문화 A0019

파이썬프로그래밍

김 태 완

kimtwan21@dongduk.ac.kr

파이썬프로그래밍
중간 과제

1번 문제

- 학수번호를 입력 받아 교과목명과 강의실 정보를 출력하는 프로그램

학수번호	교과목명	강의실
데사K0025	데이터사이언스를위한수학	숭의관1호
문화A0019	파이썬프로그래밍	인문관15호
문화A0007	데이터사이언스입문	대학원4호

>>> 학수번호를 입력하세요 : **데사K0025**
>>> 입력하신 과목은 데이터사이언스를위한수학 이며, 강의실은 숭의관 1호 입니다.

>>> 학수번호를 입력하세요 : **문화A0019**
>>> 입력하신 과목은 파이썬프로그래밍 이며, 강의실은 인문관 15호 입니다.

>>> 학수번호를 입력하세요 : **문화A0007**
>>> 입력하신 과목은 데이터사이언스입문 이며, 강의실은 대학원 4호 입니다.

>>> 학수번호를 입력하세요 : **데사B0002**
>>> 입력하신 과목은 정보에 없습니다.

2번 문제

- 정답은 25이며, 숫자를 입력 받아 UP/DOWN을 출력하여 정답이 나올 때 까지 반복하는 프로그램
  - 정답은 25로 고정
  - 정답을 맞추기 전에는 프로그램이 종료 X
  - 정답을 입력하면 “정답!” 출력 후 프로그램 종료

```
>>> 숫자를 입력하세요 : 90
>>> DOWN !

>>> 숫자를 입력하세요 : 10
>>> UP !

>>> 숫자를 입력하세요 : 30
>>> DOWN !

>>> 숫자를 입력하세요 : 20
>>> UP !

>>> 숫자를 입력하세요 : 24
>>> UP !

>>> 숫자를 입력하세요 : 25
>>> 정답 !
```

3번 문제

- $3^{79} = 4926960980478197443869440340212777 65867$이다. 각 자리의 합을 구하는 코드를 구현해 보자.
  - 4+9+2+6+ … + 8+6+7 = ?

```
sum = 0

for i in                :
        
print(sum)
```

4번 문제

- while문을 이용하여 아래와 같이 출력해보자.

****

*****

***

*

5번 문제

- 정답 리스트 내 있는 데이터인지 아닌지 판별하는 프로그램
- answer 리스트에 있으면 'O', 없으면 'X'를 출력
- 입력한 리스트가 A,B,C 가 아닌 경우 '리스트에 없습니다.' 출력

- **answer = ['apple', 39, 'music', 568.2, 'Dongduk', 145, 'hello']**

- **A = ['hello', 62, 'umbrella', 145]**
- **B = ['September', 512.3, 'coffee', 39, 'keyboard', 'notebook', 0.5, 'f12']**
- **C = ['computer', 568.2, 39, 'aPple', 111, 'Dongduk', 'water']**

```
>>> 리스트를 입력하세요 : A
>>> OXXO

>>> 리스트를 입력하세요 : B
>>> XXXOXXXX

>>> 리스트를 입력하세요 : C
>>> XOOXXOX

>>> 리스트를 입력하세요 : F
>>> 리스트에 없습니다.
```

