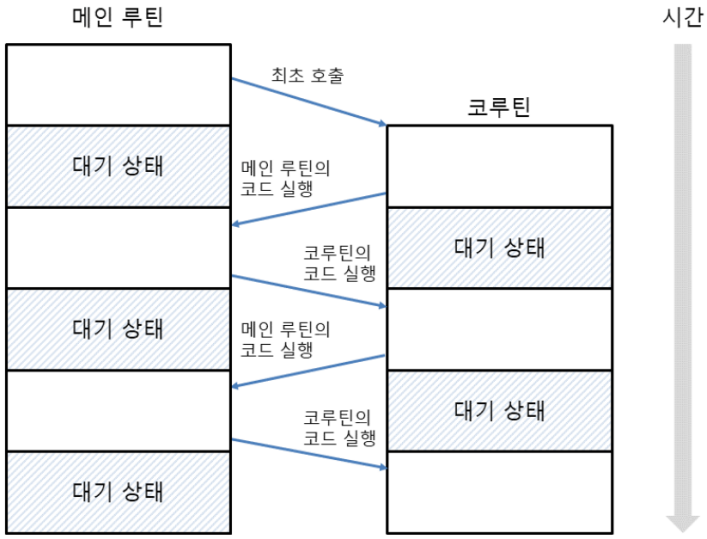
Unit 41.

Status	완료
담당자	
마감일	
완료일	@2022년 12월 2일

Unit 41. 코루틴 사용하기

지금까지는 함수를 호출한뒤에 함수가 끝나면 현재 코드로 돌아왔었는데 코루틴은 서로 대등한 관계이며 특정 시점에 상대방의 코드를 실행합니다.



코루틴은 함수가 종료되지 않은 상태에서 메인 루틴의 코드를 실행한 뒤 다시 돌아와서 코루틴의 코드를 실행한다.

41.1.1 코루틴에 값 보내기

코루틴은 제너레이터의 특별한 형태

제너레이터는 yield로 값을 발생시켰지만 코루틴은 yield로 값을 받아올 수 있다

코루틴에 값을 보내면서 코드를 실행할 때는 send 메서드를 사용

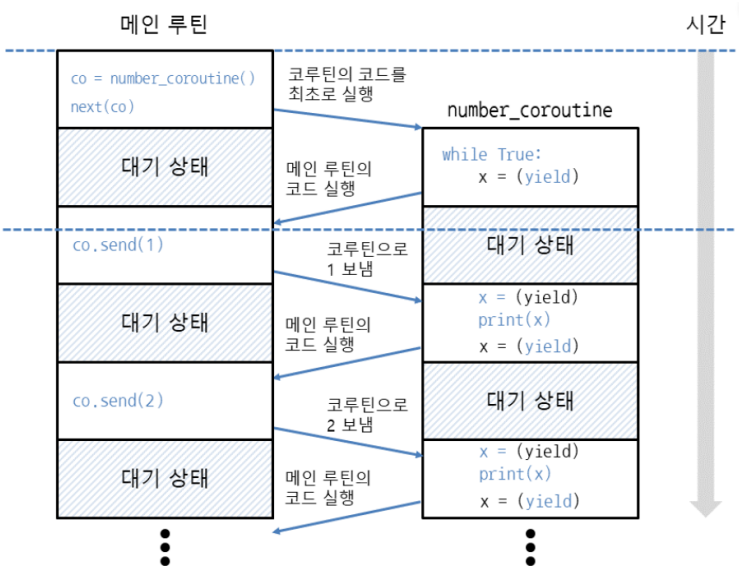
- **코루틴객체.send(값)**
- **변수 = (yield)**

```
def number_coroutine():
    while True:         # 코루틴을 계속 유지하기 위해 무한 루프 사용
        x = (yield)     # 코루틴 바깥에서 값을 받아옴, yield를 괄호로 묶어야 함
        print(x)

co = number_coroutine()
next(co)      # 코루틴 안의 yield까지 코드 실행(최초 실행)

co.send(1)    # 코루틴에 숫자 1을 보냄
co.send(2)    # 코루틴에 숫자 2을 보냄
co.send(3)    # 코루틴에 숫자 3을 보냄
```

코루틴의 코드를 최초로 실행하면 x = (yield)의 yield에서 대기하고 다시 메인 루틴으로 돌아온다.



