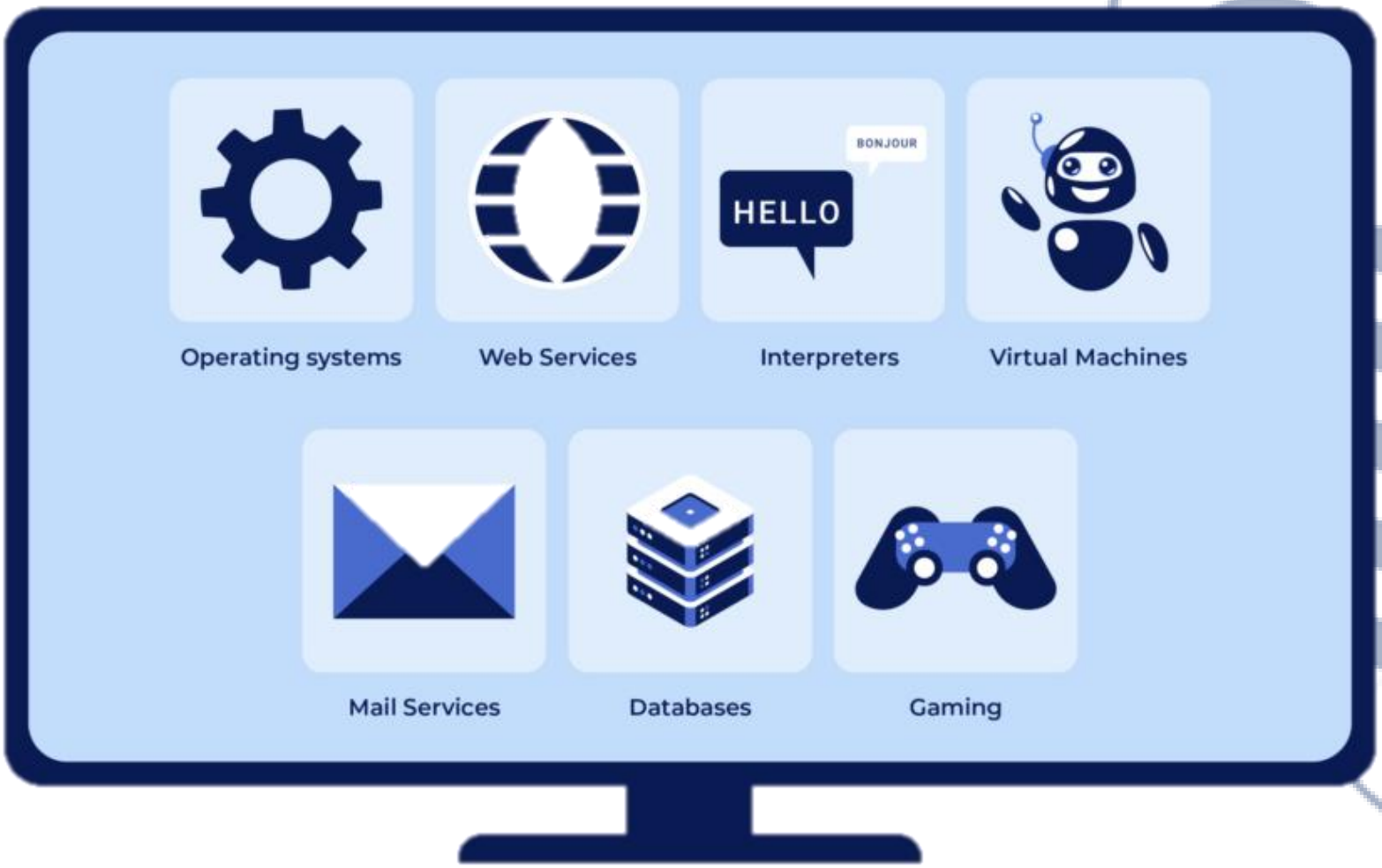
Rust focuses on three goals −

- Safety
- Speed
- Concurrency

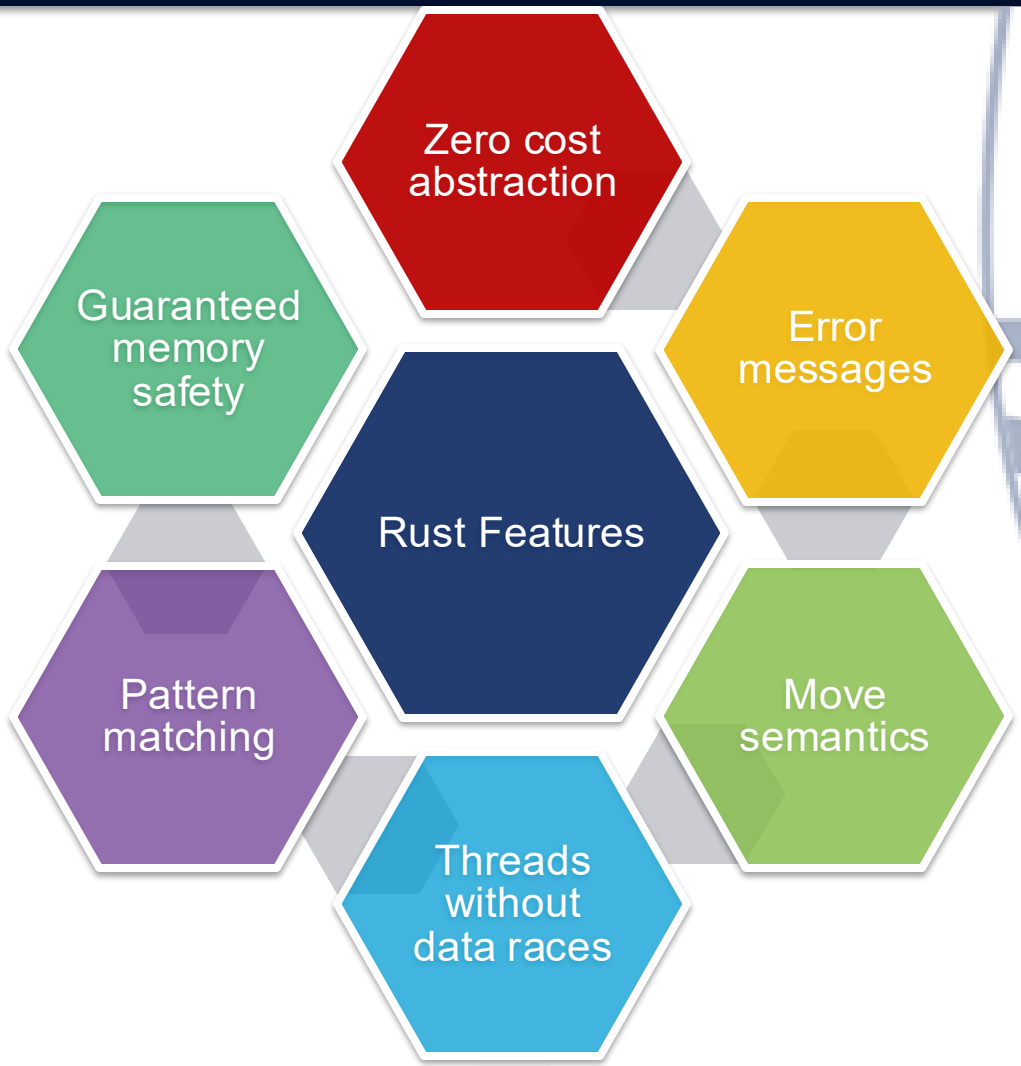
The language was designed for developing highly reliable and fast software in a simple way.

Rust can be used to write high-level programs down to hardware-specific programs.

What is Rust used for?



Rust Core Features



Rust Core Features

Zero cost abstraction

In Rust, we can add abstractions without affecting the runtime performance of the code. It improves the code quality and readability of the code without any runtime performance cost.

Error messages

In C++ programming, there is an excellent improvement in error messages as compared to GCC. Rust goes one step further in case of clarity. Error messages are displayed with (formatting, colors) and also suggest misspellings in our program.

Type inference

Rust provides the feature of a Type inference which means that it determines the type of an expression automatically.

Rust Core Features

Move semantics

Rust provides this feature that allows a copy operation to be replaced by the move operation when a source object is a temporary object.

Threads without data races

A data race is a condition when two or more threads are accessing the same memory location. Rust provides the feature of threads without data races because of the ownership system. Ownership system transmits only the owners of different objects to different threads, and two threads can never own the same variable with write access.

