**Rust Programming Lab #1**

**4th August 2022**

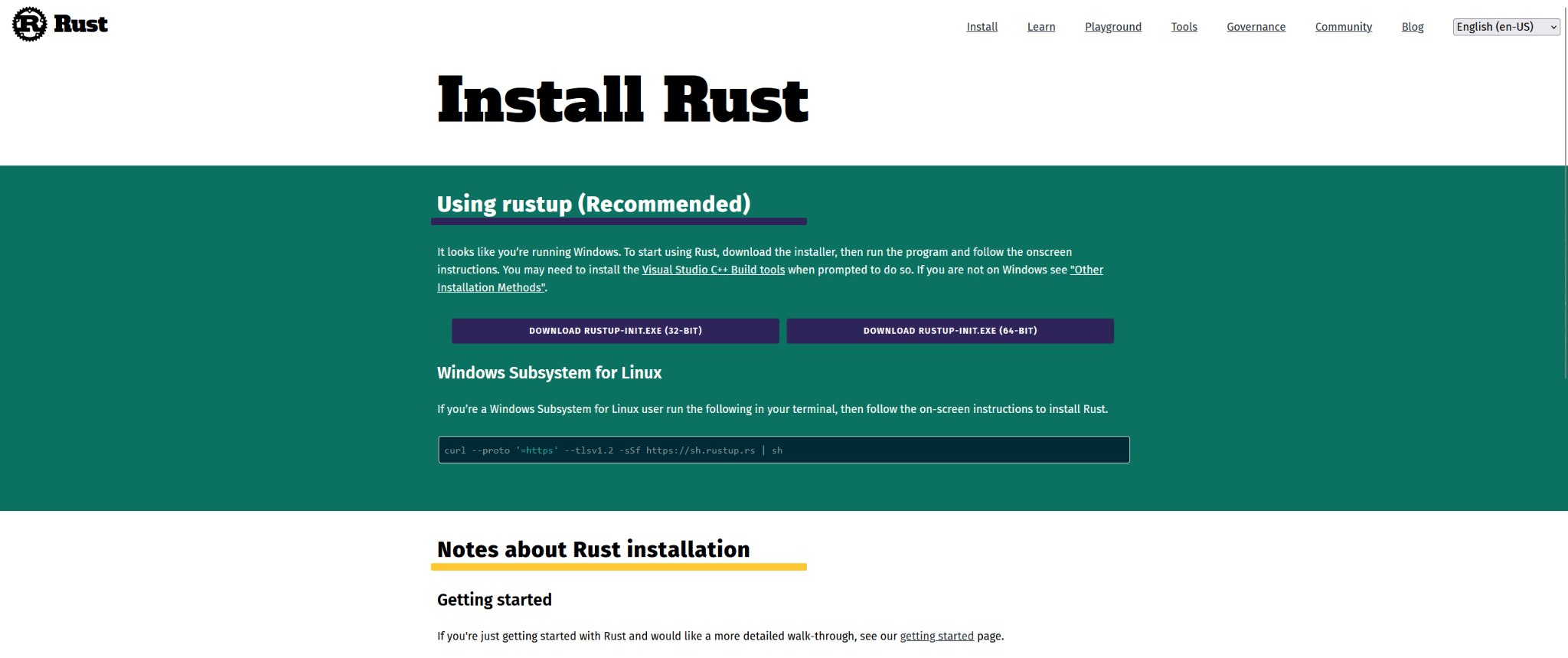
**Rust and Cargo Development Environment Installation**

# Install the development environment

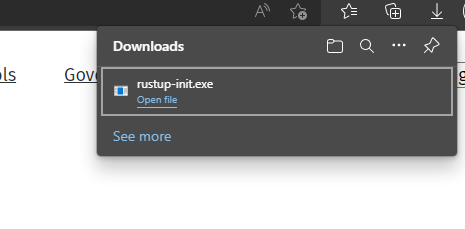
Instructions for downloading and installing Rust may be found on several websites. You can follow them or the condensed version here!

1. Open and download rust development environment from the website.

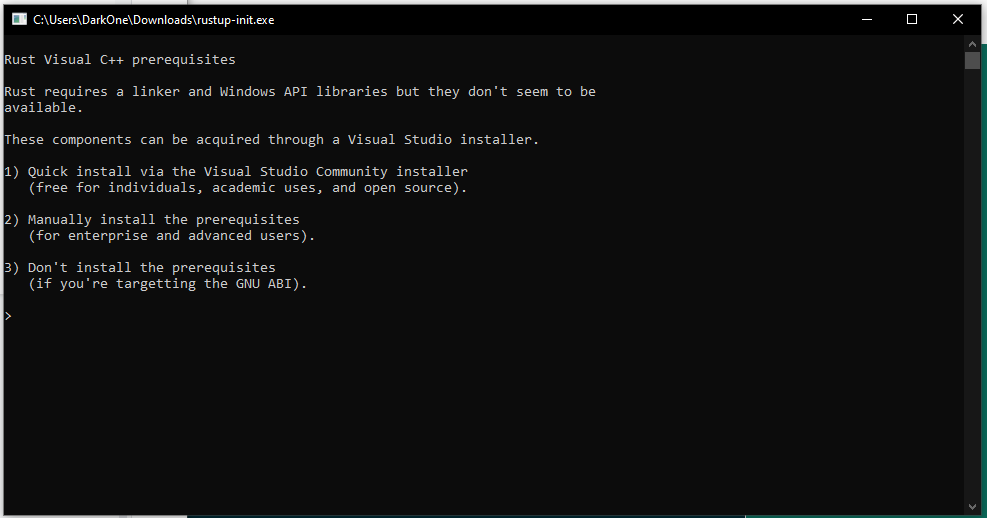
Website : <https://www.rust-lang.org/tools/install>



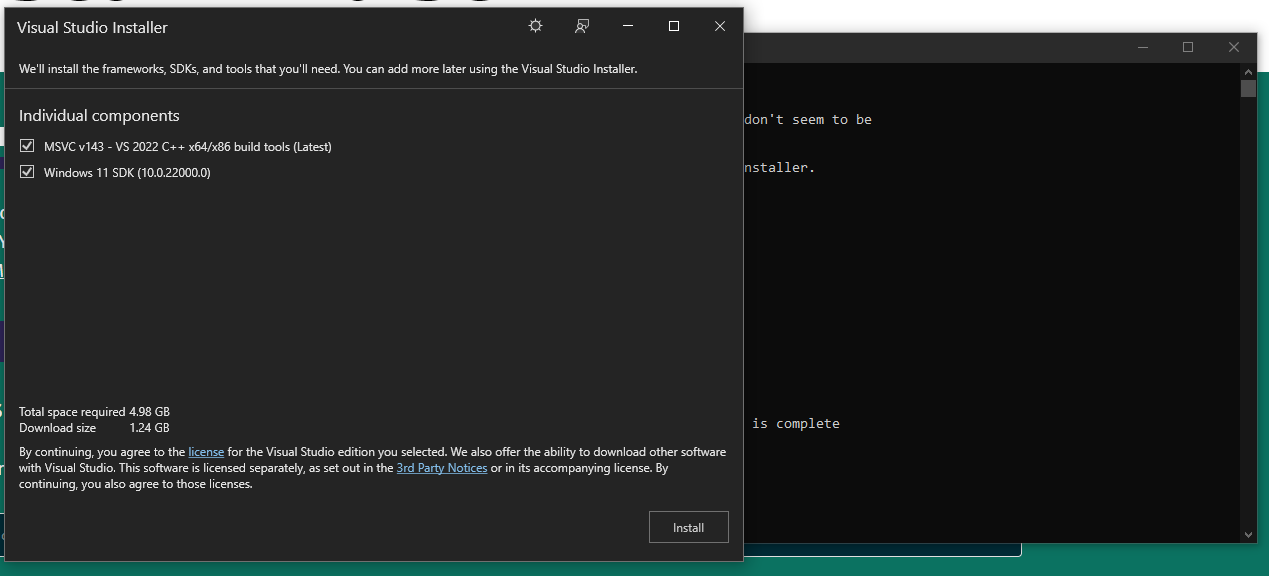
1. Download the appropriate installer for your operating system.



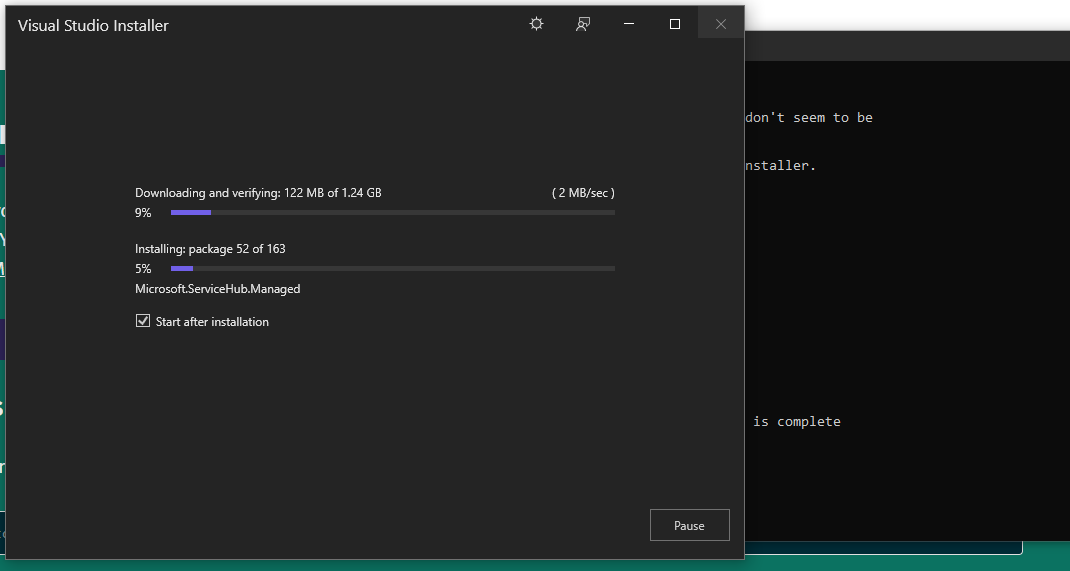
1. Open and run the file



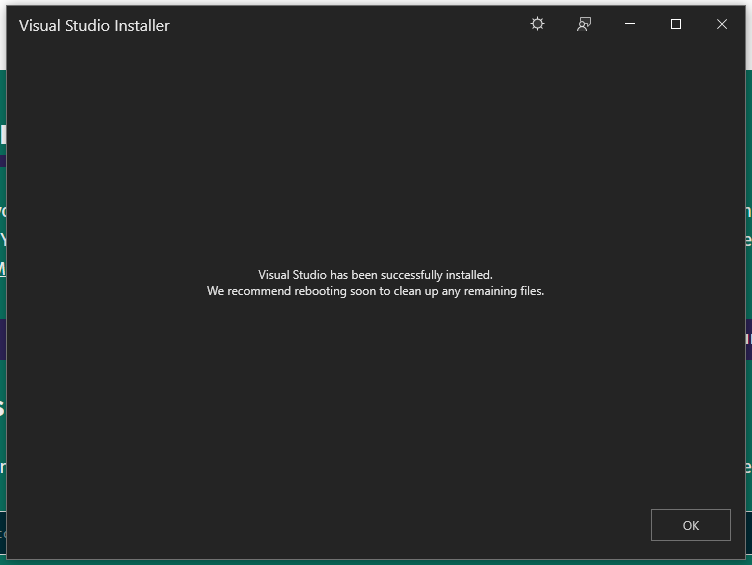
The development environment requires Visual Studio. If it is not already installed on your computer, this message will appear: choose 1 to install Visual Studio.



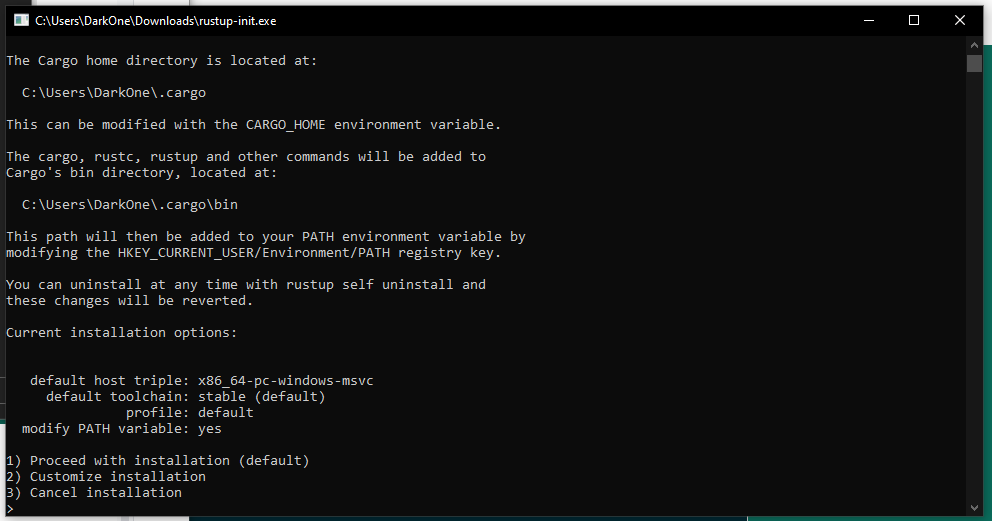
Proceed with the installation of Visual Studio.



