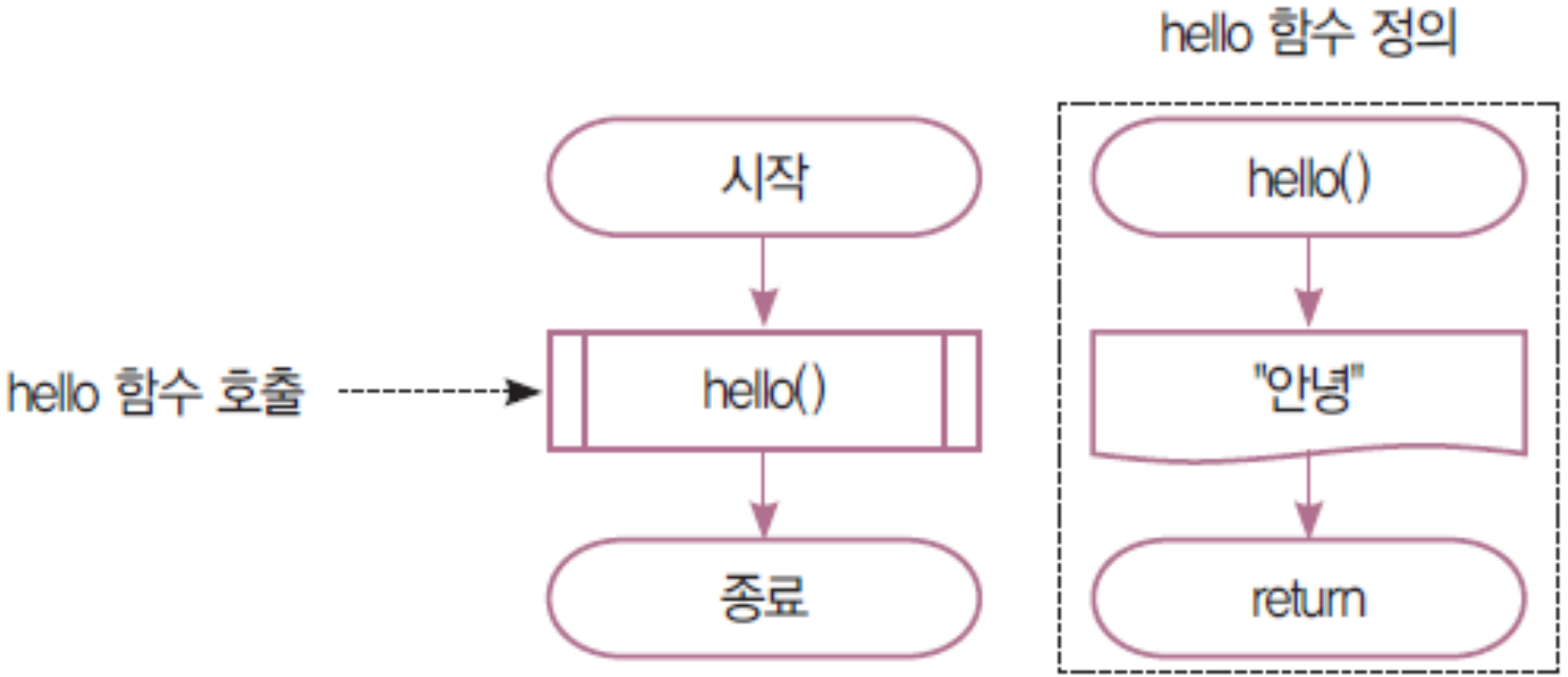
Flow Diagram V

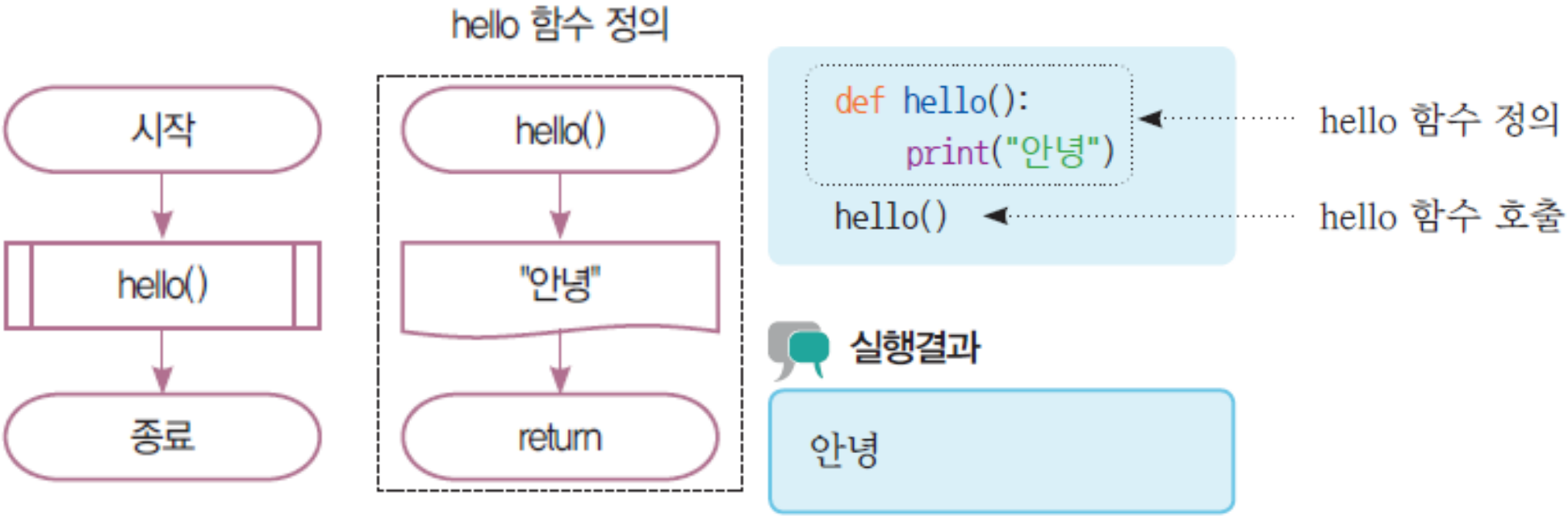
함수

함수의 개요

- 함수: 특정 동작을 수행하는 일정 코드
- 여러 명령어들을 하나의 단위로 묶어 한 가지 기능을 수행함