함수

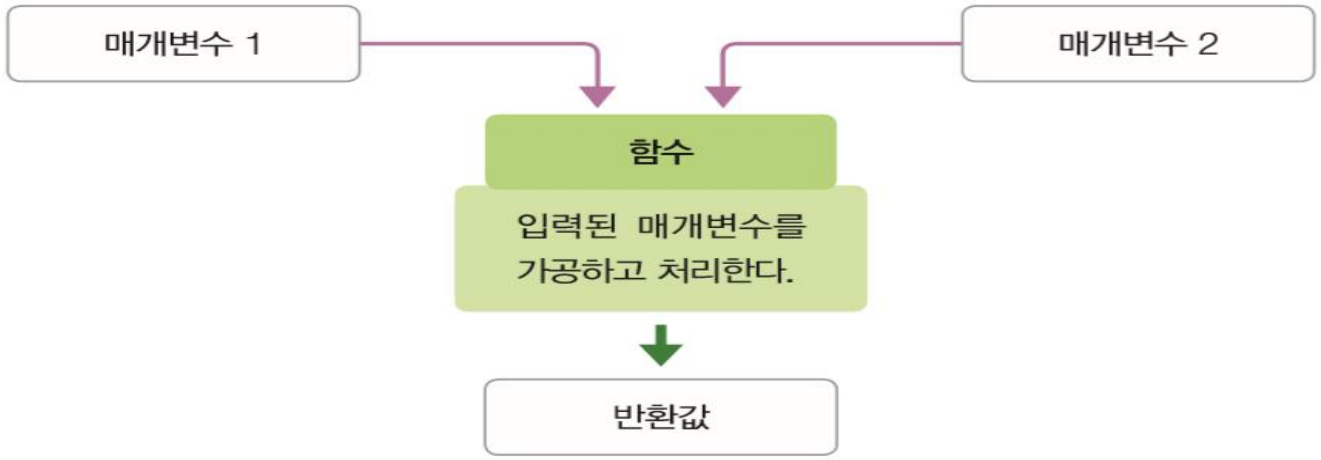
- 무엇을 넣으면 그것이 처리되어 다시 어떤 것을 돌려주는 기능을 함

```
scores = [80, 75, 91, 47, 66, 82, 57, 65, 90, 91, 33, 39, 78, 59, 40, 23, 19, 99, 75, 79, 37, 48, 82

average = sum(scores) / len(scores)
print(average)
```

```
def calculate_average(list):
    average = sum(list) / len(list)
    return average


scores = [80, 75, 91, 47, 66, 82, 57, 65, 90, 91, 33, 39, 78, 59, 40, 23, 19, 99, 75, 79, 37, 48, 82,

result = calculate_average(scores)

print(result)
```

함수

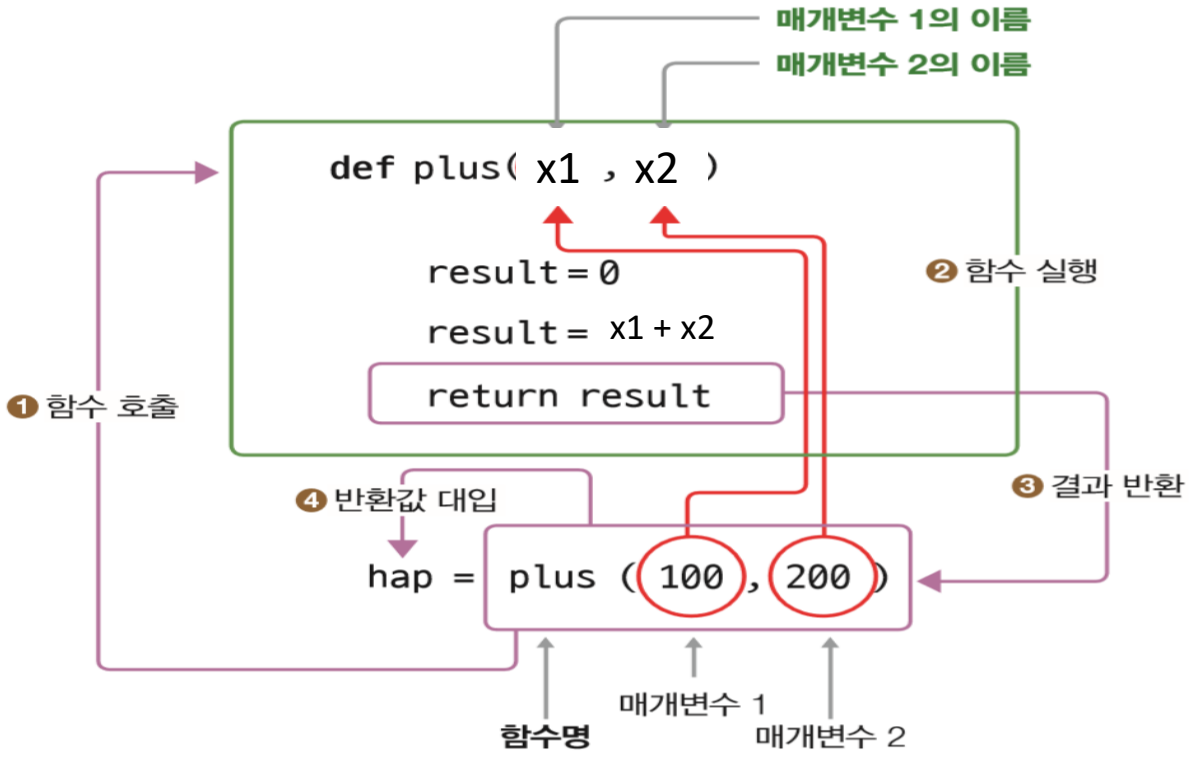
- 함수의 형식과 활용




```python
def plus(x1, x2):
    result = 0
    result = x1 + x2
    return result

hap = 0

hap = plus(100, 200)
print(hap)
```

