# 01
# 함수 기초

# 01. 함수 기초

**여러 명이 프로그램을 개발할 때, 코드를 어떻게 작성하면 좋을까?**

① 다 같이 모여 토론하며 한 줄 한 줄 작성하기

② 가장 잘하는 사람이 혼자 작성하기

③ **필요한 부분을 나누어 작성한 후 합치기** → 가장 일반적이고 많이 사용하는 방법

# 01. 함수 기초

## ■ 함수의 개념과 장점

- **함수(function) :** 어떤 일을 수행하는 코드의 덩어리, 또는 코드의 묶음
- **함수의 장점**

  ① 필요할 때마다 호출 가능

  ② 논리적인 단위로 분할 가능

  ③ 코드의 캡슐화

# 01. 함수 기초

## ■ 함수의 선언

```
def 함수이름 (매개변수 #1 ···):
    수행문 1
    수행문 2
    return <반환값>
```

① **def : 'definition'**의 줄임말로, 함수를 정의하여 시작한다는 의미이다.

② **함수 이름 :** 함수 이름은 개발자가 마음대로 지정할 수 있지만, 파이썬에서는 일반적으로 다음과 같은 규칙을 사용한다.

- 소문자로 입력한다.
- 띄어쓰기를 할 경우에는 _ 기호를 사용한다. ex) save_model
- 행위를 기록하므로 동사와 명사를 함께 사용하는 경우가 많다. ex) find_number
- 외부에 공개하는 함수일 경우, 줄임말을 사용하지 않고 짧고 명료한 이름을 정한다.

# 01. 함수 기초

## ■ 함수의 선언

```
def 함수이름 (매개변수 #1 ···):
    수행문 1
    수행문 2
    return <반환값>
```

③ **매개변수(parameter) :** 함수에서 입력값으로 사용하는 변수를 의미하며, 1개 이상의 값을 적을 수 있다.

④ **수행문 :** 수행문은 반드시 들여쓰기한 후 코드를 입력해야 한다. 수행해야 하는 코드는 일반적으로 작성하는 코드와 같다. if나 for 같은 제어문을 사용할 수도 있고, 고급 프로그래밍을 하게 되면 함수 안에 함수를 사용하기도 한다.

# 01. 함수 기초

## 함수의 선언

- 함수 선언 작성 예시를 간단한 코드로 살펴보자

```
def calculate_rectangle_area(x, y):
    return x * y
```

**① 함수 이름 : calculate_rectangle_area**

**② 매개변수 :  x와 y라는 2개의 매개변수 사용**

**③ return 값 :  x * y**

# 01. 함수 기초

여기서 잠깐! **반환**

- 약간 어렵게 느껴질 수 있는 부분이 바로 **'반환'**이라는 개념이다. 이는 수학에서의 함수와 같은 개념이라고 생각하면 된다. 예를 들어, 수학에서 f(x) = x + 1이라고 한다면 f(1)의 값은 얼마일까? 중학교 정도의 수학을 이해하고 있다면 f(1) = 2라는 것을 알 것이다. 즉, 함수 f(x)에서 x에 1이 들어가면 2가 반환되는 것이다. 파이썬의 함수도 같은 개념이다. 수학에서 x에 해당하는 것이 매개변수, 즉 입력값이고, x + 1의 계산 과정이 함수 안의 코드이며, 그 결과가 출력값이다.

# 01. 함수 기초

## 함수의 실행 순서

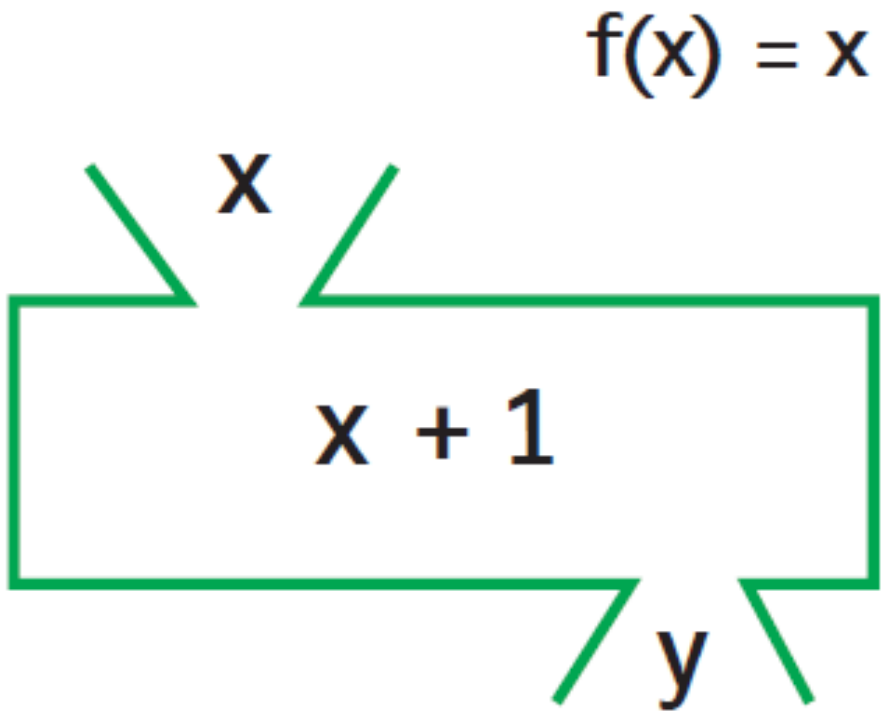
```
def calculate_rectangle_area(x,y):
    return x*y


rectangle_x = 10
rectangle_y = 20
print("사각형 x의 길이:", rectangle_x)
print("사각형 y의 길이:", rectangle_y)
area = calculate_rectangle_area(rectangle_x,rectangle_y)
print("사각형의 넓이:",area)
```

```
사각형 x의 길이: 10
사각형 y의 길이: 20
사각형의 넓이: 200
```

# 01. 함수 기초

## 프로그래밍의 함수와 수학의 함수

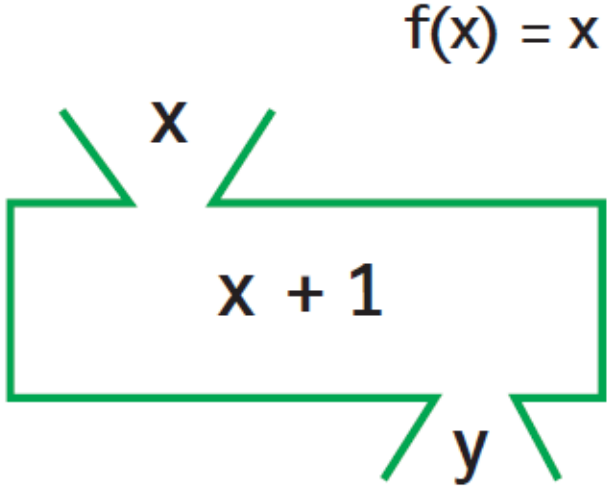
- 간단히 f(x) = x + 1을 코드로 나타낸다면, 다음과 같은 형태로 표현할 수 있다.




```
def f(x)
    y = x + 1
    return y
```

# 01. 함수 기초

## ■ 매개변수




```
def f(x)
    y = x + 1
    return y
```

**매개변수 : def f(x) : 의 x**

**인터페이스**

- 일반적으로 함수의 입력값에 대한 정의
- 매개변수는 함수에서 어떤 변수를 사용하는지를 정의

**인수**

- 실제 매개변수에 대입되는 값
- 매개변수가 설계도라면 인수는 그 설계도로 지은 건물 같은 것
- f(2)에서 2가 인수에 해당한다.

# 01. 함수 기초

## ■ 함수의 형태

| 매개변수 유무 / 반환값 유무 | 매개변수 없음 | 매개변수 있음 |
|---|---|---|
| 반환값 없음 | 함수 안 수행문만 수행 | 매개변수를 사용하여 수행문만 수행 |
| 반환값 있음 | 매개변수 없이 수행문을 수행한 후, 결과값 반환 | 매개변수를 사용하여 수행문을 수행한 후, 결과값 반환 |

# 01. 함수 기초

- 매개변수 X 반환값 X

```
def rectangle_area():
    print( 5*7)

rectangle_area()
```

```
35
```

- 매개변수 O 반환값 X

```
def rectangle_area(x,y):
    print(x*y)

rectangle_area(5,7)
```

```
35
```

# 01. 함수 기초

- 매개변수 X 반환값 O

```
def rectangle_area():
    return 5*7

print(rectangle_area())
```

```
35
```

- 매개변수 O 반환값 O

```
def rectangle_area(x,y):
 return x*y

print(rectangle_area(5,7))
```

```
35
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

- **두 개의 숫자를 입력받아 두 수의 차를 구하는 함수를 작성한다.**
- **함수 이름은 sub_number이다.**
- **인자로 number1, number2를 입력 받는다.**
- **결과값으로 result를 반환한다.**

```
첫 번째 숫자를 입력하세요 : 100
두 번째 숫자를 입력하세요 : 50
두 수의 차이는 50 입니다.
```

# 00. 실습문제

```
def sub_number(number1, number2):
    result = number1 - number2
    return result


number1 = int(input("첫번째 숫자를 입력하세요 : "))
number2 = int(input("두번째 숫자를 입력하세요 : "))
diff_number = sub_number(number1, number2)
print("두수의 차이는", diff_number, "입니다")
```

# 00. 실습문제

```
def sub_number(number1, number2):
    result = number1 - number2
    print("두수의 차이는", result, "입니다")


number1 = int(input("첫번째 숫자를 입력하세요 : "))
number2 = int(input("두번째 숫자를 입력하세요 : "))
sub_number(number1, number2)
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