함수의 개요


hello 함수 정의
시작
hello()
종료
hello()
"안녕"
return
def hello():
print("안녕")
hello()
hello 함수 정의
hello 함수 호출
실행결과
안녕

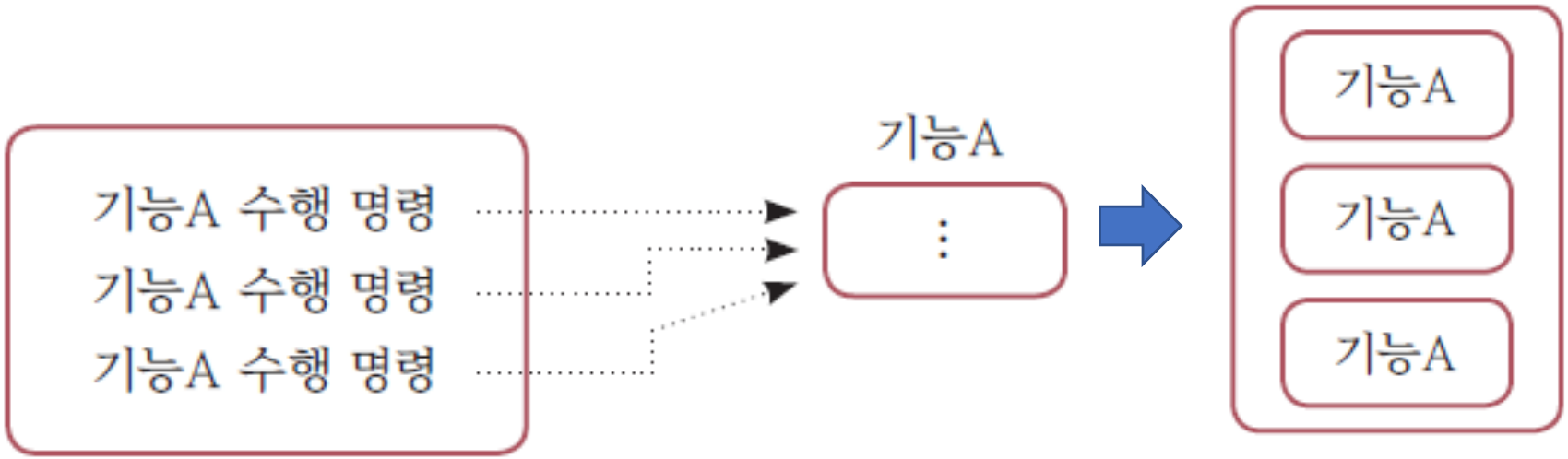
함수는 언제 쓰는가?

