persons = [('김기수', 30), ('홍대길', 35), ('강찬수', 25)]

mydict = dict(persons)

mydict.update({'홍대길':33,'강찬수':26})

함수는 문들을 하나로 묶는 코드블록

반복적으로 실행 가능

기능 중심으로 프로그램 구성

함수는 동사 구?????????????

코드 집합이어서 :꼭 써줘야함

return에 값 안쓰면 none return

끝나면 아래꺼가 실행됨?

함수는 계층 구조. 고급 함수들을 더 위에 정의

여러 개를 return하려면 tuple하면됨

def swap(a,b):

return b,a

a=10

b=20

실제로 변수 a, b값이 바뀌는건 아님

return값을 넣어줘야함

c는 call by reference 없음

call by reference 효과 내려면 포인터 활용

pass by pointer

call by reference는 원본에서 함수의 내용을 변경 가능한데

파이썬은 꼭 그렇지 않음. 그렇다고 call by value도 아님

call by reference에 가까움

call by object reference 또는 call by shareing 이라고 부름

int - 리터럴, 변경 불가능, 다이렉트

레퍼런스를 전달해도 원본 값을 바꿀 수가 없음

>>> def h(t):

print(t)

t=(1,2,3)

print(t)

>>> a=(5,6,7);h(a);a

(5, 6, 7)

(1, 2, 3)

(5, 6, 7)

레퍼런스를 복사받았을 때는 직후에는 같은 값을 가리키는데 t에다가 새로 넣으면 기존꺼는 덮어써지니깐 지워짐

리스트로 하면 바꿀 수 있음

>>> def g(t):

t[1]=10

>>> a=[1,2,3];g(a)

>>> print(a)

[1, 10, 3]

>>> def h(t):

t=[5,6,7]

>>> a=[1,2,3];h(a)

>>> print(a)

[1, 2, 3]

이 경우는 새로운 리스트를 만든거라 래퍼런스를 교체해줘서(a랑 연결 끊고 t에 새로운 레퍼런스 연결) a값이 안바뀜

함수가 안긑났는데 또 함수 정의할 수 있음

enclosing function local - 함수가 안끝났는데 또 다른 함수를 내포하는 또 다른 함수 영억

함수 안에 global 변수명 이렇게 전역변수 선언하면 함수 호출 후부터 선언한 전역변수 사용하겠다는거

>>> g=10

>>> def f():

a=g

g=20

return a

>>> f()

이러면 에러남

g를 지역변수로 취급해서

>>> def outer():

x=1

def inner():

nonlocal x

x=2

print('inner : ',x)

inner()

print('outter : ',x)

>>> outer()

inner : 2

outter : 2

이렇게하면 x가 지역변수가 아니라는걸 알려줌

오보로딩 효과도 있음

인수의 기본값 줘서

기본값 지정된 인수 오른쪽에는 기본값으로 하며ㅑㄴ

가변변수 ??????????????????????????

]

내장함수

사용자 정의 함수

3rd praty 제공

키와 벨류형식????

depth와 dimensinon 지원하지 않음

가변인수ㅗ아 정의되ㅣ않은 거 넣는다는거 가장 오른쪽