함수

- 함수의 형식과 활용




```python
def plus(x1, x2):
    result = 0
    result = x1 + x2
    return result

hap = 0

hap = plus(100, 200)
print(hap)
```

함수

- 함수의 형식과 활용
  - 별도의 반환 값이 없는 함수
    - 함수를 실행한 결과, 돌려줄 것이 없는 경우에는 return문을 생략함
    - 또는 반환 값 없이 return만 써도 됨
    - 대체로 return 없이 함수를 끝내는 경향이 있음

```
함수이름( )
```

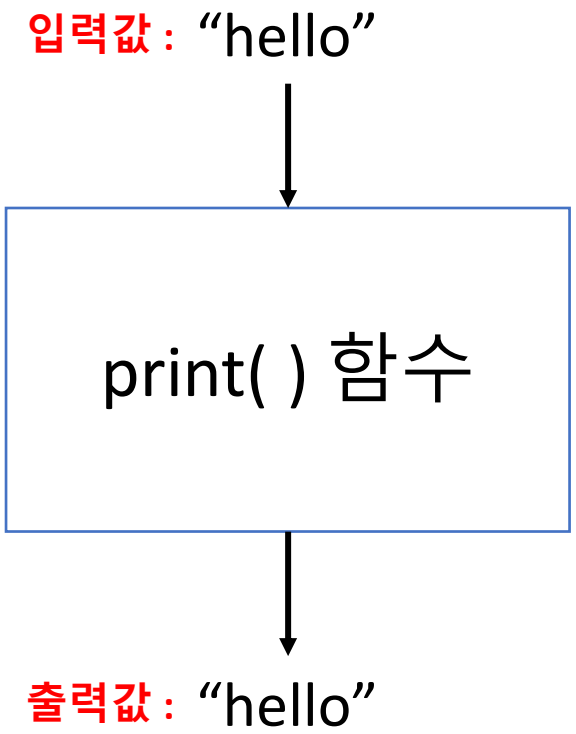
    - 예시 : print( ) → 괄호 안에 들어있는 내용을 화면에 출력

  - 함수에 별도의 반환 값이 있다면 변수에 반환 값을 받아야 함
    - 함수에서 어떤 계산이나 작동을 한 후에 반환할 값이 있으면
      'return 반환 값' 형식으로 표현함

```
변수 이름 = 함수이름( )
```

