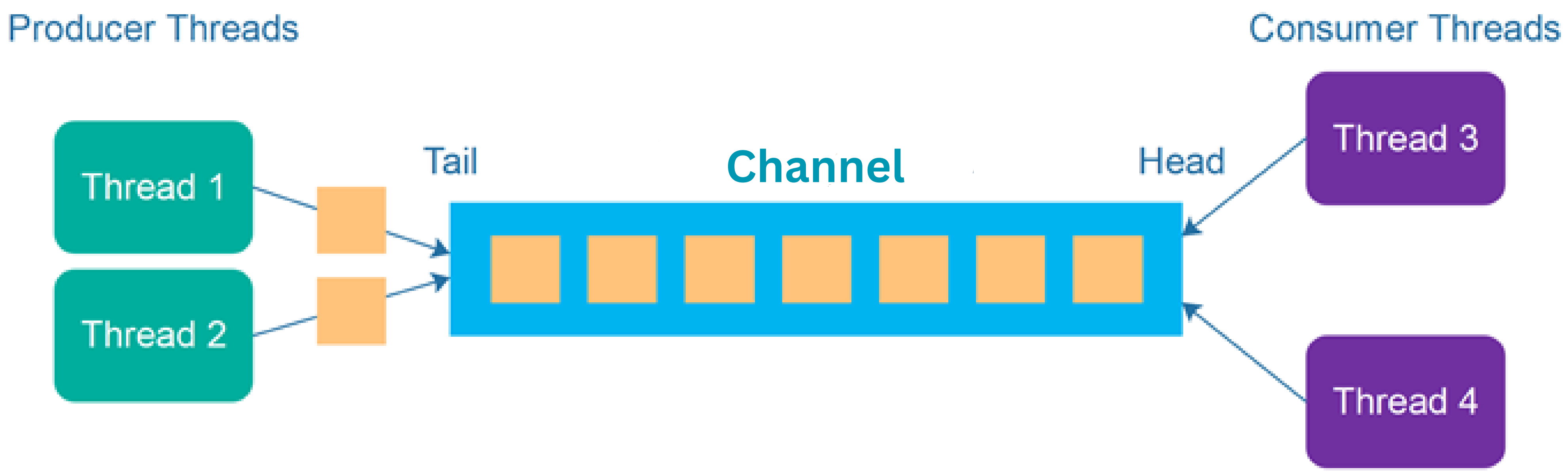
Consumer-Producer pattern

with examples in Python and Golang



Next Slide

Channel Does Not Mean Queue

The consumer-producer pattern is a concurrency-related design pattern. As the figure in previous page illustrates, in order to create a consumer-producer pattern, channels play crucial role. A producer and consumer communicate via channel.

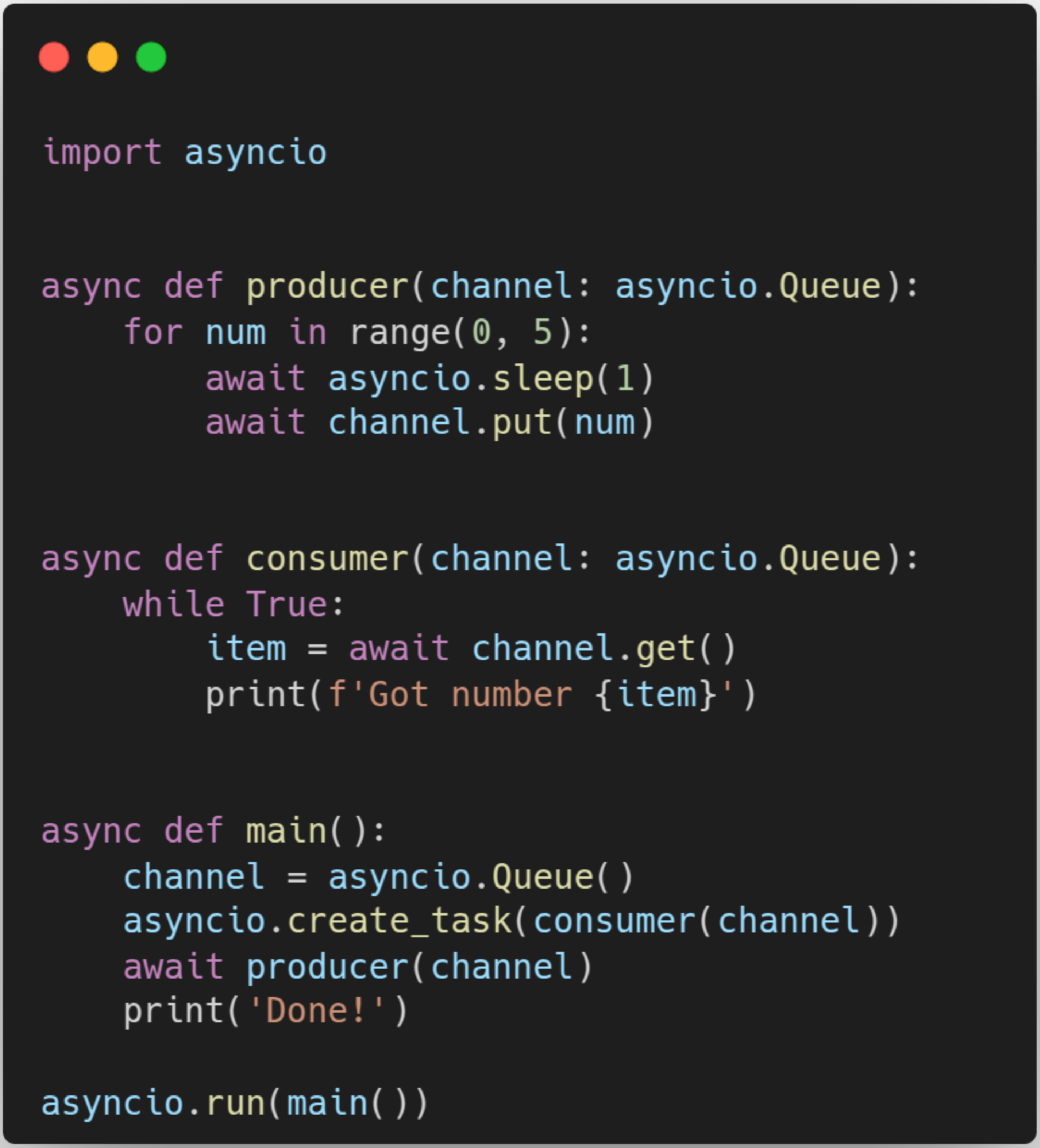
The term channel is used in languages like Golang and Kotlin. They are not exactly the same as queue. In short, a queue is a storage mechanism, while a channel is a communication mechanism (which may internally use a queue).

All three following examples would result in the same output, buffering numbers between 0 to 4 from the consumer to the producer.

Next Slide

A Basic Python Example

This is a very basic example (from cprieto) of consumer producer in Python using asyncio library which uses queue as a channel.

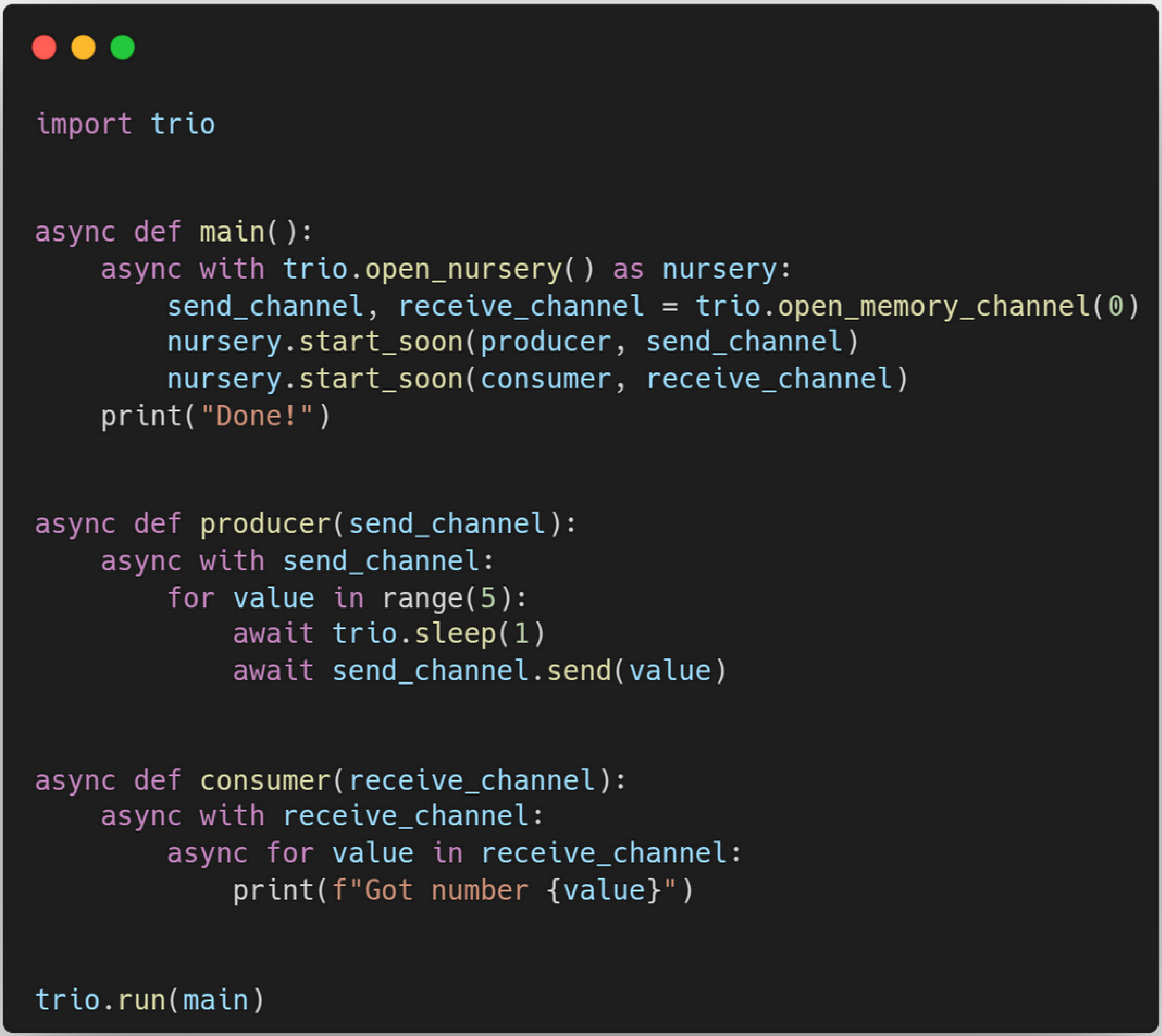


```python
import asyncio


async def producer(channel: asyncio.Queue):
    for num in range(0, 5):
        await asyncio.sleep(1)
        await channel.put(num)


async def consumer(channel: asyncio.Queue):
    while True:
        item = await channel.get()
        print(f'Got number {item}')


async def main():
    channel = asyncio.Queue()
    asyncio.create_task(consumer(channel))
    await producer(channel)
    print('Done!')

asyncio.run(main())
```

Next Slide

A More Advanced Example in Python (Trio)

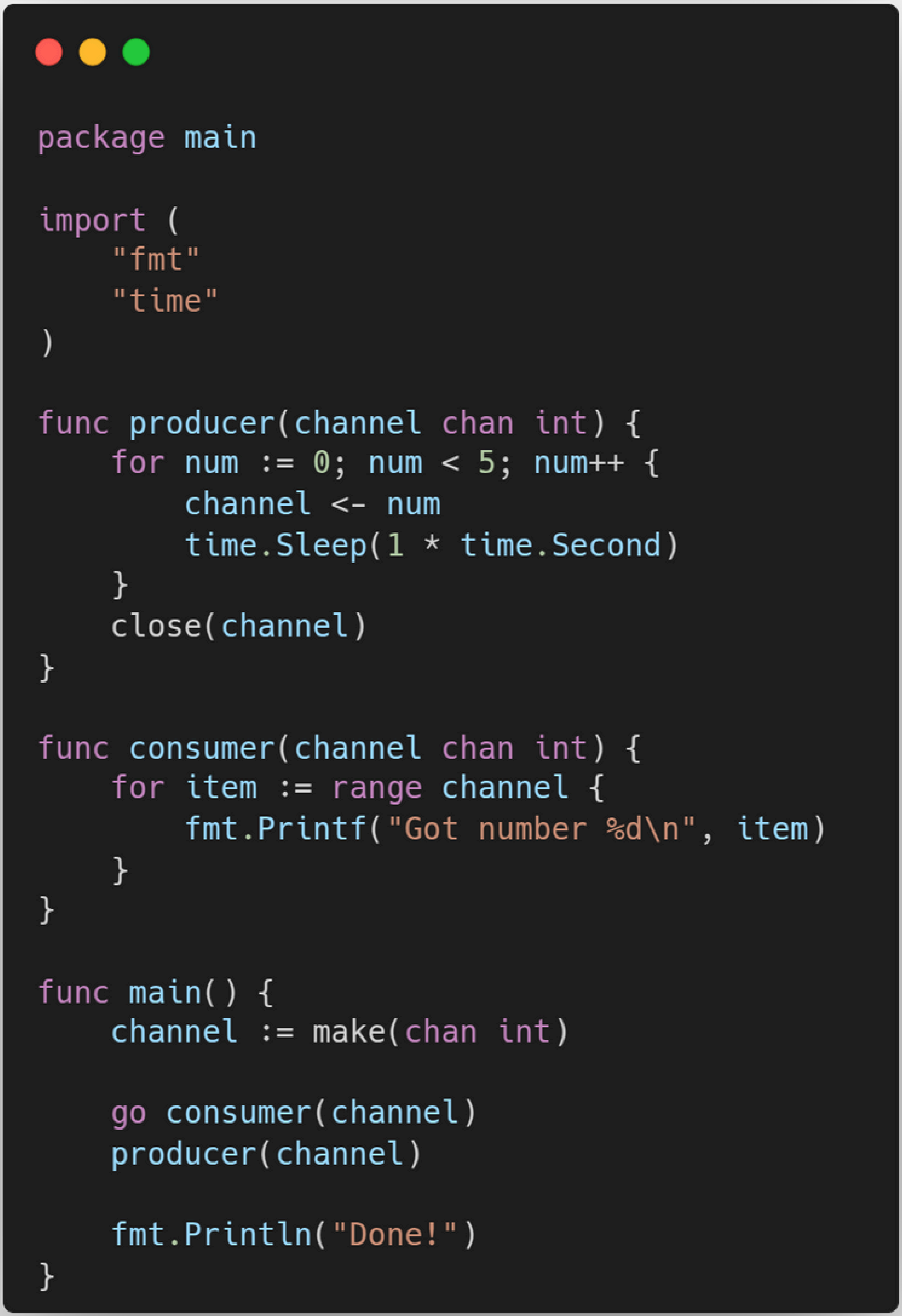
This example shows how to create a channel in python using Trio and its open_memory_channel API.



```python
import trio


async def main():
    async with trio.open_nursery() as nursery:
        send_channel, receive_channel = trio.open_memory_channel(0)
        nursery.start_soon(producer, send_channel)
        nursery.start_soon(consumer, receive_channel)
    print("Done!")


async def producer(send_channel):
    async with send_channel:
        for value in range(5):
            await trio.sleep(1)
            await send_channel.send(value)


async def consumer(receive_channel):
    async with receive_channel:
        async for value in receive_channel:
            print(f"Got number {value}")


trio.run(main)
```

Next Slide

Equivalent Golang Example



```go
package main

import (
    "fmt"
    "time"
)

func producer(channel chan int) {
    for num := 0; num < 5; num++ {
        channel <- num
        time.Sleep(1 * time.Second)
    }
    close(channel)
}

func consumer(channel chan int) {
    for item := range channel {
        fmt.Printf("Got number %d\n", item)
    }
}

func main() {
    channel := make(chan int)

    go consumer(channel)
    producer(channel)

    fmt.Println("Done!")
}
```