An Introduction to Concurrency

Let's begin our learning by getting familiar with the basics of concurrency.

WE'LL COVER THE FOLLOWING

What is concurrency?

Concurrency, by definition, is the ability to break down a computer program or algorithm into individual parts, which can be executed independently. The final outcome of a concurrent program is the same as that of a program which has been executed sequentially. Using concurrency, we are able to achieve the same results in lesser time, thus increasing the overall performance and efficiency of our programs.

The trick with writing concurrent programs is to ensure the *correctness* of your program yourself. Therefore, keep in mind that all the individual chunks of your concurrent program should be executed independently and have access to the shared memory i.e. they can read from or write to the same memory location. Also, you need to take care of the allocation of resources so that no process **starves** due to a lack of resources. In addition, you will have to synchronize and coordinate between processes so that you can prevent a **deadlock** if they are dependent on each other.

In short, concurrency is all about designing and structuring your algorithm. It is widely used in computer science today to solve problems efficiently. One such example is that of bank operations in which someone deposits and withdraws cash from their bank account at the same time. The bank account is the shared resource and both the operations have to be executed concurrently in order to get the correct bank balance at the end. Hence, you have to write your code concurrently, where `deposit()` and `withdraw()` are

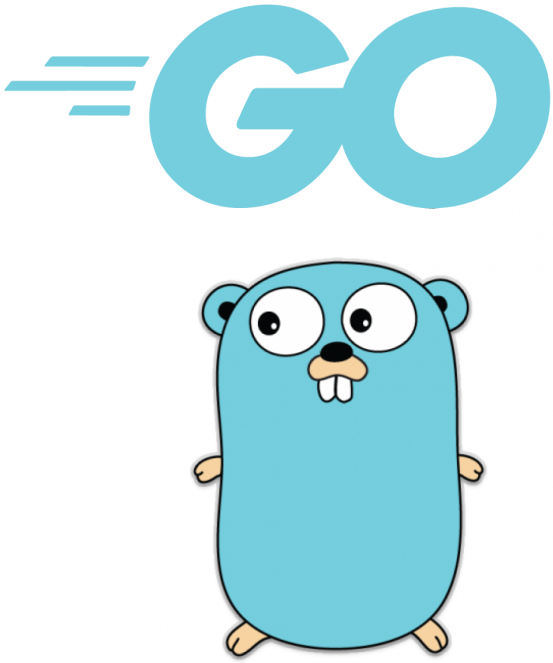
have to write your code concurrently, where `deposit()` and `withdraw()` are

independent operations which will read/write to the same bank account at the same time and obtain the exact balance after the execution.

The bottom line is that the concept of concurrency is tricky to understand and implement. In general, concurrency implies that a lot of things are happening at the same time so people tend to confuse it with parallelism. Please remember that concurrency and parallelism are two separate concepts in computer science, so make sure that you don't mix them up!

What to expect from this course?

For the sake of this course, we'll explore concurrency using Go. Go is syntactically similar to C and contains built-in commands and features that help in implementing concurrency with ease. It provides programmers with the abstraction of OS-level operations, such as multithreading so that we can invest more in thinking about concurrent designs rather than worrying about the underlying hardware. In this course, you'll learn about core concurrency concepts in the first chapter while in the second chapter, you will discover how to implement these concepts using the Go language. After getting familiar with the basics, you'll get to know about the different concurrency patterns in Go, which will prove to be very useful in sharpening your concurrent programming skills.
Happy learning!



Let’s dive into understanding concurrency and differentiating between concurrency and parallelism in the next lesson. See you there!