**• int형 숫자 1개를 인자로 입력 받아, 그 인자보다 3이 큰 값을 반환하는 함수를 정의하여 보자.**

**① 함수 이름은 increase_3, 인자는 number → def increase_3(number):**

**② number에 3을 더한 결과값인 result 구하기 → result = number + 3**

**③ 실행의 결과값인 result를 반환한다. → return result**

```
정수를 입력하세요 : 100
100 보다 3큰 수는 103 입니다.
```

# 00. 실습문제

```
def increase_3(number1):
    result = number1 + 3
    return result


number1 = int(input("정수를 입력하세요 : "))
print(number1, "보다 3 큰 수는", increase_3(number1), "입니다")
```

# 00. 실습문제

```
def increase_3(number1):
    result = number1 + 3
    print(number1, "보다 3 큰 수는", result, "입니다")


number1 = int(input("정수를 입력하세요 : "))
increase_3(number1)
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

- **괄호 안의 인자(parameters)인 number1, number2를 입력 값으로 받아 result = (num1 + num2)을 수행하고, 그의 결과 값 result를 반환 (return문) 한 후 종료하는 이름이 sum라는 함수를 정의해 보자.**

```
첫번째 정수를 입력하세요 : 3
두번째 정수를 입력하세요 : 5
두수의 합은 3 + 5 = 15 입니다.
```

# 00. 실습문제

```
def sum(num1, num2):
    result = num1 + num2
    return result


number1 = int(input("첫번째 정수를 입력하세요 : "))
number2 = int(input("두번째 정수를 입력하세요 : "))
sum_number = sum(number1, number2)
print("두 수의 합은 %d + %d = %d 입니다." %(number1, number2, sum_number))
```

# 00. 실습문제

```
def sum(num1, num2):
    result = num1 + num2
    print("두 수의 합은 %d + %d = %d 입니다." %(num1, num2, result))

number1 = int(input("첫번째 정수를 입력하세요 : "))
number2 = int(input("두번째 정수를 입력하세요 : "))
sum(number1, number2)
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

- **괄호 안의 인자(parameters)인 number1, number2를 입력 값으로 받아 result = (num1 + num2)/2을 수행하고, 그의 결과 값 result를 반환 (return문) 한 후 종료하는 이름이 average라는 함수를 정의해 보자. (평균값은 정수로 나온다)**

```
첫번째 정수를 입력하세요 : 10
두번째 정수를 입력하세요 : 20
두 수의 평균은 (10+20)/2 = 15 입니다.
```

# 00. 실습문제

```
def average(num1, num2):
    result = int((num1 + num2)/2)
    return result


number1 = int(input("첫번째 정수를 입력하세요 : "))
number2 = int(input("두번째 정수를 입력하세요 : "))
avg_number = average(number1, number2)
print("두 수의 평균은 (%d+%d)/2 = %d 입니다." %(number1, number2, avg_number))
```

# 00. 실습문제

```
def average(num1, num2):
    result = int((num1 + num2)/2)
    print("두 수의 평균은 (%d+%d)/2=%d 입니다." %(num1, num2, result))


number1 = int(input("첫번째 정수를 입력하세요 : "))
number2 = int(input("두번째 정수를 입력하세요 : "))
average(number1, number2)
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**•사각형의 폭과 높이 인자(parameters)인 width, height를 입력 값으로 받아 사각형의 둘레를 result로 반환 (return문) 한 후 종료하는 함수(이름을 사용자가 정의)를 정의해 보자.**

```
사각형의 폭을 입력하세요 : 20
사각형의 높이를 입력하세요 : 10
사각형의 둘레의 길이는 (20+10)*2=60 입니다.
```

# 00. 실습문제

```
def cirsum(width, height):
    result = (width + height)*2
    return result


width = int(input("사각형의 폭을 입력하세요 : "))
height = int(input("사각형의 높이를 입력하세요 : "))
rect_round = cirsum(width, height)
print("사각형의 둘레의 길이는 (%d+%d)*2=%d 입니다." \
      %(width,height, rect_round))
```

**백슬래쉬(\)를 한 문장의 끝에 적어주면 파이썬 인터프리터가 다음 라인을 같은 라인으로 인식한다.**
**백슬래시를 이용해서 여러줄에 걸쳐서 한 문장을 작성해 줄 수 있다.**

# 00. 실습문제

```
def cirsum(width, height):
    result = (width + height)*2
    print("사각형의 둘레의 길이는 (%d+%d)*2=%d 입니다."%(width,height,result))


width = int(input("사각형의 폭을 입력하세요 : "))
height = int(input("사각형의 높이를 입력하세요 : "))
cirsum(width, height)
```

# 02. 함수 심화

## ■ 변수의 사용 범위

- **변수의 사용 범위(scoping rule) :** 변수가 코드에서 사용되는 범위
- **지역 변수(local variable) :** 함수 안에서만 사용
- **전역 변수(global variable) :** 프로그램 전체에서 사용

# 02. 함수 심화

## ■ 변수의 사용 범위

```
def test(t):
    print(x)
    t=20
    print("In Function:",t)


x=10
test(x)
print("In Main:",x)
print("In Main:",t)
```

```
10
Traceback (most recent call last):
In Function: 20
  File "D:\_Works\PyCharm\Test001\test_flake.py", line 9, in <module>
In Main: 10
    print("In Main:",t)
NameError: name 't' is not defined

Process finished with exit code 1
```

# 02. 함수 심화

## 변수의 사용 범위

```
def f():
    s = "I love London!"
    print(s)


s = "I love Paris!"
f()
print(s)
```

```
I love London!
I love Paris!
```

# 02. 함수 심화

## 변수의 사용 범위

```
def f():
    global s
    s = "I love London!"
    print(s)


s = "I love Paris!"
f()
print(s)
```

```
I love London!
I love London!
```

# 02. 함수 심화

## ■ 변수의 사용 범위

```
def calculate(x,y):
    total = x+y
    print("In Function =>total : %d" % (total))
    return total


a=5; b=7; total =0
print("In Main => total : %d"%(total))
sum = calculate(a,b)
print("After Calculation => Total : %d"%(total))
```

```
In Main => total : 0
In Function => total : 12
After Calculation => Total : 0
```

# 03
# 함수의 인수

# 03. 함수의 인수

- 파이썬에서 인수를 사용하는 방법

| 종류 | 내용 |
| --- | --- |
| 키워드 인수 | 함수의 인터페이스에 지정된 변수명을 사용하여 함수의 인수를 지정하는 방법 |
| 디폴트 인수 | 별도의 인수값이 입력되지 않을 때, 인터페이스 선언에서 지정한 초깃값을 사용하는 방법 |
| 가변 인수 | 함수의 인터페이스에 지정된 변수 이외의 추가 변수를 함수에 입력할 수 있게 지원하는 방법 |
| 키워드 가변 인수 | 매개변수의 이름을 따로 지정하지 않고 입력하는 방법 |

# 03. 함수의 인수

## ■ 키워드 인수(keyword arguments)

- 함수에 입력되는 매개변수의 변수명을 사용하여 함수의 인수를 지정하는 방법

```
def print_something(my_name, your_name):
    print("Hello {0}, My name is {1}".format(your_name,my_name))


print_something("Gildong","PYTHON")
print_something(your_name="PYTHON",my_name="Gildong")
```

```
Hello PYTHON, My name is Gildong
Hello PYTHON, My name is Gildong
```

# 03. 함수의 인수

## 디폴트 인수

- **디폴트 인수(default arguments)** : 매개변수에 기본값을 지정하여 사용하고, 아무런 값도 인수로 넘기지 않으면 지정된 기본값을 사용하는 방식이다

```
def print_something_2(my_name, your_name="PYTHON"):
    print("Hello {0}, My name is {1}".format(your_name, my_name))


print_something("Gildong","PYTHON")
print_something(my_name="Gildong")
```

```
Hello PYTHON, My name is Gildong
Hello PYTHON, My name is Gildong
```

# 03. 함수의 인수

## ■ 가변 인수

- 함수의 매개변수 개수가 정해지지 않고 진행해야 하는 경우가 있다. 이때 사용하는 것이 바로 가변 인수(variable-length arguments)이다.
- 가변 인수는 *(asterisk라고 부름)로 표현할 수 있는데, *는 파이썬에서 기본적으로 곱셈 또는 제곱 연산 외에도 변수를 묶어 주는 가변 인수를 만든다

# 03. 함수의 인수

## 가변 인수

