

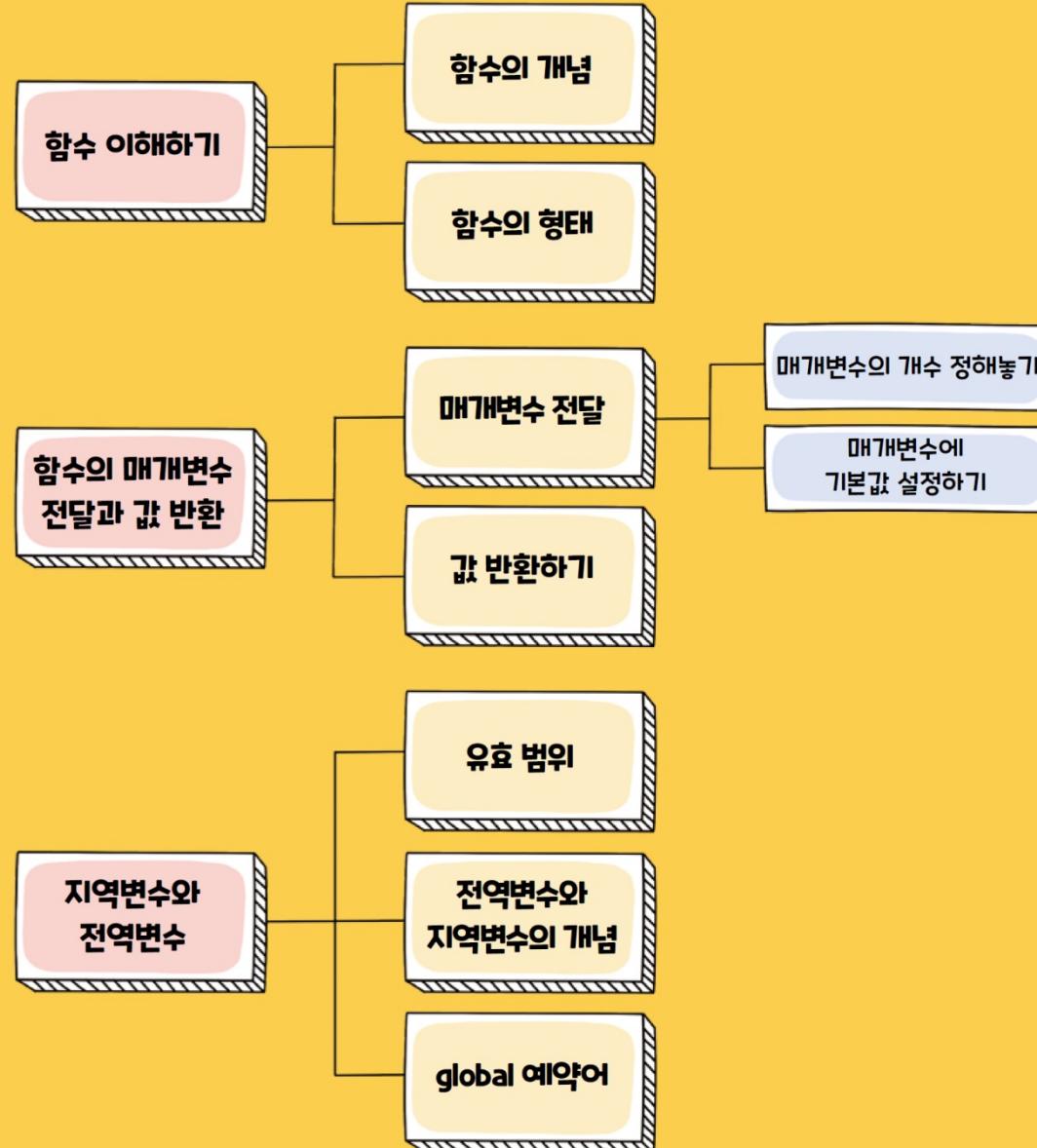
# Chapter 8

## 함수

# 임경태



## Preview



# 학습목표

- 함수에 대한 이해와 작성법을 익힙니다.
- 함수에서 전달되는 매개변수의 작동원리를 이해합니다.
- 함수에서 반환하는 반환값을 이해합니다.
- 지역변수와 전역변수의 차이점과 활용법을 익힙니다

# Section 01

## 함수 이해하기



# 함수의 개념

## 함수

- 무엇을 넣으면 그것이 처리되어 다시 어떤 것을 돌려주는 기능을 함
- 함수의 예시) '음료수 자판기'
  - 자판기에 동전을 넣고 콜라 버튼을 누르면, 그 안에서 어떤 처리가 되어서 콜라가 나옴
  - 이때 동전의 금액이 콜라를 뽑을 수 있는 액수인지, 만약 콜라를 뽑을 수 있는 금액이라면, 내부에서 기계가 어떤 방식으로 작동하는지 등은 알 수 없음
  - 따라서 함수를 블랙박스라고 표현함



그림 8-1 함수의 개념

# 파이썬에서 제공하는 함수



## ■ 파이썬에서 제공하는 함수의 사용법

- ▶ 별도의 반환값이 없는 함수의 경우

함수이름()

- ▶ 함수에 별도의 반환값이 있다면 변수에 반환값을 받아야 함

변수 이름 = 함수이름()

# 파이썬에서 제공하는 함수



## ■ 반환값이 없는 함수

- print() 함수는 괄호 안에 들어있는 내용을 화면에 출력함

```
>>> print("난생처음 파이썬")  
난생처음 파이썬
```

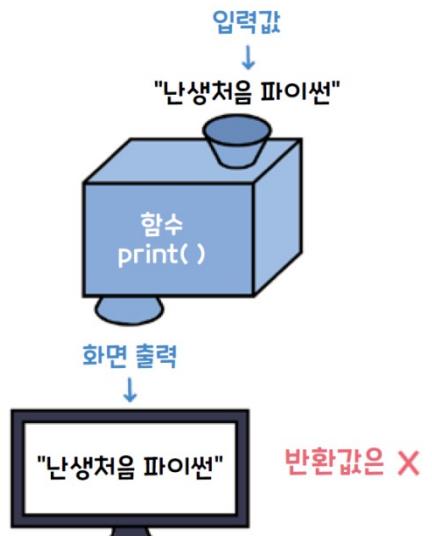


그림 8-2 반환값이 없는 `print()` 함수의 작동

# 파이썬에서 제공하는 함수



## ■ 반환값이 있는 함수

- int() 함수는 입력받은 문자열을 변환해서 정수로 반환함

```
>>> num = int("1234")
>>> print(num)
1234
```