Pattern matching

Rust provides the feature of pattern matching. In pattern matching, patterns in Rust are used in conjunction with the 'match' expressions to give more control over the program's control flow. Following are the combinations of some patterns:

- Literals
- Arrays, enums, structs, or tuples
- Variables
- Wildcards
- Placeholders

Rust Core Features

Guaranteed memory safety

Rust guaranteed the memory safety by using the concept of ownership. Ownership is a middle ground between the memory control of C and the garbage *collection of java*. In Rust programs, memory space is owned by the variables and temporarily borrowed by the other variables. This allows Rust to provide the memory safety at the compile time without relying on the garbage collector.

Efficient C bindings

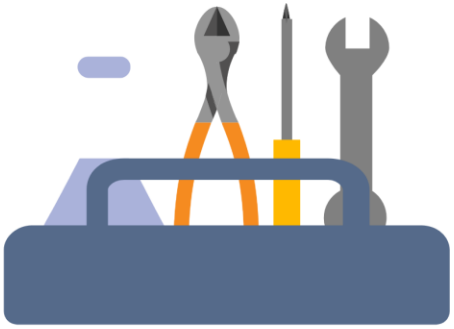
Rust provides the feature of *'Efficient C bindings'* means that the Rust language can be able to interoperate with the C language as it talks to itself. Rust provides a 'foreign function interface' to communicate with C API's and leverage its **ownership** system to guarantee the memory safety at the same time.

Safe memory space allocation

In Rust, memory management is manual, i.e., the programmer has explicit control over where and when memory is allocated and deallocated. In C language, we allocate the memory using malloc function and then initialize it but Rust refuses these two operations by a single **'~'** operator. This operator returns the smart pointer to int. A smart pointer is a special kind of value that controls when the object is freed. Smart pointers are *"smart"* because they not only track where the object is but also know how to clean it up.

Installing Rust

1. Installation of Visual Studio Build tools is mandatory to run the rust program. Goto: https://go.microsoft.com/fwlink/?LinkId=691126



2. Installation the Rust compiler.

Goto: https://www.rust-lang.org/install.html

Rust Installation on Windows

On Windows, open the link https://www.rust-lang.org/install.html and follow the instructions for installing Rust. After following all the instructions, Rust will get installed.

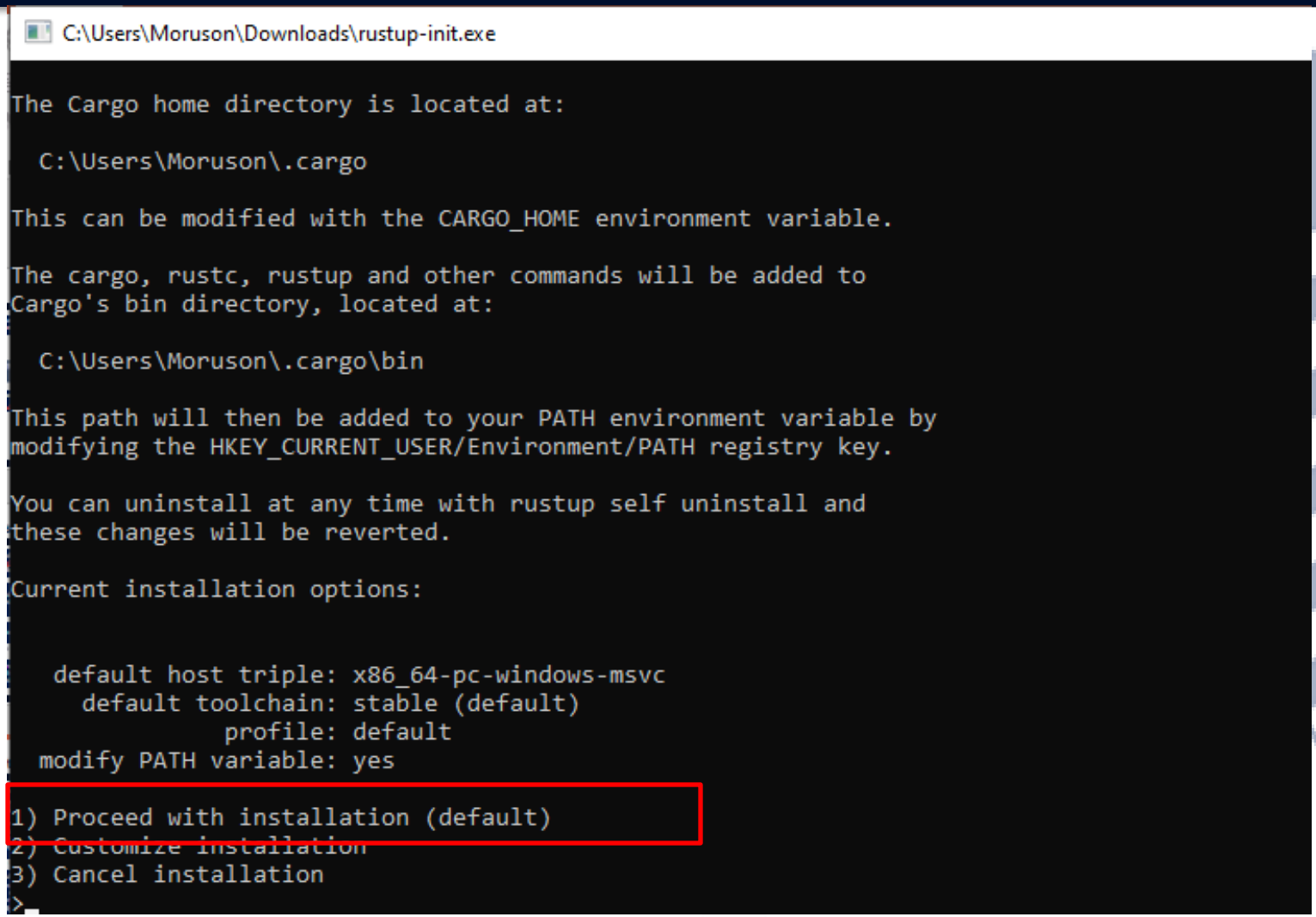
After installation, PATH variable of Rust automatically adds in your system PATH.