- 반복적으로 활용되는 부분을 별도의 실행 단위로 만들어 가독성을 높일 때 사용
- 세부 내용 보다 전체적인 구성을 볼 수 있도록 함
- 어디서 오류가 발생했는지 검사가 쉬워짐

- 사용 예: 특정 조건에 거북이로 이름을 쓰는 작업
  - 이름을 쓰는 데 필요한 문장의 수: 40
  - 특정 조건 발생 횟수: 50
  - 함수 없이 코드를 작성할 때 최소 코드의 길이 40줄 x 50번 = 200 줄

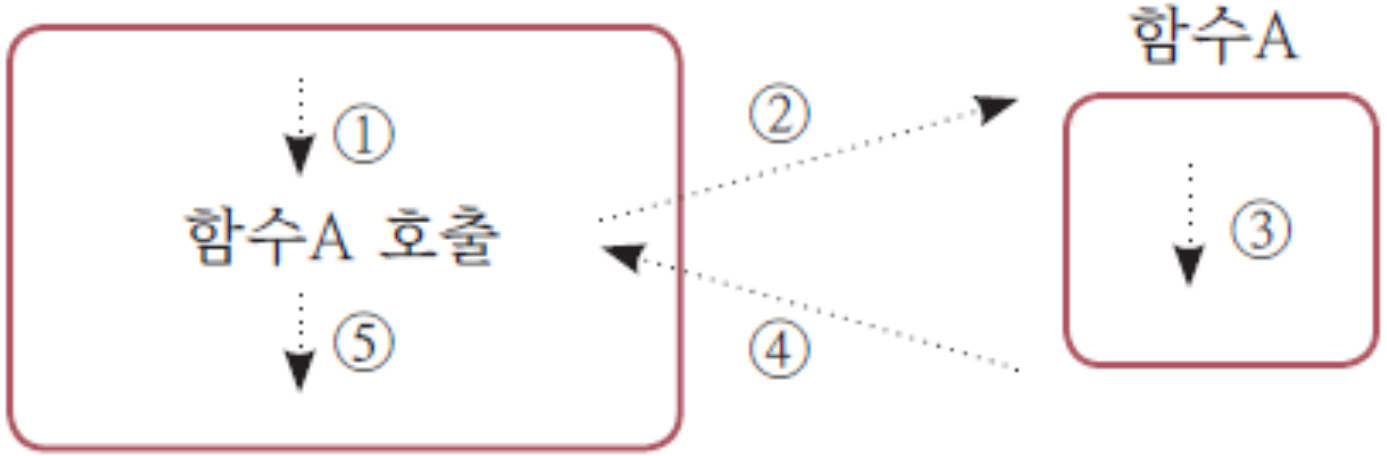
함수는 언제 쓰는가?

- 사용 예: 특정 조건에 거북이로 이름을 쓰는 작업
  - 함수로 만드는데 드는 추가 코드 수: 1 (총 41줄)
  - 특정 조건에 함수를 호출하는 횟수: 50
  - 함수로 코드를 자성할 때 최소 코드의 길이 41줄 + 50줄 = 91줄

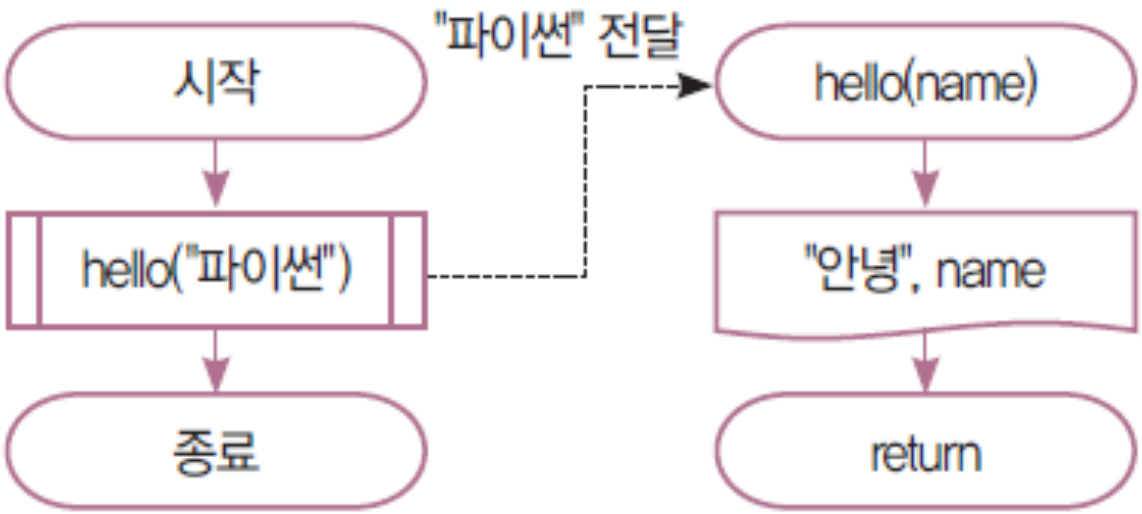


함수의 실행 흐름

- 함수는 하나의 기능을 수행하도록 짜여진 코드의 실행 순서임
- 함수에 이름이 있어 변수처럼 재사용이 쉬움
- 함수 내에서 다른 함수를 호출하는 것도 가능



