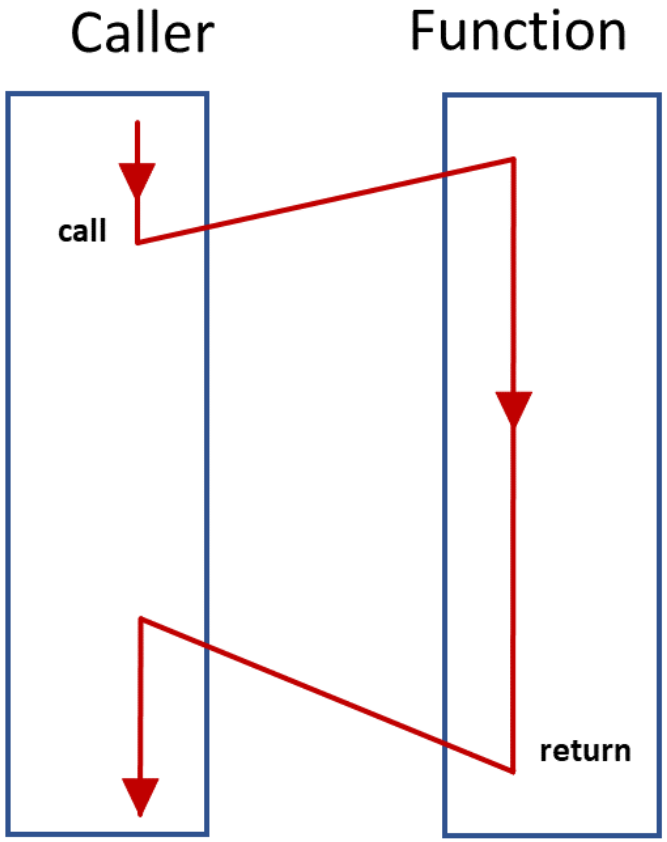
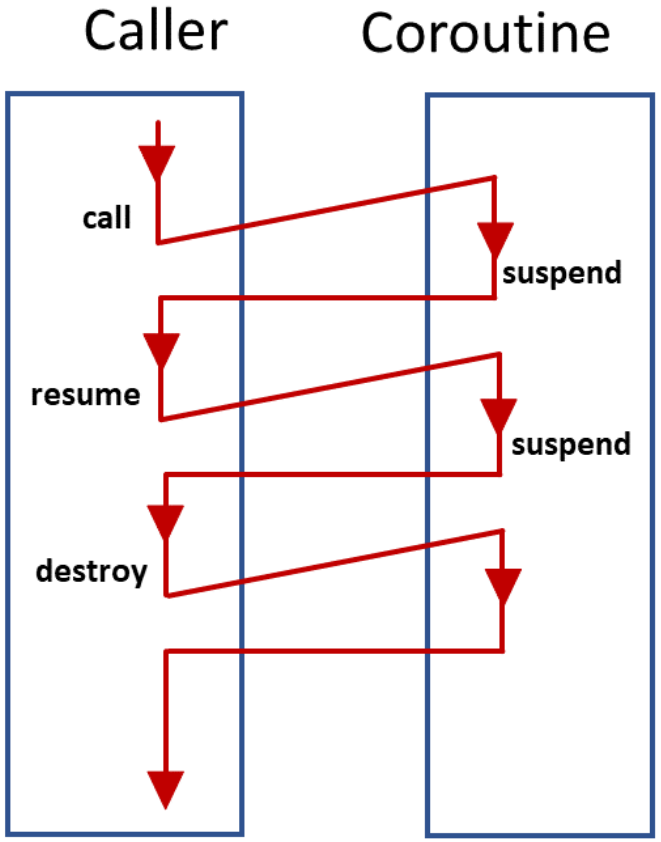
Coroutines

Coroutines are generalized functions that can be paused and resumed while saving their state.





Characteristics

- Two new concepts
  - `co_await expression`: suspend and resume expression; expression is an awaitable
  - `co_yield expression`: support generators

- Typical use cases
  - Cooperative Tasks
  - Event loops
  - Infinite data streams
  - Pipelines

Characteristics

Design Principles (James McNellis)

- **Scalable**, to billions of concurrent coroutines
- **Efficient**: Suspend/resume operations comparable in cost to function call overhead
- **Open-Ended**: Library designers can develop coroutine libraries
- **Seamless Interaction** with existing facilities with no overhead
- **Usable** in environments where exceptions are forbidden or not available

Characteristics

	Function	Coroutine
`invoke`	`func(args)`	`func(args)`
`return`	`return statement`	`co_return statement`
`suspend`		`co_await expression` `co_yield expression`
`resume`		`coroutine_handle<>::resume()`

A function is a coroutine if it contains a call `co_return`, `co_await`, `co_yield`, or a range-based for loop `co_await`.