Unit 33.

Status	완료
담당자	
마감일	
완료일	@2022년 11월 28일

Unit 33. 클로저 사용하기

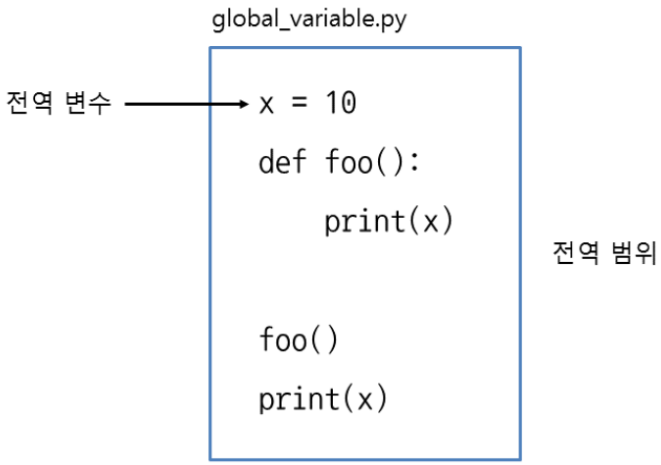
33.1 변수의 사용 범위 알아보기

- 파이썬 스크립트에서 변수를 만들면 다음과 같이 함수 안에서도 사용할 수 있다.

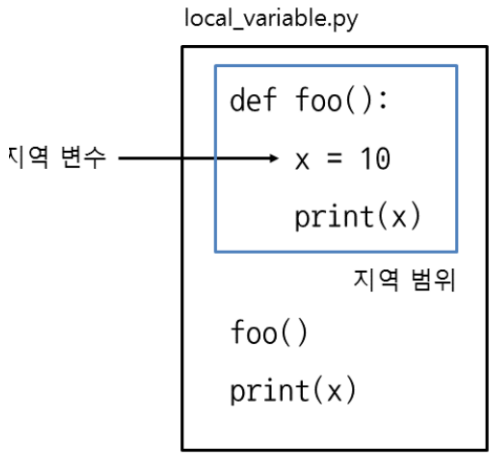
```
x = 10          # 전역 변수
def foo():
    print(x)    # 전역 변수 출력

foo()
print(x)        # 전역 변수 출력
```

- 함수를 포함하여 스크립트 전체에서 접근할 수 있는 변수를 전역 변수(global variable)라고 부른다
- 전역 변수에 접근할 수 있는 범위를 전역 범위라고 한다.



- 지역 변수는 변수를 만든 함수 안에서만 접근할 수 있고, 함수 바깥에서는 접근할 수 없습니다. 특히 지역 변수를 접근할 수 있는 범위를 지역 범위(local scope)라고 한다.



33.1.1 함수 안에서 전역 변수 변경하기

```
x = 10          # 전역 변수
def foo():
    x = 20      # x는 foo의 지역 변수
    print(x)    # foo의 지역 변수 출력

foo()
print(x)        # 전역 변수 출력
```

- 겉으로 보기에는 foo 안의 x는 전역 변수인 것 같지만 실제로는 foo의 지역 변수다. 즉, 전역 변수 x가 있고, foo에서 지역 변수 x를 새로 만들게 된다. 이 둘은 이름만 같을 뿐 서로 다른 변수다.
- 함수 안에서 전역 변수의 값을 변경하려면 global 키워드를 사용해야 한다.
- **global 전역변수**

```
x = 10          # 전역 변수
def foo():
    global x    # 전역 변수 x를 사용하겠다고 설정
    x = 20      # x는 전역 변수
    print(x)    # 전역 변수 출력

foo()
print(x)        # 전역 변수 출력
```

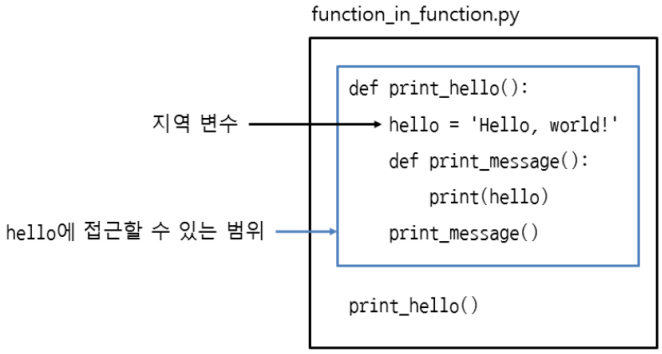
33.2 함수 안에서 함수 만들기

```
def 함수이름1():
    코드
    def 함수이름2():
        코드
```

33.2.1 지역 변수의 범위

- 안쪽 함수 print_message에서는 바깥쪽 함수 print_hello의 지역 변수 hello를 사용할 수 있다.
- 바깥쪽 함수의 지역변수는 그 안에 속한 모든 함수에서 접근할 수 있다.

```
def print_hello():
    hello = 'Hello, world!'
    def print_message():
        print(hello)    # 바깥쪽 함수의 지역 변수를 사용
```