```
def asterisk_test(a,b, *args):
    return a + b + sum(args)


print(asterisk_test(1, 2, 3, 4, 5))
```

```
15
```

- asterisk_test() 함수는 변수 a, b를 받고, 나머지 변수는 *args로 받음
- *args 는  가변 인수
- asterisk_test(1, 2, 3, 4, 5)에서 1과 2는 각각 a와 b에 할당되고, 나머지 인수인 3, 4, 5가 모두 *args에 할당 됨.

# 03. 함수의 인수

## 키워드 가변 인수

- 키워드 가변 인수(keyword variable-length arguments)는 매개변수의 이름을 따로 지정하지 않고 입력하는 방법으로, 이전 가변 인수와는 달리 *를 2개 사용하여 함수의 매개변수를 표시한다.
- 입력된 값은 튜플 자료형이 아닌 딕셔너리 자료형(dictionary type)으로 사용할 수 있다.
- 키워드 가변 인수는 반드시 모든 매개변수의 맨 마지막, 즉 가변 인수 다음에 선언되어야 한다.

# 04

# 좋은 코드를 작성하는 방법

# 04. 좋은 코드를 작성하는 방법

## ■ 좋은 코드의 의미

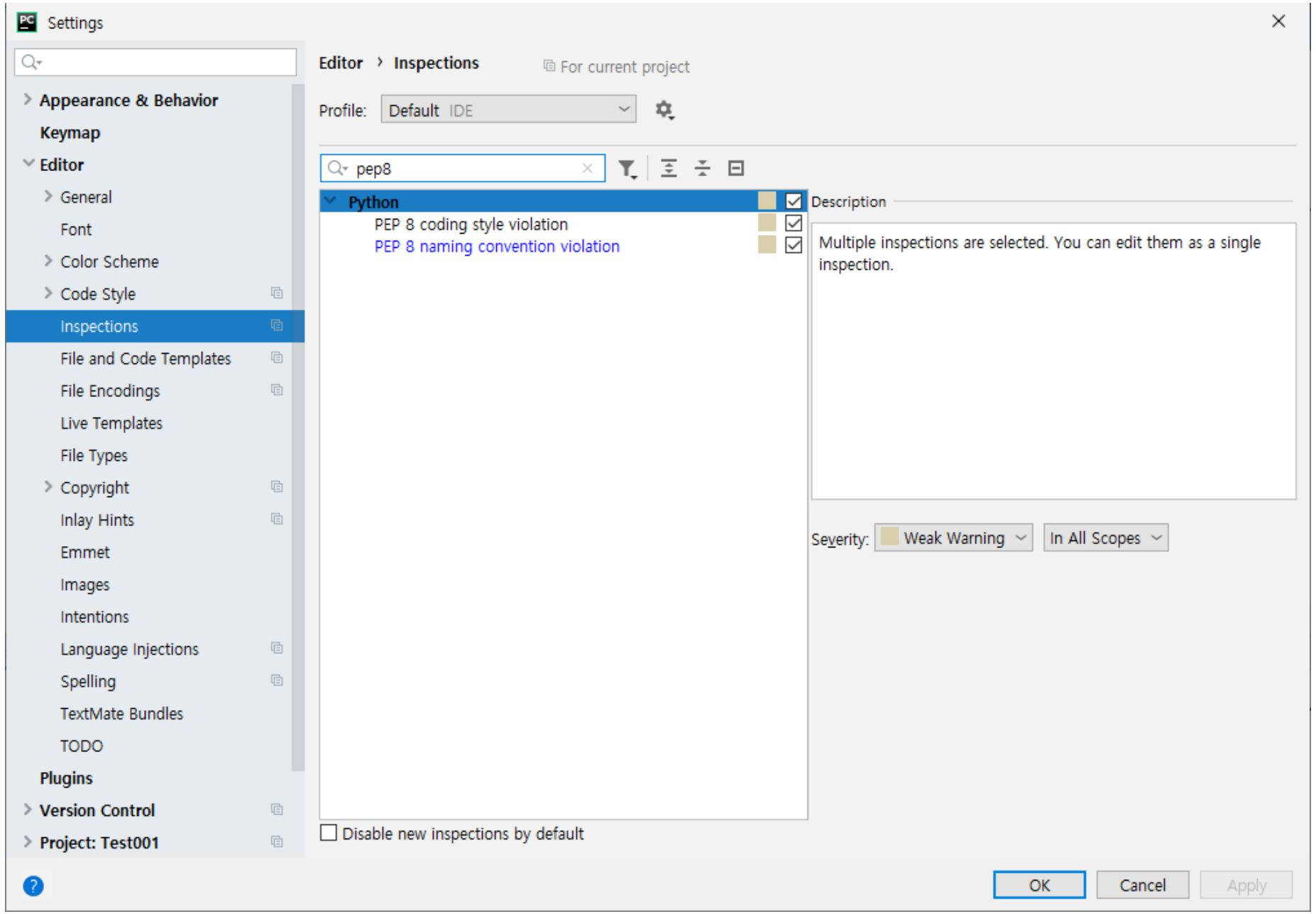
- 가독성 좋은 코드를 작성하기 위해서는 여러 가지가 필요하지만, 일단 여러 사람의 이해를 돕기 위한 규칙이 필요하다.
- 프로그래밍에서는 이 규칙을 일반적으로 코딩 규칙(coding convention)이라고 한다.

> "컴퓨터가 이해할 수 있는 코드는 어느 바보나 다 짤 수 있다. 좋은 프로그래머는 사람이 이해할 수 있는 코드를 짠다."
>
> - 마틴 파울러

# 04. 좋은 코드를 작성하는 방법

## ■ 코딩 규칙

- 들여쓰기는 4 스페이스
- 한 줄은 최대 79자까지
- 불필요한 공백은 피함
- 파이썬에서는 이러한 규칙 중 파이썬 개발자가 직접 정한 것이 있다. 이를 **(PEP 8 Python Enhance Proposal 8)**이라고 하는데, 이는 파이썬 개발자들이 앞으로 필요한 파이썬의 기능이나 여러 가지 부수적인 것을 정의한 문서이다.


Settings
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Inlay Hints
Emmet
Images
Intentions
Language Injections
Spelling
TextMate Bundles
TODO
Plugins
Version Control
Project: Test001
Editor › Inspections
For current project
Profile: Default IDE
pep8
Python
PEP 8 coding style violation
PEP 8 naming convention violation
Description
Multiple inspections are selected. You can edit them as a single inspection.
Severity: Weak Warning
In All Scopes
Disable new inspections by default
OK
Cancel
Apply

# 04. 좋은 코드를 작성하는 방법

## PEP 8의 코딩 규칙

- = 연산자는 1칸 이상 띄우지 않는다

```
variable_example  =  12          # 필요 이상으로 빈칸이 많음
variable_example = 12            # 정상적인 띄어쓰기
```

- 주석은 항상 갱신하고, 불필요한 주석은 삭제한다.
- 소문자 l, 대문자 O, 대문자 I는 사용을 금한다.

```
lIO0 = "Hard to Understand"      # 변수를 구분하기 어려움
```

- 함수명은 소문자로 구성하고, 필요하면 밑줄로 나눈다.

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수 이름

- 함수는 가능하면 짧게 작성할 것(줄 수를 줄일 것)
- 함수 이름에 함수의 역할과 의도를 명확히 드러낼 것

```
def print_hello_world():
    print("Hello, World")
def get_hello_world():
    return"Hello, World"
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수의 역할

- 하나의 함수에는 유사한 역할을 하는 코드만 포함해야 한다. 함수는 한 가지 역할을 명확히 해야 한다. 다음 코드처럼 두 변수를 더하는 함수라면 굳이 그 결과를 화면에 출력할 필요는 없다. 이름에 맞는 최소한의 역할을 할 수 있도록 작성해야 한다.

```
def add_variables(x, y):
    return x + y


def add_variables(x, y):
    print(x, y)
    return x + y
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 공통으로 사용되는 코드를 함수로 변환

```
a = 5

if(a>3):
    print("Hello World")
    print("Hello PYTHON")
if(a>4):
    print("Hello World")
    print("Hello PYTHON")
if(a>5):
    print("Hello World")
    print("Hello PYTHON")
```

```
Hello World
Hello PYTHON
Hello World
Hello PYTHON
Hello World
Hello PYTHON
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 공통으로 사용되는 코드를 함수로 변환

```
a = 5

if(a>3):
    print("Hello World")
    print("Hello PYTHON")
if(a>4):
    print("Hello World")
    print("Hello PYTHON")
if(a>5):
    print("Hello World")
    print("Hello PYTHON")
```

```
Hello World
Hello PYTHON
Hello World
Hello PYTHON
Hello World
Hello PYTHON
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 공통으로 사용되는 코드를 함수로 변환

```
def print_hello():
    print("Hello World")
    print("Hello PYTHON")

a = 5
if(a>3):
    print_hello()
if(a>4):
    print_hello()
if(a>5):
    print_hello()
```

```
Hello World
Hello PYTHON
Hello World
Hello PYTHON
Hello World
Hello PYTHON
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 복잡한 로직이 사용되었을 때, 식별 가능한 이름의 함수로 변환

```
import math
a=1; b=-2; c=1
print((-b + math.sqrt(b**2 -(4*a*c)))/(2*a))
print((-b - math.sqrt(b**2 -(4*a*c)))/(2*a))
```

```
1.0
1.0
```

# 04. 좋은 코드를 작성하는 방법

## ■ 함수 개발 가이드라인 : 함수를 만드는 경우

- 복잡한 로직이 사용되었을 때, 식별 가능한 이름의 함수로 변환

```
import math
def get_result_quadratic_equation(a,b,c):
    values = []
    values.append((-b + math.sqrt(b**2 -(4*a*c)))/(2*a))
    values.append((-b - math.sqrt(b**2 -(4*a*c)))/(2*a))
    return values


a=1; b=-2; c=1
print(get_result_quadratic_equation(a,b,c))
```

```
[1.0, 1.0]
```