함수

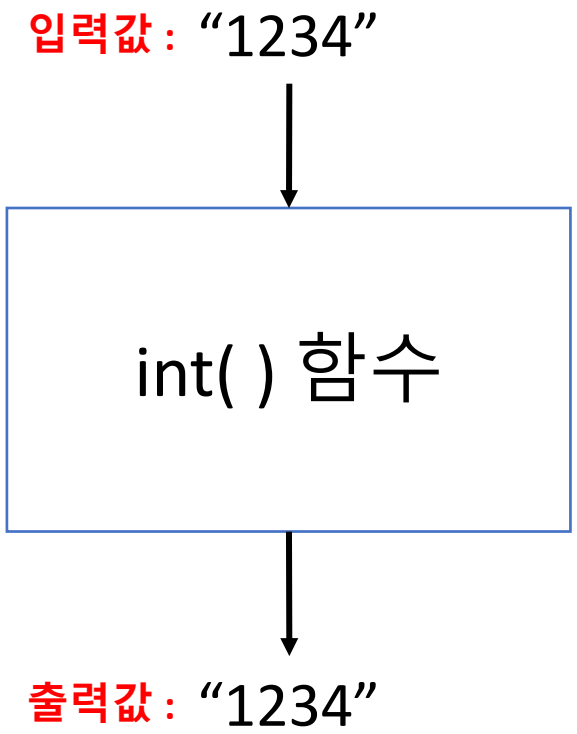
- 별도의 반환 값이 없는 함수



반환값은 없음

```
print(“hello”)
```

- 반환 값이 있는 함수

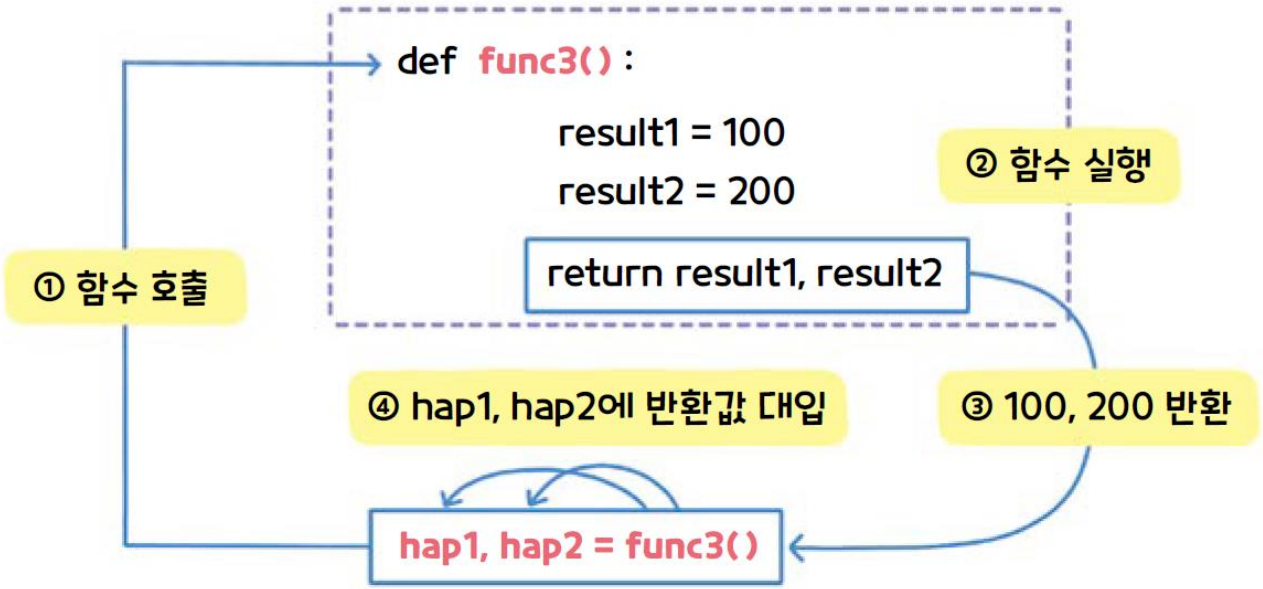


반환값 있음

```
num = int(“1234”)
```

함수

- 반환 값이 2개 있는 함수
  - 반환할 값이 2개라면 return 반환 값1, 반환 값2 형식으로 표현




```
## 함수 정의 부분
def func3() :
    result1 = 100
    result2 = 200

    return result1, result2

hap1, hap2 = 0, 0

## 메인 코드 부분
hap1, hap2 = func3()
print("func3()에서 돌려준 값 ==> ",
hap1, hap2)
```

함수

- 함수가 필요한 이유 : 중복되는 기능 효율적 처리
  - 예시 : 3명의 사용자 A, B, C가 두 숫자를 입력하고 합을 구하는 python 코드

```
print("A님. 두 숫자를 입력하세요")
num1 = int(input("정수1 ==>"))
num2 = int(input("정수2 ==>"))
hap = num1 + num2
print("결과 :", hap)

print("B님. 두 숫자를 입력하세요")
num1 = int(input("정수1 ==>"))
num2 = int(input("정수2 ==>"))
hap = num1 + num2
print("결과 :", hap)

print("C님. 두 숫자를 입력하세요")
num1 = int(input("정수1 ==>"))
num2 = int(input("정수2 ==>"))
hap = num1 + num2
print("결과 :", hap)
```

```
def hapFunc() :
    num1 = int(input("정수1 ==>"))
    num2 = int(input("정수2 ==>"))
    return num1 + num2

print("A님. 두 숫자를 입력하세요")
hap = hapFunc()
print("결과 :", hap)

print("B님. 두 숫자를 입력하세요")
hap = hapFunc()
print("결과 :", hap)

print("C님. 두 숫자를 입력하세요")
hap = hapFunc()
print("결과 :", hap)
```

함수

- 함수의 작업
  - 정의하기(define) : 0개 또는 1개 이상의 매개변수를 가짐
    - def 와 함수 이름, 괄호를 입력
    - 괄호 안에는 옵션으로 매개변수parameter를 입력할 수 있음
    - 마지막으로 콜론 (:) 을 붙임
    - 함수 이름은 변수 이름과 동일한 규칙으로 작성

      (이름의 첫 글자는 반드시 영문자나 언더바(_)를 사용해야 함. 영문자, 숫자, 언더바만 사용할 수 있음)

  - 호출하기(call) : 0개 또는 1개 이상의 결과를 획득

함수

- 숫자 2개의 합과 3개의 합을 구하는 함수
  - 함수에 매개변수의 개수를 정해 놓으면 함수를 호출할 때, 매개변수의 개수를 정확히 맞춰서 호출해야 함

```
## 함수 정의 부분
def para2_func(v1, v2) :
    result = 0
    result = v1 + v2
    return result

def para3_func(v1, v2, v3) :
    result = 0
    result = v1 + v2 + v3
    return result

hap = 0

## 메인 코드 부분
hap = para2_func(10, 20)
print("매개변수 2개 함수 호출 결과 ==> ", hap)
hap = para3_func(10, 20, 30)
print("매개변수 3개 함수 호출 결과 ==> ", hap)
```

함수

- 매개변수의 개수를 정해 놓는 방법
  - 함수에 매개변수의 개수를 정해 놓으면 함수를 호출할 때는 정확히 매개변수의 개수에 맞춰서 호출해야 함
  - 매개변수의 개수가 다르면 별도의 함수를 만들어야 함

- 매개변수에 기본값을 설정해 놓는 방법
  - 가장 많이 전달될 매개변수 개수를 준비해 놓고 각 매개변수에 기본값을 설정하기

```
def para_func(v1, v2, v3 = 0) :
    result = 0
    result = v1 + v2 + v3
    return result


hap = 0

## 메인 코드 부분
hap = para_func(10, 20)
print("매개변수 2개 함수 호출 결과 ==> ", hap)
hap = para_func(10, 20, 30)
print("매개변수 3개 함수 호출 결과 ==> ", hap)
```