Open command prompt then runs the following command:

- rustc --version

After running this command, you should see the version number, commit hash, and commit date. If you do, it means Rust has been installed successfully. Congratulations!!!

Rust Installation on Windows



Rust Installation on Linux/Mac

open a terminal then use the following command:

- $ curl https://sh.rustup.rs -sSf | sh

The above command downloads a script and starts the installation of rustup tool. This installs the latest version of Rust. If the installation is done successfully, then the following message will appear:

- Rust is installed now. Great!

This installation adds automatically Rust to your system path after your next login. If you want to run Rust just right away without restarting the terminal, then run the following command to your shell to add the path to your system PATH manually:

- $ source $HOME/.cargo/env

Run a Rust program

Exercise 1: week-2/practice_1.rs

Goto Tools Project Preferences Help

practice-1.rs

```
1  fn main(){
2      println!("Welcome to COS 101!");
3
4      println!("The course learning outcome is as follows:");
5
6      println!(
7
8
9
10
11
12
13 }
```

fn main(): The main() function is always the first code in every Rust executable code. The main() function is enclosed in curly braces{}. The main() function does not contain any parameter as well as it does not return any value.

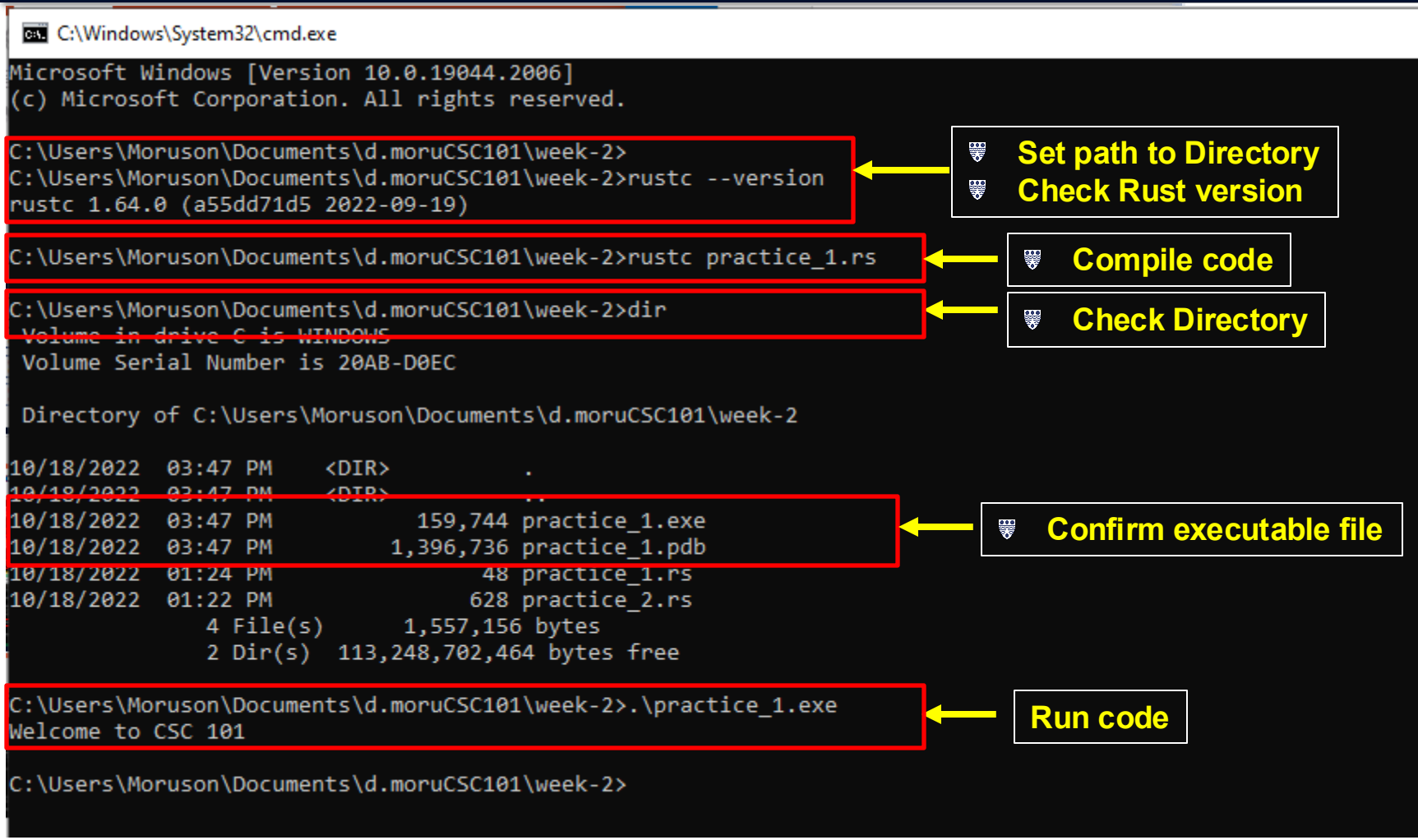
println!: It is a Rust macro. If it calls the function, then it does not contain '!'.

"Welcome to CSC 101!": It is a string passed as a parameter to the println!, and the string is printed to the console.

Procedure to create, compile and run the program

1. Open the command prompt from cloned repository