첫 예제 다시 보기

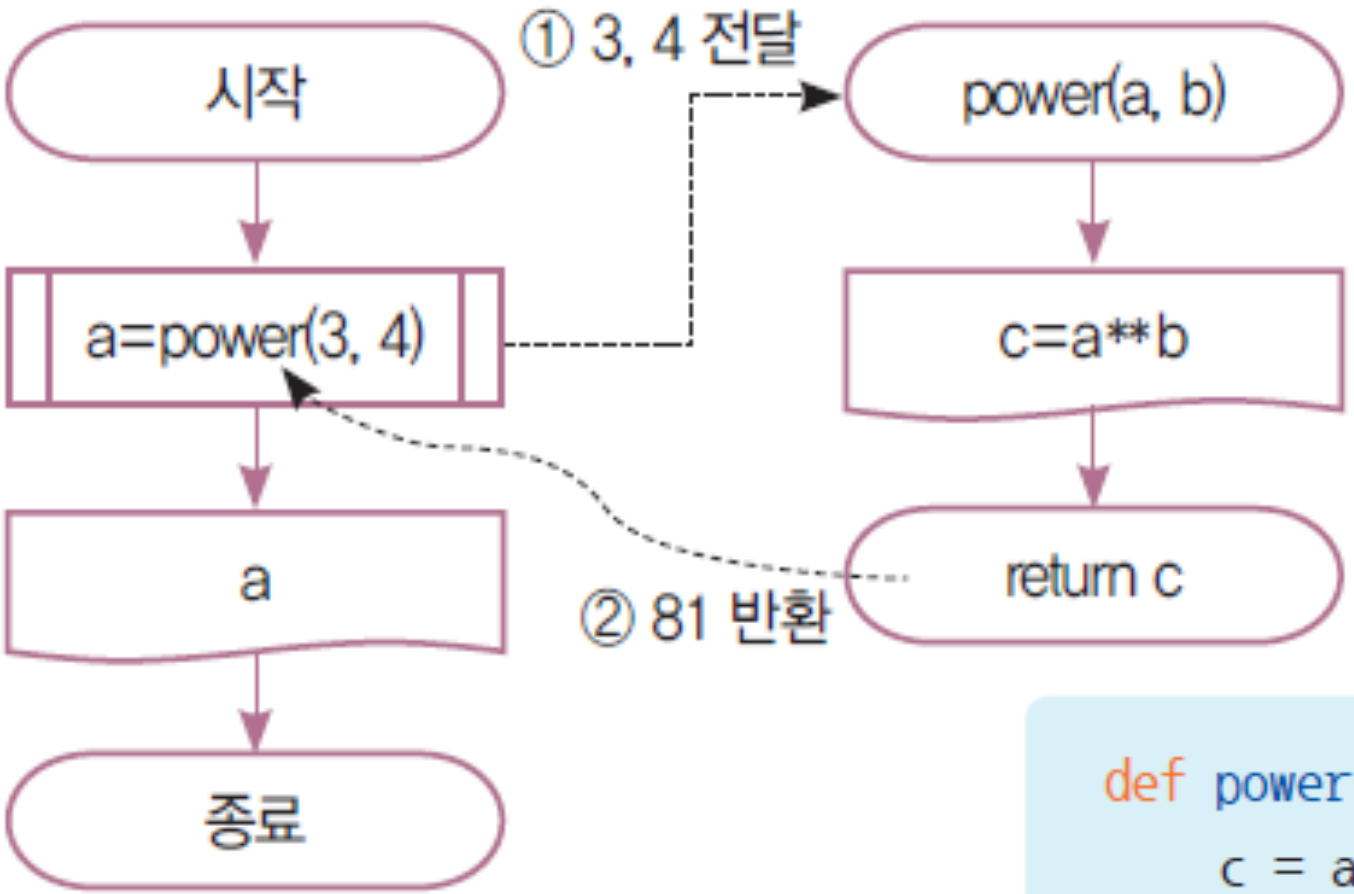



```
def hello(name):
    print("안녕", name)
hello("파이썬")
```