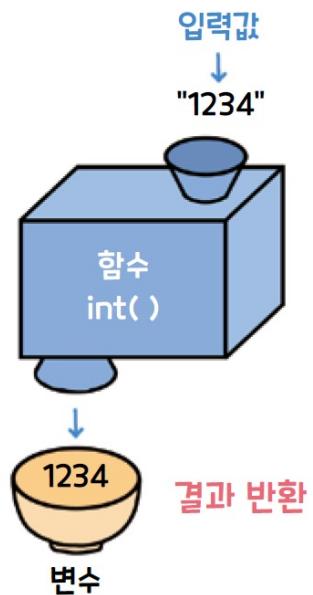


그림 8-3 반환값이 있는 `int()` 함수의 작동

# 사용자 정의 함수의 필요성



## ■ 사용자 정의 함수가 필요한 이유

- 3명의 사용자 A, B, C가 두 숫자를 입력하고,  
입력한 값을 더해서 그 결과를 출력하는 프로그램

### [코드 8-1]

```
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)
```

### [실행결과]

A님. 두 숫자를 입력하세요  
정수1 ==>10  
정수2 ==>20  
결과 : 30

B님. 두 숫자를 입력하세요  
정수1 ==>30  
정수2 ==>40  
결과 : 70

C님. 두 숫자를 입력하세요  
정수1 ==>50  
정수2 ==>60  
결과 : 110

사용자 입력

# 사용자 정의 함수의 필요성



## ■ 사용자 정의 함수가 필요한 이유

- 3명의 사용자 A, B, C가 두 숫자를 입력하고,  
입력한 값을 더해서 그 결과를 출력하는 프로그램을 함수로 구현

### [코드 8-2]

```
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)
```

### [실행결과]

A님. 두 숫자를 입력하세요  
정수1 ==>10  
정수2 ==>20 └── 사용자 입력  
결과 : 30

B님. 두 숫자를 입력하세요  
정수1 ==>30  
정수2 ==>40 └── 사용자 입력  
결과 : 70

C님. 두 숫자를 입력하세요  
정수1 ==>50  
정수2 ==>60 └── 사용자 입력  
결과 : 110

# 사용자 정의 함수의 필요성



## ■ 사용자 정의 함수가 필요한 이유

- [코드 8-1]과 [코드 8-2]는 완전히 동일한 기능을 하는 코드이기 때문에 실행 결과가 같음
- [코드 8-1]에서 반복되는 2~4행, 8~10행, 14~16행 부분을 [코드 8-2]에서 1~4행으로 한 곳에 모아 함수 hapFunc()로 만듦
- int()를 float()로 변경해야 한다면, 어디만 고치면 될까요?

Click!

```
def hapFunc() :  
    num1 = float(input("실수1 ==>"))  
    num2 = float(input("실수2 ==>"))  
    return num1 + num2
```

# 사용자 정의 함수의 필요성



## 확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

블랙박스처럼 그 내부가 보이지는 않지만, 입력된 값을 처리해서 반환하는 기능을 하는 것을  
\_\_\_\_\_ (이)라고 부른다.

2. 다음 중 잘못된 것을 모두 고르시오.

- ① 함수를 만들면 전체 소스 코드가 더 길어진다.
- ② 함수를 사용하면 유지보수가 편해진다.
- ③ 함수를 만든 후에 변경할 사항이 생기면 수정할 부분이 더 많아진다.
- ④ 함수에는 파이썬에서 제공하는 함수와 사용자가 직접 정의하는 함수가 있다.

## 정답

Click!



# 함수의 형태

## 함수의 기본 형태

- 함수는 매개변수(Parameter)를 입력받은 후 그 매개변수를 가공 및 처리한 후에 반환값을 돌려줌

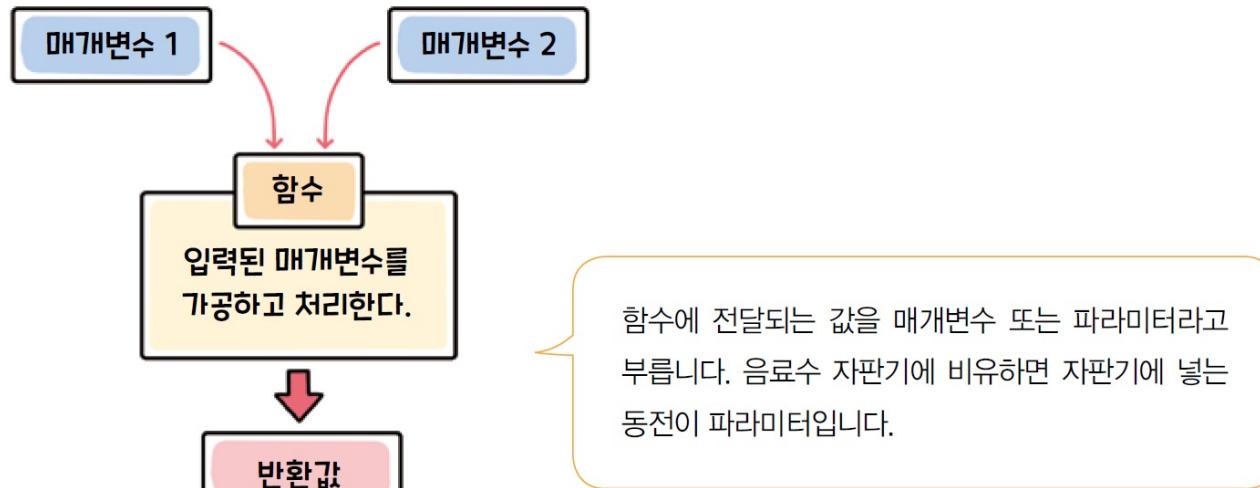


그림 8-4 함수의 형태



# 함수의 형태

## ■ 함수의 기본 형태

- 두 정수를 입력받아 두 정수의 합계를 반환하는 plus() 함수

### [코드 8-3]

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

```
## 전역변수 선언 부분
hap = 0
```

```
## 메인 코드 부분
hap = plus(100, 200)
print("100과 200의 plus() 함수 결과는 ", hap)
```