2. Set the path of the directory. E.g. ***C:\Users\Moruson\Documents\d.moruCSC101\week-2***
3. Compile the above program using the ***rustc*** command + ***filename.rs***.
4. Finally, run the program by using the command ***filename.exe***

Procedure to create, compile and run the program



Runtime Error

Practice 1:

practice-1.rs

```
fn mainn(){
    println!("Welcome to COS 101!");

    println!("The course learning outcome is as follows:");

    println!("1. Distinguish between computer and computer programming,/n
              2. Develop some techniques in computer science,/n
              3. Understand the different areas of study in computer science,/n
              4. Conversant with applications of computer science,/n
              5. Navigate the career prospects in computer science./n
              6. Conversant with computer programming concepts." );
```

Misspelt *main*

Output:

em32\cmd.exe

```
C:\Users\Moruson\Documents\d.moruCSC101\week-2>rustc practice_1.rs
error[E0601]: `main` function not found in crate `practice_1`
 --> practice_1.rs:3:2
  |
3 | }
  |  ^ consider adding a `main` function to `practice_1.rs`

error: aborting due to previous error

For more information about this error, try `rustc --explain E0601`.

C:\Users\Moruson\Documents\d.moruCSC101\week-2>_
```

Runtime error message

What a macros?

- Rust provides a powerful macro system that allows meta-programming.
- Macros look like functions, except that their name ends with a bang(!).
- Instead of generating a function call, macros are expanded into source code that gets compiled with the rest of the program.
- Macros provide more runtime features to a program unlike functions.
- Macros are an extended version of functions.

