x(1, c = 3, b = 3) -> 정상작동

x(a = 1, 3, 3) -> 오류발생 (keyword 방식 뒤에는 일반 arguments가[positional 방식] 올 수 없음.)

def a(a, b, c = 3):

return a + b + c

* 함수 정의를 할 때에는 default값을 주어 인자로 입력을 받지 않고도 사용 가능

def s(a, \*, b):

return a + b

* \* 이후는 keyword방식을 사용해 호출해야함
* s(1, 2) -> 오류 발생
* s(1, b = 2) -> 정상작동

def x(\*a):

return a

* 인자의 개수에 상관없이 입력 받을 수 있음

def z(\*\*a):

return a

* Keyword 방식으로 원하는 만큼 인자 호출 가능,
* z(a = 1, b = 2, c = 3)
* {'a': 1, 'b': 2, 'c': 3}

()를 붙일 수 있는애들

* Type이 function,
* Class인 애들
* Class 안에 선언된 \_\_call\_\_ Duck typing을 통해서 붙일 수 없는 것에도 붙이게 만들 수 있음

Local -> Enclosing -> Global -> Bulitin

Nonlocal

* D

a = 1

def x():

a = 2

def y():

nonlocal a

a = a + 1

return a

return y()

nonlocal을 쓰면 자신을 감싸고 있는 애 거를 사용

Higher-order-function -> 인자로 함수를 입력받아 함수를 리턴함

Python은 모든 연산자에 대한 그에 상당한 메소드 지원

@staticmethod

* ㅇ

class A:

@staticmethod

def t():

print('A')

staticmethod 이름만 빌려와서 사용함 (범용적으로 사용할 때에 사용되는 것) 클래스와 연동 x 상관 x 다른데에 만들어놓고 연결만 시켜놓은 것

@property method를 attribute로써(변수처럼) 사용 가능하게 함