Lecture #14 Coroutine II

How to write a coroutine-based iterator.

① Write a function f that prints the sequence

② change print to yield.

③ Use f as the body of the coroutine.

Lexical Scope.

Merge 3 trees.

```
fun merge3 (t1, t2, t3)
    it1 = treeIter(t1)
    it2 = treeIter(t2)
    it3 = treeIter(t3)
    v1 = it1()
    v2 = it2()
    v3 = it3()
    while v1 or v2 or v3 do
        select = 0
        if (v1 != nil) {
            t = v1   select = 1.
        if (v2 != nil andalso v2 < t)
            t = v2   select = 2
        if (v3 != nil andalso v3 < t)
            t = v3   select = 3
        print(t, " ")
```

merge :: tree -> tree -> iter.

If the program position is a part of the context. then use a coroutine.

If it's easy to store from the beginning -> then use closure.