Using the println! Macro - Syntax

Practice 2: week-2/practice_2.rs ← **mkdir week-2**

Goto Tools Project Preferences Help

practice_2.rs

```
1  fn main() {
2
3      println!(); // prints just a
4
5      println!("hello ");//prints
6
7      println!("format {} arguments", "some"); //prints format some arguments
8
9      println!("My name is {}. I am the class rep of stream 1", "Dagogo William-Jumbo");
10
11 }
12
```

Output:

n32\cmd.exe

```
C:\Users\Moruson\Documents\d.moruCSC101\week-2>rustc practice_2.rs

C:\Users\Moruson\Documents\d.moruCSC101\week-2>.\practice_2.exe

hello
format some arguments
My name is Dagogo William-Jumbo. I am the class rep of stream 1

C:\Users\Moruson\Documents\d.moruCSC101\week-2>
```

Comments in Rust

Comments are a way to improve the readability of a program. Comments can be used to include additional information about a program like author of the code, hints about a function/ construct, etc. The compiler ignores comments.

Rust supports the following types of comments :

- Single-line comments (//) − Any text between a // and the end of a line is treated as a comment
- Multi-line comments (/* */) − These comments may span multiple lines.

Example:

```
1
2   //this is single line comment
3
4
5   /* This is a
6      Multi-line comment
7   */
8
9
```

Rust Variables

A variable is a named storage that programs can manipulate. It helps programs to store values.

Variables in Rust are associated with a specific data type. The data type determines the size and layout of the variable's memory, the range of values that can be stored within that memory and the set of operations that can be performed on the variable.

Rules for Naming a Variable:

- In this section, we will learn about the different rules for naming a variable.
- The name of a variable can be composed of letters, digits, and the underscore character.
- It must begin with either a letter or an underscore.
- Upper and lowercase letters are distinct because Rust is case-sensitive.

Rust Variables

Syntax: The data type is inferred from the value assigned to the variable.

The syntax for declaring a variable is given below.

```
• let variable_name = value;              // no type specified
• let variable_name:dataType = value;     //type specified
```

Practice 3:

Help

practice_3.rs

```
1  fn main() {
2      let fees = 25_000;
3      let salary:f64 = 35_000.00;
4      println!("fees is {} and salary is {}",fees,salary);
5  }
6
```

Output:

em32\cmd.exe

```
C:\Users\Moruson\Documents\d.moruCSC101\week-2>rustc practice_3.rs

C:\Users\Moruson\Documents\d.moruCSC101\week-2>.\practice_3.exe
fees is 25000 and salary is 35000

C:\Users\Moruson\Documents\d.moruCSC101\week-2>
```

Rust Variables

Practice 4:

Goto Tools Project Preferences Help

practice_4.rs

```
1  fn main() {
2      let p:f64 = 1000.0;
3      let r:f64 = 1.0;
4      let t:f64 = 2.0;
5
6      // simple interest
7      let a = p * ( 1.0 + (r / 100.0)) * t;
8      println!("Amount is {}", a);
9      let si = a - p;
10     println!("Simple Interest is {}", si);
11
12 }
13
```

Output:

m32\cmd.exe

```
C:\Users\Moruson\Documents\d.moruCSC101\week-2>rustc practice_4.rs

C:\Users\Moruson\Documents\d.moruCSC101\week-2>.\practice_4.exe
Amount is 2020
Simple Interest is 1020

C:\Users\Moruson\Documents\d.moruCSC101\week-2>
```

Class Projects

SCHOOL OF
SCIENCE AND
TECHNOLOGY
PAN-ATLANTIC UNIVERSITY

Project I

The Ibeju Local Government Chairman has received a mortgage loan of N520,000,000 from Sterling Bank for the construction of the Lekki Free Trade Zone industrial estate. Find the compound interest for 5 years at 10% per annum compounded annually.

Hint: Use the formula of CI

$A = P[1 + (R/100)]n$

$CI = A - P$

Project II

Chief Donatus and Sons Ltd is downsizing and readjusting their products sales due to an ongoing recession. You have been consulted to write a rust program to calculate the sum and average of the following sales record.

S/N	Item	Qty	Amount
1	Toshiba	2	450,000.00
2	Mac	1	1,500,000.00
3	HP	3	750,000.00
4	Dell	3	2,850,000.00
5	Acer	1	250,000.00

Project III

Ms. Anjola Olowokere has recently acquired a brand new TV set. The TV was bought for N510,000. The value of the TV was depreciated by 5% per annum. Write a rust program to find the value of the TV after 3 years. (Depreciation means the reduction of value due to use and age of the item).

Hint: Use the formula of CI for depreciation.

$A = P[1 - (R/100)]^n$