### [실행결과]

100과 200의 plus() 함수 결과는 300



# 함수의 형태

## ■ 함수를 정의하고 호출하는 동작

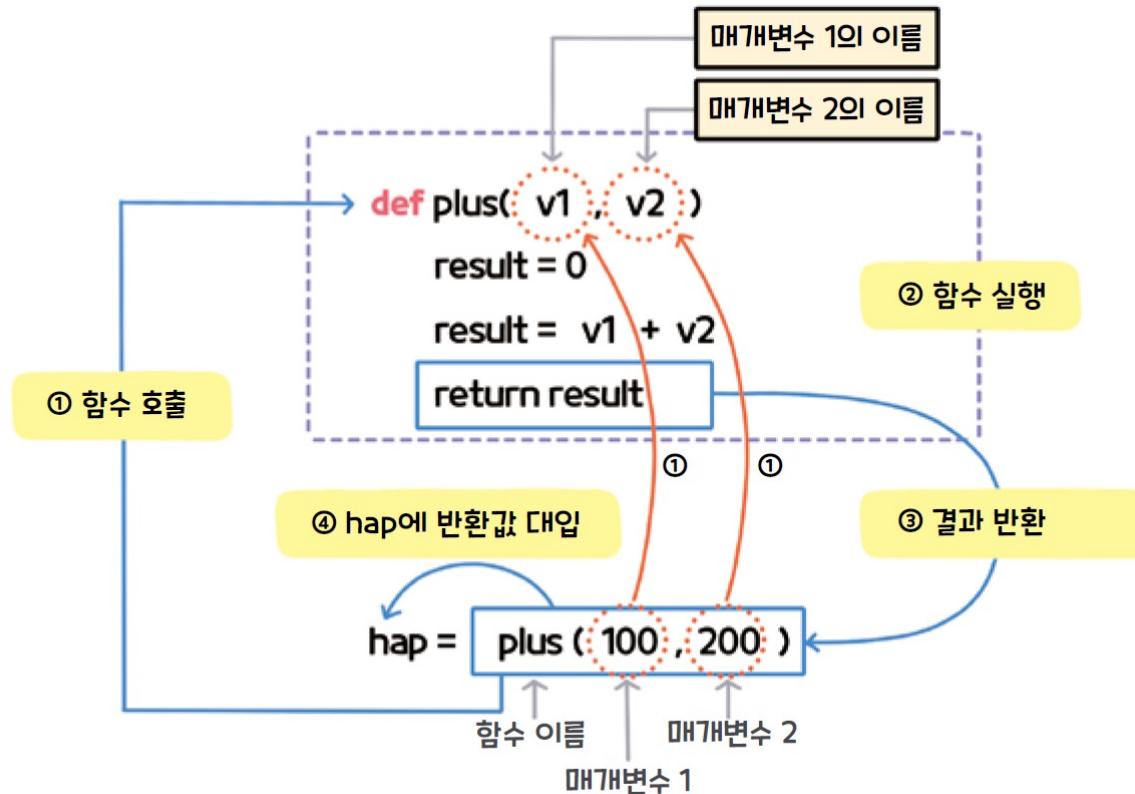


그림 8-5 `plus()` 함수의 형태와 호출 순서

# 함수의 형태



하나 더 알기 ▾

## 프로그램의 틀

앞으로 작성할 코드는 기존 코드에 비해서 코드가 좀 길어질 수 있기 때문에 다음과 같이 코드를 세 부분으로 나눕니다. 아래와 같이 굳이 나누지 않아도 프로그램 작동에는 문제가 없지만, 프로그램을 작성하는 좋은 구조이므로 이 책에서는 아래 방식을 주로 사용하겠습니다.

## 함수 정의 부분

## 전역변수 선언 부분

## 메인(main) 코드 부분

전역변수에 대한 것은 잠시 후에 다루므로 지금은 그냥 변수라고 이해해도 됩니다. 그리고 전에 언급했듯이 #은 주석(remark)으로 #이 나오면 그 줄은 설명 줄이 됩니다. 즉, 없어도 되는 문장이지만 주석을 많이 달아 놓으면 프로그램 코드를 보기 가 편해지므로, 주석을 다는 것은 좋은 프로그래밍 습관입니다.



계산을 위한 기호 +, -, \*, /를 입력받은 후에, 두 수를 입력하면 선택한 계산이 되는 계산기를 함수를 사용해 만들어 봅시다.

## 실행 결과

계산 입력 ( +, -, \* , / ) : -

첫 번째 숫자 입력 : 55

두 번째 숫자 입력 : 33

## 계산기 :  $55 - 33 = 22$

사용자 입력



1. lab08-01.py 파일을 만들고, 매개변수를 3개 받는 calc() 함수를 정의하기
  - 이후 계산이 필요할 때 calc() 함수를 호출해서 사용함

```
## 함수 정의 부분
def calc(v1, v2, op) :
    result = 0
    if op == '+':
        result = v1 + v2
    elif op == '-':
        result = v1 - v2
    elif op == '*':
        result = v1 * v2
    elif op == '/':
        result = v1 / v2
    return result
```

2. 계산 결과를 저장할 변수(res) 및 연산자(op)와 두 숫자(var1, var2)를 저장할 변수 준비하기. 그리고 어떤 연산을 할지 연산자를 입력받은 후에, 계산할 숫자 2개를 입력받기

```
## 전역변수 선언 부분
```

```
res = 0  
var1, var2, oper = 0, 0, ""
```

```
## 메인 코드 부분
```

```
oper = input("계산 입력 ( +, -, *, / ) : ")  
var1 = int(input("첫 번째 숫자 입력 : "))  
var2 = int(input("두 번째 숫자 입력 : "))
```

### 3. calc() 함수에 3개의 매개변수를 넘겨주며 호출함

- 이 매개변수는 calc() 함수가 정의된 1의 var1(→ v1), var2(→ v2), oper(→ op)에 각각 대응됨
- 사용자가 입력한 연산자에 따라 필요한 연산을 수행한 후에, 계산된 값을 return으로 반환함
- calc()에서 반환한 값을 res에 넣고, 계산식을 출력함

```
res = calc(var1, var2, oper)
print("## 계산기 : ", var1, oper, var2, "=", res)
```

### 4. <Ctrl>+<S>를 눌러 저장한 후, <F5>를 눌러서 실행하고 결과를 확인하기

## Section 02

# 함수에 매개변수를 전달하고 값 반환하기



# 함수의 매개변수 전달

## ■ 숫자 2개의 합과 3개의 합을 구하는 함수

- 함수에 매개변수의 개수를 정해 놓으면 함수를 호출할 때, 매개변수의 개수를 정확히 맞춰서 호출해야 함

### [코드 8-4]

```
## 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
```

```
## 전역변수 선언 부분
```

```
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)
```

