전역 이름공간에 정의되어 프로그램 어디서든 부를 수 있는 이름을 **전역변수(global variable)**라고 한다. 함수 밖에서 변수를 정의하면 전역변수가 된다. 반면에 지역 이름공간에 정의되어 그 문맥 속에서만 부를 수 있는 이름을 **지역변수(local variable)**라고 한다. 모든 함수는 자신만의 지역 이름공간을 가지며, 함수 속에서 작성한 변수는 그 함수의 지역변수가 된다.

변수 - [지역변수 (local) / 전역변수 (global)]

3.4.1 지역변수는 함수만의 것

함수 내에서 만들어진 변수가 '지역변수'임.

함수의 지역변수는 함수가 실행되는 동안에만 존재한다. 각 함수가 호출되어 실행될 때 만들어지고, 함수의 실행이 끝나면 모두 삭제된다. 그래서 지역변수는 그 변수가 속한 함수의 밖이나 다른 함수에서는 부를 수 없다. 매개변수도 함수 안에 정의되므로 지역변수다. 다음 예제에서 전역변수와 지역변수를 구별해 보자.

return 으로 끝나면 해당 지역 변수들은 사라짐.

핵심!!!!!

코드 3-11 전역변수와 지역변수가 함께 사용된 프로그램

```
seconds_per_minute = 60  # 1분은 60초 ❶
def minutes_to_seconds(minutes):
    """분을 입력받아 같은 시간만큼의 초를 반환한다."""
    seconds = minutes * seconds_per_minute  # ❷
    return seconds
print(minutes_to_seconds(3))  # 화면에 180이 출력된다
print(seconds)  # ❸ 오류! 함수 밖에서 지역변수를 불렀다
```

전역변수

지역변수

실행 결과:

```
180
NameError: name 'seconds' is not defined
```

❶에서 정의한, 1분이 몇 초인지 나타내는 seconds_per_minute 변수는 프로그램 어디에서든 사용할 수 있는 전역변수다. minutes_to_seconds() 함수 안의 ❷에서도 이 변수를 읽고 있다. 반면, 분을 입력받는 매개변수 minutes 와 함수 안에서 중간 계산 결과를 저장하는 변수 seconds 는 지역변수다. ❸과 같이 seconds 변수를 그 변수가 존재하는 문맥(함수) 밖에서 읽으려고 하면, 문맥 밖에는 그 변수가 존재하지 않기 때문에 이름 오류가 발생한다.

중요!!!

전역변수는 어디에서나 읽을 수 있지만, 함수 안에서 전역변수에 새로운 값을 대입하는 것은 금지된다. (잠시후 설명할 global 문을 사용하면 예외적으로 가능해진다.) 표 3-1은 지역변수와 전역변수의 접근 조건을 표로 정리한 것이다.