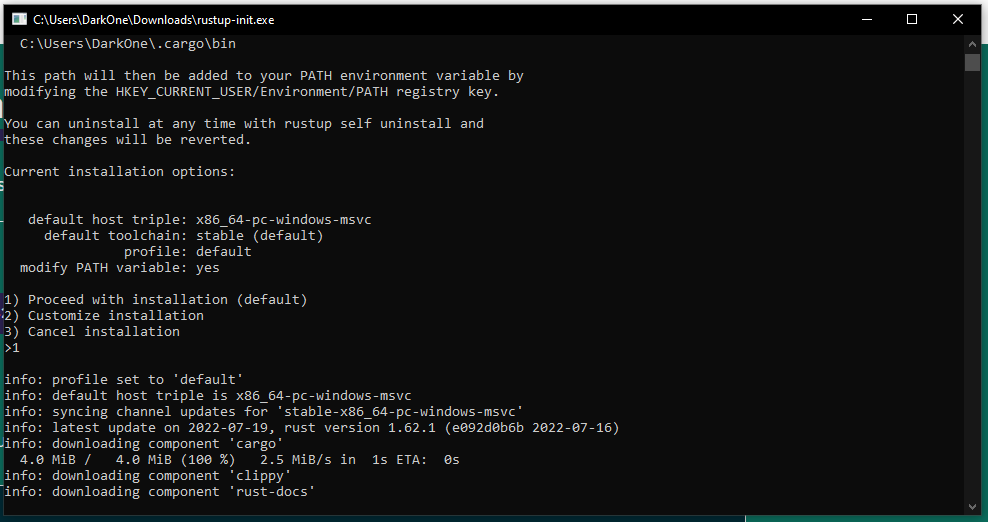
After finishing, click “OK” to continue



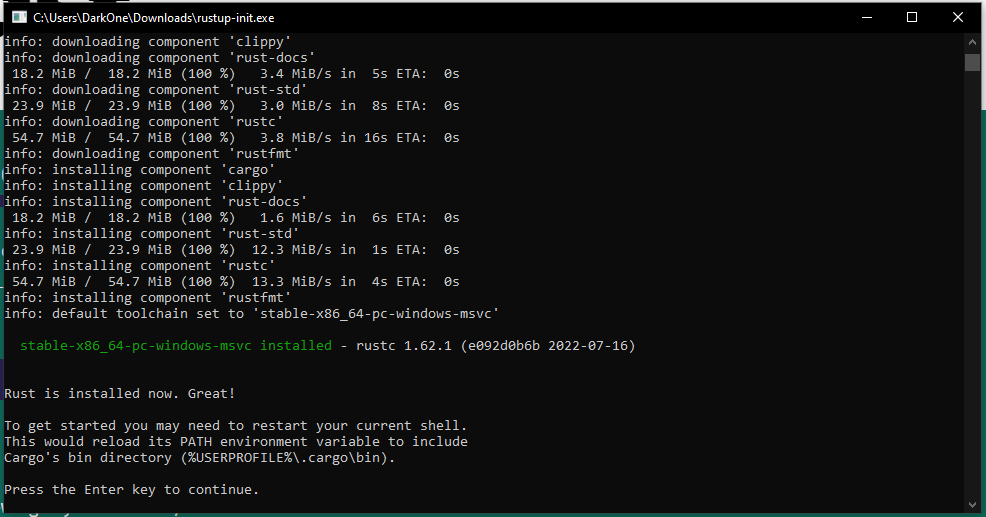
With Visual Studio installed, the installer will display choices for installing the environment. Choose 1.



The installer will download and install the necessary modules for rust development



When finished, press the enter key to close the installer.



# Rust Development using command line

## Start “hello\_world” program with rust

Open the (old fashioned 😊) command line program.

Text

Description automatically generated

Make a directory for your Rust progams:

mkdir <rusty\_work>

Change to your working directory of “hello\_world.rs” using

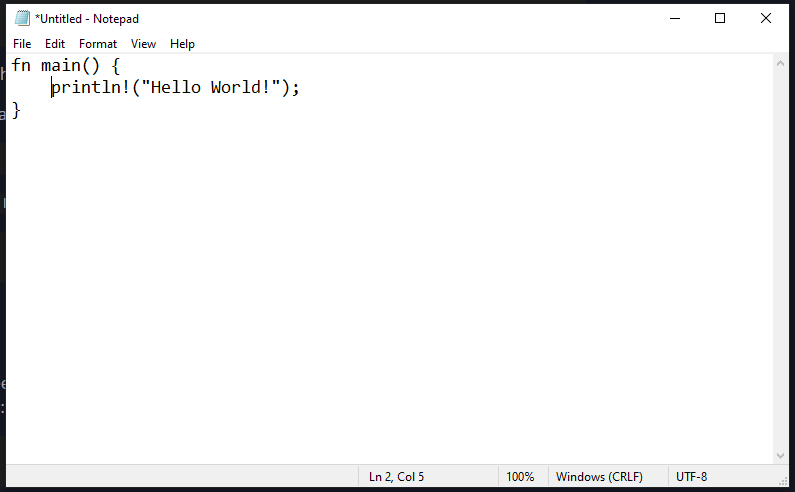
cd <rusty\_work>

Open a basic text editor, for example Notepad – but you might find it more convenient to use the same editor for Python and Rust programs, and input the code below.

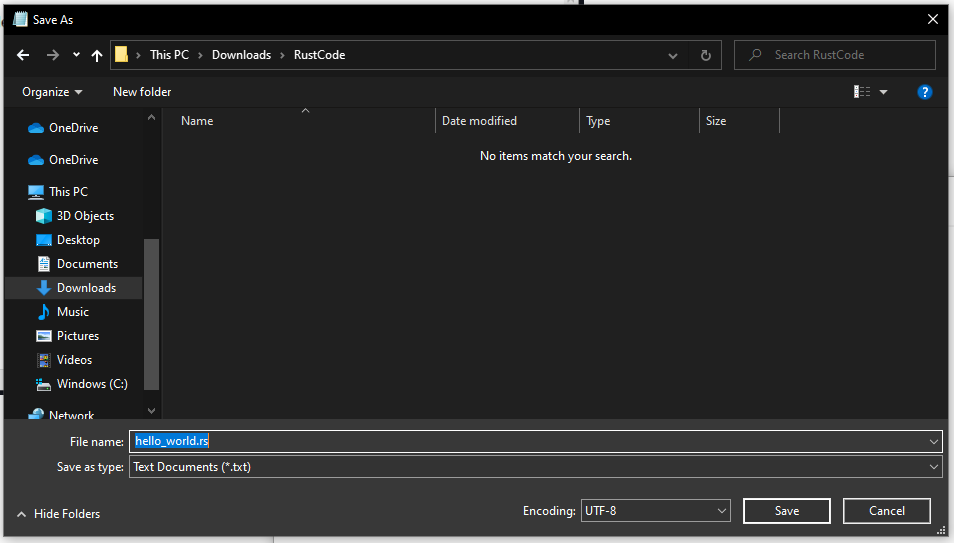
fn main() {

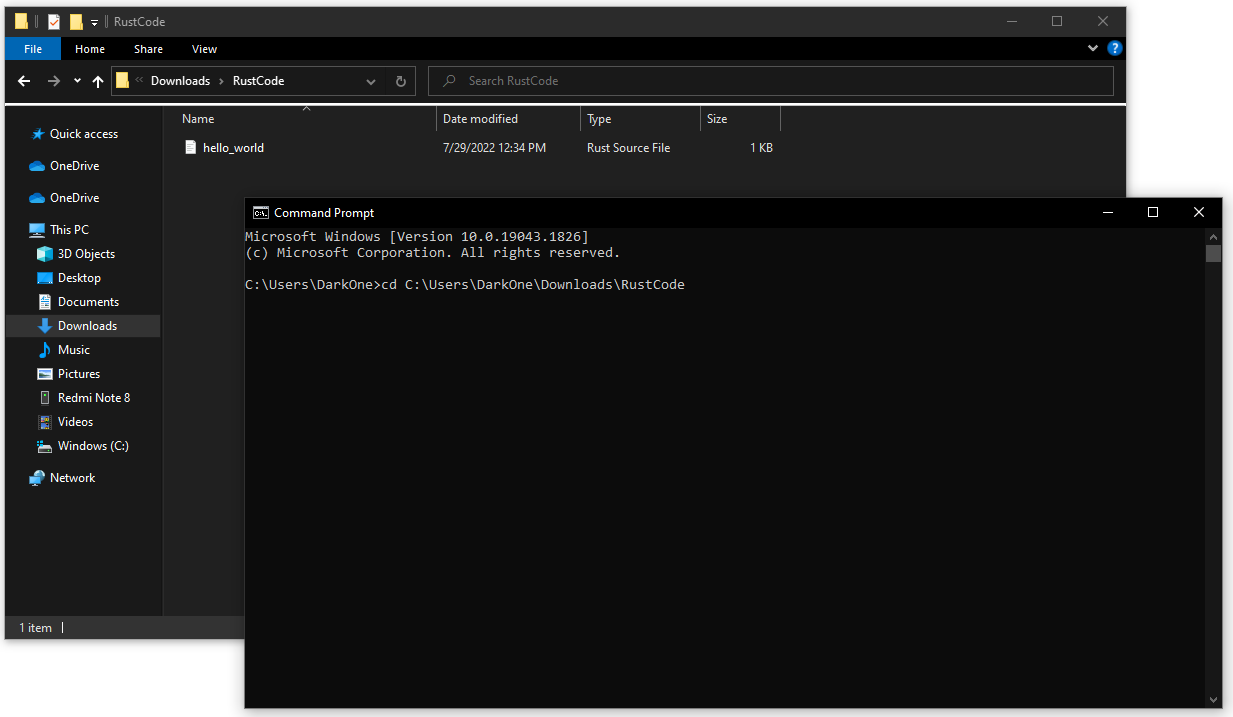
println!("Hello World!");

}



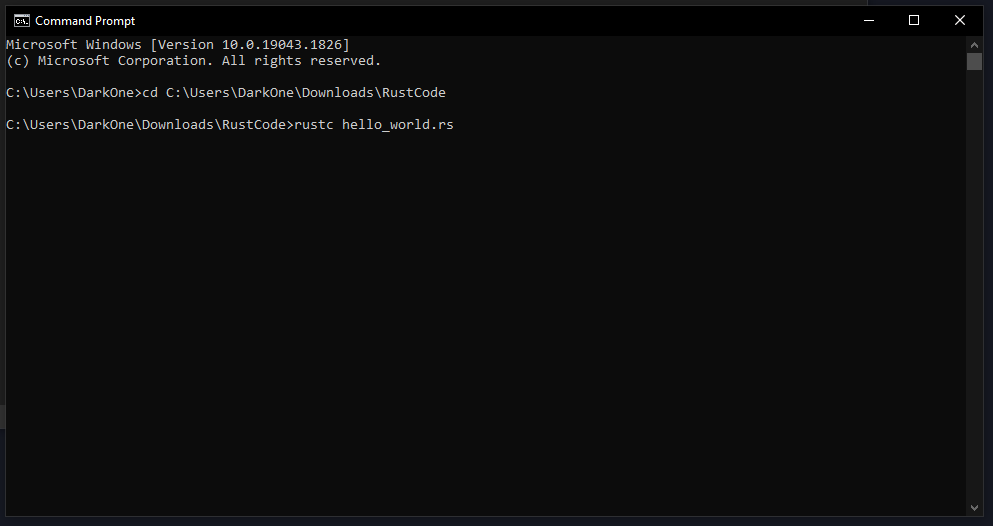
Save the file as “hello\_world.rs”.