실행결과

```
안녕 파이썬
```

예제: 승수 구하기



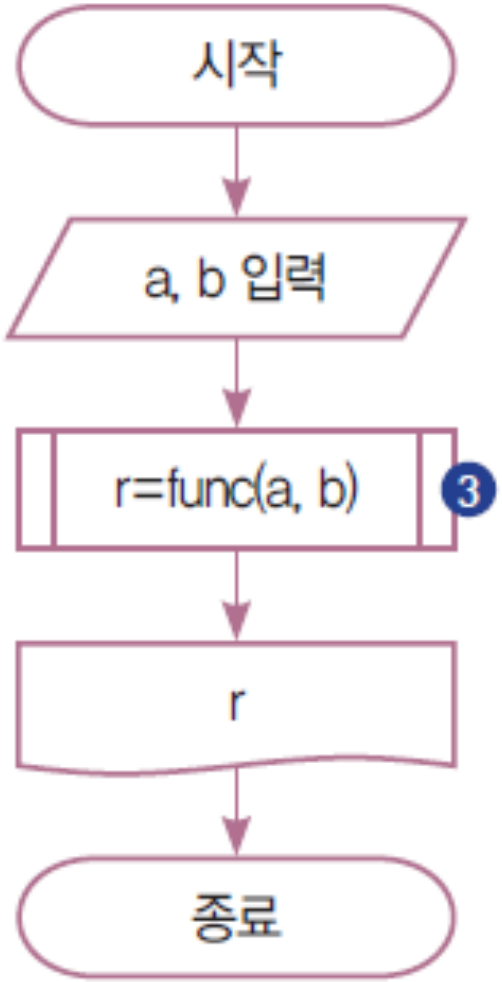
실행결과

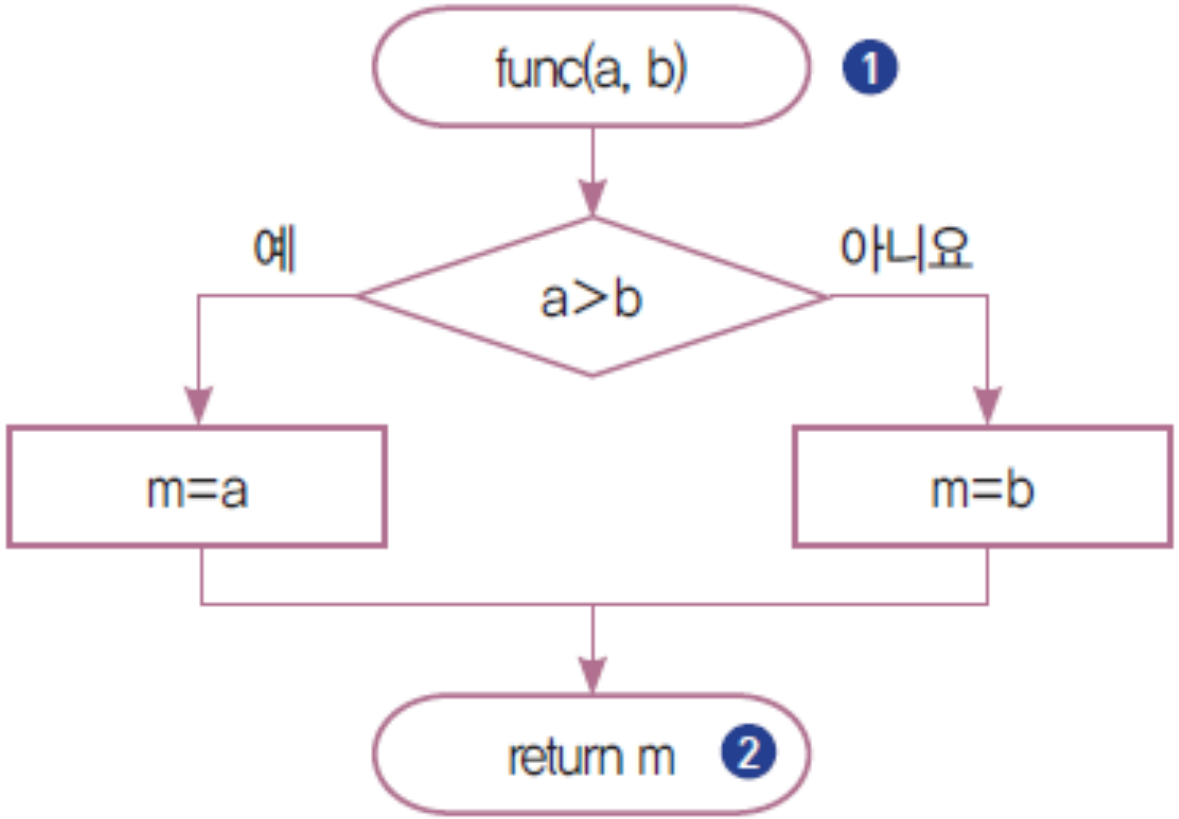
```
81
```

```
def power(a,b):
    c = a**b
    return c
a = power(3,4)
print(a)
```

