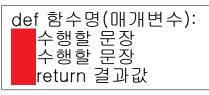
- 어떠한 값을 입력하면 정의된 절차에 따라 일을 수행한 후 결과물을 내는 것
 - 수학에서의 함수 : y = x + 3
 - 파이썬에서의 함수



- 함수를 사용하는 이유
 - 반복되는 코드의 수를 줄일 수 있음
 - 누군가가 만들어 놓은 기능을 손쉽게 활용할 수 있음
 - 팀 단위로 협업 / 분업 가능
 - 함수의 코드만 수정하면 실제 사용되는 여러 부분들이 자동으로 수정되므로 유지보수가 용이함

● 두 수를 입력받아 합을 구하는 함수

```
def plus(a, b):
    return a + b

x = 10
y = 5

result = plus(x, y) # result = plus(10, 5)
print(result)
```

- plus 함수로 정의된 변수 a, b → 매개변수(parameter), 입력값
- plus 함수를 부르면서 입력해주는 변수 x, y → 인자(arguments)
- return a + b → 반환 / 출력 / 결과값
- 함수의 형태는 입력값과 결과값의 존재 여부에 따라서 4가지로 나누어짐
 - 입력값 O, 결과값 O: 일반적으로 많이 사용되는 형태
 - 입력값 O, 결과값 X
 - 입력값 X, 결과값 O
 - 입력값 X, 결과값 X

● 입력값 O, 결과값 O

```
def total(a, b, c): # 합을 구하는 함수
return a + b + c

def average(num, count): # 평균을 구하는 함수
return num / count

num = total(90, 80, 70) # 합을 계산한 결과 반환
avg = average(num, 3) # 평균을 계산한 결과 반환
print(num, avg)
```

● 입력값 O, 결과값 X

```
def total(a, b, c): # 합을 구하는 함수
print(a + b + c) # return 없음

def average(num, count): # 평균을 구하는 함수
print(num / count) # return 없음

num = total(90, 80, 70) # num = None
avg = average(240, 3) # num을 사용할 수 없으므로 직접 계산
# avg = None

print(num, avg) # None, None
```

● 입력값 X, 결과값 O

```
import random

def getNumber(): # 0 ~ 10 사이 임의의 숫자를 구하는 함수
  return random.randint(0, 10)

num = getNumber();
print(num)
```

● 입력값 X, 결과값 X

```
def hello(): # 화면에 Hello 라는 문자열만 출력하는 함수 print('Hello') hello()
```

● 함수를 호출 할 때 매개변수를 지정하여 호출

```
def total(a, b, c):
  return a + b + c

def average(num, count):
  return num / count

num = total(a=90, b=80, c=70) # 매개변수 지정 후 값 입력
avg = average(count=3, num=num) # 순서 상관없이 입력 가능

print(num, avg)
```

- 입력되는 값의 개수에 따라 동적으로 처리
 - 입력값을 모두 더하는 함수 (1, 2, 3 이면 6, 1, 2, 3, 4, 5 이면 15)

```
def total(*args): # Tuple 형태로 저장
total = 0

for n in args:
total += n

return total

num = total(1, 2, 3)
print(num)

num = total(1, 2, 3, 4, 5)
print(num)

6515
```

- 입력되는 값의 개수에 따라 동적으로 처리
 - 일반 입력값과 가변 입력값 동시 사용

```
def calculator(operator, *args):
  if operator == '+':
    result = 0
    for n in args:
      result += n
  elif operator == '*':
    result = 1
    for n in args:
      result *= n
  return result
num = calculator('+', 1, 2, 3, 4)
print(num)
num = calculator('*', 1, 2, 3, 4)
                                                         10
print(num)
                                                         24
```

- 키워드 파라미터
 - 매개변수명이 지정된 입력값을 개수에 따라 동적으로 처리

```
def func(**kwargs): # Dictionary 형태로 저장
print(kwargs)

func(a=1, b=2, c=3)
{'a': 1, 'b': 2, 'c': 3}
```

- 인자를 key=value 형태로 넘겨주고 함수 내부에서는 Dictionary 형태로 활용
- 가변 인자 방식과 키워드 파라미터 방식을 함께 사용

- 결과값의 여러가지 반환 형태
 - 일반적인 형태

```
def funcA():
    return 'a'

retA = funcA()
    print(retA)
    a
```

- 2개 이상의 값 반환

```
def funcB():
  return 'a', 'b' # Tuple 형태로 반환

retB = funcB() # Tuple 형태로 저장
  print(retB) ('a', 'b')

retB1, retB2 = funcB() # 각 변수에 따로 저장
  print(retB1, retB2) a b
```