Compile “hello\_world.rs” file using

rustc hello\_world.rs

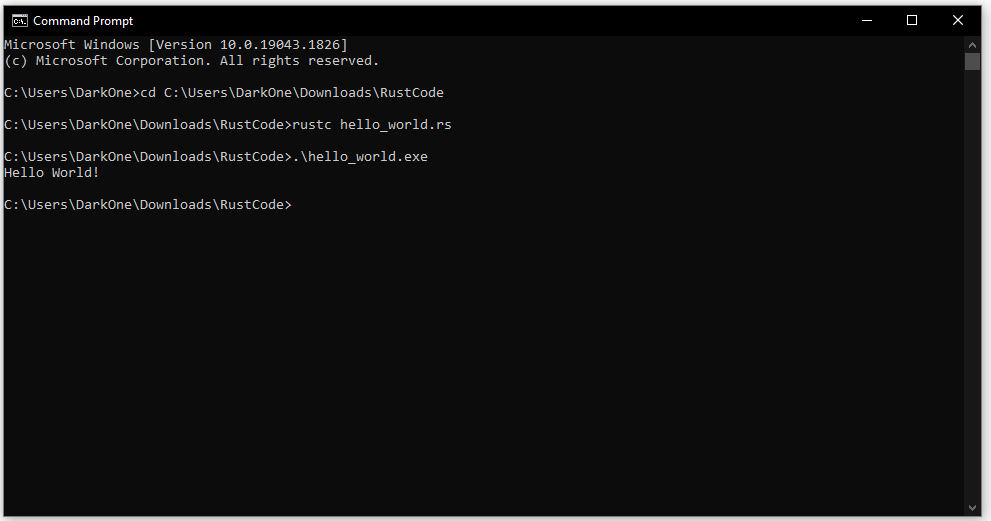


If there are no errors, the compiler will produce an executable file for the “hello\_world” program



Execute that file to run the “hello\_world” program:

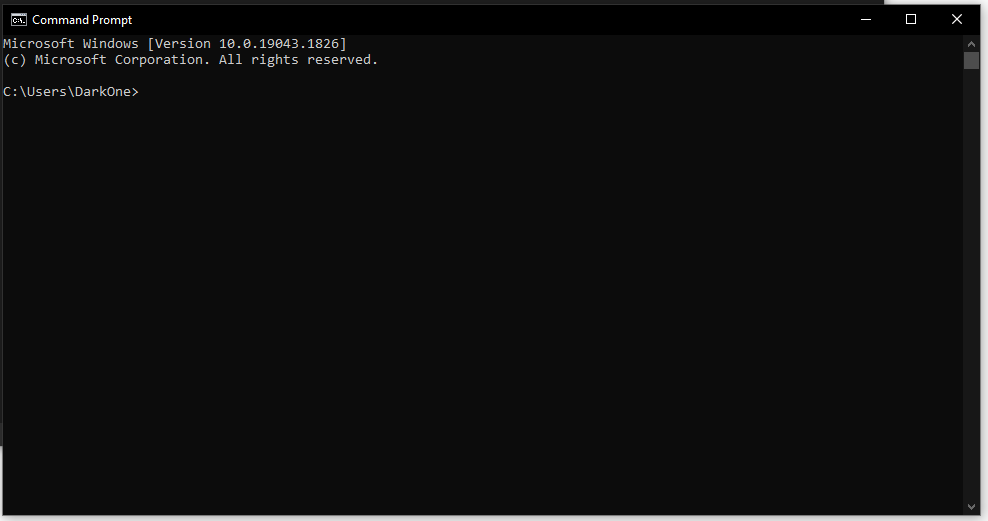
.\hello\_world.exe



Every time you modify the code, you need to repeat the whole process again (save, compile, execute) to reflect the change in coding.

## Start “hello\_world” project with cargo

Open the command line program.

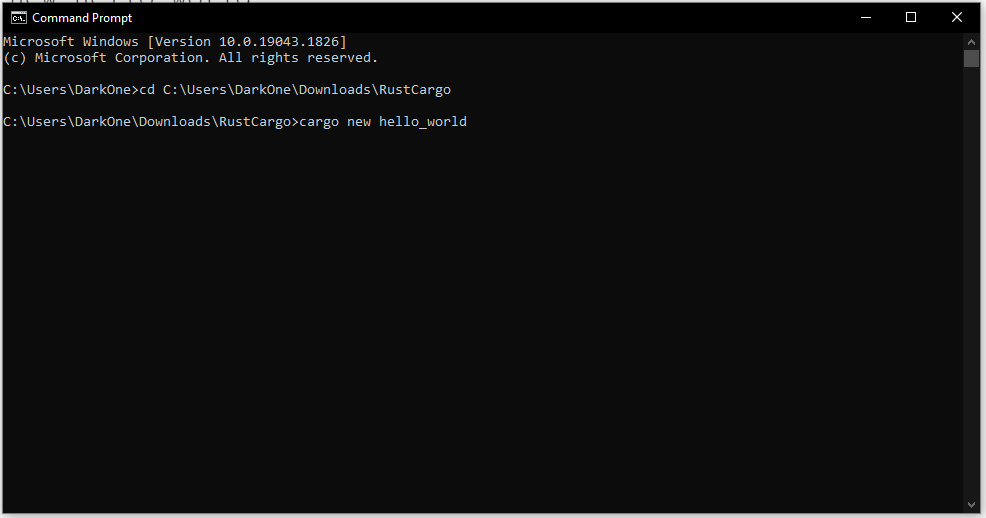


Change to your working directory for the project using

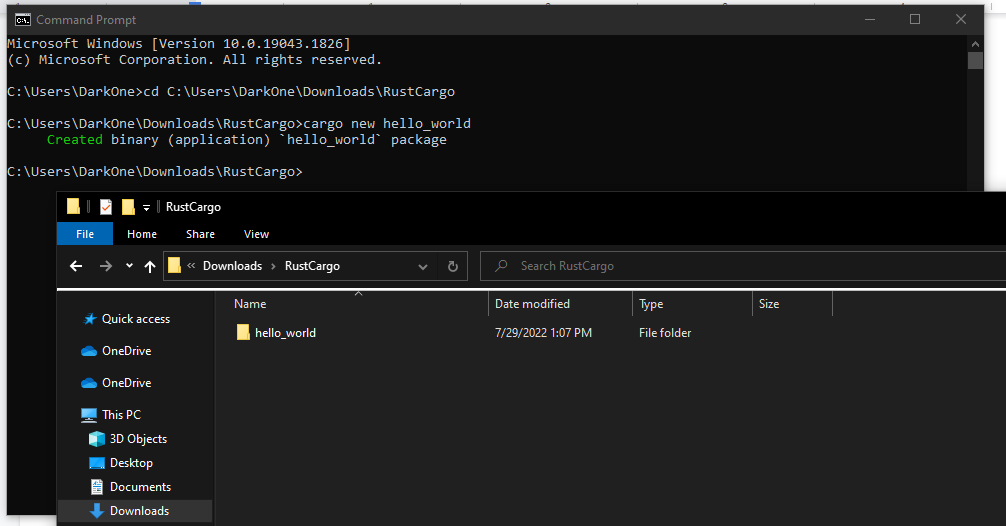
cd <rusty\_work>

Run ‘cargo’ to create new “hello\_world” project using

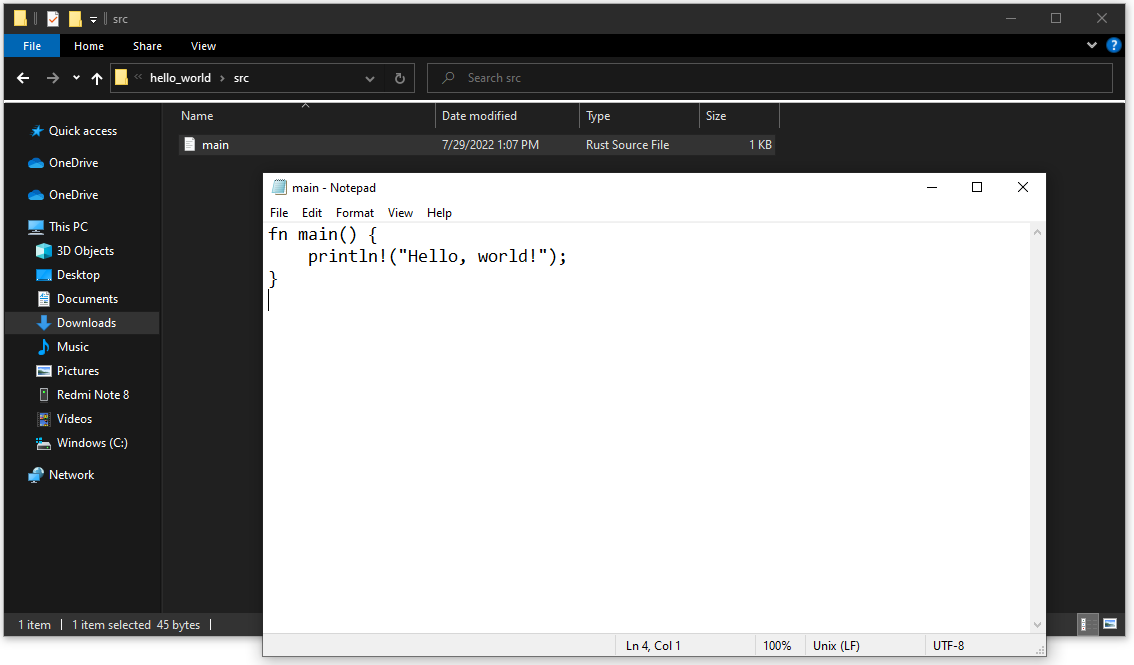
cargo new hello\_world



When finished, cargo will generate a new directory



In the directory “/hello\_world/src”, you will find the “main.rs” file. Open it with a text editor.



Modify “main.rs” to

use std::io::stdin;

fn main(){

let mut line = String::new();

println!("Enter your name :");

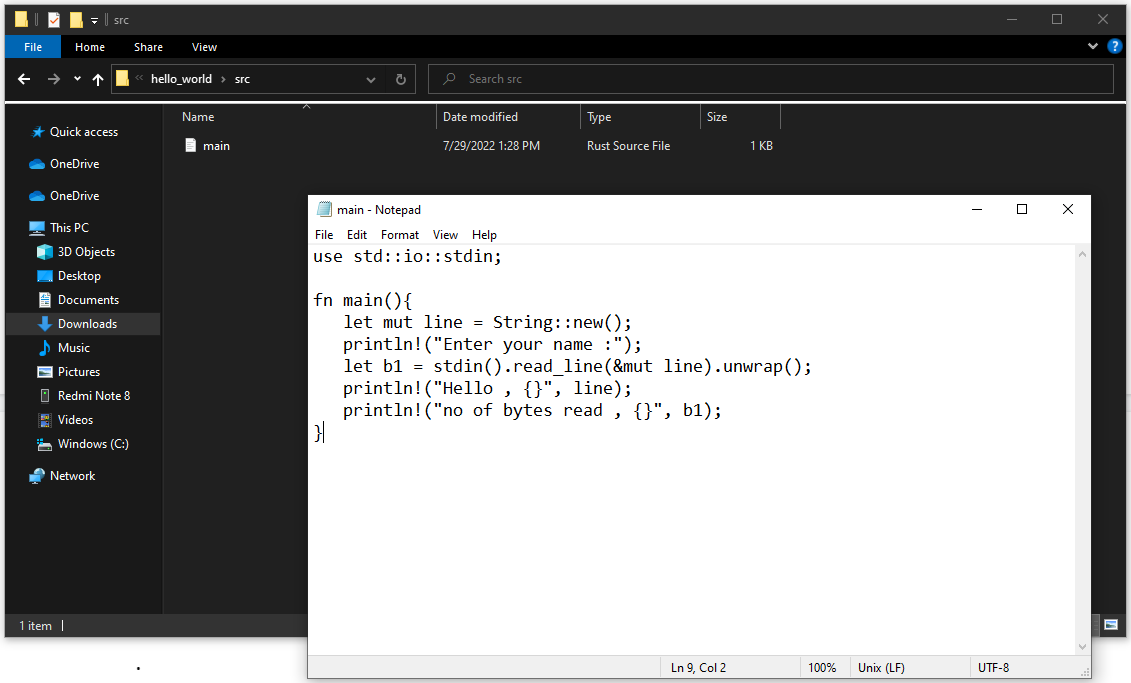
let b1 = stdin().read\_line(&mut line).unwrap();

println!("Hello , {}", line);

println!("no of bytes read , {}", b1);

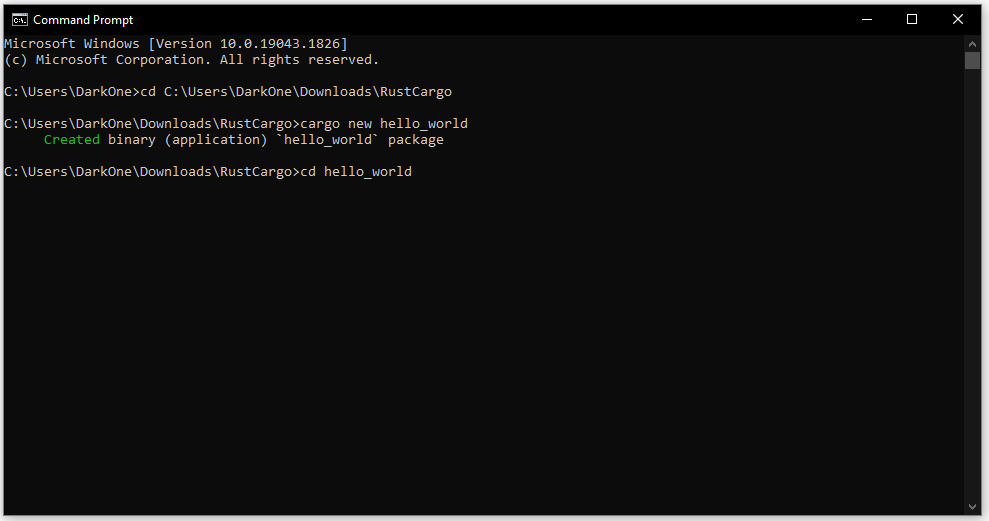
}

then save the file.

