IDEAS – Async Rust

*This is actually an attempt to extract ideas from the main article on* [*Async Rust*](https://thomashartmann.dev/blog/async-rust/) *by Thomas Heartman. It is written with a student’s perspective and my own interpretation of the article and also what I’ve read on the subject from other sources.*

[](https://thomashartmann.dev/blog/async-rust/)Source: https://thomashartmann.dev/blog/async-rust/

# Concept Building:

Before we get on with what exactly Asynchronous Programming in Rust is, we should invest a little bit of our time in building some concepts.

**Asynchronous Programming** in Rust is when we have multiple tasks running concurrently in a single thread. **Multithreading** is also a related concept but is distinct from Asynchronous Programming. Multithreading is different because it deals with CPU-bound tasks that are normally separated over multiple cores. IO-bound tasks are however better dealt with using the concurrency in Rust Programming Language through async programming.

So multiple tasks can be run simultaneously like for example if one IO-bound task is waiting for a response from a server other tasks are run concurrently and when we reach a point where we need the result of an asynchronous task, we must *“.await”* it. In Rust, values that are *‘awaitable’* are known as *‘futures’*.

# Async in Rust is Different:

* An *async function* does not (necessarily) start executing immediately.
* To start it you must either *“.await”* it or launch it using an *executor*.
* Until then you only have something called *“Future”*.
* You need an external library to use *async/.await.*

**Then there are two code examples in the main article which are self explanatory with some definitions.**