# 00. 실습문제

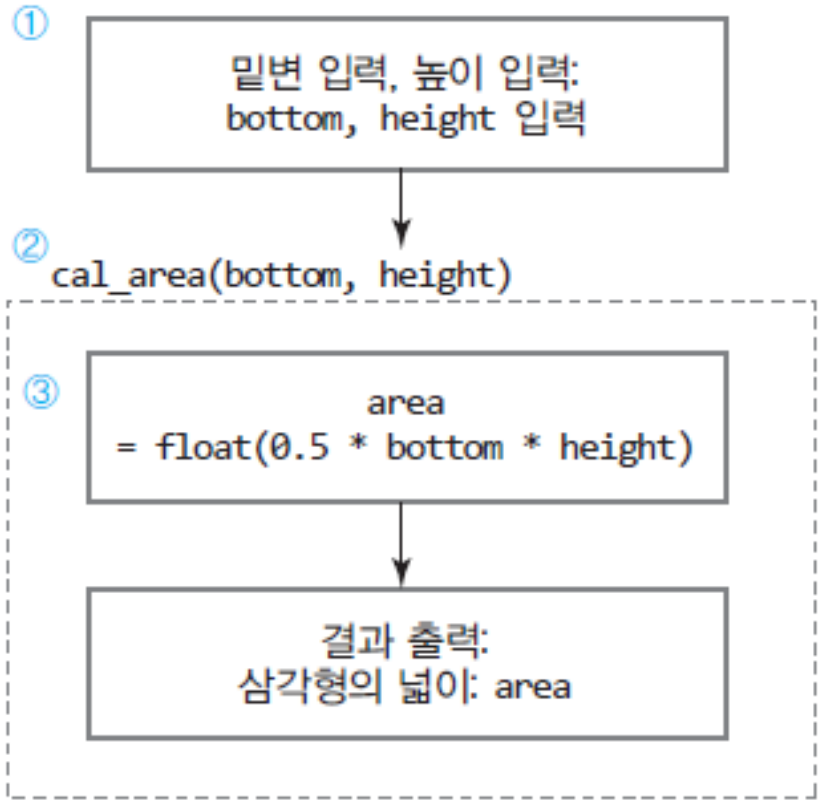
## PRACTICE

**I 요구사항 I**
**삼각형의 밑변과 높이를 입력받아, 삼각형의 면적을 출력하는 프로그램을 작성해보자.**
**프로그램의 상세 요구사항은 다음과 같다.**
**1. 사용자가 밑변 bottom, 높이 height를 입력한다. 이때, 입력받는 인자는 정수형 (integer type)이다.**
**2. 다음과 같은 수식을 사용하여 면적을 계산한다.**
**→ area = float(0.5 * bottom * height)**
**3. 삼각형의 면적을 출력한다.(소수점 한자리로 결과를 표시하라)**

```
삼각형의 밑변의 길이를 입력하세요 : 20
삼각형의 높이를 입력하세요 : 10
삼각형의 면적은 20 * 10 * 0.5 = 100.0 입니다.
```

# 00. 실습문제

① 사용자에게 bottom, height를 입력 받는다.
② 입력받은 bottom, height로 삼각형의 면적 area를 return하는 cal_area() 함수를 정의한다.
③ bottom, height를 이용하여, area를 계산하여 출력한다.



# 00. 실습문제

```
def cal_area(bottom, height):
    area = float(0.5 * bottom * height)
    return area


bottom = int(input("밑변을 입력하세요 : "))
height = int(input("높이를 입력하세요 : "))
triangle_area = cal_area(bottom, height)
print("삼각형의 면적은 (%d * %d * 0.5) = %.1f" \
      % (bottom, height, triangle_area))
```

**백슬래쉬(\)를 한 문장의 끝에 적어주면 파이썬 인터프리터가 다음 라인을 같은 라인으로 인식한다.**
**백슬래시를 이용해서 여러줄에 걸쳐서 한 문장을 작성해 줄 수 있다.**

# 00. 실습문제

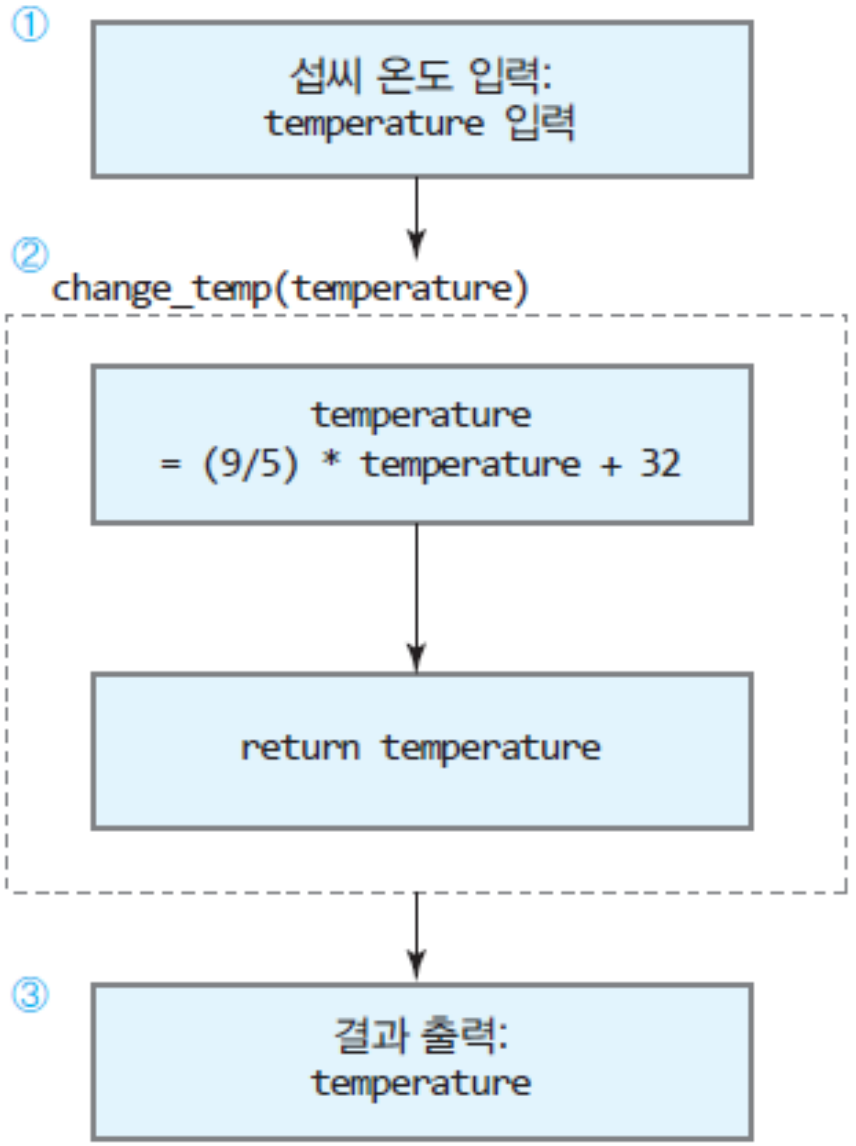
PRACTICE

**I 요구사항 I**
**섭씨 온도를 입력으로 받아서, 화씨 온도로 변환하는 함수를 작성해보자.**
**프로그램의 상세 요구사항은 다음과 같다.**
**1. 사용자가 변환할 섭씨 온도를 입력한다.**
**2. 다음과 같은 수식을 사용하여 화씨 온도로 변환한다.**
**- 화씨 온도 = ((9/5) * 섭씨 온도 + 32)**
**3. 화씨 온도를 출력한다.(소수점 1자리표시)**

```
섭씨온도를 입력하세요(실수형) : 32
화씨온도는 89.6 입니다.
```

# 00. 실습문제

① 사용자에게서 섭씨 온도 temperature를 입력받는다.
② temperature를 입력받고, 변환 수식에 따라 화씨 온도로 변한 값을 return하는 change_temp() 함수를 정의한다.
③ 화씨 온도를 출력한다



# 00. 실습문제

```
def change_temp(temperature):
    temperature = (9/5) * temperature + 32
    return temperature


temperature=float(input('섭씨 온도를 입력하세요(실수형) : '))
print('화씨 온도는', temperature, '입니다'))
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**meter를 입력 받아 yard와 feet로 변환하여 출력하는 함수를 작성해보자.**
**1. 사용자가 변환할 meter를 입력한다.**
**2. 다음과 같은 수식을 사용하여 feet와 yard로 변환한다.**
- **feet = meter / 0.305**
- **yard = meter * 1.0936**

**3. 변환된 feet와 yard를 다음과 같이 출력한다.**

```
meter를 입력 하시오 : 50
50.0 meter = 54.67 yard
50.0 meter = 163.93 feet
```

# 00. 실습문제

**• 함수의 이름은 change_meter, meter를 인자로 받는 함수를 정의한다.**

**→ def change_meter(meter)**

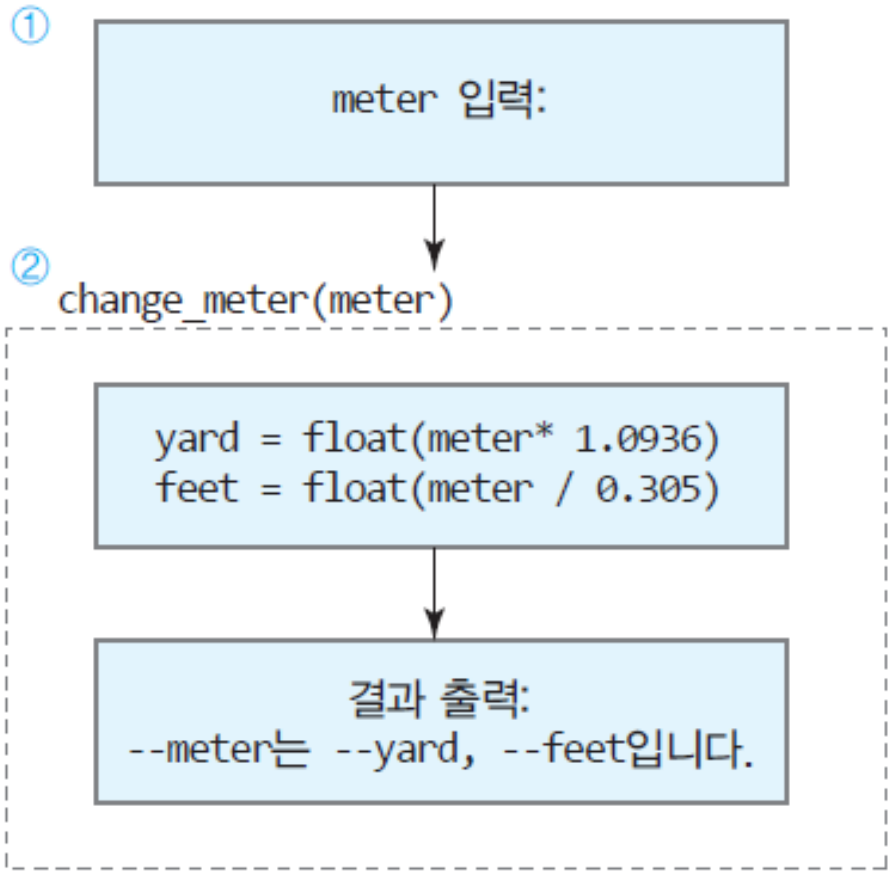
**• meter를 입력받을 때, 숫자(float)로 형 변환을 미리 해주어야 한다.**

**→ float(input('meter 입력: '))**

• 다음과 같은 알고리즘으로 프로그램을 작성한다.

① 사용자에게 meter를 입력받는다.

② meter를 입력받고, 변환한 yard와 feet를 출력하는 change_meter() 함수를 정의한다.
입력받은 meter을 인자로 넣어 change_meter() 함수를 호출하여 결과를 출력한다.



# 00. 실습문제