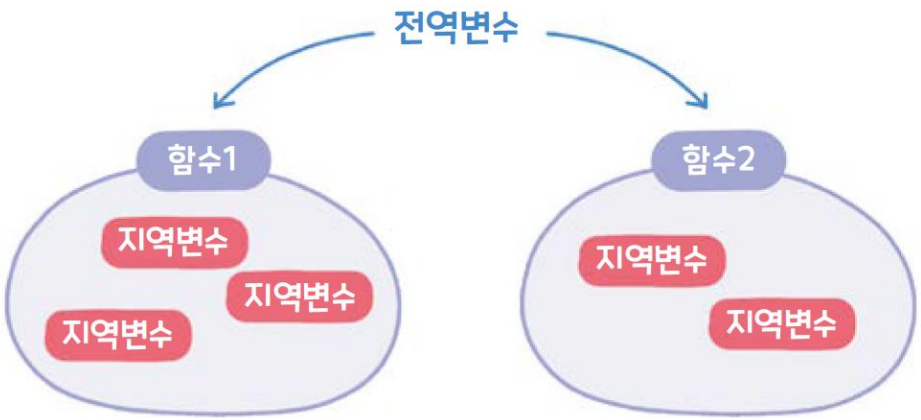
함수

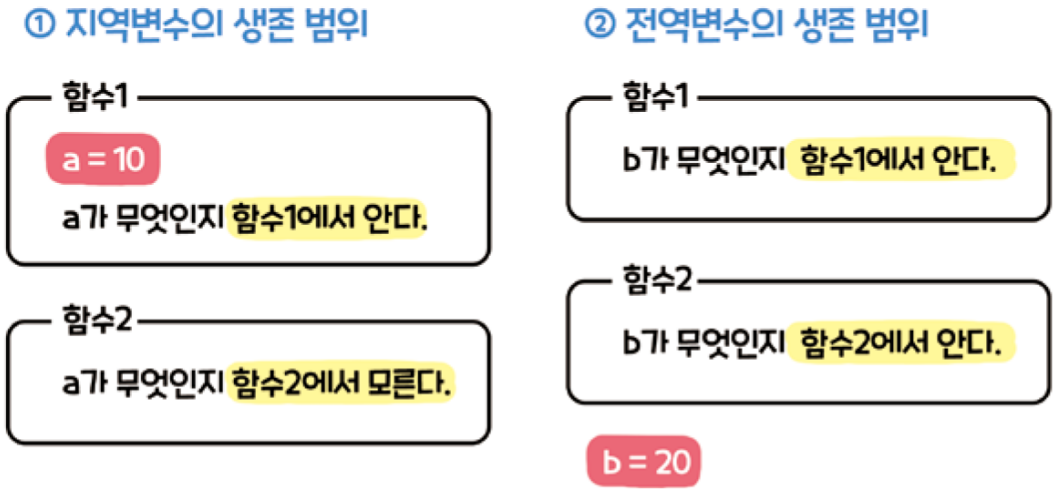
- 매개변수의 개수를 지정하지 않고 전달

```
def plus(*param):
    result = 0
    for num in param :
        result += num
    return result

hap = 0

hap = plus(100, 200, 300, 400)
print(hap)
```

함수

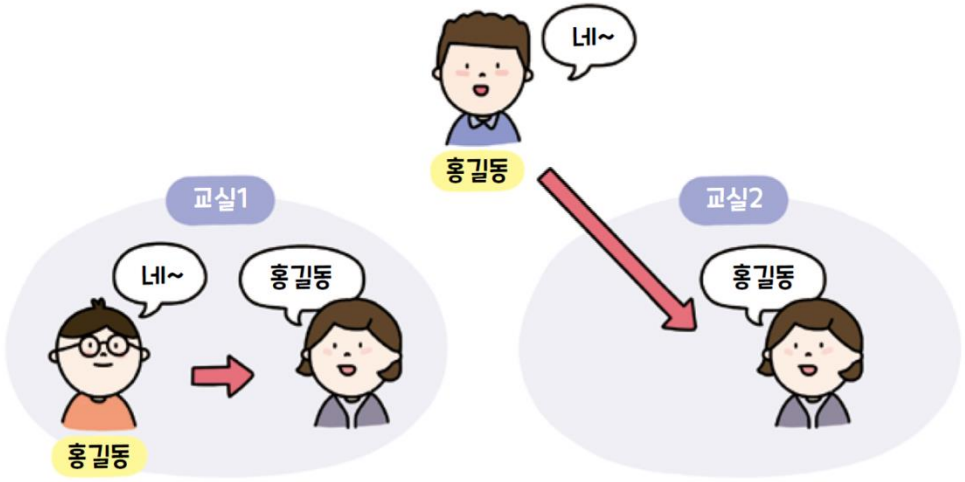
- 지역변수와 전역변수
  - 유효 범위 : 변수가 활동할 수 있는 범위
  - 지역변수
    - 말 그대로 한정된 지역(local)에서만 사용되는 변수
  - 전역변수
    - 프로그램 전체(global)에서 사용되는 변수