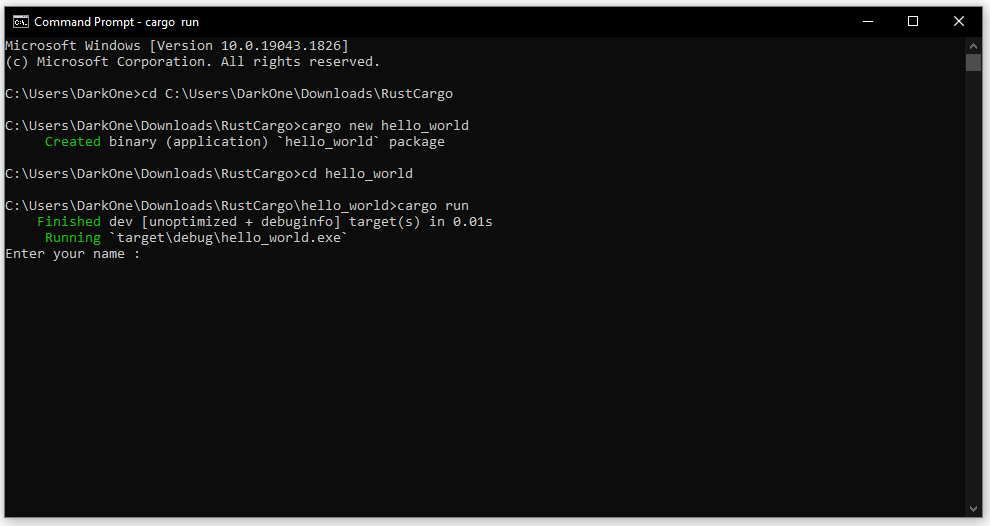


To run the new program change to the “hello\_world” directory

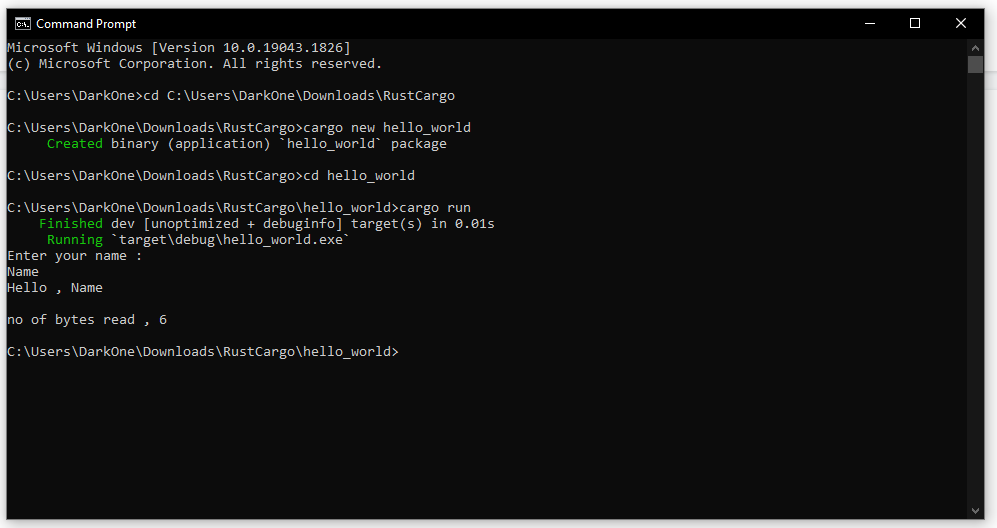
cd hello\_world



then enter the command cargo run



Following the program prompt, enter your name for example.



At this point, you can either

1. Continue using your chosen editor. This is suggested if you have not already used some other development tool, e.g. the Visual Studio tools. ***Skip to the*** ‘**A Real Program**’ section.

***OR***

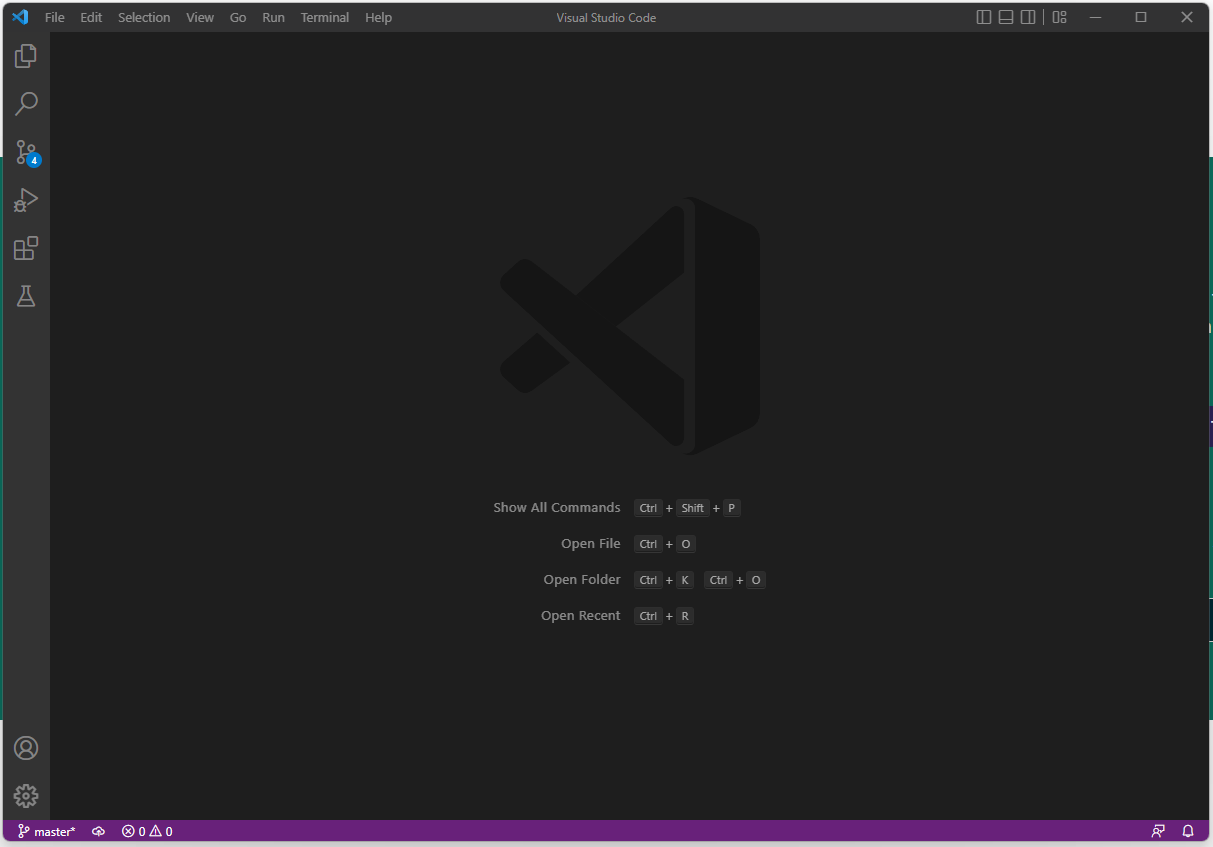
1. Use a more complex program development tool, e.g. Visual Studio. ***See*** the following notes.

# Rust Development using Visual Studio Code

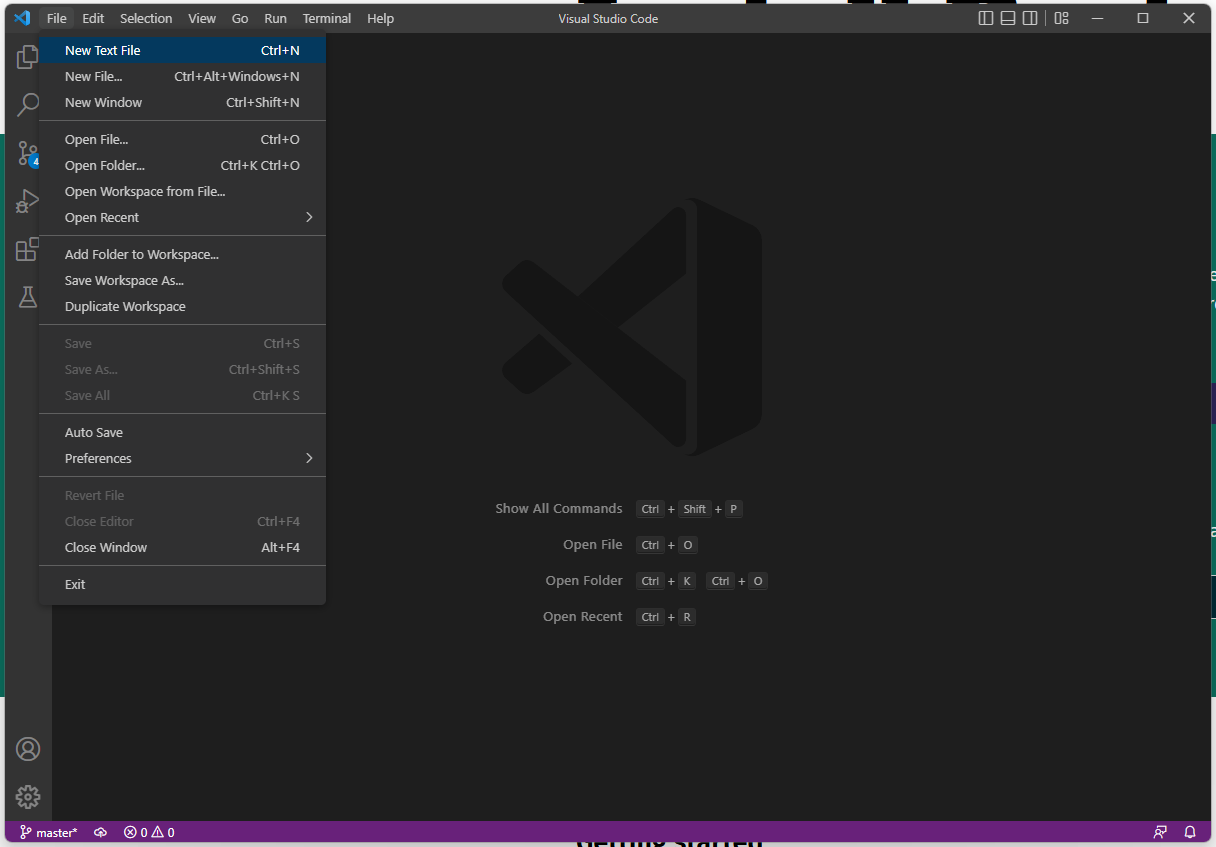
website : <https://code.visualstudio.com/download>

## Start “Hello\_world” program with rust on VS Code

Open Visual Studio Code



Then create file by File >>> New Text File

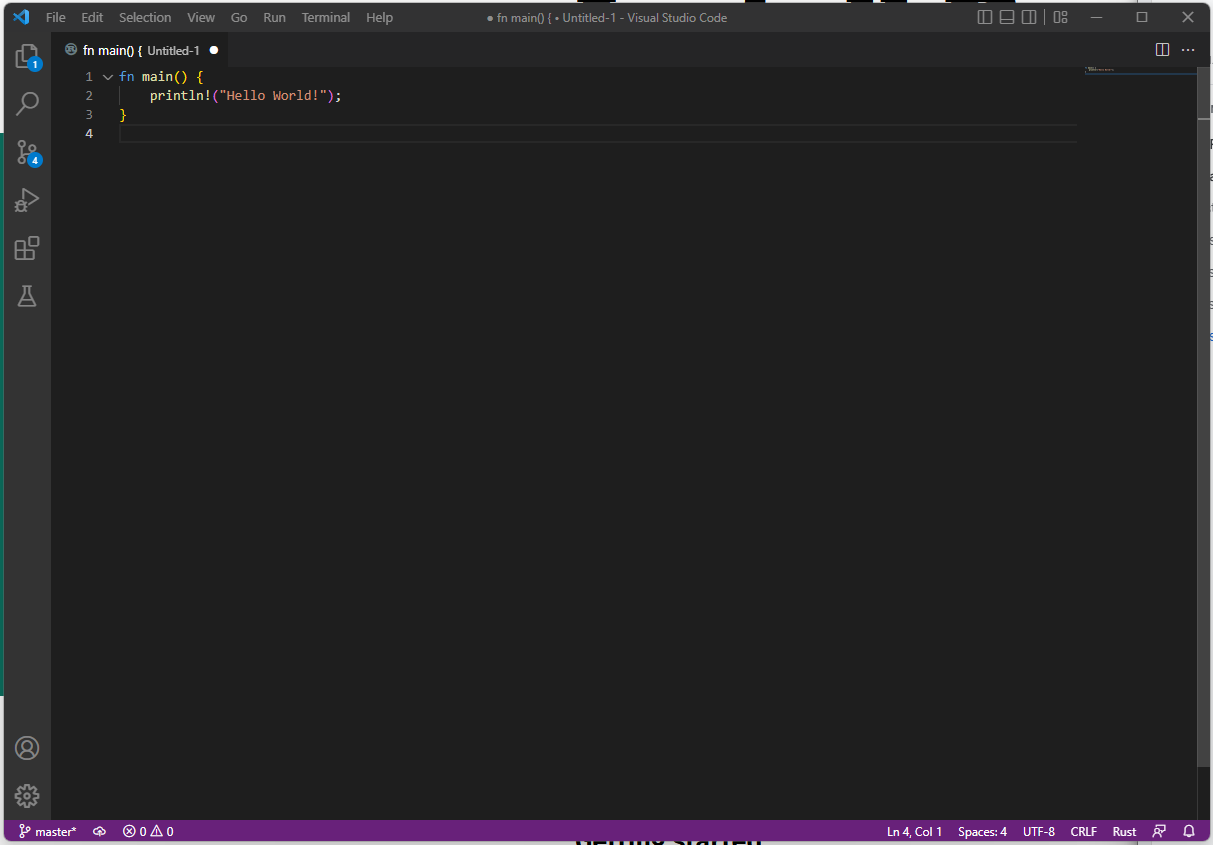


Program create Untitled file, and input the code below.

