**Async Programming in Rust**

Rust is a synchronous programming language, which means that the code runs in a sequential manner and one function/operation doesn’t start until the former ends.

However, we can implement asynchronous programming in rust. Async programming involves execution of a single thread to perform multiple tasks instead of many threads. There are various reasons why we would need an asynchronous program, the most important one being the latency in switching between threads. Also, multiple threads consume a lot of computer resources even if they are not running. That’s why we need async rust to counter all the above-mentioned problems.

Now we can run async code by defining an async function. This function returns a trait which is called a future trait. In order to execute the function, the future trait much be executed to completion. This async transforms the code to a form of a state machine.

Synchronous programming is already supported by all the languages therefore we do not have to import any libraries. On the other hand, async is not supported by many languages and so we need some specific languages and libraries.

It is because of this, that async functions require an executor for its execution. These executors include

* Block\_on
* Async/.await



Block\_on blocks current thread and run other future traits. It is used outside the async function while in async.await wait is used for writing async code which looks like sync code.It waits for the completion of another type that implement the future trait. It is used inside of the async function.



Although it may seem like nothing could be better than async programming it is important to note that async is very complex and therefore, should not be used for small projects. If the project is small then simply use multi-threaded approach as the complexity might not be worth it.