Lab7. Functions II and Set

CSED101 LAB

기타

List Comprehension, zip()

List Comprehension

- 리스트에 for 문을 사용하여 반복적으로 표현식을 실행해서 리스트 원소들을 정의하는 용법

[표현식 for 요소 in 시퀀스자료형 [if 조건문]]

```
>>> L = [ i for i in range(1, 6) ]
>>> L
[1, 2, 3, 4, 5]

>>> [ i ** 2 for i in range(1, 6) ]

>>> [ i for i in range(1, 6) if i % 2 == 0 ]
```

```
# 중첩 for 문
>>> L1 = [1, 2, 3]
>>> L2 = [3, 4, 5]

>>> [ x * y for x in L1 for y in L2 ]
```

- 한 줄 if ~ else : 조건문을 만족하면 A, 아니면 B를 수행

A if 조건문 else B

zip()

- 2개 이상의 리스트를, 각 리스트의 같은 인덱스 원소끼리 묶은 튜플을 원소로 하는 리스트를 만들어 줌

```
>>> list( zip([1,2,3], [4,5,6]) )    # 리스트
[(1, 4), (2, 5), (3, 6)]

>>> list( zip([1,2,3], [4,5,6], ['a','b','c']) )
[(1, 4, 'a'), (2, 5, 'b'), (3, 6, 'c')]

>>> list( zip((1,2,3), (4,5,6)) )    # 튜플
[(1, 4), (2, 5), (3, 6)]

>>> [ sum(x) for x in zip((1,2,3), (4,5,6)) ]
```

실습

- 아래와 같이 주어진 2개의 리스트로 딕셔너리를 만들어 출력하시오. 단, 순서는 다를 수 있음
- zip() 사용할 것

```
L1 = ['one', 'two', 'three', 'four']
L2 = [1, 2, 3, 4]
```

실행 결과)

```
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

함수 2

가변 매개변수, 디폴트 매개변수

가변 매개변수

- 매개변수가 몇 개인지 모를 경우

```
def 함수이름 (*매개변수) :
    수행할 문장
    …
```

- 예) 짝수 개수 구하기

```
def count_even(*n):
    cnt = 0
    for v in n:
        if v % 2 == 0:
            cnt += 1
    return cnt
```

```
>>> count_even(1,2,3,4,5)
```

```
def f1(a, *b):    # 정상
def f2(a, *b, c): # 에러
def f3(*a, *b):   # 에러
def f4(a, b, *c):
```

- 가변 매개변수 뒤에는 일반 매개변수가 올 수 없음
- 가변 매개변수는 하나만 사용 가능

실습 – 가변 매개변수

- 1개 이상의 2차원 벡터들을 전달받아 벡터들의 합을 구하여 반환하는 함수를 작성하시오.

```
def vector_sum(vector, *vectors):
    pass
```

```
v1=[0, 1]
v2=[0.5, 0.5]
v3=[1, 0]
v4=[6, 4]
v5=[3.13, 2.72]
m1 = vector_sum(v1, v2, v3)
m2 = vector_sum(v1, v2, v3, v4)
m3 = vector_sum(v3, v5)
print(m1,m2,m3)

# [1.5, 1.5], [7.5, 5.5], [4.13, 2.72]
```

디폴트 매개변수

- 매개변수의 기본값을 지정해줄 수 있다
  - 함수 호출 시, 해당 매개변수에 대응되는 값을 주지 않으면 정의할 때 지정한 기본값을 가짐

```
def print_name(first, second='Kim'): #second 변수 기본값 'Kim'
    print ('My name is', first, second + '.')

#함수 호출
print_name('Gildong', 'Hong')
print_name('Gildong') #second 변수에 대응되는 값을 주지 않았으므로
                      #정의할 때 지정한 'Kim'을 가짐
```

디폴트 매개변수

- 앞 매개변수가 디폴트 값을 가지면, 뒤에 오는 매개변수는 반드시 디폴트 값을 가져야 함

```
def print_name(first, second='Kim'): # 정상

def print_name(first='Kim', second): # 에러 발생

def print_name(first, second, third='Kim'):    # 정상

def print_name(first, second='Kim', third='M') # 정상
```

키워드 매개변수

- 함수 호출 시, 해당 매개변수 이름을 명시적으로 지정해서 전달