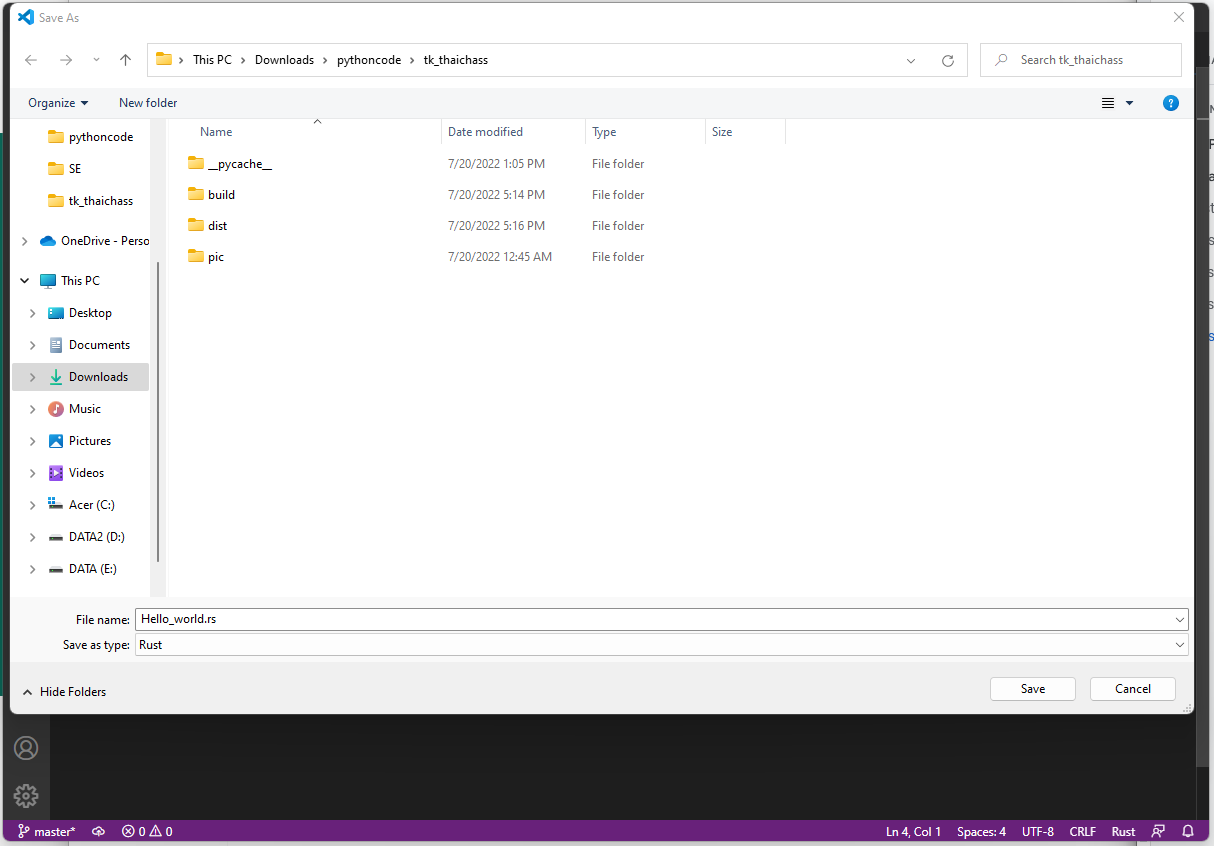
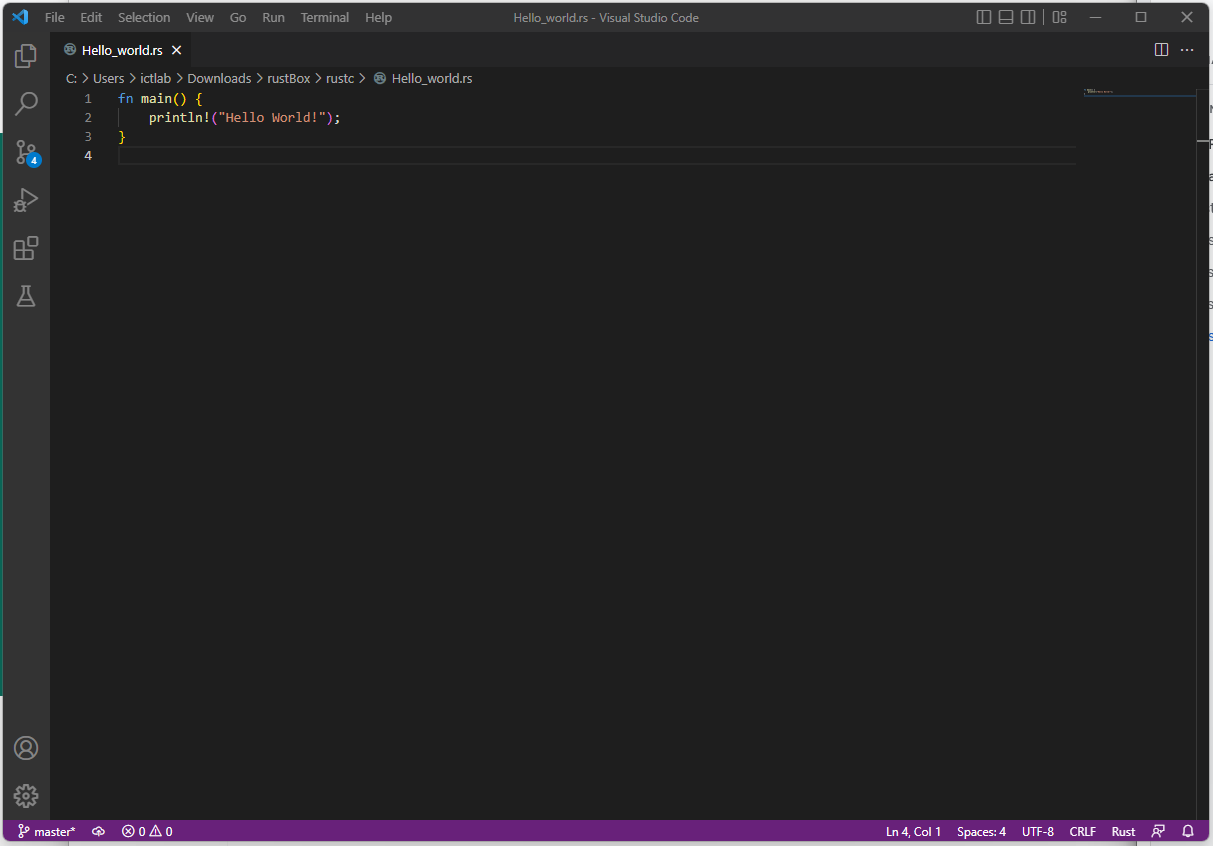
Fn main() {

println!(“Hello World!”);

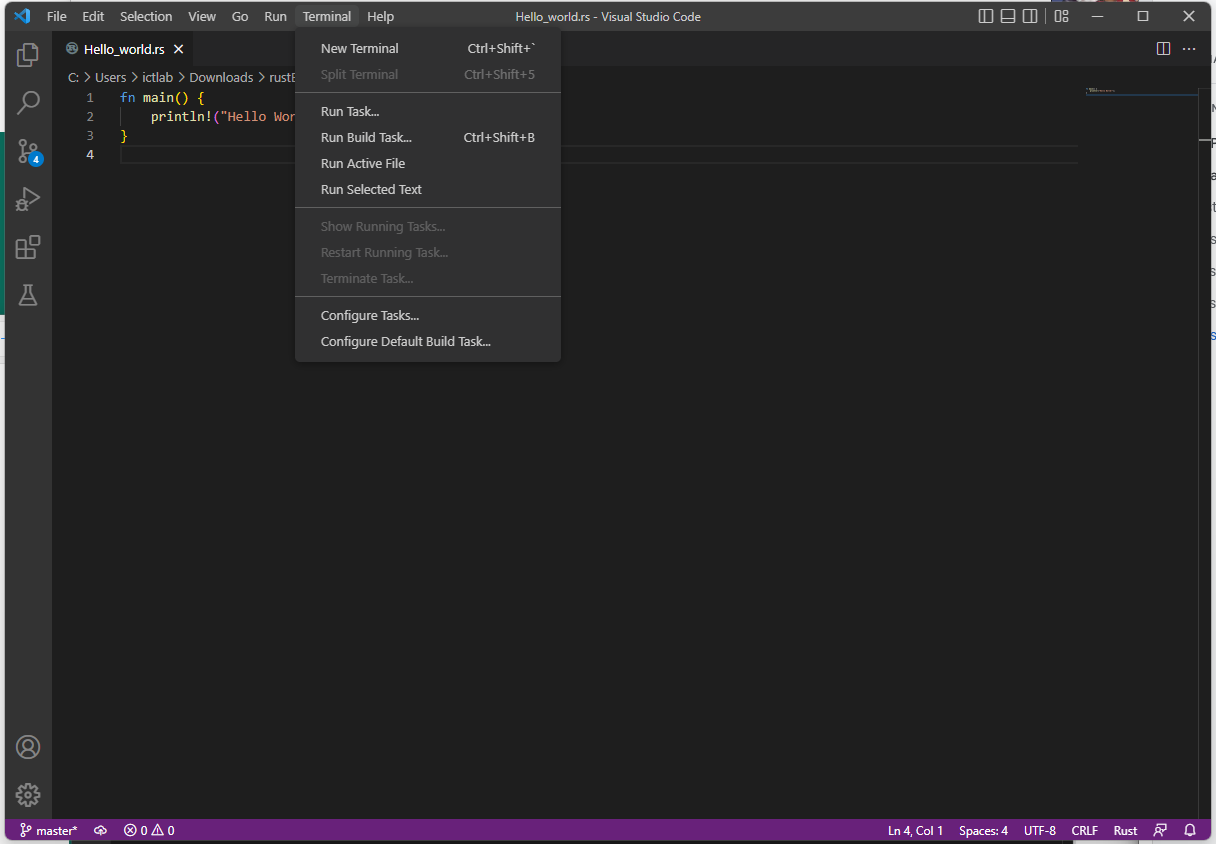
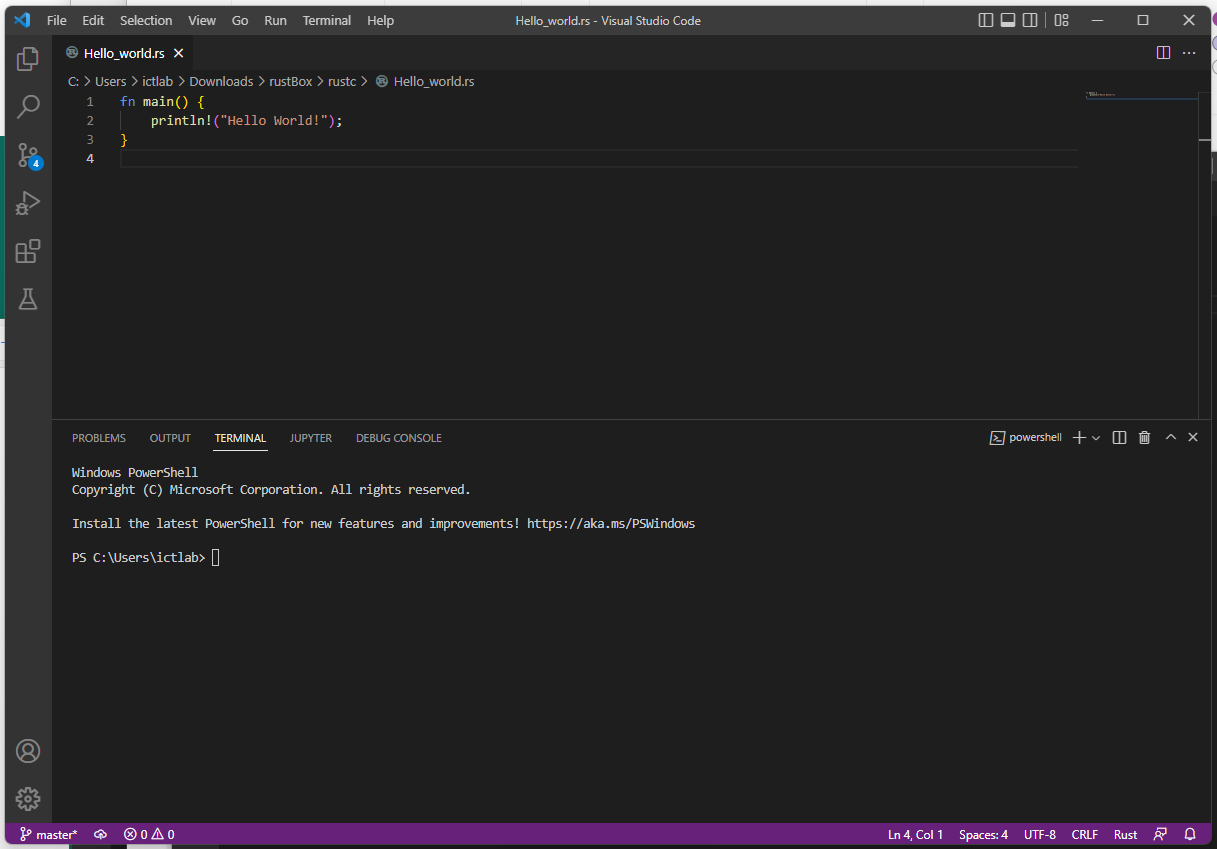
}