41.2 코루틴 바깥으로 값 전달하기

(yield 변수) 형식으로 yield에 변수를 지정한 뒤 괄호로 묶어주면 값을 받아오면서 바깥으로 값을 전달한다.

yield를 사용하여 바깥으로 전달한 값은 next 함수(**next** 메서드)와 send 메서드의 반환값으로 나온다.

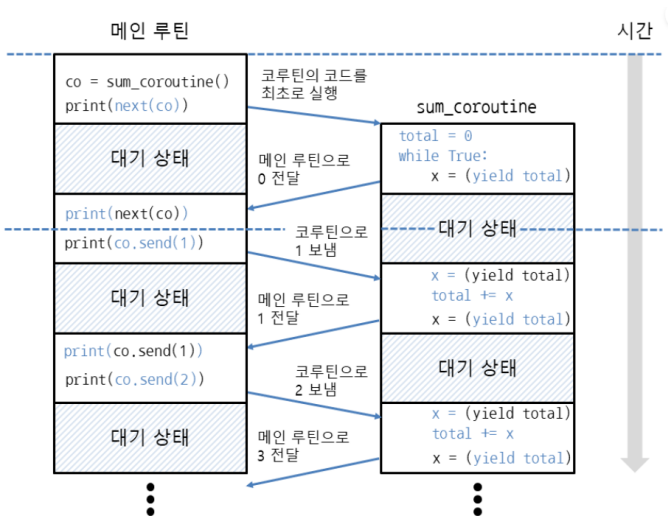
- **변수 = (yield 변수)**
- **변수 = next(코루틴객체)**
- **변수 = 코루틴객체.send(값)**

```
def sum_coroutine():
    total = 0
    while True:
        x = (yield total)    # 코루틴 바깥에서 값을 받아오면서 바깥으로 값을 전달
        total += x

co = sum_coroutine()
print(next(co))      # 0: 코루틴 안의 yield까지 코드를 실행하고 코루틴에서 나온 값 출력

print(co.send(1))    # 1: 코루틴에 숫자 1을 보내고 코루틴에서 나온 값 출력
print(co.send(2))    # 3: 코루틴에 숫자 2를 보내고 코루틴에서 나온 값 출력
print(co.send(3))    # 6: 코루틴에 숫자 3을 보내고 코루틴에서 나온 값 출력
```

next와 send의 차이를 살펴보면 next는 코루틴의 코드를 실행하지만 값을 보내지 않을 때 사용하고, send는 값을 보내면서 코루틴의 코드를 실행할 때 사용한다.



- 제너레이터는 next 함수(__next__ 메서드)를 반복 호출하여 값을 얻어내는 방식
- 코루틴은 next 함수(__next__ 메서드)를 한 번만 호출한 뒤 send로 값을 주고 받는 방식

41.3 코루틴을 종료하고 예외 처리하기

보통 코루틴은 실행 상태를 유지하기 위해 while True:를 사용해서 끝나지 않는 무한 루프로 동작

코루틴을 강제로 종료하고 싶다면 close 메서드를 사용

• 코루틴객체.close()

```
def number_coroutine():
    while True:
        x = (yield)
        print(x, end=' ')

co = number_coroutine()
next(co)

for i in range(20):
    co.send(i)

co.close()    # 코루틴 종료/.
```

41.3.1 GeneratorExit 예외 처리하기

코루틴 객체에서 close 메서드를 호출하면 코루틴이 종료될 때 GeneratorExit 예외가 발생하므로 내가 가

예외를 처리하면 코루틴의 종료 시점을 알 수 있습니다.

```
def number_coroutine():
    try:
        while True:
            x = (yield)
            print(x, end=' ')
    except GeneratorExit:    # 코루틴이 종료 될 때 GeneratorExit 예외 발생
        print()
        print('코루틴 종료')

co = number_coroutine()
next(co)

for i in range(20):
    co.send(i)

co.close()
```

41.3.2 코루틴 안에서 예외 발생시키기

코루틴 안에 예외를 발생시켜서 코루틴을 종료시키기

코루틴 안에 예외를 발생 시킬 때는 throw 메서드를 사용

• 코루틴객체.throw(예외이름, 에러메시지)

```
def sum_coroutine():
    try:
        total = 0
        while True:
            x = (yield)
            total += x
    except RuntimeError as e:
        print(e)
        yield total    # 코루틴 바깥으로 값 전달

co = sum_coroutine()
next(co)

for i in range(20):
    co.send(i)

print(co.throw(RuntimeError, '예외로 코루틴 끝내기')) # 190
                                                    # 코루틴의 except에서 yield로 전달받은 값
```