- 결과값의 여러가지 반환 형태
 - return 키워드를 2번 이상 사용 (사용 불가)

```
def funcC():
    return 'a'
    return 'b' # 실행되지 않음

retC = funcC() # 문자열 'a' 저장
    print(retC)
```

- return 키워드 단독 사용

```
def divide(a, b):

if b == 0: # 나누는 수(제수)가 0 인 경우
  print('0 으로 나눌 수 없습니다.')
  return # 강제 종료

return a / b

result = divide(4, 2) # 정상 실행, result = 2.0
print(result)

result = divide(4, 0) # 실행 도중 종료, result = None
print(result)
```

● 매개변수 기본값 지정

```
def setInfo(name, age, isPass=True): # 매개변수의 마지막에 위치 print(name, age, isPass)

setInfo('ggoreb', 20)
setInfo('ggoreb', 20, False)

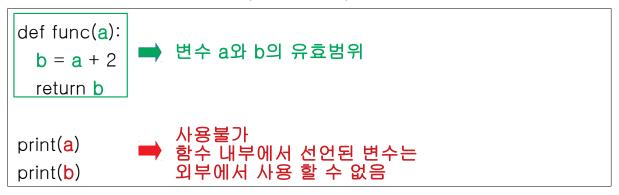
ggoreb 20 True
ggoreb 20 False
```

- 주의사항



■ 변수의 유효범위

● 함수 안에서 변수 선언 (지역변수)



● 함수 밖에서 변수 선언 (전역변수)

```
outer_a = 0

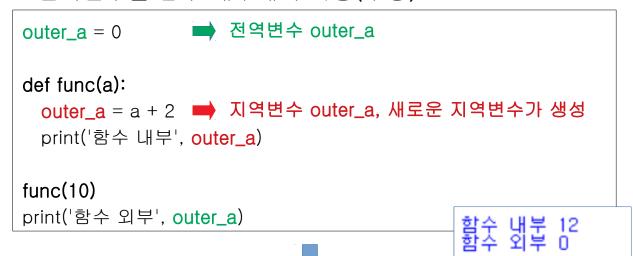
def func(a):
  b = a + 2
  return b

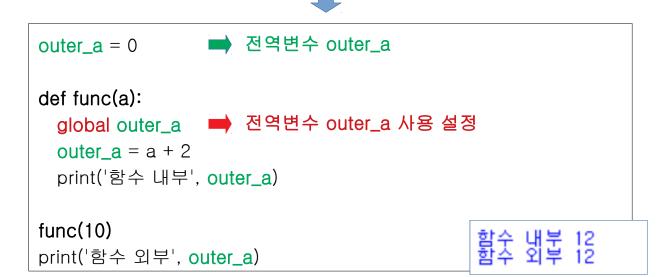
世수 outer_a의 유효범위

outer_a = func(10)

print(outer_a)
```

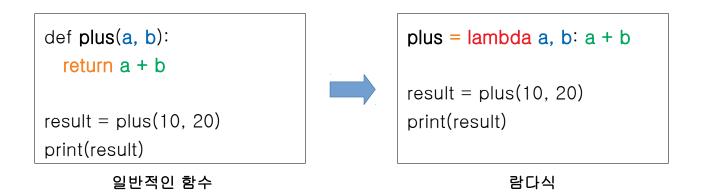
- 변수의 유효범위
 - 함수 밖에서 변수 선언 (전역변수)
 - 전역변수를 함수 내부에서 사용(수정)





lambda

- 함수를 생성할 때 사용
- def 예약어와 동일한 역할
 - 함수를 간결하게 한줄로 만들때
 - 함수가 함수를 반환할때
- 기본 형태
 - lambda 인자1, 인자2, ··· , 인자n : 표현식



lambda

● List에 람다식 저장 후 사용

```
lambda_list = [ lambda a, b: a + b, lambda a, b: a * b ]
print(lambda_list)

result1 = lambda_list[0](3, 4)
print(result1)

result2 = lambda_list[1](3, 4)
print(result2)

[<function < lambda> at 0x0000002A61B9932F0>, <function < lambda> at 0x0000002A61B993378>]
7
12
```

● 함수가 함수를 반환

```
def makePrdCode(name):
  return lambda score: name + str(score)
prd1 = makePrdCode('A')
code1 = prd1(100)
code2 = prd1(200)
print(code1)
print(code2)
prd2 = makePrdCode('B')
                                                         A100
                                                         A200
code3 = prd2(100)
code4 = prd2(200)
                                                         B100
print(code3)
                                                         B200
print(code4)
```