예제: 함수를 이용해 두 수 중 큰 수 찾기

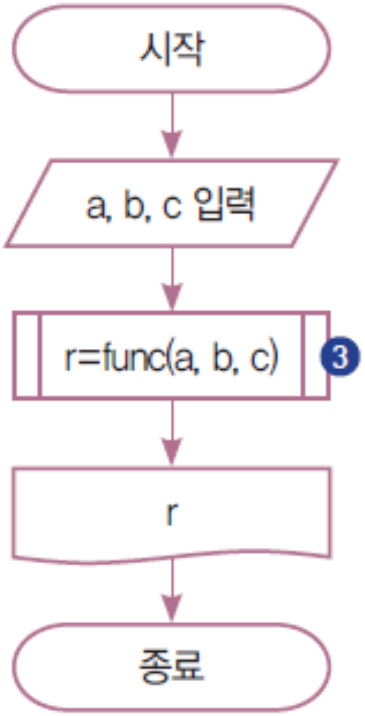
- LAB: 프로그램으로 표현해보자

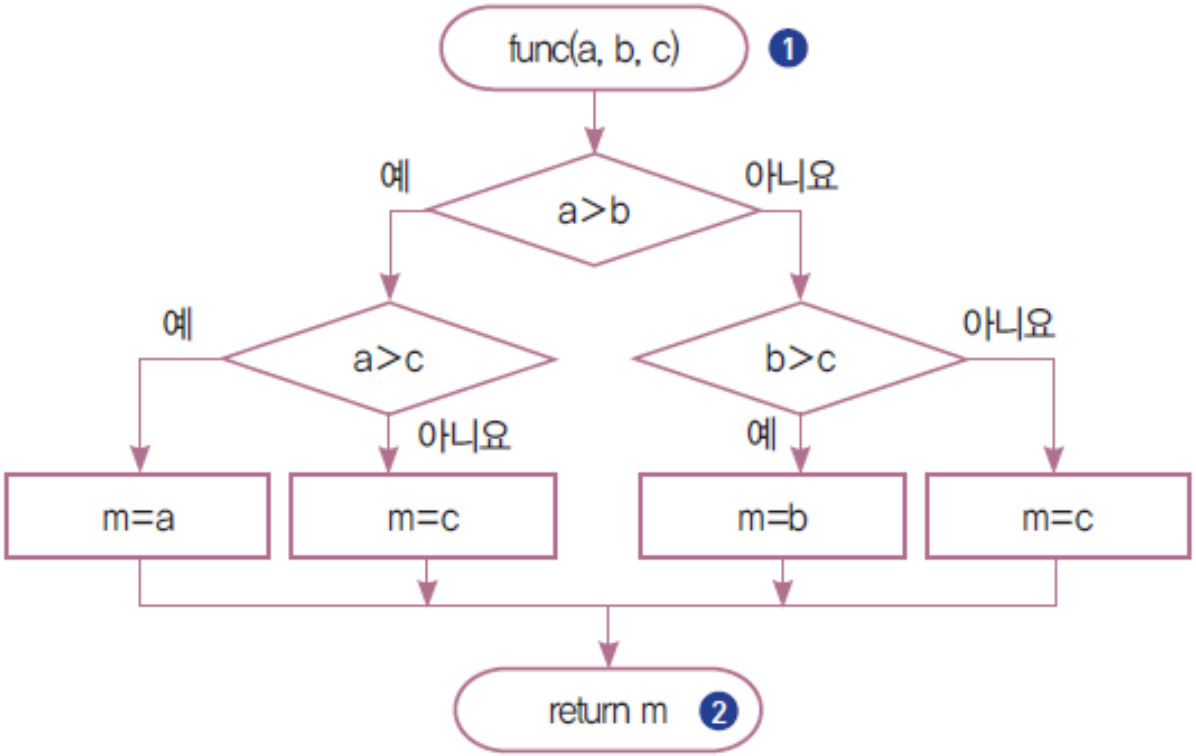




실습: 세수 중 큰 수 찾기

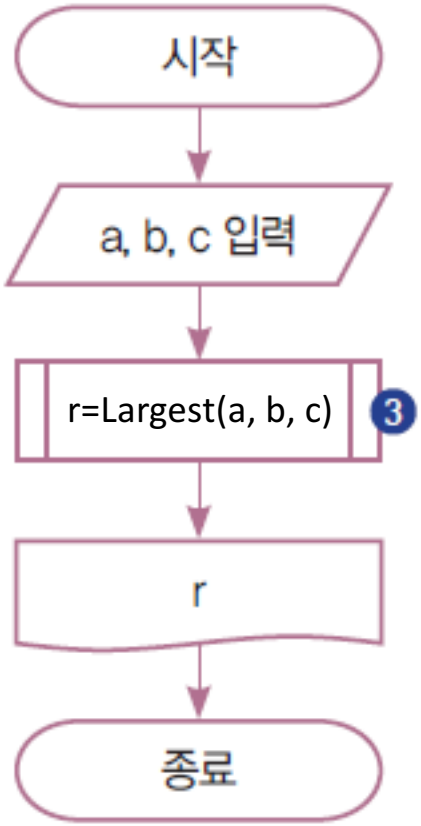
- 2명이 하나의 그룹을 구성한다.
- 함수의 입력을 이 몇 개고 무엇인지 논의한다.