```
def calc(x, y=0, z=0) :
    return x+y+z

result = calc(y=20, x=10, z=30)  # 매개변수 순서 상관 없음
print(result)

>>> calc(y=20, x=10)      # 정상
>>> calc(10, y=30, z=20) # 정상
>>> calc(10, 30, y=20)   # 에러 발생
```

```
<Example in built-in>
print("Output", end=" ")
sorted_list = sorted(l, reverse=True)
```

실습 – 디폴트 매개변수

- 숫자로 구성된 리스트 2개를 받아서 이 두 리스트를 합친 후에 정렬한 새로운 리스트를 반환하는 함수를 작성하시오.
- 단, 매개변수의 값이 주어지지 않으면, [0]이 기본적으로 주어진다.

```
def merge_list(??):
    pass
```

```
l = [3, 5, 9, 1, 2]
ml1 = merge_list(l,[2,1])
ml2 = merge_list([6,9,4])
ml3 = merge_list()
print(ml1) # [1, 1, 2, 2, 3, 5, 9]
print(ml2) # [0, 4, 6, 9]
print(ml3) # [0, 0]
```

집합

집합

- 집합(set)
  - 순서 없이 항목을 저장, 항목이 중복 될 수 없음

```
>>> s1 = set([1,2,3,3,3,4,2])  # 리스트기반
>>> s1
{1, 2, 3, 4}
>>> s2 = {3,4,5}

>>> word = "Hello" * 2
>>> word
'HelloHello'
>>> word_list = list(word)
>>> word_list
['H','e','l','l','o','H','e','l','l','o']
```

```
>>> s3 = set(word_list)
>>> s3
{'H', 'e', 'l', 'o'}



>>> s4 = set("Hello") # 문자열기반
>>> s4
{'H', 'e', 'l', 'o'}
```

※ 자료형의 중복을 제거하기 위한 필터로 종종 사용

집합 관련 함수

```
>>> s = {1, 2, 3, 4, 5, 3, 4}
>>> s
{1, 2, 3, 4, 5}

>>> s.add(10)
>>> s
{1, 2, 3, 4, 5, 10}

>>> s.remove(2)
>>> s
{1, 3, 4, 5, 10}
```

```
>>> s.update([1,3,5,7,9])
>>> s
{1, 3, 4, 5, 7, 9, 10}

>>> s.discard(7)
>>> s
{1, 3, 4, 5, 9, 10}

>>> s.clear()
>>> s
set()
```

집합 관련 함수

- 합집합, 교집합, 차집합

```
>>> s1 = {1, 2, 3}
>>> s2 = {3, 4, 5}

# 합집합
>>> s1.union(s2)
{1, 2, 3, 4, 5}

# 교집합
>>> s1.intersection(s2)
{3}

# 차집합
>>> s1.difference(s2)
{1, 2}
```

```
# 합집합( | )
>>> s1 | s2
{1, 2, 3, 4, 5}

# 교집합( & )
>>> s1 & s2
{3}

# 차집합( - )
>>> s1 - s2
{1, 2}
```

실습 – 집합1

- 1부터 100까지 숫자 중 3과 5의 공배수를 집합 형태로 만들어 아래와 같이 출력하시오. (단, 집합의 원소 출력 순서는 다를 수 있음)

```
{15, 30, 45, 60, 75, 90}
3와 5의 공배수: 6개
```

아래 코드를 채워서 완성할 것!

```
…

s = s3 & s5
print(s)