Then save the file as “Hello\_world.rs”

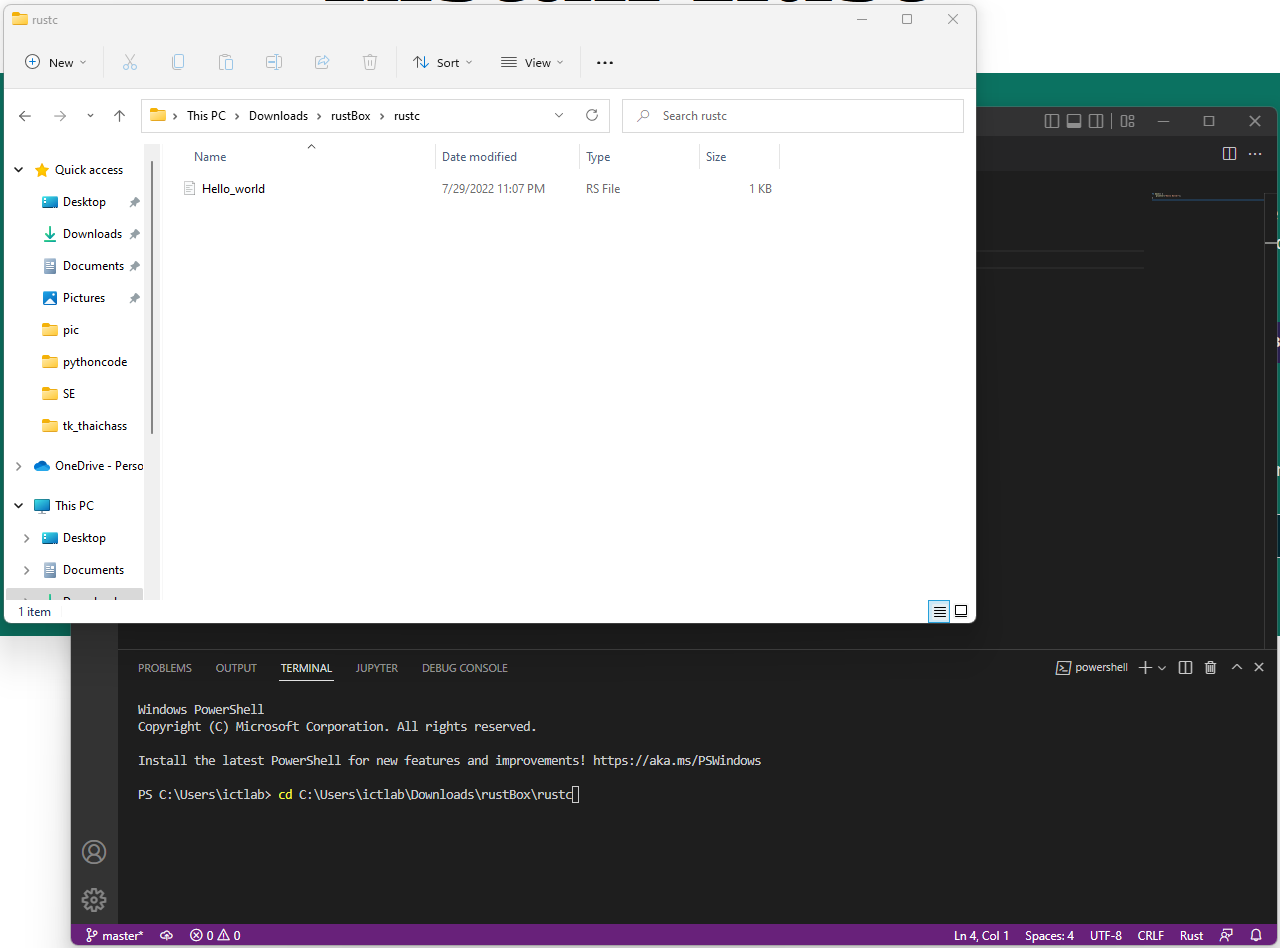
 

To compile and run the program, choose “Terminal” >>> “New Terminal” to open a command line.

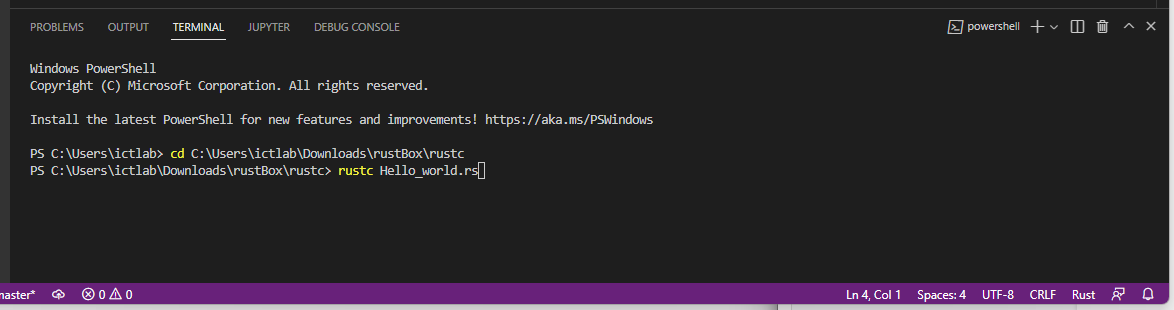
Change to the directory we going to use for the program using