```
def change_meter(meter):
    yard = float(meter * 1.0936)
    feet = float(meter / 0.305)
    print("%.1f meter = %.2f yard\n%.1f meter = %.2f feet" \
          %(meter, yard, meter, feet))


meter = float(input('meter 입력: '))
change_meter(meter)
```

# 00. 실습문제

```
def change_meter(meter):
    yard = float(meter * 1.0936)
    feet = float(meter / 0.305)
    return (yard,feet)


meter = float(input('meter 입력: '))
yard, feet = change_meter(meter)
print("%.1f meter = %.2f yard\n%.1f meter = %.2f feet" \
      %(meter,yard,meter,feet))
```

# 00. 실습문제

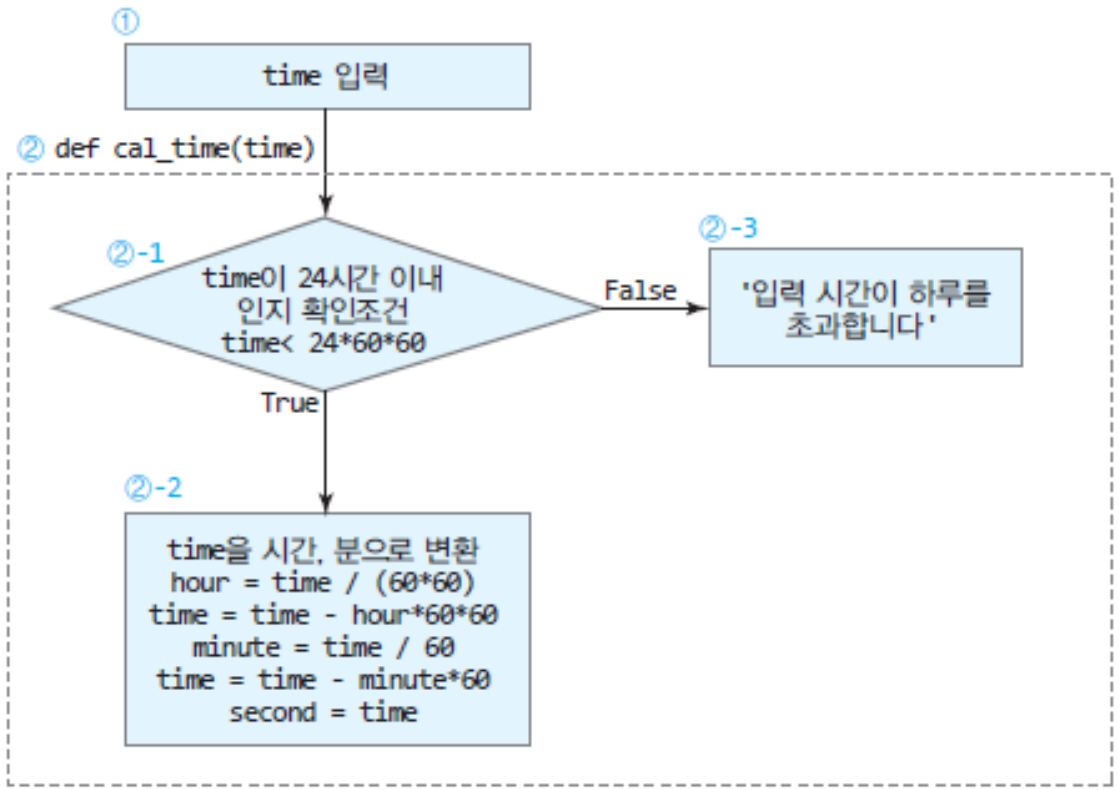
PRACTICE

**I 요구사항 I**
**초를 입력하면 몇 시, 몇 분, 몇 초인지를 출력하는 함수를 작성해보자.**
**1. 사용자가 입력 받은 시간은 하루 내의 시간이어야 한다. (24*60*60 보다 작아야 함)**
**2. 1분은 60초, 1시간은 3,600초이다**

```
초를 입력 하시오 : 12000
3 시 20 분 0 초 입니다
초를 입력 하시오 : 1000000
입력 시간이 하루를 초과합니다.
```

# 00. 실습문제

① time을 입력 받는다,
② 함수의 이름은 cal_time, time을 인자로 받는 함수를 정의한다. → def cal_time(time)
②-1 입력받은 time이 하루 내의 시간인지 아닌지를 확인하는 조건을 정의한다.
→ if time < 24*60*60
②-2 time이 하루 내의 시간인 경우 수행을 계속한다.
②-3 time이 하루 내의 시간을 초과하는 경우 "입력 시간이 하루를 초과합니다." 를 출력한다



# 00. 실습문제

```
def cal_time(time):
    if time < 24*60*60:
        hour = int(time/(60*60))
        time = time - hour*60*60
        minute = int(time/60)
        time = time - minute*60
        second = time
        print(hour,'시',minute,'분',second,'초 입니다')
    else :
        print('입력 시간이 하루를 초과합니다.')


time = int(input('초를 입력 하시오 : '))
cal_time(time)
```

# 00. 실습문제

PRACTICE

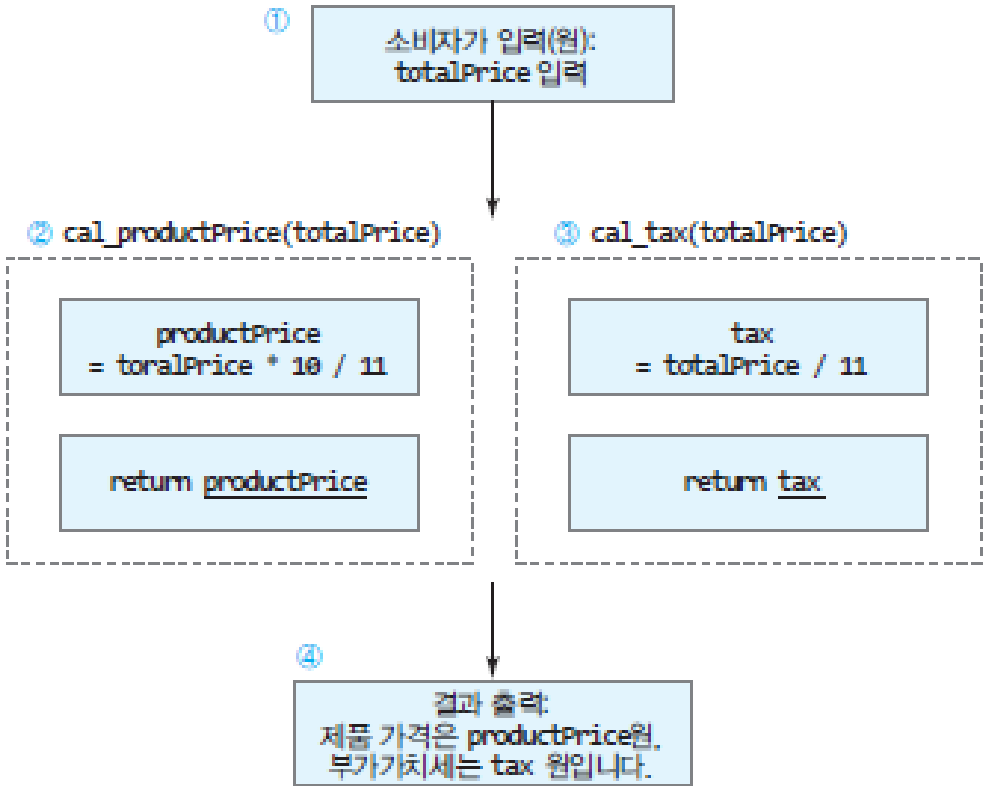
**I 요구사항 I**

**우리가 구매하는 제품의 소비자가에는 제품 가격과 제품 가격의 10%에 해당하는 부가가치세가 포함되어 있다.**
**제품의 소비자가를 입력 받아 제품 가격과 부가가치세를 출력하는 프로그램을 작성해보자.**
**프로그램의 상세 요구사항은 다음과 같다.**
**1. 사용자가 제품의 소비자가 totalPrice를 입력한다. 이때, totalPrice는 integer 타입이다.**
**2. 구해진 제품 가격 productPrice와 부가가치세 tax에 대해 다음과 같이 출력한다.**
**"제품 가격은 productPrice원, 부가가치세는 tax원입니다."**

