Lecture #15 Coroutines

Range Iterator:

- when does Range create?
  → an iterator
- how to get next element?
  → call the iterator → a closure.
- how to determine 'end':
  → return null.

Implement range(s, e)

```
function range(s, e) {
  cntr = s - 1;
  return function() {
    cntr++;
    if (cntr < e) cntr
    else null
  };
}
```

Desugar for:

```
for v in E { Stmt }.
```

→

```
var it = E
var v = it()
while (v != null) {
  S
  v = it()
}.
```

for (it = E.v = it(); v != null; v = it()) { S }

yield & resume.

return & call.

Asymmetric coroutine.

co = create_coroutine(body)

resume(co, args)

return at the end → implicit yield.