함수의 지역 변수에 접근할 수 있는 범위

33.2.2 지역 변수 변경하기

- 바깥쪽 함수의 지역 변수를 안쪽 함수에서 변경

```
def A():
    x = 10        # A의 지역 변수 x
    def B():
        x = 20    # x에 20 할당

    B()
    print(x)      # A의 지역 변수 x 출력

A()
```

- 파이썬에서는 함수에서 변수를 만들면 항상 현재 함수의 지역 변수가 된다.

```
def A():
    x = 10        # A의 지역 변수 x
    def B():
        x = 20    # B의 지역 변수 x를 새로 만듦
```

- 현재 함수의 바깥쪽에 있는 지역 변수의 값을 변경하려면 nonlocal 키워드를 사용해야 한다.
- **nonlocal 지역변수**

```
def A():
    x = 10        # A의 지역 변수 x
    def B():
        nonlocal x    # 현재 함수의 바깥쪽에 있는 지역 변수 사용
        x = 20        # A의 지역 변수 x에 20 할당

    B()
    print(x)      # A의 지역 변수 x 출력

A()
```

33.2.3 nonlocal이 변수를 찾는 순서

- nonlocal은 현재 함수의 바깥쪽에 있는 지역 변수를 찾을 때 가장 가까운 함수부터 먼저 찾는다.

```
def A():
    x = 10
    y = 100
    def B():
        x = 20
        def C():
            nonlocal x
            nonlocal y
            x = x + 30
            y = y + 300
            print(x)
            print(y)
        C()
    B()

A()
```

33.2.4 global로 전역 변수 사용하기

- 함수가 몇 단계든 상관없이 global 키워드를 사용하면 무조건 전역 변수를 사용하게 된다.

```
x = 1
def A():
    x = 10
    def B():
        x = 20
        def C():
            global x
            x = x + 30
            print(x)
        C()
    B()

A()
```

전역 변수는 코드가 복잡해졌을 때 변수의 값을 어디서 바꾸는지 알기가 힘들기 때문에 전역 변수는 가급적 사용하지 않는 것이 좋다.

33.3 클로저 사용하기

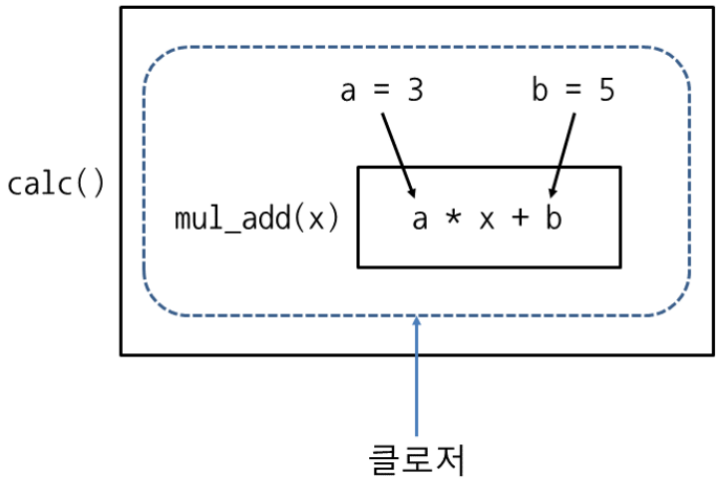
```
def calc():
    a = 3
    b = 5
    def mul_add(x):
        return a * x + b    # 함수 바깥쪽에 있는 지역 변수 a, b를 사용하여 계산
    return mul_add          # mul_add 함수를 반환

c = calc()
print(c(1), c(2), c(3), c(4), c(5))
```

- 함수 mul_add를 만든 뒤에는 이 함수를 바로 호출하지 않고 return으로 함수 자체를 반환(함수를 반환할때는 이름만 반환해야 한다 ( ) 붙이면 안된다.

33.3.1 클로저의 개념

- 클로저는 지역 변수와 코드를 묶어서 사용하고 싶을 때 활용합니다. 또한, 클로저에 속한 지역 변수는 바깥에서 직접 접근할 수 없으므로 데이터를 숨기고 싶을 때 활용한다.



33.3.1 lambda로 클로저 만들기

```
def calc():
    a = 3
    b = 5
    return lambda x: a * x + b    # 람다 표현식을 반환

c = calc()
print(c(1), c(2), c(3), c(4), c(5))
```

람다는 이름이 없는 익명 함수를 뜻하고, 클로저는 함수를 둘러싼 환경을 유지했다가 나중에 다시 사용하는 함수를 뜻한다.

33.3.2 클로저의 지역 변수 변경하기

- 클로저의 지역 변수를 변경하고 싶다면 nonlocal을 사용하면 된다.

```
def calc():
    a = 3
    b = 5
    total = 0
    def mul_add(x):
        nonlocal total
        total = total + a * x + b
        print(total)
    return mul_add

c = calc()
c(1)
c(2)
c(3)
```