```
소비자가 입력(원): 11000
제품가격은 10000.0원, 부가가치세는 1000.0원 입니다.
```

# 00. 실습문제

① 사용자에게 totalPrice를 입력받는다.
② totalPrice를 인자로 받고, 제품 가격 productPrice를 return하는 cal_product_price() 함수를 정의한다.
③ totalPrice를 인자로 받고, 부가가치세 tax를 return 하는 cal_tax() 함수를 정의한다.
④ 결과인 productPrice와 tax를 출력한다.



# 00. 실습문제

```
def cal_product_price(totalPrice):
    productPrice = totalPrice * 10 / 11
    return productPrice


def cal_tax(totalPrice):
    tax = totalPrice / 11
    return tax


totalPrice = int(input('소비자가 입력(원): '))
productPrice = cal_product_price(totalPrice)
tax = cal_tax(totalPrice)
print('제품 가격은 %.1f원, 부가가치세는 %.1f원입니다.'%(productPrice, tax))
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**통신사에서는 사용 개월 수와 신용카드 종류에 따라 할인율을 정하고 있다. 아래의 할인 제도에 따라 사용 개월 수, 신용카드 코드, 계약 금액을 입력받아 최종 요금을 출력하는 프로그램을 작성해보자.**
**(다음페이지의 할인제도를 참조하라)**

```
계약 금액 입력: 50000
사용 개월 수 입력: 10
카드 코드 입력: 11
최종 요금은 42500.0 원입니다.
```

# 00. 실습문제

프로그램의 상세 요구사항은 다음과 같다.

1. 사용자가 다음을 입력한다.
   계약 금액 contractPrice
   사용 개월 수 period
   카드 코드 cardCode

2. 입력한 값을 다음 수식에 따라 최종요금으로 계산한다.
   - 사용 개월 수에 따른 할인 금액
     = (계약 금액) * (사용 개월 수 할인율)

   - 신용카드 종류에 따른 할인 금액
     = (계약 금액) * (신용카드 할인율)

3. 최종요금 finalPrice를 다음과 같이 출력한다.
   "최종 요금은 finalPrice원입니다."

| 사용 개월 수 | 할인율 |
|---|---|
| 6개월 이하 | 0% |
| 6개월 초과 12개월 이하 | 10% |
| 12개월 초과 | 20% |

| 카드 코드 | 할인율 |
|---|---|
| 11 | 5% |
| 12 | 8% |
| 13 | 12% |

# 00. 실습문제

① 사용자에게 계약 금액 contractPrice, 사용 개월 수 period, 카드 코드 cardcode를 입력 받는다.

② contractPrice와 period를 인자로 받아, 사용 개월 수에 따른 할인금액을 return 하는 periodDiscount() 함수를 정의한다. 위의 상세 요구사항에 따라 입력된 사용 개월 수 기준으로 할인율이 결정된다.

③ contractPrice와 cardCode를 인자로 받아, 신용카드 종류에 따른 할인금액을 return하는 creditCardDiscount() 함수를 정의한다. 위의 상세 요구사항에 따라 입력된 카드 코드 기준으로 할인율이 결정된다.

④ 최종 요금 finalPrice는 계약 금액에서 각각의 함수 결과인 '사용 개월 수에 따른 할인금액'과 '신용카드 종류에 따른 할인금액'을 뺀 금액이다.

⑤ 계산이 완료된 결과값 finalPrice를 출력한다.

# 00. 실습문제

```
def period_discount(contractPrice, period):
    if period > 12:
        discount = 20
        discounted_price = contractPrice * discount/100
    elif period > 6 :
        discount = 10
        discounted_price = contractPrice * discount/100
    else:
        discount = 0
        discounted_price = contractPrice * discount/100
    return int(discounted_price)
```

# 00. 실습문제

```
def credit_card_discount(contractPrice, cardCode):
    if cardCode == 11:
        credit_discount = contractPrice * 5/100
    elif cardCode == 12:
        credit_discount = contractPrice * 8/100
    elif cardCode == 13:
        credit_discount = contractPrice * 12/100
    return int(credit_discount)


contractPrice = int(input('계약 금액 입력: '))
period = int(input('사용 개월 수 입력: '))
cardCode = int(input('카드 코드 입력: '))
finalPrice = contractPrice - period_discount(contractPrice, period) -
credit_card_discount(contractPrice, cardCode)
print('최종 요금은', finalPrice,'원입니다.')
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**수지는 오늘 저녁 메뉴를 고민하다가 동전 던지기를 하여 앞면이 나오면 중국요리를, 뒷면이 나오면 일본요리를 먹기로 결정했다. 동전의 앞/뒷면을 입력하여 무슨 메뉴를 선택할지 결정해주는 프로그램을 return문 없이 작성해보자.**
**함수이름은 `decision_food` 로 하자**

중국 요리로 결정되었습니다.

# 00. 실습문제

- if 동전의 면 == '앞' → '중국 요리'
- else → '일본 요리'
- 결정된 요리 출력
  - 동전의 면이 앞면인 경우에는, '중국 요리'로 결정되어 출력된다.
  - 동전의 면이 뒷면인 경우에는, '일본 요리'로 결정되어 출력된다.

# 00. 실습문제

```
import random

def decision_food(coin):
    if coin == '앞':
        print("중국 요리로 결정되었습니다.")
    else:
        print("일본 요리로 결정되었습니다.")

coin = random.choice(['앞','뒤'])
decision_food(coin)
```

```
중국 요리로 결정되었습니다.
```

# 함수 + 제어문

# 00. 실습문제

PRACTICE

**I 요구사항 I**

**6층짜리 건물이 있다. 사용자가 가고 싶은 층을 선택하면, 사용자가 서 있는 층보다 높은지 낮은지 계산하여 높을 경우 올라가고 낮을 경우 내려가며 현재 층을 출력하는 프로그램을 작성해보자. 프로그램의 상세 요구사항은 다음과 같다.**

**1. 사용자가 가고자 하는 층 inputLocation을 입력한다. 사용자가 현재 있는 층 nowLocation을 입력한다.**
**2. 엘리베이터 시스템의 설계는 다음과 같다.**
  **1) 사용자가 1~6 이외의 숫자는 누를 수 없다.**
  **2) 사용자가 현재 엘리베이터가 서 있는 층을 선택하면 다른 층을 선택하라는 메시지를 출력한다.**
**3. 각 경우에 해당하는 출력 메시지는 다음과 같다.**
  **1) 사용자가 가고자 하는 층을 현재 층으로 입력하거나 범위 외의 층을 입력한 경우**
    **→ "다른 층(1~6)을 입력해주세요."**
  **2) 엘리베이터의 층 이동이 있는 경우 → "현재 층은 floor 입니다."**
  **3) 가고자 하는 층에 도착한 경우 → "inputLocation층에 도착하였습니다. 안녕히 가세요."**

```
소비자가 입력(원): 11000
제품가격은 10000.0원, 부가가치세는 1000.0원 입니다.
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**6층짜리 건물이 있다. 사용자가 가고 싶은 층을 선택하면, 사용자가 서 있는 층보다 높은지 낮은지 계산하여 높을 경우 올라가고 낮을 경우 내려가며 현재 층을 출력하는 프로그램을 작성해보자. 프로그램의 상세 요구사항은 다음과 같다.**

```
가고자 하는 층 입력: 9
현재 위치 입력: 1
다른 층(1~6)을 눌러주세요.

가고자 하는 층 입력: 5
현재 위치 입력: 1
현재 층은 1 입니다.
현재 층은 2 입니다.
현재 층은 3 입니다.
현재 층은 4 입니다.
현재 층은 5 입니다.
5 층에 도착하였습니다. 안녕히 가세요.
```

# 00. 실습문제

```
def goDownfloor(now, target):
    for floor in range(now, target-1, -1):
        print('현재 층은', floor,'입니다.')
    print(target, '층에 도착하였습니다. 안녕히 가세요.')

def goUpfloor(now, target):
    for floor in range(now, target+1):
        print('현재 층은', floor,'입니다.')
    print(target, '층에 도착하였습니다. 안녕히 가세요.')

inputLocation = int(input('가고자 하는 층 입력: '))
nowLocation = int(input('현재 위치 입력: '))
if ((inputLocation==nowLocation) or (inputLocation<1) or (inputLocation>6)):
    print("다른 층(1~6)을 눌러주세요.")
else:
    if(nowLocation > inputLocation):
        goDownfloor(nowLocation,inputLocation)
    else:
        goUpfloor(nowLocation,inputLocation)
```