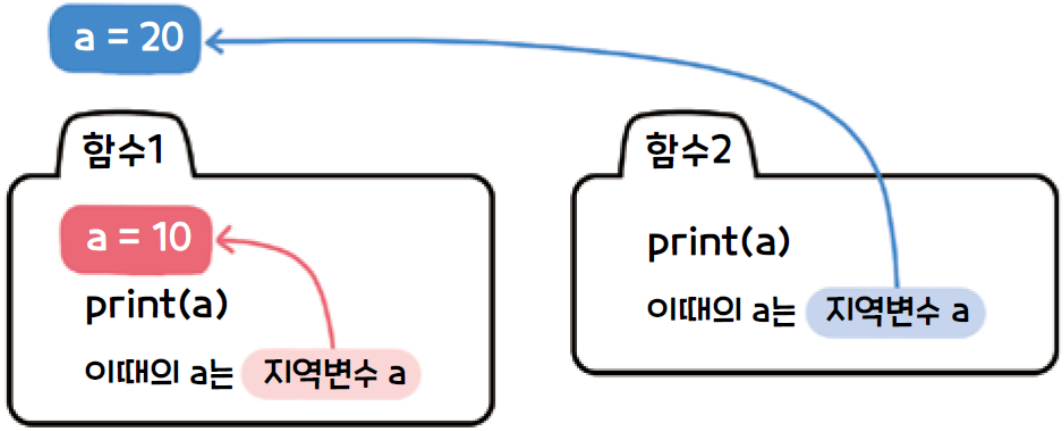




함수

- 지역변수와 전역변수의 이름이 같은 경우
  - 지역변수가 우선됨
  - 함수 내에 변수가 정의되어 있는가를 확인하면 간단히 구분할 수 있음
    - 같은 a라고 해도 함수1의 a는 함수 내에서 따로 정의했으므로 지역변수임
    - 함수2의 a는 함수 안에 정의된 것이 없으므로 전역변수임





함수

- 지역변수와 전역변수의 이름이 같은 경우

```
## 함수 정의 부분
def func1() :
    a = 10      # 지역변수
    print("func1()에서 a의 값 ", a)

def func2() :
    print("func2()에서 a의 값 ", a)

## 전역변수 선언 부분
a = 20

## 메인 코드 부분
func1()
func2()
```

예제 #1

- **예제 1 : 함수에 정의와 다르게 함수를 호출**

```
def taewan(s):
    print(s)

taewan("hello")
taewan()
```

- **실행 결과**

```
Hello

Traceback (most recent call last):
  File "c:₩Users₩enoug₩Desktop₩python₩tmp2.py", line 17, in <module>
    taewan()
TypeError: taewan() missing 1 required positional argument: 's'
```

예제 #2

- **예제 2 : 전역변수와 지역변수 오류 예시**

```
def n_plus_1 (n) :
    result = n + 1

n_plus_1(3)

print(result)
```

- **실행 결과**

```
NameError: name 'result' is not defined
```

예제 #3

- **예제 3 : 문자열 하나를 입력 받아 인터넷 주소를 반환하는 make_url 함수를 정의해보자**

```
def make_url(string):
    ### coding here ###


make_url(“naver”)
make_url(“facebook”)
```

- **실행 결과**

```
www.naver.com
www.facebook.com
```

예제 #4

- **예제 4 : 문자열을 입력받아 각 문자들로 구성된 리스트로 반환하는 make_list 함수를 정의해보자**

```
def make_list(string):
    ### coding here ###


make_list(“naver”)
make_list(“facebook”)
```

- **실행 결과**

```
[‘n', ‘a', ‘v’, ‘e', ‘r’]
[‘f', ‘a’, ‘c’, ‘e’, ‘b’, ‘o’, ‘o’, ‘k’]
```

예제 #5

- **예제 5 : 숫자로 구성된 하나의 리스트를 입력받아, 짝수들을 추출하여 리스트로 반환하는 pickup_even 함수를 정의**

```
def pickup_even(items):
    ### coding here ###


pickup_even([3, 4, 5, 6, 7, 8])
```

- **실행 결과**

```
[4, 6, 8]
```

예제 #6

- **예제 6 : 주어진 리스트의 평균 구하는 함수**

```
scores = [80, 75, 91, 47, 66, 82, 57, 65, 90, 91, 33, 39, 78, 59, 40, 23, 19, 99, 75, 79, 37, 48, 82

average = sum(scores) /
print(average)
```

```
def average(list):
 ### coding here ###

scores = [80, 75, 91, 47, 66, 82, 57, 65, 90, 91, 33, 39, 78, 59, 40,
23, 19, 99, 75, 79, 37, 48, 82, 60, 60, 63, 100, 8, 12, 92, 32, 50, 61,
28, 84, 40, 100, 25, 94, 74, 88, 94, 100, 5, 26]

print(average(scores))
```

- **실행 결과**

```
61.13333333333333333333333
```

감사합니다

kimtwan21@dongduk.ac.kr

김 태 완