cd <file directory>

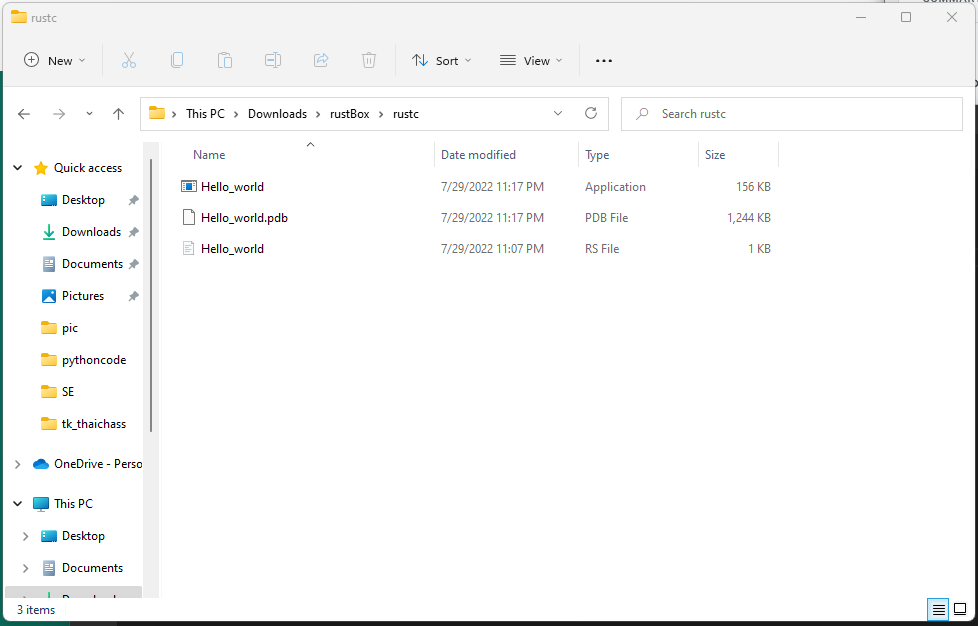


Compile “Hello\_world.rs” using

rustc Hello\_world.rs

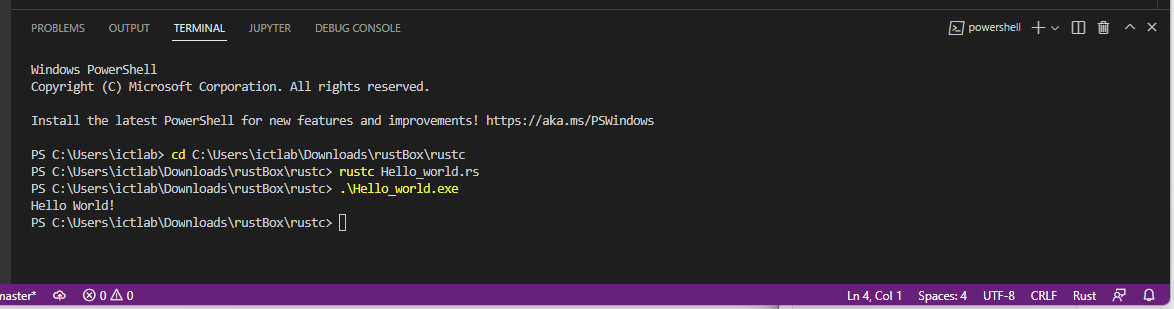


Once finished, the compiler will produce an executable “Hello\_world” program



Go back to the visual studio code to run the program using the result file

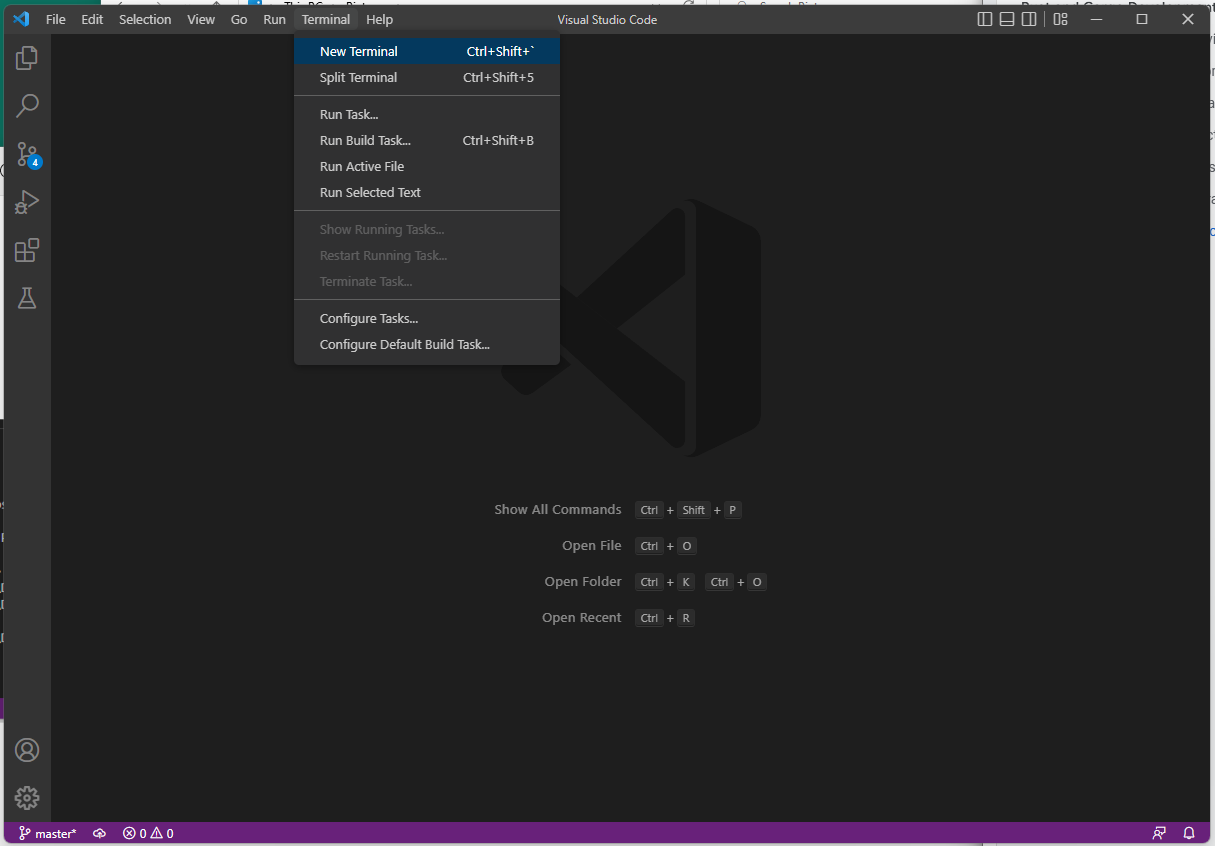
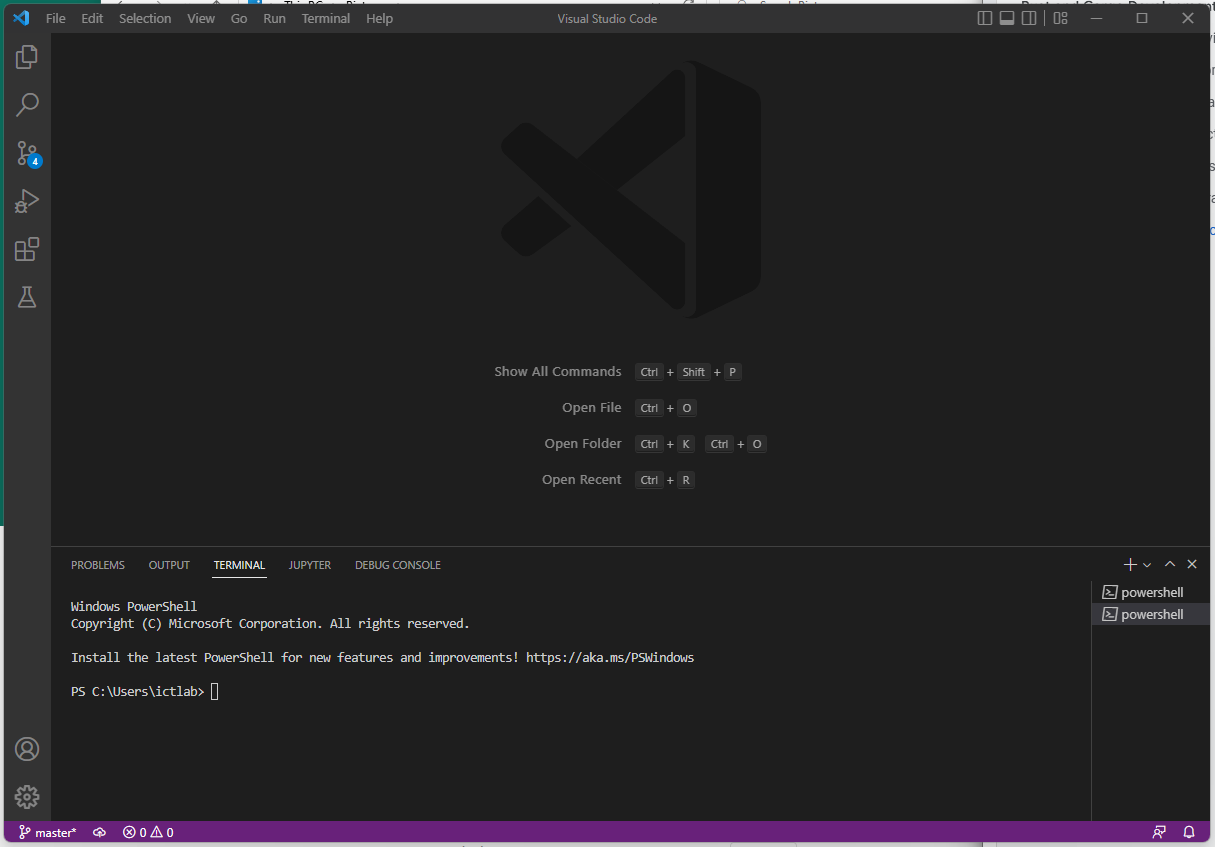
.\Hello\_world.exe



## 

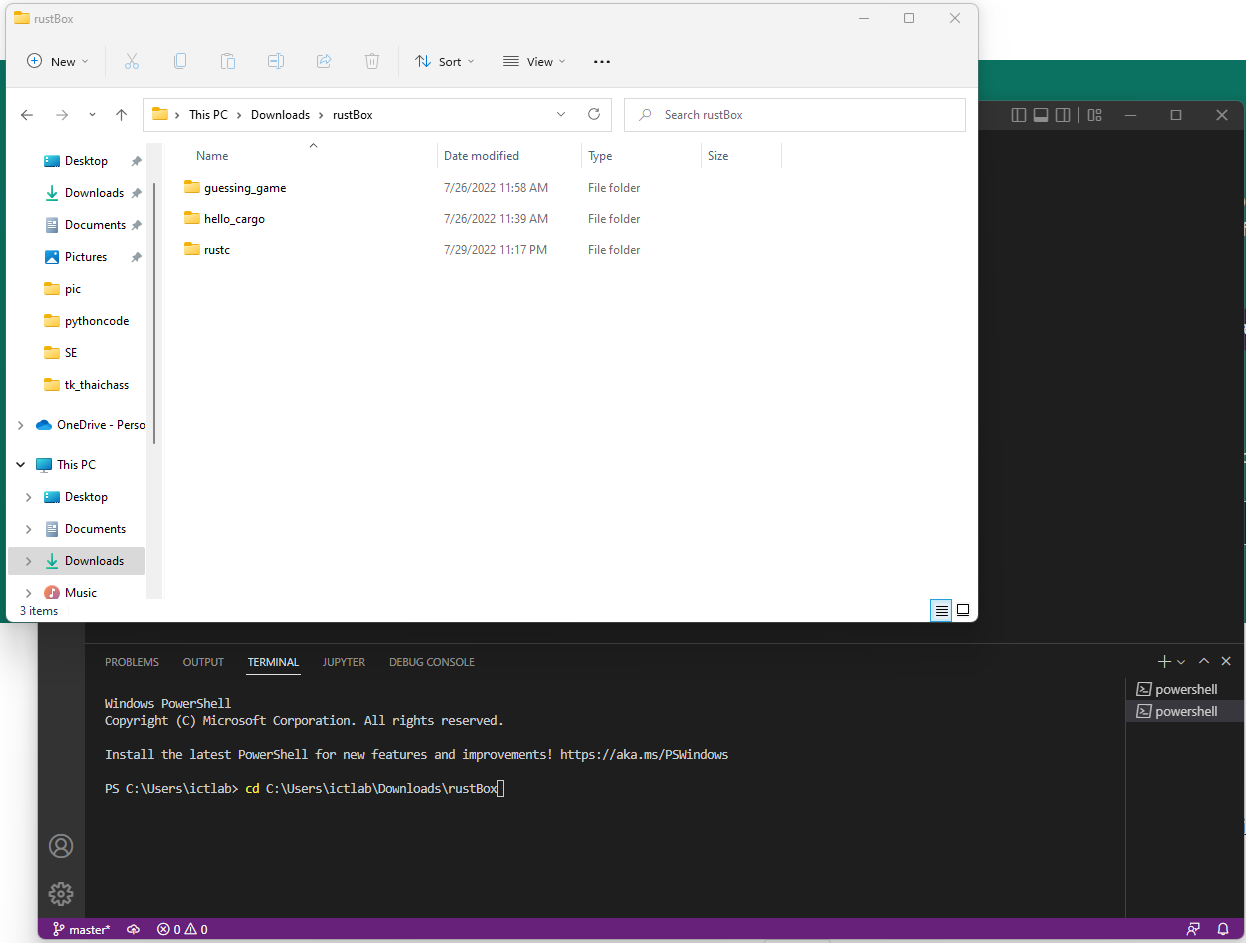
## Start “Hello\_world” project with cargo on VS Code

Use “Terminal” >>> “New Terminal” to open the terminal.

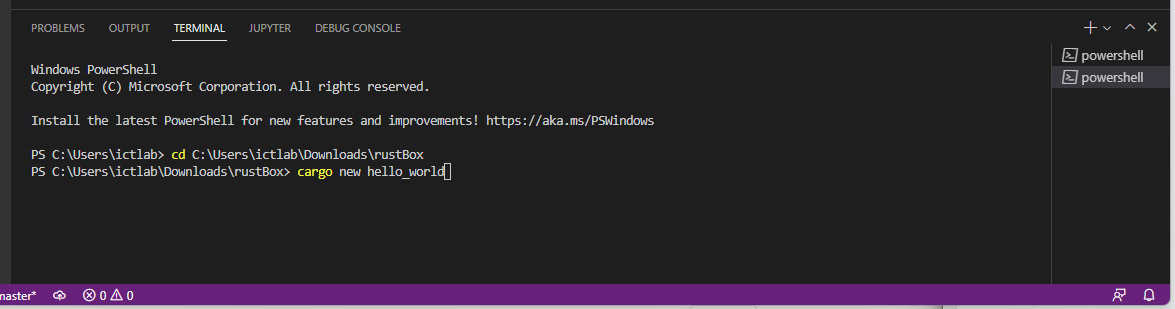
Change to the working directory for the project using