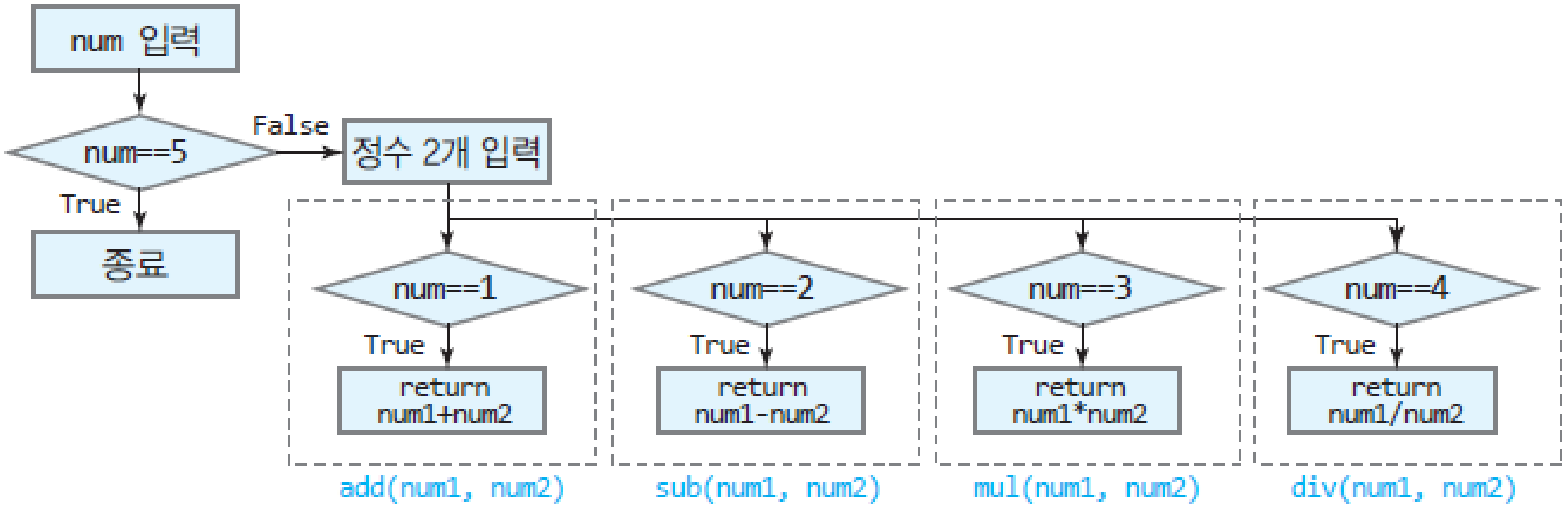
# 00. 실습문제

## PRACTICE

**I 요구사항 I**
**덧셈, 뺄셈, 곱셈, 나눗셈을 하는 계산기 프로그램을 만들려고 한다. 프로그램의 상세 요구사항은 다음과 같다.**
**1. 모든 함수는 결과를 반환한다.**
**2. 뺄셈 함수의 경우 어떤 식으로 입력이 들어오든 큰 수에서 작은 수를 뺀다.**
**3. 두 정수의 입력 값은 float형으로 받는다.**
**4. 연산 결과를 출력할 수 있도록 한다.**
**5. 무한히 돌 수 있도록 하며, 종료가 가능하도록 한다.**

```
*** 계산기 ***
1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 5.종료 :3
정수 2개 입력:
5
6
결과: 30.0
1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 5.종료 :2
정수 2개 입력:
3
6
결과: 3.0
```

# 00. 실습문제


num 입력
num==5
False
정수 2개 입력
True
종료
num==1
True
return
num1+num2
num==2
True
return
num1-num2
num==3
True
return
num1*num2
num==4
True
return
num1/num2
add(num1, num2)
sub(num1, num2)
mul(num1, num2)
div(num1, num2)

# 00. 실습문제

```
def add(num1, num2):
    result = num1 + num2
    return result

def sub(num1, num2):
    if num1 > num2:
        result = num1 - num2
    else:
        result = num2 - num1
    return result

def mul(num1, num2):
    result = num1 * num2
    return result

def div(num1, num2):
    result = num1 / num2
    return result
```

# 00. 실습문제

```
print("*** 계산기 ***")
num_result = 0
while(True):
    num = int(input("1.덧셈 2.뺄셈 3.곱셈 4.나눗셈 5.종료 :"))
    if num==5:
        break
    else:
        print("정수 2개 입력: ")
        num1 = float(input())
        num2 = float(input())
        if num==1:
            num_result = add(num1, num2)
        elif num == 2:
            num_result = sub(num1, num2)
        elif num == 3:
            num_result = mul(num1, num2)
        else:
            num_result = div(num1, num2)
print("결과: ", num_result)
```

# 00. 실습문제

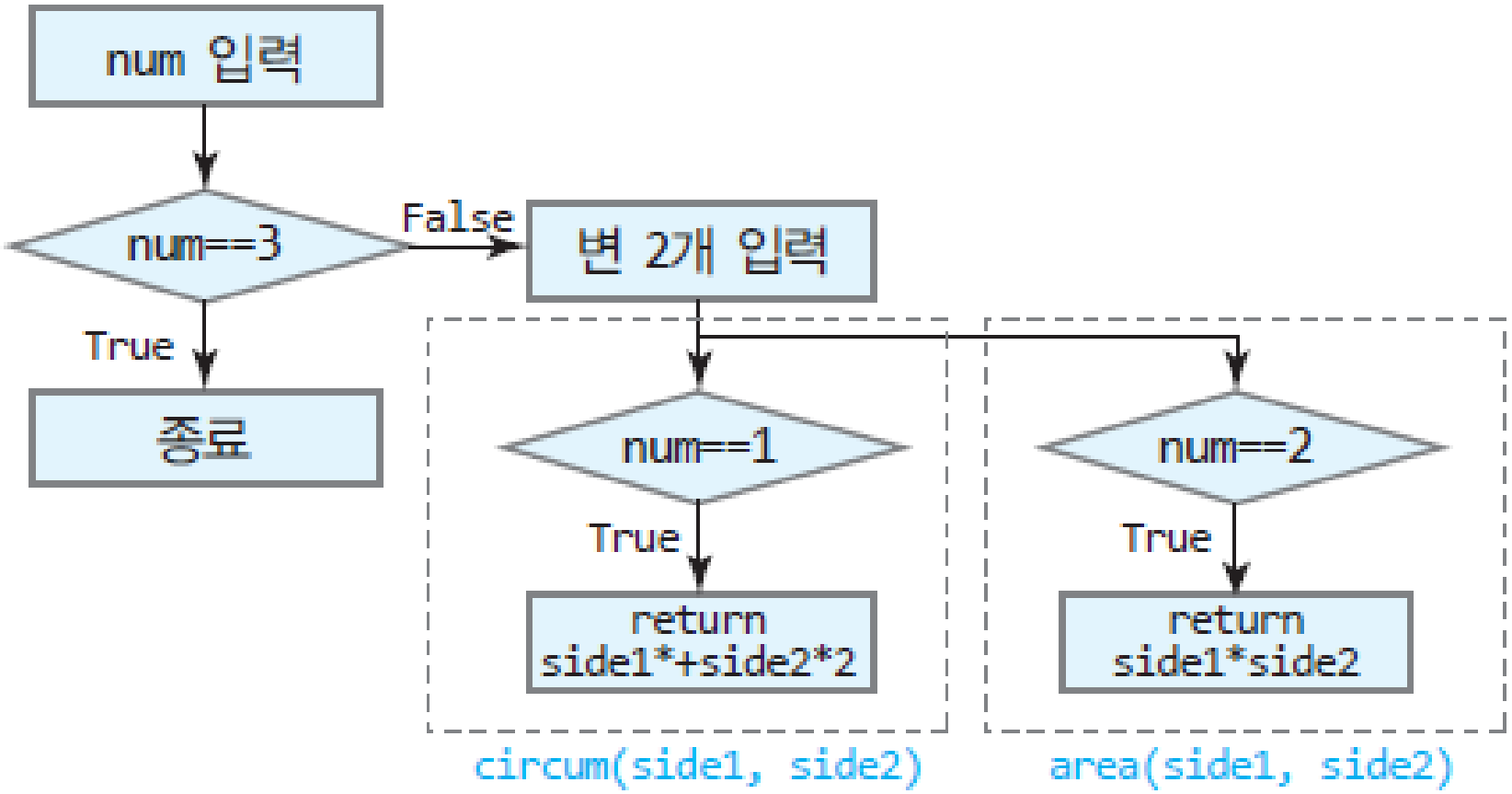
PRACTICE

**I 요구사항 I**
**사각형의 둘레 구하는 함수를 이용해서 사각형의 둘레와 넓이를 구하는 프로그램을 작성해 보자**

**1. 사용자가 종료를 입력하기 전엔 무한히 돌아야 한다.**
**2. 결과를 출력해야한다.**

```
1.둘레 2.넓이 3.종료 : 2
두 변 길이 입력:
2
3
넓이는 6
1.둘레 2.넓이 3.종료 : 1
두 변 길이 입력:
1
1
둘레는 4
1.둘레 2.넓이 3.종료 : 3
```

# 00. 실습문제


num 입력
num==3
False
변 2개 입력
True
종료
num==1
num==2
True
True
return
side1*+side2*2
return
side1*side2
circum(side1, side2)
area(side1, side2)

# 00. 실습문제

```
def circum(side1, side2):
    result = side1*2+side2*2
    return result


def area(side1, side2):
    result = side1 * side2
    return result
```

# 00. 실습문제

```
squ_result = 0
while(True):
    num = int(input("1.둘레 2.넓이 3.종료 : "))
    if num==3:
        break
    else:
        print("두 변 길이 입력: ")
        side1 = int(input())
        side2 = int(input())
        if num==1:
            squ_result = circum(side1, side2)
            print("둘레는 ", squ_result)
        else:
            squ_result = area(side1, side2)
            print("넓이는 ", squ_result)
```

# 연습문제

PRACTICE

**I 요구사항 I**
**6명의 학생이 시험을 치렀다. 각 시험 점수를 score라는 리스트에 넣고 이에 대해 최고점을 구하는 함수를 for문을 이용해서 작성해 보시오.**
**score = [95, 90, 45, 10, 80, 100]**

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**아래의 프로그램에서 어느 부분을 따로 함수로 정의하면 효율적일지 생각해보고, 함수를 새로 정의해보자.**

```
num1 = 2; num2 = 3; num3 = 4; num4 = 5 ; num5 = 6
square_num1 = num1 * num1
print(square_num1)
square_num2 = num2 * num2
print(square_num2)
square_num3 = num3 * num3
print(square_num3)
square_num4 = num4 * num4
print(square_num4)
square_num5 = num5 * num5
print(square_num5)
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**아래의 프로그램에서 어느 부분을 따로 함수로 정의하면 효율적일지 생각해보고, 함수를 새로 정의해보자.**