- 함수 안에서의 판별문을 어떻게 구성할지 논의한다.
- 최대값을 어떻게 판별해야 하는지 논의한다.
- 리턴값을 무엇을 정의해야 하는지 논의한다.
- 순서도를 바탕으로 코드를 작성한다.





실습: 세수 중 큰 수 찾기

- 함수를 재활용하는 방법은?



Largest Larger

재귀 함수

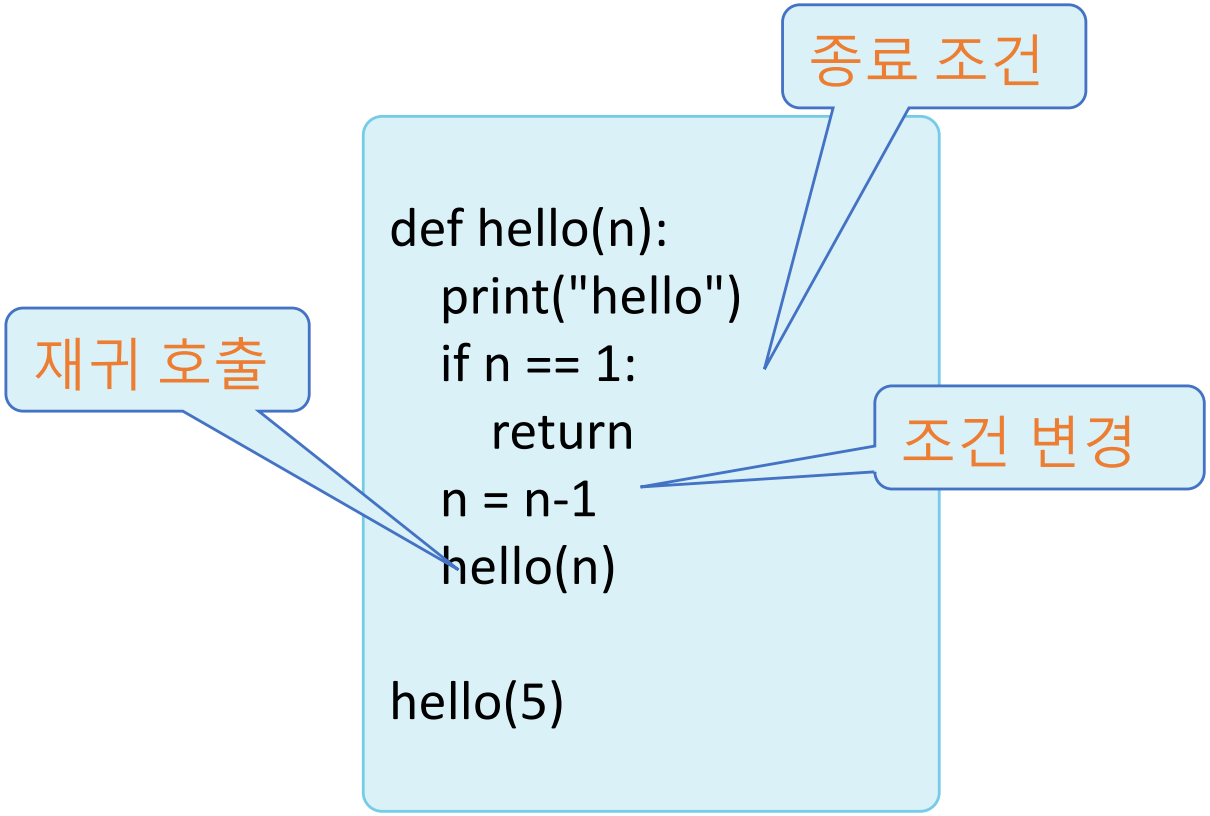
- 자기 자신을 호출하는 함수

- 아래 코드의 결과는?

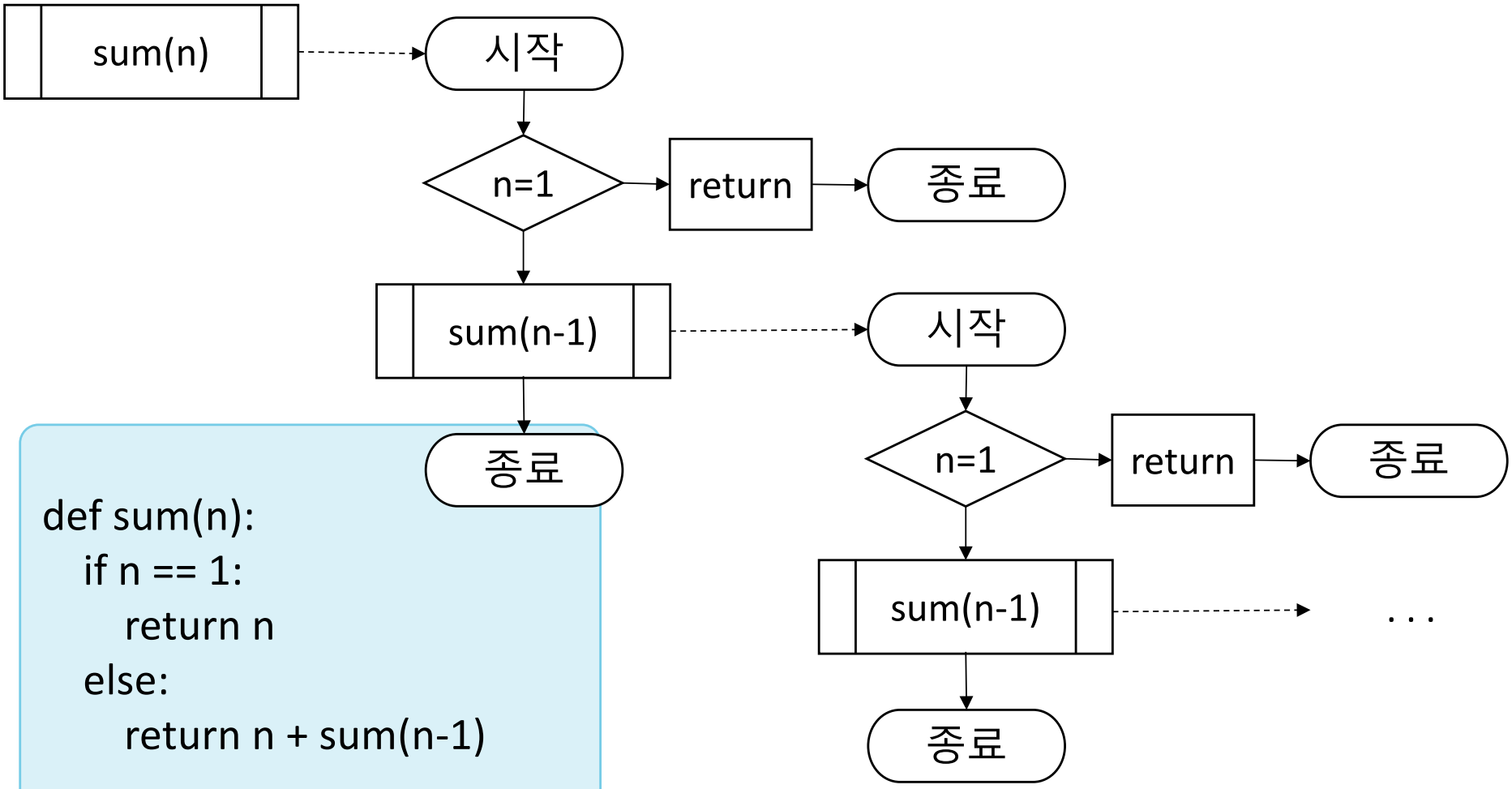
```
def hello():
    print("hello")
    hello()

hello()
```

재귀 함수

- 보통은 무한반복을 원하지 않으므로, 종료 조건을 넣는다.



재귀 함수 예제: 1 부터 10까지의 합




```
def sum(n):
    if n == 1:
        return n
    else:
        return n + sum(n-1)

c = sum(10)
print(c)
```

LAB: 코드 작성 실습: 재귀 함수 정의

- 연습문제: n! 은 1*2*3…*n까지 모든 수를 곱한 결과이다.
- 이를 재귀 함수를 이용해서 작성해 보도록 하자.
    1. 반복되는 부분을 명시하자.
    2. 종료 조건을 명시하자.
    3. 명시한 부분을 재귀 함수를 위한 순서도를 작성해서 체크하자.
    4. 순서도를 코드로 명기하자
        1. 함수를 정의하자
        2. 함수를 호출하자