cd <file directory>

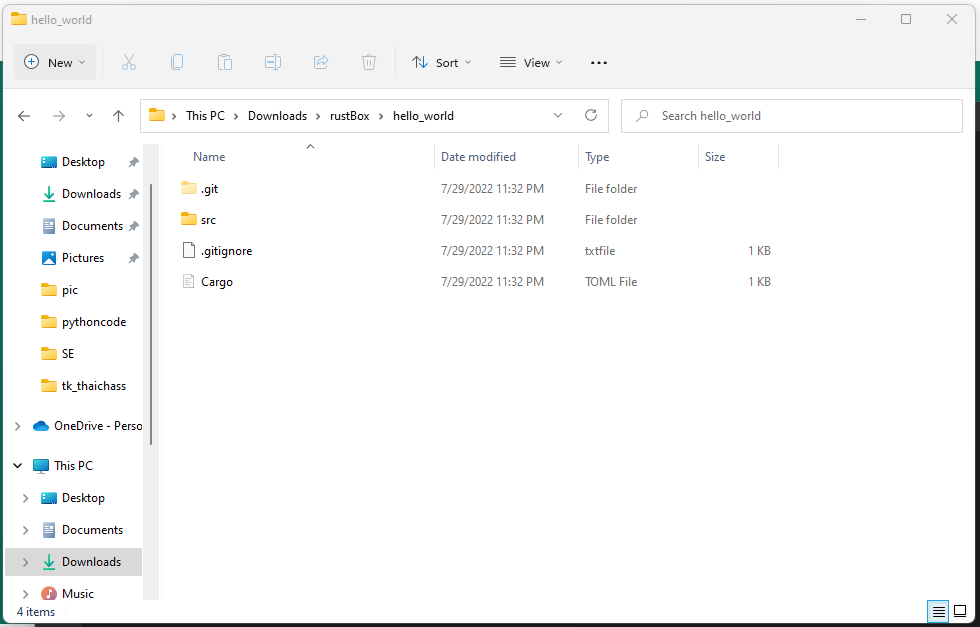


Create a new “Hello\_world” project using

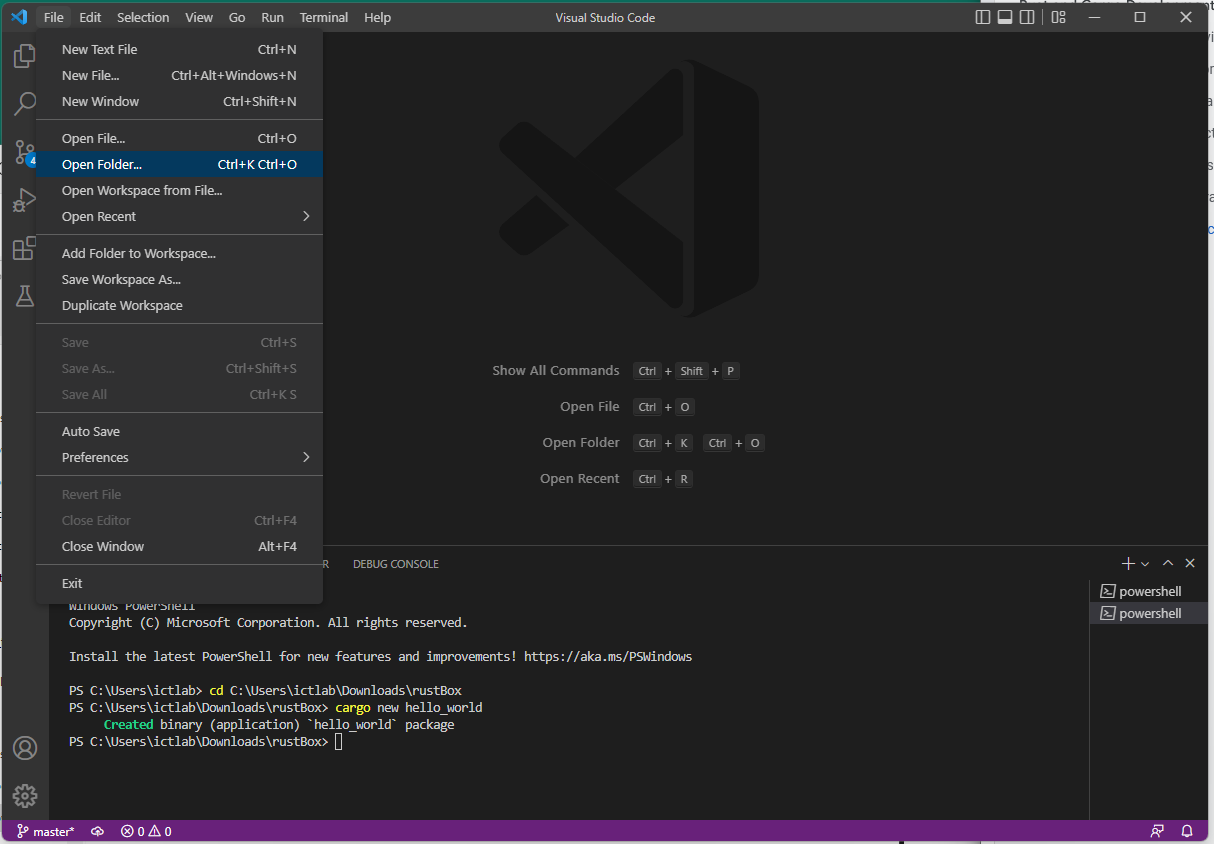
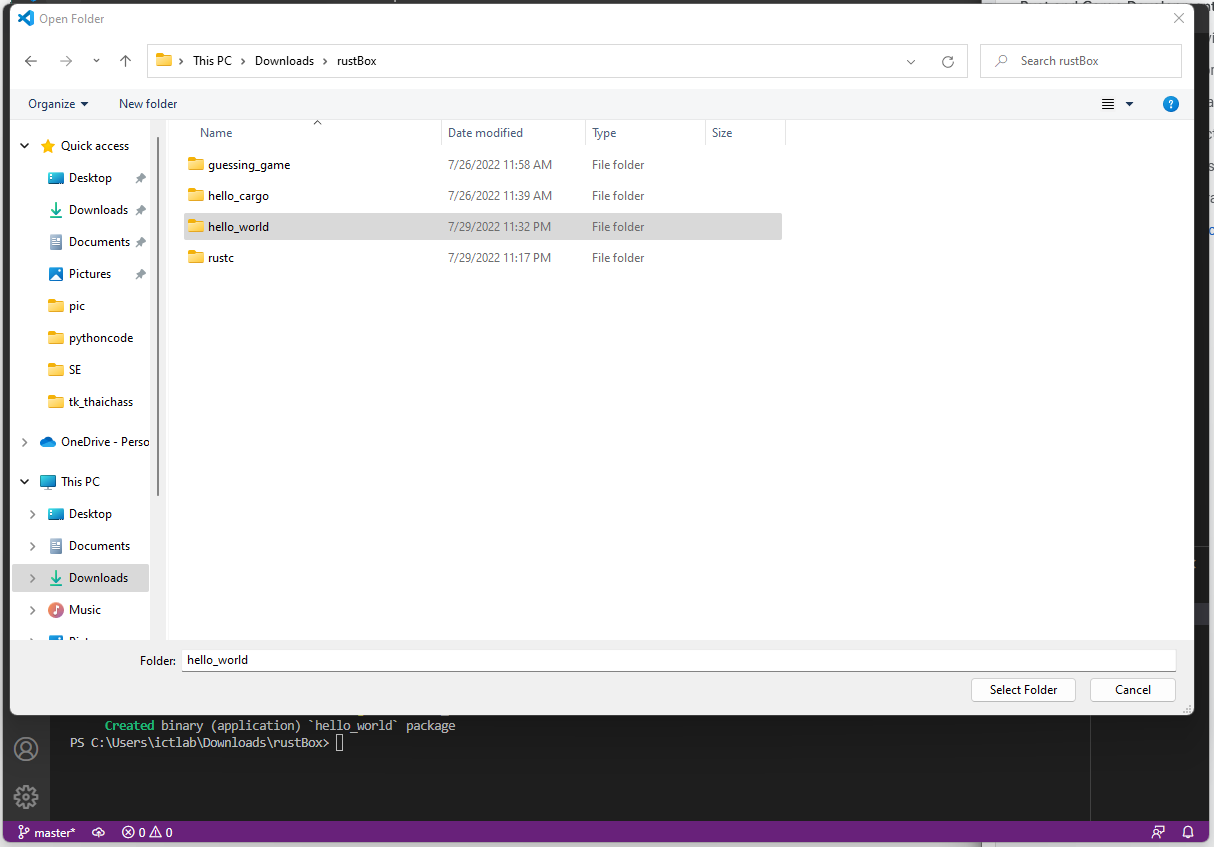
cargo new Hello\_world



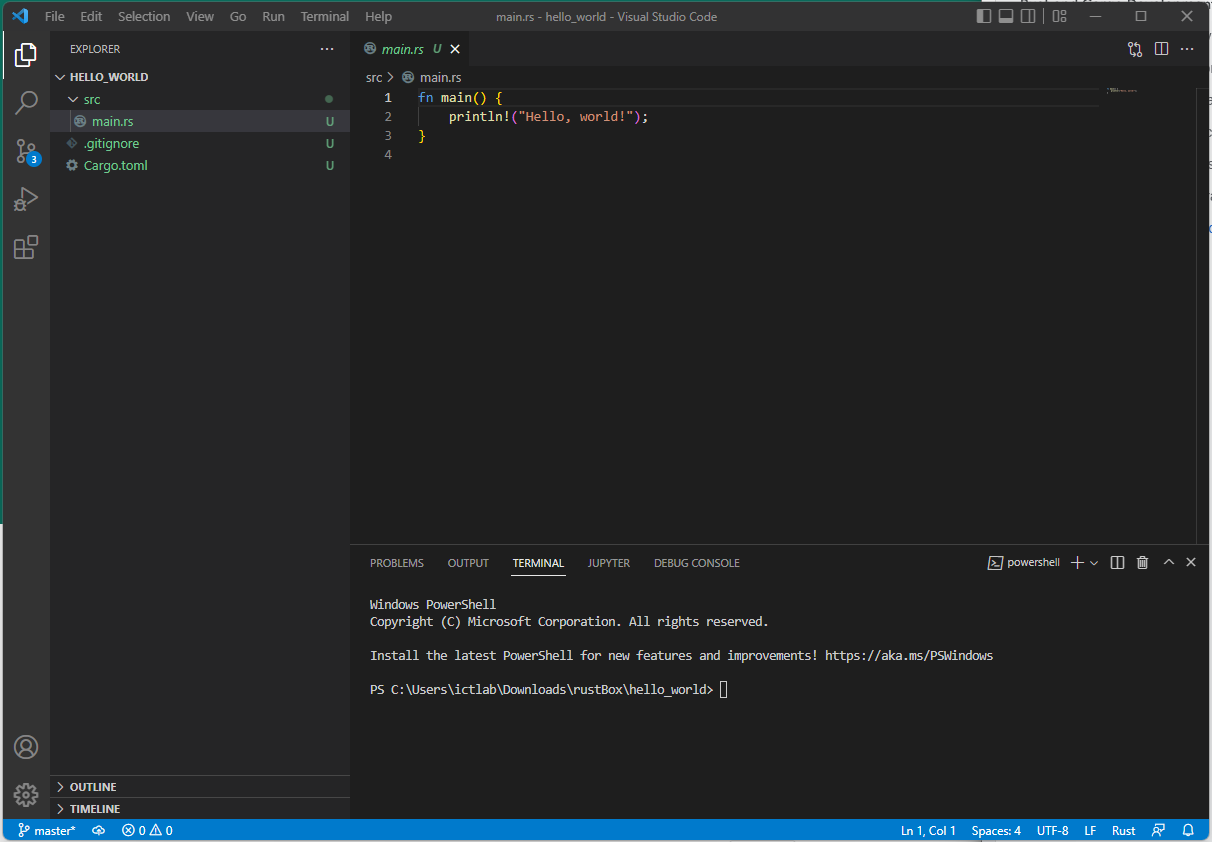
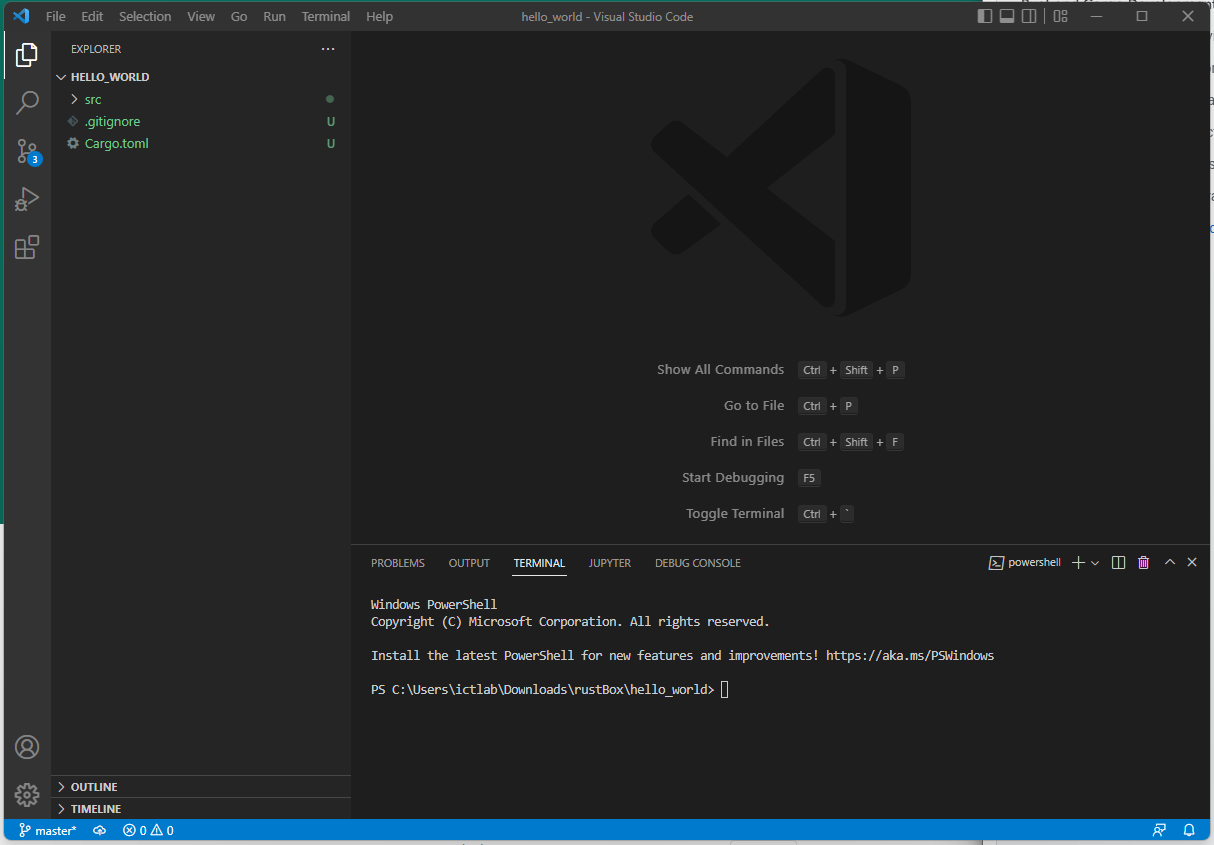
When finished, cargo will generate a new directory



Open the directory of the project in VS code by “File” >>> “Open Folder”

The project directory and files will be displayed, go to “src” >>> “main.rs” to open the file.



# A ‘Real’ program

Now that you have succeeded in creating a somewhat trivial program, let’s try to make one that we can learn something from!!

Create a new cargo project and edit the default source code (found in the src directory) to

fn main() {

println!("Squares!");

for n in 1..100 {

let x = n as f64;

let sqrt\_x:f64 = x.sqrt();

let xx:f64 = sqrt\_x \* sqrt\_x;

print!("{} ", x);

if x == xx {

println!(" OK");

}

else {

let dx = x - xx;

print!("{}", dx);

println!(" diff");

}

}

}

This code will introduce several features of Rust, e.g.

1. Making a loop – the for command will execute the loop 100 times
2. Declaring variables with let - x. sqrt\_x, xx, dx.
3. Setting variable types – f64 is a 64 bit floating point value
4. Calling library functions – sqrt() - to calculate a square root.
5. Adding an if .. else .. statement
6. Code blocks in braces – { …. }
7. Basic print commands – print! and println!
8. Converting variables from one type to another .. as converts the integer n to an f64 type

***Do not worry too much*** about details at this point – they will be covered (in excruciating detail ☹) in lectures. Run the program and compare the output with the code.

Observe that you might have believed that ***x =* √*x* ×√*x* 😉**

Do you understand why Rust thinks differently (*sometimes*!)? For homework, you can look up the representation of floating point variables (in any programming language) and see if you can understand what happened. This will prepare you for the first lecture.