```
def square_num(num):
    return num*num


num1 = 2; num2 = 3; num3 = 4; num4 = 5 ; num5 = 6
print(square_num(num1))
print(square_num(num2))
print(square_num(num3))
print(square_num(num4))
print(square_num(num5))
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**
**길이가 서로 다른 3개의 단어를 입력받아 가장 길이가 긴 단어와 가장 길이가 짧은 단어를 찾는 함수를 만들어 보자**
**1. 가장 길이가 긴 단어를 찾는 함수 → longest()**
**2. 가장 길이가 짧은 단어를 찾는 함수 → shortest()**

```
print(longest('one', 'second', 'third'))
print(shortest('one', 'second','third'))
```

# 00. 실습문제

```
def longest(str1,str2,str3) :
    longest = ''
    if ((len(str1) > len(str2)) and (len(str1) > len(str3))) :
        longest = str1
    elif ((len(str2) > len(str1)) and (len(str2) > len(str3))) :
        longest = str2
    elif ((len(str3) > len(str1)) and (len(str3) > len(str2))) :
        longest = str3
    else :
        print('wrong input')
    return longest

def shortest(str1,str2,str3) :
    shortest = ''
    if ((len(str1) < len(str2)) and (len(str1) < len(str3))) :
        shortest = str1
    elif ((len(str2) < len(str1)) and (len(str2) < len(str3))) :
        shortest = str2
    elif ((len(str3) < len(str1)) and (len(str3) < len(str2))) :
        shortest = str3
    else :
        print('wrong input')
    return shortest

print(longest('one','second','third'))
print(shortest('one','second','third'))
```

# 퀴즈

3가지 단어로 구성된 wordlist

list_fruit = ['apple', 'banana', 'tomato']

list_animal = ['bear', 'elephant', 'monkey']

list_instrument = ['guitar', 'piano', 'harmonica']

일 때, 속한 단어들의 길이의 합을 나타내는 프로그램을 작성해 보자

예를 들어 ['age', 'salt', 'zoo'] 라는 리스트가 있다면 단어 길이의
합은 3+4+3 = 10 이다

- 문제 해결 알고리즘

- **단어로 구성된 리스트** wordlist**를 인자로 받아 단어들의 길이의 합을 반환해주는 함수를 정의한다**. → def len_word(wordlist)

- **리스트에 속하는 단어를 탐색한다**. → for word in wordlist:

- **단어들의 길이를 합한다**. → len(word)

# 퀴즈

```
def len_word(wordlist):
    lensum = 0
    for word in wordlist:
        lensum += len(word)
    return lensum


list_fruit = ['apple','banana','tomato']
list_animal = ['bear','elephant','monkey']
list_instrument = ['guitar','piano','harmonica']

print(len_word(list_fruit))
print(len_word(list_animal))
print(len_word(list_instrument))
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

**여러 개의 점수를 list로 받아서, 그중 가장 높은 점수를 구하는 함수를 작성해 보자.**

- **리스트는 math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89] 이다**
- **가장 높은 점수를 구하는 함수 max_score 를 사용한다.**

```
[45, 66, 70, 83, 50, 77, 87, 92, 73, 89] 의 가장 높은 점수는 92 입니다.
```

# 00. 실습문제

```
def max_score(score_list):
    max_num = 0
        for score in score_list:
   if score > min_num:
       max_num = score
   return max_num

math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89]
max_value = max_score(math_score)
print(math_score,"의 가장 높은 점수는",max_value,"입니다.")
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

**여러 개의 점수를 list로 받아서, 그중 가장 낮은 점수를 구하는 함수를 작성해 보자.**

- **리스트는 math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89] 이다**
- **가장 높은 점수를 구하는 함수 min_score 를 사용한다.**

```
[45, 66, 70, 83, 50, 77, 87, 92, 73, 89] 의 가장 낮은 점수는 45 입니다.
```

# 00. 실습문제

```
def min_score(score_list):
    min_num = 9999
        for score in score_list:
   if score < min_num:
       min_num = score
   return min_num

math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89]
min_value = min_score(math_score)
print(math_score,"의 가장 낮은 점수는",min_value,"입니다.")
```

# 00. 실습문제

PRACTICE

**I 요구사항 I**

**여러 개의 점수를 list로 받아서, 그중 가장 높은 점수와 가장 낮은 점수의 차를 구하는 함수를 작성해 보자.**

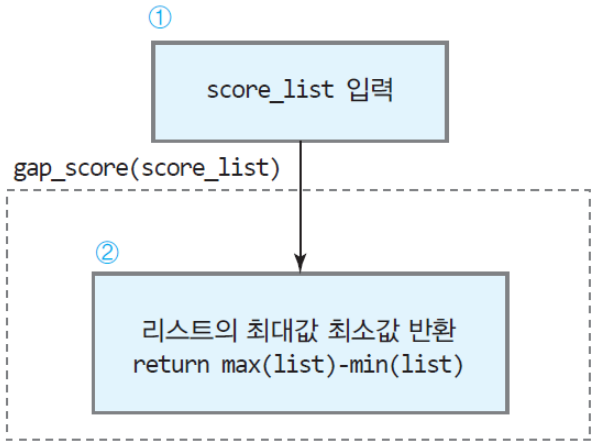
**math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89] 를 사용한다.**

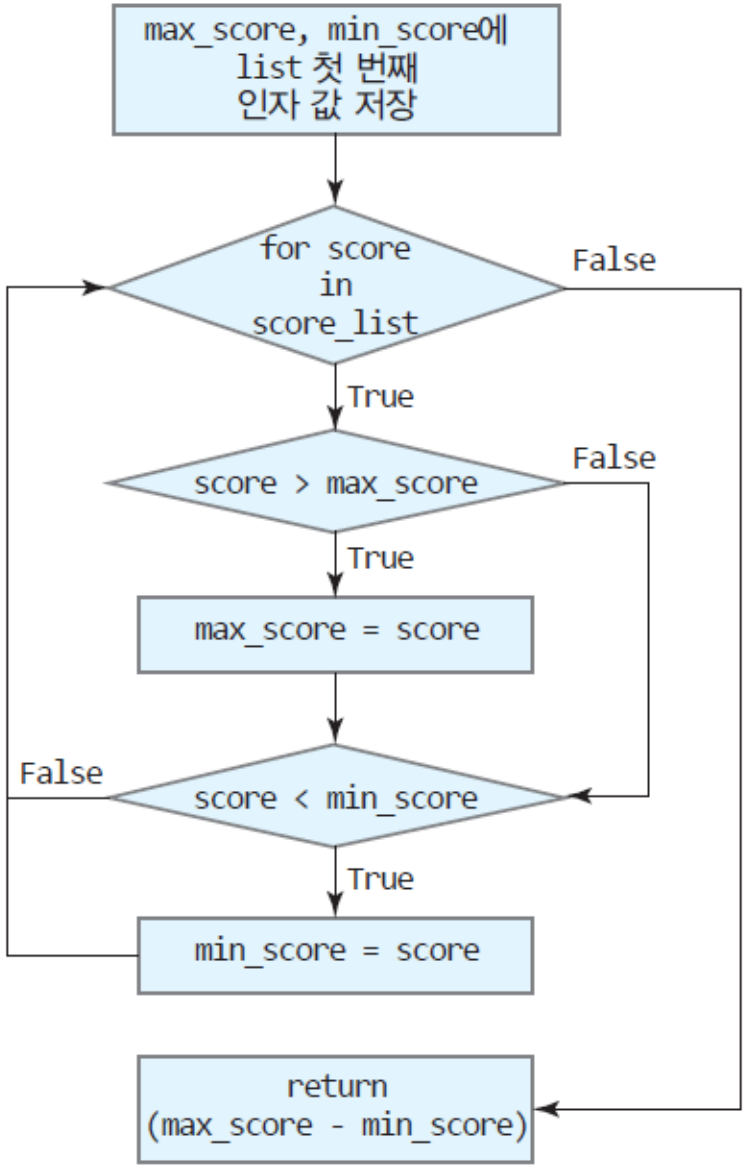
**① list score_list를 입력한다.**

**② 가장 높은 점수와 가장 낮은 점수를 찾아 그 둘의 차를 구하여 return하는 함수를 정의한다. → def gap_score(score_list)**

```
list = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89]
max : 92
min : 45
gap_score : 47
```

# 00. 실습문제


①
score_list 입력
gap_score(score_list)
②
리스트의 최대값 최소값 반환
return max(list)-min(list)


max_score, min_score에
list 첫 번째
인자 값 저장
for score
in
score_list
False
True
score > max_score
False
True
max_score = score
False
score < min_score
True
min_score = score
return
(max_score - min_score)

# 00. 실습문제

```
def max_score(score_list):
    max_num = 0
        for score in score_list:
   if score > max_num:
       max_num = score
   return max_num


def min_score(score_list):
    min_num = score_list[0]
        for score in score_list:
   if score < min_num:
       min_num = score
   return min_num
```

# 00. 실습문제

```
def diff_score(max_value, min_value):
    diff_num = 0
    if max_value > min_value:
        diff_num = max_value-min_value
    else:
        diff_num = min_value-max_value
    return diff_num

math_score = [45, 66, 70, 83, 50, 77, 87, 92, 73, 89]
max_value = max_score(math_value)
min_value = min_score(math_value)
diff_value = diff_score(max_value,min_value)
print(max_value,min_value, diff_value)
```