…
```

실습 – 집합2

- 숫자로 구성된 리스트에서 중복된 원소들만 반환하는 함수를 작성하시오. 단, 중복 원소 없으면 빈 리스트 반환

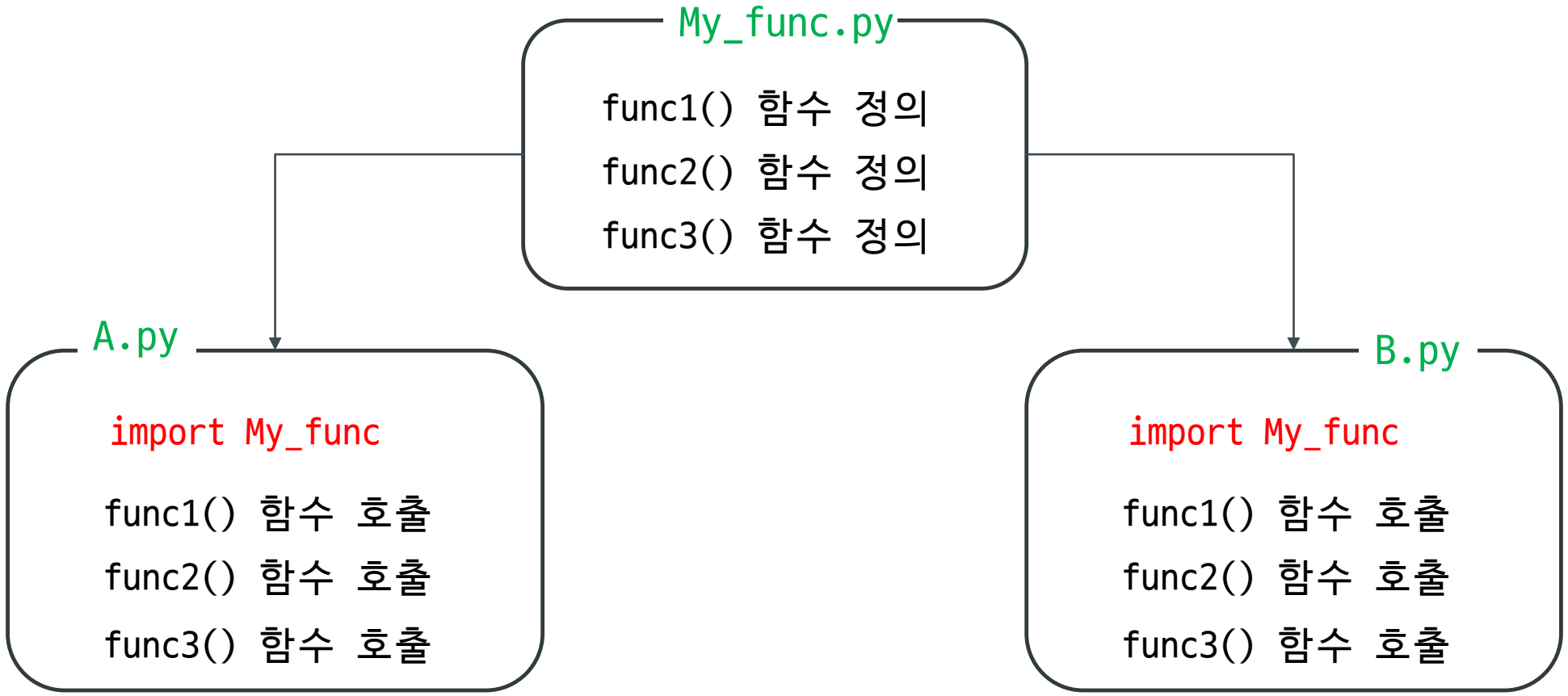
```
def find_duplicates(l):
    pass
```

```
l1 = [1, 2, 3, 2, 5, 5, 5, 6]
l2 = [1, 3, 4]

dl1 = find_duplicates(l1)
dl2 = find_duplicates(l2)
dl3 = find_duplicates(l1 + l2)

print(dl1) # [2, 5]
print(dl2) # []
print(dl3) # [1, 2, 3, 5]
```

사용자 정의 모듈 실습

모듈

- 많이 사용하는 함수를 만들어 놓고, 프로그램에서 해당 함수 사용시, import하여 사용하면 편리함



모듈

- 모듈의 작성과 사용
  - 모듈로 사용할 파일과 호출하는 파일은 모두 같은 폴더에 저장

```
#FILE: test.py
import arth

numbers = input('Enter two integers: ')

a, b = numbers.split()
a = int(a)
b = int(b)
```

모듈이름.함수이름() 형식으로 사용

```
print ("%d + %d = %2d"  % (a, b, arth.add(a, b)))
print ("%d - %d = %2d"  % (a, b, arth.sub(a, b)))
print ("%d * %d = %2d"  % (a, b, arth.mul(a, b)))
print ("%d / %d = %.1f" % (a, b, arth.div(a, b)))
```

```
#FILE: arth.py

def add(a, b):
    return a + b


def sub(a, b):
    return a - b


def mul(a, b):
    return a * b


def div(a, b):
    return a / b
```

모듈이름 생략하고 함수이름만으로 사용하고 싶다면 **from 모듈명 import ***