**Go Channels**

# What are channels in Go?

Go Channels provide concurrency for goroutines. Channels allow goroutines to exchange values with each other by allowing them to send and receive data via the channels, effectively acting as a pipeline between the two goroutines. Channels are declared using the chan keyword, followed by the variable type that is to be exchanged.

A basic example of a Go channel:



# Unbuffered Channels

An unbuffered channel is a channel that does not have a declared capacity. As a result, an unbuffered channel must immediately have a receiver ready, otherwise the sender routine will be blocked, which allows for synchronous communication between the two routines.

An example of an unbuffered Go channel *(Note: The basic example is also an unbuffered channel)*:



# Buffered Channels

A buffered channel is a channel that has a specified capacity, which is given as an extra parameter at the channel’s declaration. Unlike an unbuffered channel, a buffered channel does not need an immediate receiver after accepting a sent value. Instead, a buffered channel can accept a limited number of values to be sent to it without a receiver being immediately available. The buffered channel will be blocked only when the buffer capacity is exceeded.

An example of a buffered Go channel:



# For-loop and Channel

Using a for-loop makes it possible to iterate over a buffered channel without needing multiple receive statements. Using the range keyword, it is possible to iterate through each sent value, but the channel must be closed before iterating through the channel, as range only stops when the channel is told to close.

An example of a for-loop iterating through a channel:



# Channel close

Closing a channel is useful when there are no more values that need to be sent to the channel. Closing a channel also can indicate completeness to the channel’s receivers. Only the sender should be closing the channel, as sending data to a closed channel will cause a panic. It is still possible however, to read data from a closed channel, it is even required to close a channel before iterating through it using for and range.

An example of closing a Go channel *(Note: the for-loop example is also an example of a channel close)*:

