20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by M NAGA RAVI CHANDRA
CB.EN.U4CYS21044
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Concurrent

- Concurrent programming paradigm is about ability to handle multiple tasks at the same time(even on a single-core processor), even if they're not truly executing parallelly.

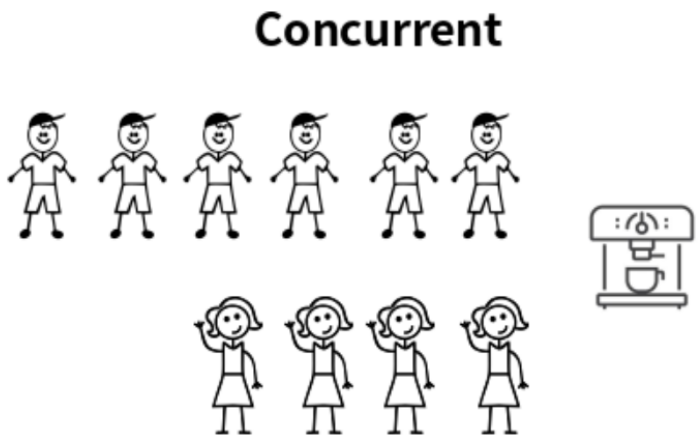


Figure: example of concurrent in real-world

```
package main

import (
        "fmt"
        "time"
)

func main() {
        start := time.Now()
        defer func() {
                fmt.Println(time.Since(start))
        }()
        evilninjas := []string{"Tommy", "jhonny", "bobby", "andy"}
        for _, evilninja := range evilninjas {
                go attack(evilninja)
        }
        time.Sleep(time.Second * 1)
}

func attack(target string) {
        fmt.Println("throw ninja stars", target)
        time.Sleep(time.Second)
}
```

Figure: example of concurrency in Go

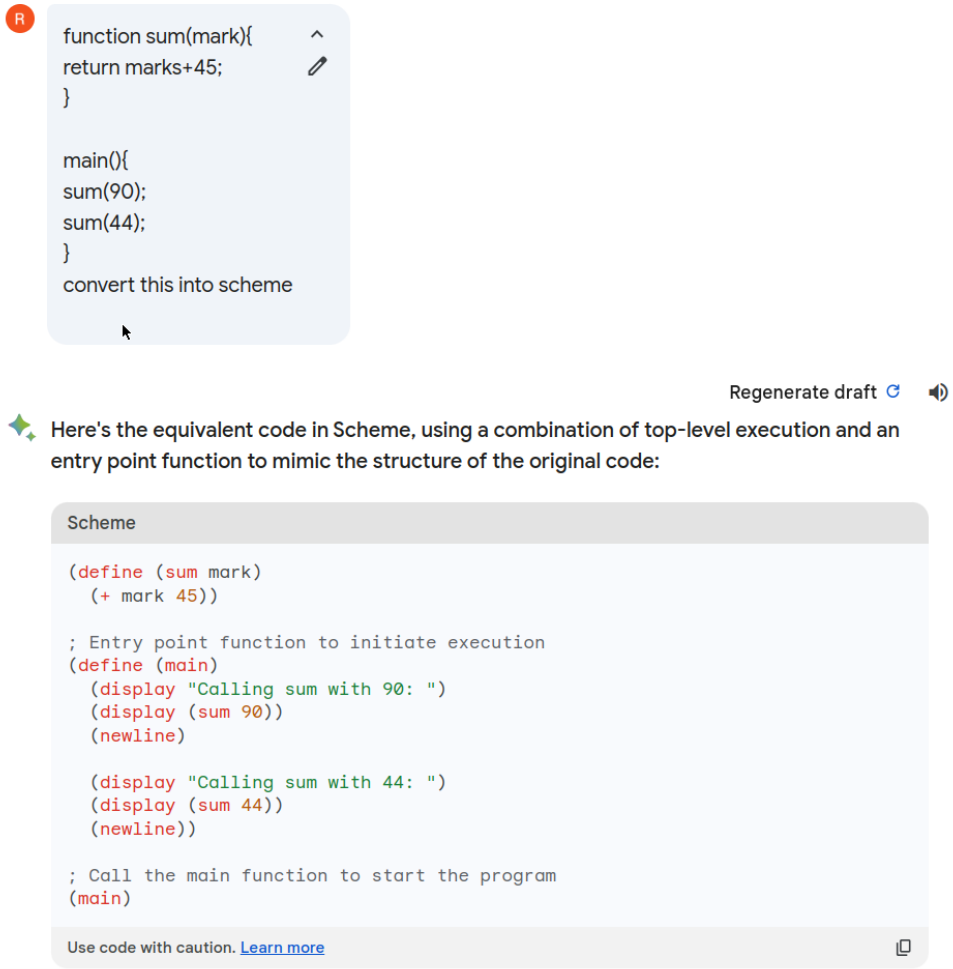
Go concepts

- Goroutines: Lightweight threads managed by the runtime.
- Channels: Typed message-passing channels for synchronization and communication between goroutines.

Functional

- Functional programming treats functions as first class citizens
  - They can be passed as parameters
- It encourage to keep the data and functions separate.
- It encourage us to create a new variable instead of overwriting the variable.

Scheme



R

```
function sum(mark){
return marks+45;
}

main(){
sum(90);
sum(44);
}
convert this into scheme
```

Regenerate draft

Here's the equivalent code in Scheme, using a combination of top-level execution and an entry point function to mimic the structure of the original code:

Scheme

```
(define (sum mark)
  (+ mark 45))

; Entry point function to initiate execution
(define (main)
  (display "Calling sum with 90: ")
  (display (sum 90))
  (newline)

  (display "Calling sum with 44: ")
  (display (sum 44))
  (newline))

; Call the main function to start the program
(main)
```

Use code with caution. Learn more

Figure: example of functional in Scheme

Scheme concepts

- iteration through recursion
- delayed evaluation

Comparison and Discussions

Functional	Concurrent
declarative style	execution of multiple tasks
Pure functions,immutability	handling shared resources
Sequential	Concurrent
Predictability, easier reasoning and testing	Improved performance, efficient resource utilization
Can be less efficient for certain tasks	Increased complexity, potential for race conditions and synchronization issues

Table: Differences between Functional and Concurrent

References

- I took the code for Go concurrent from `https://youtu.be/oHIbeTmmTaA?feature=shared`
- I got scheme code for functional using Bard.
- I took image used for concurrent from `https://github.com/nikhilkumarsingh/concurrent-programming-in-python/blob/master/concurrency.ipynb`
- I took reference from this video `https://youtu.be/dAPL7MQGjyM?feature=shared` for functional.