### [실행결과]

매개변수 2개 함수 호출 결과 ==> 30

매개변수 3개 함수 호출 결과 ==> 60



# 함수의 매개변수 전달

## ■ 숫자 2개의 합과 3개의 합을 구하는 함수

- [코드 8-4]의 2~5행은 매개변수를 2개, 7~10행은 매개변수를 3개 받아 합계를 반환하는 함수의 표준임

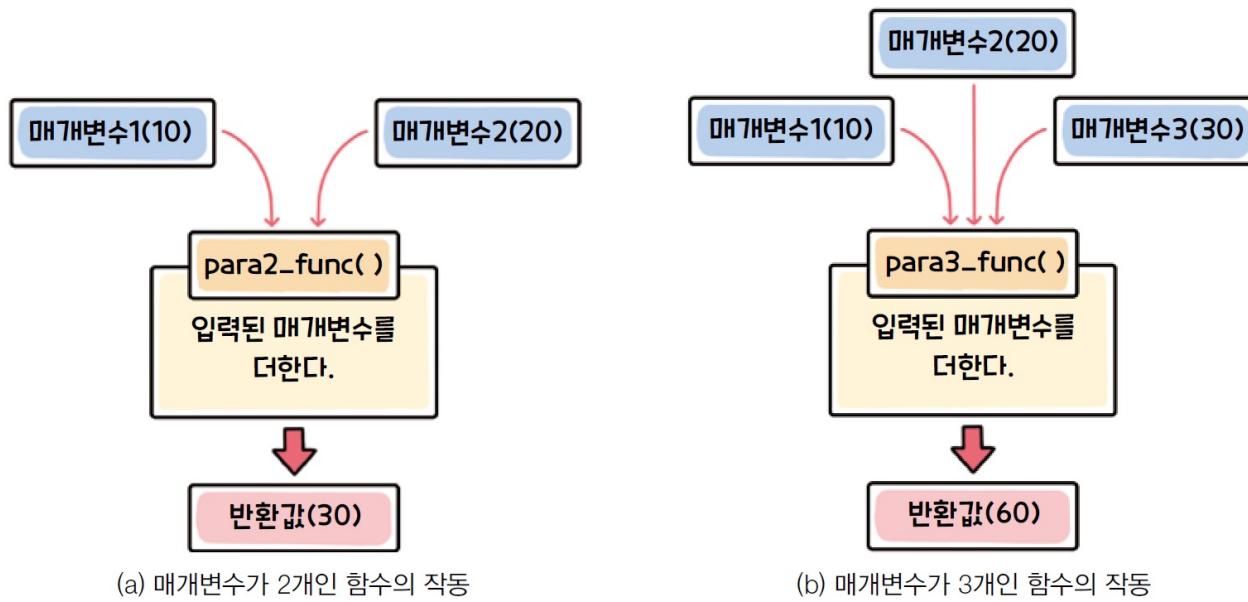


그림 8-6 매개변수의 개수에 따른 함수의 작동



# 매개변수 전달의 다양한 방법

## ■ 매개변수의 개수를 정해 놓는 방법

- ▶ 함수에 매개변수의 개수를 정해 놓으면 함수를 호출할 때는 정확히 매개변수의 개수에 맞춰서 호출해야 함
- ▶ 이와 같이 매개변수의 개수가 다르면 별도의 함수를 만들어야 함

## ■ 매개변수에 기본값을 설정해 놓는 방법

- ▶ 가장 많이 전달될 매개변수 개수를 준비해 놓고 각 매개변수에 기본값을 설정하기



# 매개변수 전달의 다양한 방법

## ■ 매개변수에 기본값을 설정해 놓는 방법의 예시

- [코드 8-4]에서 para2\_func()과 para3\_func() 중 3개의 파라미터를 사용하는 para3\_func()를 기준으로 하나의 함수만 작성하도록 수정하기

### [코드 8-5]

```
## 함수 정의 부분
def para_func(v1, v2, v3 = 0) :
    result = 0
    result = v1 + v2 + v3
    return result

## 전역변수 선언 부분
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)
```

### [실행결과]

```
매개변수 2개 함수 호출 결과 ==> 30
매개변수 3개 함수 호출 결과 ==> 60
```

# 매개변수 전달의 다양한 방법



## 확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

함수에 매개변수가 2개라면 호출할 때도  개를 넘겨줘야 하고, 함수에 매개변수가 3개라면 호출할 때도  개를 넘겨줘야 한다.

2. 다음은 매개변수 v4에 기본값을 0으로 설정하는 함수의 정의이다. 빈칸에 들어갈 단어를 채우시오.

```
def myFunc(v1, v2, v3, ) :  
    함수 내용
```

## 정답

Click!

# 값 반환하기



## ■ 반환값이 없는 함수

- 함수를 실행한 결과, 돌려줄 것이 없는 경우에는 return문을 생략함
- 또는 반환값 없이 return만 써도 됨
  - 대체로 return 없이 함수를 끝내는 경향이 있음
- 반환값 없이 함수를 마치면 아무것도 반환하지 않고 함수가 종료됨

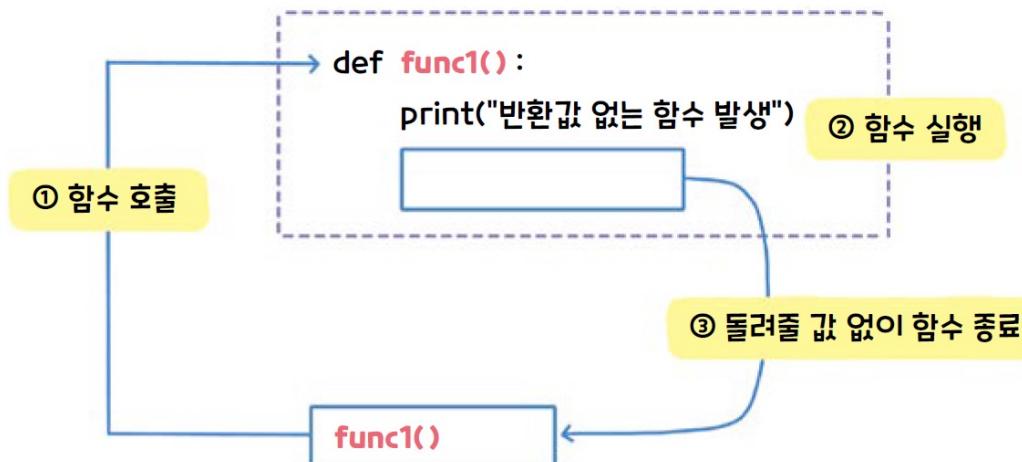


그림 8-7 반환값이 없는 함수의 작동

# 값 반환하기



## ■ 반환값이 없는 함수

### [코드 8-6]

```
## 함수 정의 부분
def func1() :
    print("반환값이 없는 함수 실행")

## 전역변수 선언 부분

## 메인 코드 부분
func1()
```

### [실행결과]

반환값이 없는 함수 실행

# 값 반환하기



## 반환값이 1개 있는 함수

- 함수에서 어떤 계산이나 작동을 한 후에 반환할 값이 있으면 'return 반환값' 형식으로 표현함

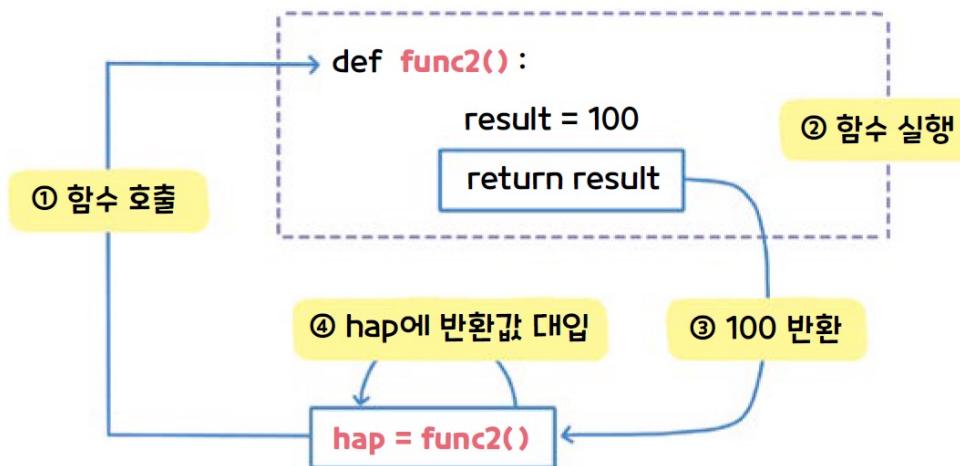


그림 8-8 값 1개 반환하는 함수의 작동

# 값 반환하기



## ■ 반환값이 1개 있는 함수

### [코드 8-7]

```
## 함수 정의 부분
def func2() :
    result = 100
    return result

## 전역변수 선언 부분
hap = 0

## 메인 코드 부분
hap = func2()
print("func2()에서 돌려준 값 ==> ", hap)
```

### [실행결과]

```
func2()에서 돌려준 값 ==> 100
```



# 값 반환하기

## 반환값이 2개 있는 함수

- 반환할 값이 2개라면 return 반환값1, 반환값2 형식으로 표현

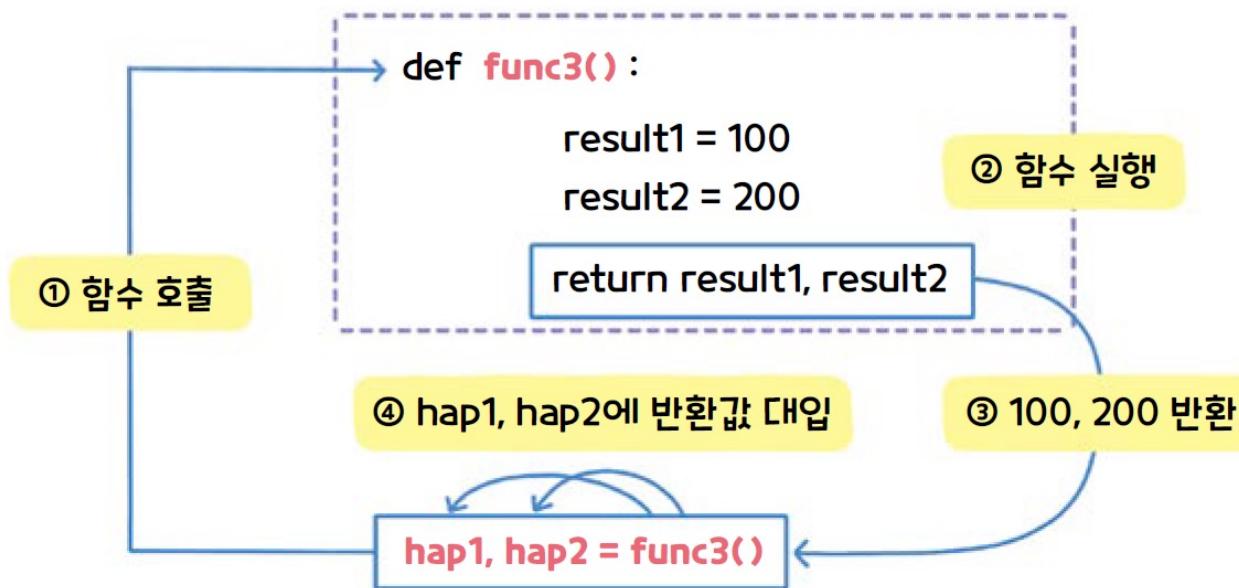


그림 8-9 값을 2개 반환하는 함수의 작동



# 값 반환하기

## ■ 반환값이 2개 있는 함수

### [코드 8-8]

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

## 전역변수 선언 부분
hap1, hap2 = 0, 0

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

### [실행결과]

```
func3()에서 돌려준 값 ==> 100 200
```

# 값 반환하기



## ■ pass 키워드

- 함수의 이름과 형태만 만들어 놓고, 내부는 나중에 코딩하고 싶은 경우에 사용하는 키워드

```
def myFunc() : <--- 오류 발생
```

```
def myFunc() :  
    pass
```

- pass 키워드는 함수뿐 아니라, if문이나 반복문에서도 아무것도 안 하는 코드로 채울 때에도 사용함

```
if True :  
    <--- 오류 발생  
else :  
    print('거짓입니다')
```

```
if True :  
    pass  
else :  
    print('거짓입니다')
```

# 값 반환하기



## 확인문제

1. 다음 중 잘못된 것을 고르시오.

- ① 반환값이 없는 함수는 함수의 끝에 아무것도 쓰지 않거나, 그냥 return만 써도 됩니다.
- ② 반환값이 2개라도 돌려받는 변수는 1개여야 합니다.
- ③ 반환값이 없는 함수는 오류를 발생합니다.
- ④ 함수의 빈 부분이나, if문의 빈 부분을 채울 때 아무것도 안하는 의미의 키워드는 pass입니다.

2. 다음은 2개의 값을 반환하는 myFunc() 함수를 호출하는 구문입니다. 빈칸에 들어갈 단어를 채우시오.

```
res1, res2 = 0, 0
```

```
 = myFunc()
```

## 정답

Click!

로또 복권은 1~45 숫자 중에서 6개를 추첨하는 게임입니다. 로또를 추첨하는 함수를 작성해서 함수를 호출할 때마다, 숫자를 하나씩 뽑아서 반환하도록 코드를 작성해 봅시다.

주의할 점은 새로 뽑은 숫자가 기존에 뽑은 적이 있는지 확인해야 하고 기존에 뽑은 적이 있는 숫자라면 무시하고 새로 뽑아야 합니다.

#### 실행 결과

\*\* 로또 추첨을 시작합니다. \*\*

오늘의 로또 번호 ==> 3 5 24 41 44 45



1. lab08-02.py 파일을 만들고, 로또 숫자를 랜덤하게 뽑기 위해서 random 임포트하기
  - 로또 추첨 프로그램은 실행할 때마다 다른 6개의 숫자가 출력돼야 함

```
import random
```

2. lottoFunc() 함수를 정의하기

- 이 함수는 추첨된 로또 번호 1개를 돌려주는 함수임
- random.randint(1, 45) 함수로 1~45 중에서 임의의 숫자 하나를 추출하고 반환함

```
## 함수 정의 부분
def lottoFunc() :
    lottoNum = random.randint (1, 45)
    return lottoNum
```

3. 추첨된 로또 숫자를 저장할 lottoList 리스트와 추첨된 숫자를 임시로 저장할 num 변수를 준비하기

```
## 전역변수 선언 부분
lottoList = []
num = 0
```

#### 4. 로또 추첨을 시작하는 부분을 구현하기

- 우선 무한 반복을 실행함. 임의의 숫자를 lottoFunc() 함수로 뽑는데, 뽑힌 숫자(num)가 이전에 이미 뽑힌 숫자일 수 있기 때문임

```
## 메인(main) 코드 부분
```

```
print("** 로또 추첨을 시작합니다. ** \n");
while True :
    num = lottoFunc()
```

#### 5. 만약 뽑은 숫자가 lottoList 리스트에 이미 들어 있으면 무시하고 lottoFunc() 함수를 호출하여 다시 숫자를 추출함.

그렇지 않고 뽑힌 숫자(num)가 lottoList에 없다면 lottoList.append(num) 함수로 lottoList에 숫자를 추가함.

```
if num in lottoList :
    continue
else :
    lottoList.append(num)
```

6. lottoList에 6개 이상이 저장되면 무한반복을 종료함

```
if len(lottoList) == 6 :  
    break
```

7. 뽑힌 6개의 숫자를 lottoList.sort() 함수로 정렬한 후, 차례대로 출력하기

```
print("오늘의 로또 번호 ==> ", end=' ')  
lottoList.sort()  
for i in range(0,6) :  
    print(lottoList[i], " ", end=' ')
```

8. <Ctrl>+<S>를 눌러 저장한 후, <F5>를 눌러서 실행하고 결과를 확인하기

# Section 03

## 지역변수와 전역변수



# 유효 범위 이해하기

## ■ 유효 범위

- 자신이 활동할 수 있는 범위

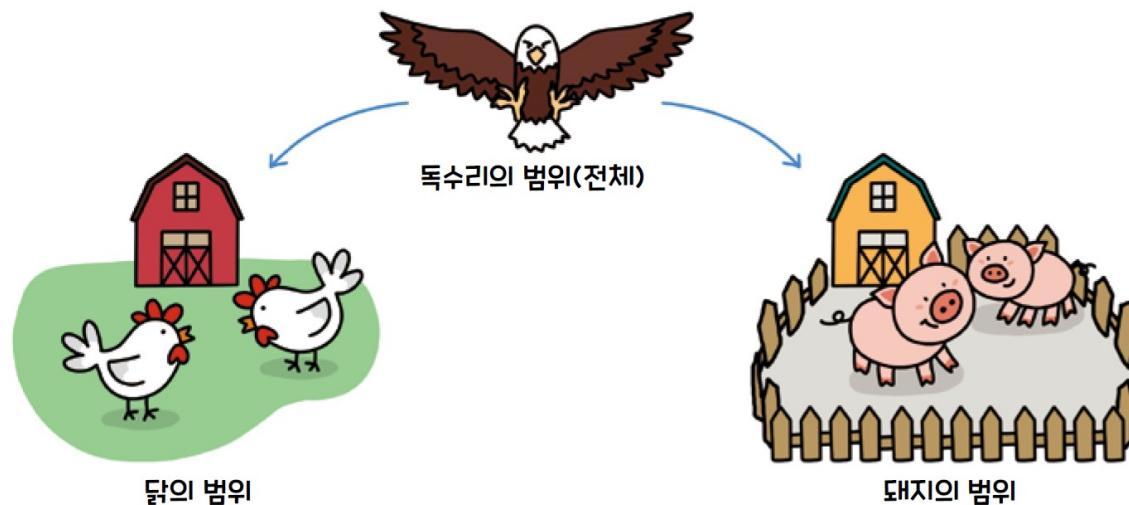


그림 8-10 동물의 유효 범위

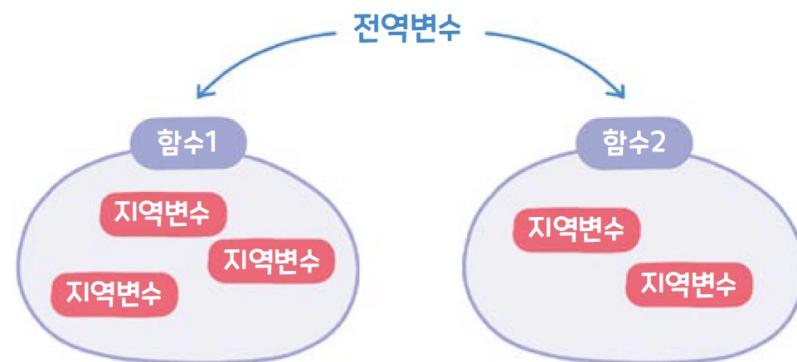


그림 8-11 변수의 유효 범위



# 지역변수와 전역변수 이해하기

## ■ 지역변수

- 말 그대로 한정된 지역(local)에서만 사용되는 변수

## ■ 전역변수

- 프로그램 전체(global)에서 사용되는 변수

### ① 지역변수의 생존 범위

함수1

a = 10

b가 무엇인지 함수1에서 안다.

함수2

b가 무엇인지 함수2에서 모른다.

### ② 전역변수의 생존 범위

함수1

b가 무엇인지 함수1에서 안다.

함수2

b가 무엇인지 함수2에서 안다.

b = 20

그림 8-12 지역변수와 전역변수의 유효 범위

# 지역변수와 전역변수 이해하기



## ■ 지역변수와 전역변수의 이름이 같은 경우

- 지역변수가 우선됨
- 1반 교실과 복도에 홍길동이라는 같은 이름을 가진 학생이 있을 경우
  - 1반에서 홍길동 학생을 부르면 복도의 홍길동이 아닌 1반의 홍길동이 대답함
  - 하지만 2반에서 선생님이 홍길동을 부르면 복도에 있는 홍길동이 대답함

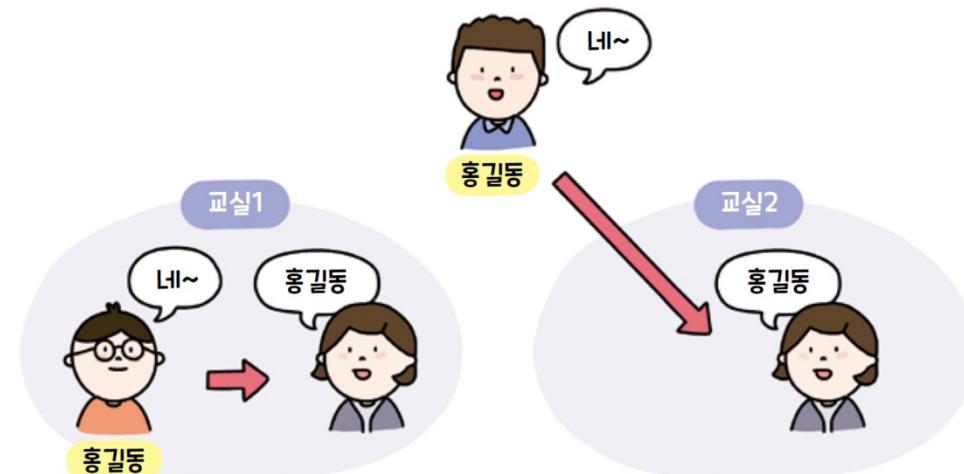


그림 8-13 이름이 같은 경우



# 지역변수와 전역변수 이해하기

## ■ 지역변수와 전역변수의 이름이 같은 경우

- 함수 내에 변수가 정의되어 있는가를 확인하면 간단히 구분할 수 있음

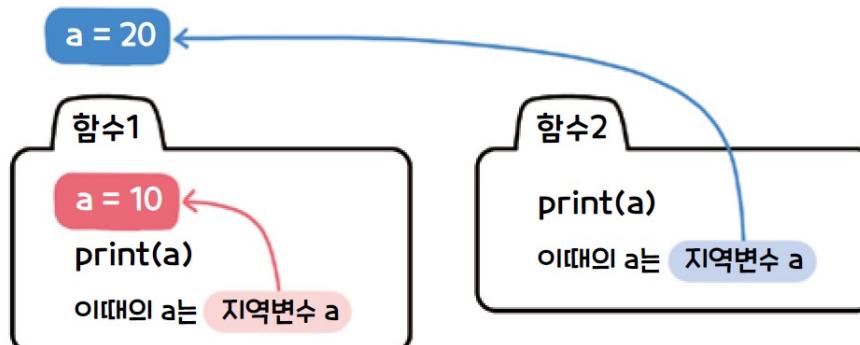


그림 8-14 지역변수와 전역변수의 공존

- 같은 a라고 해도 함수1의 a는 함수 내에서 따로 정의했으므로 지역변수임
- 함수2의 a는 함수 안에 정의된 것이 없으므로 전역변수임



# 지역변수와 전역변수 이해하기

## ■ 지역변수와 전역변수의 이름이 같은 경우

### [코드 8-9]

```
## 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()
```

### [실행결과]

```
func1()에서 a의 값 10
func2()에서 a의 값 20
```



# 지역변수와 전역변수 이해하기

## 확인문제

\* 다음 지문에서 옳은 것에는 ○, 틀린 것에는 ×를 표시하시오.

1. 전체 프로그램에서 인식되는 변수를 전역변수라 부르고, 함수 안에서만 인식되는 변수를 지역변수라 부른다. ( )
2. 함수 안에서 선언한 변수는 전역변수이고, 함수 밖에서 선언한 변수는 지역변수이다. ( )
3. 지역변수와 전역변수의 이름이 동일할 경우 전역변수가 우선된다. ( )

## 정답

Click!



# global 예약어

## ■ global 예약어

- 함수 안에서 사용되는 변수로 지역변수 대신에 무조건 전역변수로 사용하고 싶은 경우에 사용하는 예약어
- global 예약어와 함께 나오는 변수명은 무조건 전역변수임

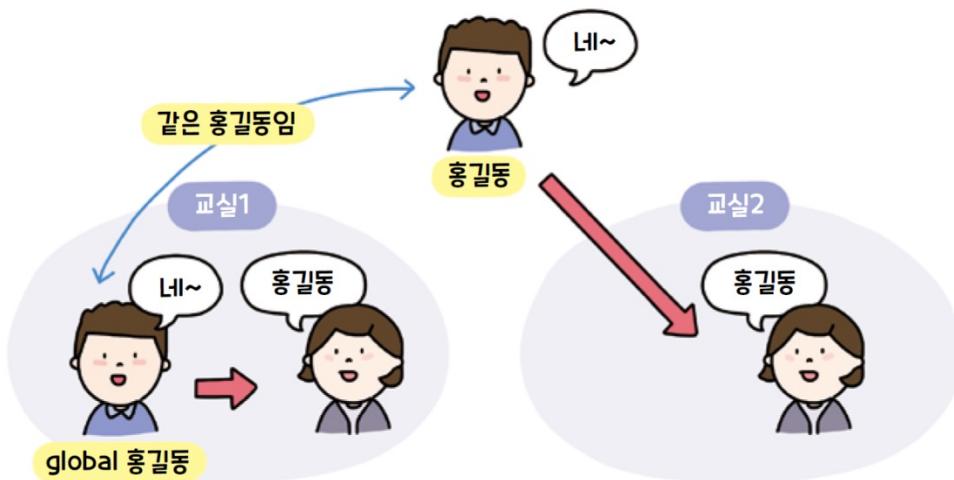


그림 8-15 global 예약어의 개념

# global 예약어



## ■ global 예약어

### [코드 8-10]

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

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

## 전역변수 선언 부분
a = 20    # 전역변수

## 메인 코드 부분
func1()
func2()
```

### [실행결과]

```
func1()에서 a의 값 10
func2()에서 a의 값 20
```

# 100일 기념일 날짜 구하기

현재 시간을 구하고, 현재 시간부터 일정 날짜가 지난 후의 날짜를 구하는 코드를 작성해 봅시다.

예를 들어 연인과 오늘부터 1일일 경우에, 100일 기념일이 언제인지 체크하는 코드를 작성합니다.

## 실행 결과

현재 날짜와 시간      ==> 2021-04-14 22:46:42.366982

100일 후 날짜와 시간 ==> 2021-07-23 22:46:42.366982



1. lab08-03.py 파일을 만들고, 날짜와 관련된 라이브러리를 임포트하기
  - datetime 및 timedelta와 관련된 함수를 사용할 수 있음

```
from datetime import datetime, timedelta
```

2. 현재 날짜를 구해서 반환하는 함수를 만들기
  - datetime.now()는 현재 날짜 및 시간을 구함

```
## 함수 정의 부분
def getCurrent() :
    curDate = datetime.now()
    return curDate
```

3. 전달받은 날짜(now)부터 지정 일자(day) 후의 날짜를 구해서 반환하는 함수 만들기
  - 전달받은 날짜인 now에 timedelta(days=일자)를 더해 지난 일자를 알아냄

```
def getAfterDate(now, day) :
    retDate = now + timedelta(days=day)
    return retDate
```

# 100일 기념일 날짜 구하기

4. 오늘 날짜 및 100일 후의 날짜를 저장할 변수를 준비함

```
## 전역변수 선언 부분
```

```
nowDate, afterDate = None, None
```

5. 현재 날짜를 구하고 출력하고 현재 날짜에서 100일이 지난 날짜를 구하고 출력하기

```
## 메인 코드 부분
```

```
nowDate = getCurrent()
print("현재 날짜와 시간 ==>", nowDate)
afterDate = getAfterDate(nowDate, 100)
print("100일 후 날짜와 시간 ==>", afterDate)
```

6. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

입력한 비밀번호가 비밀번호 규칙에 맞으면 생성되고, 규칙에 맞지 않으면 다시 생성하도록 메시지를 출력하는 프로그램을 작성해 봅시다.

#### [비밀번호의 규칙]

- (1) 8글자 이상이어야 함
- (2) 영문자 및 숫자로만 생성해야 하며 기호는 불가능

Password

abc!



weak

#### 실행 결과

새로운 비밀번호를 입력하세요 :abcd ●———— 사용자 입력

오류! 비밀번호가 규칙에 맞지 않습니다

새로운 비밀번호를 입력하세요 :abcd1234\$\$ ●———— 사용자 입력

오류! 비밀번호가 규칙에 맞지 않습니다

새로운 비밀번호를 입력하세요 :1234abcd ●———— 사용자 입력

Good~ 비밀번호가 올바르게 생성되었어요

1. lab08-04.py 파일을 만들고, 비밀번호를 전달받아서 규칙에 맞는지 체크하는 함수를 만들기
  - 비밀번호 길이가 8글자 미만이면 False를 반환하고 더 이상 진행할 필요가 없음

## 함수 정의 부분

```
def checkPassword(pwd) :  
    if len(pwd) < 8 :  
        return False
```

2. 비밀번호가 8글자 이상이면 문자 및 숫자로만 구성되었는지 체크하기

- 문자열.isalnum() 함수는 문자열이 숫자 또는 문자로만 이루어져 있으면 True를 반환
- 그 외에 문자열에 기호 등이 섞여 있다면 False

```
if pwd.isalnum() :  
    return True  
else :  
    return False
```

4. 비밀번호를 저장할 변수를 준비함

```
## 전역변수 선언 부분  
password = ""
```

5. 비밀번호를 입력받고, 비밀번호 체크함수를 호출하기

- True가 반환되면 'Good~'을 출력
- False가 반환되면 '오류!~'를 출력

```
## 메인 코드 부분  
password = input("새로운 비밀번호를 입력하세요 :")  
  
if checkPassword(password) :  
    print("Good~ 비밀번호가 올바르게 생성되었어요")  
else :  
    print("오류! 비밀번호가 규칙에 맞지 않습니다")
```

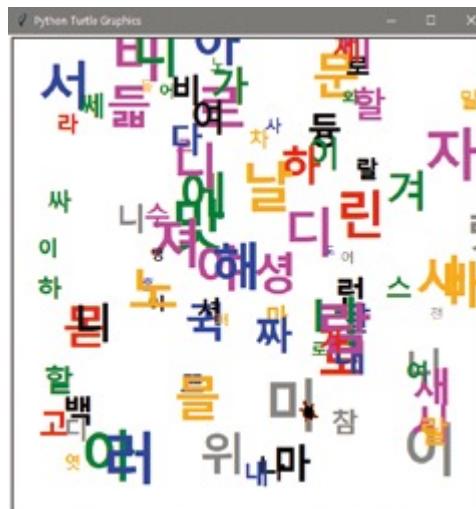
6. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

# [실전 예제] 훈민정음 그래픽 디자인

# 실전 예제 훈민정음 그래픽 디자인

## [문제]

- 세종대왕께서 창제하신 훈민정음을 거북이가 화면에 예쁘게 쓰며 그래픽 디자인스럽게 보이도록 프로그램을 작성해 보자
- 그래픽 디자인처럼 보이기 위해 랜덤한 위치와 크기 그리고 랜덤한 색상으로 한 글자씩 써지도록 구현한다.



# 실전 예제 훈민정음 그래픽 디자인

## [해결]

```
import turtle
import random

def getXYAS() :
    x, y, angle, size = 0, 0, 0, 0
    x = random.randint(-250, 250)
    y = random.randint(-250, 250)
    size = random.randint(10, 50)
    return x, y, size

## 전역변수 선언 부분 ##
koreanStr = """ 나랏말싸미 등굵에 달아 문자와로 서르 사맛디 아니할쎄
이런 전차로 어린 백성이 니르고져 홀 배 이셔도
마참내 제 뜨들 시러펴디 몯 할 노미 하니라
내 이랄 위하여 어엿비 너겨 새로 스물 여돐 짜랄 맹가노니
사람마다 해여 수비 니겨 날로 쑤메 뻔한크 하고져 할따라미니라
"""

colorList = ["red", "green", "blue", "black", "magenta", "orange", "gray"]
tX, tY, txtSize = 0,0,0

## 메인 코드 부분 ##
turtle.shape('turtle')
turtle.setup(550, 550)
turtle.screensize(500, 500)
turtle.penup()
turtle.speed(5)
```

# 실전 예제 훈민정음 그래픽 디자인

## [해결]

```
for ch in koreanStr :  
    tX, tY, txtSize = getXYAS( )  
    color = random.choice(colorList)  
    turtle.goto(tX,tY)  
    turtle.pencolor(color)  
    turtle.write(ch, font=('맑은고딕', txtSize, 'bold'))  
  
turtle.done( )
```

Preview

감사합니다 :)