

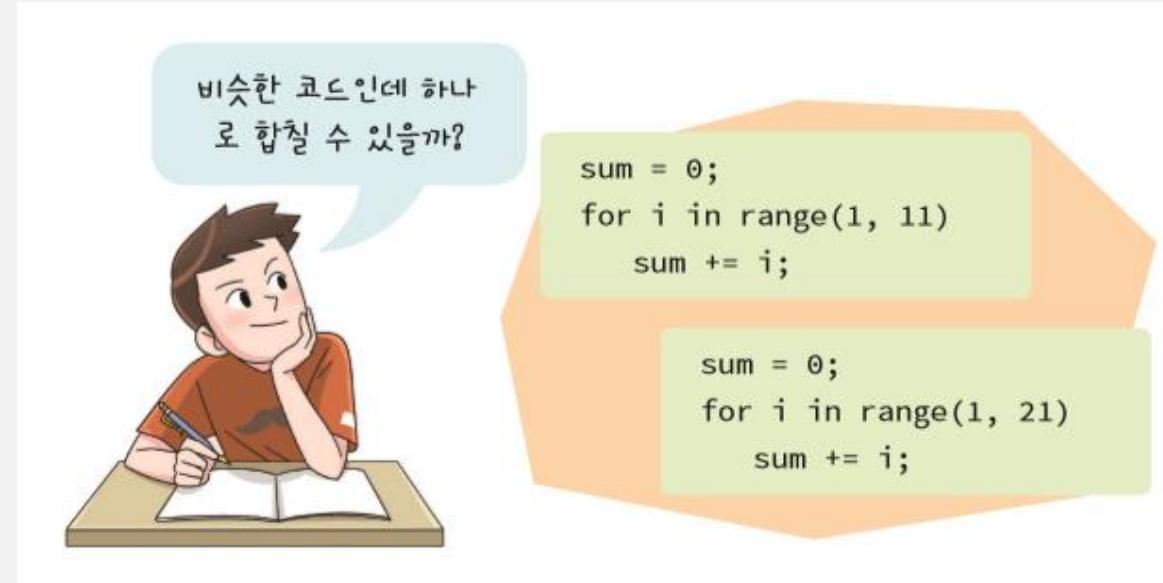
# 함수 (Function)

소프트웨어적 사고

# 수업목표

- Why functions ?
- What is a function ?
- Divide-and-conquer (& combine)
- Calling a function
- Passing arguments to a function
- Returning a value
- Variable scope
- Naming variables
- Recursive Functions
- lambda function

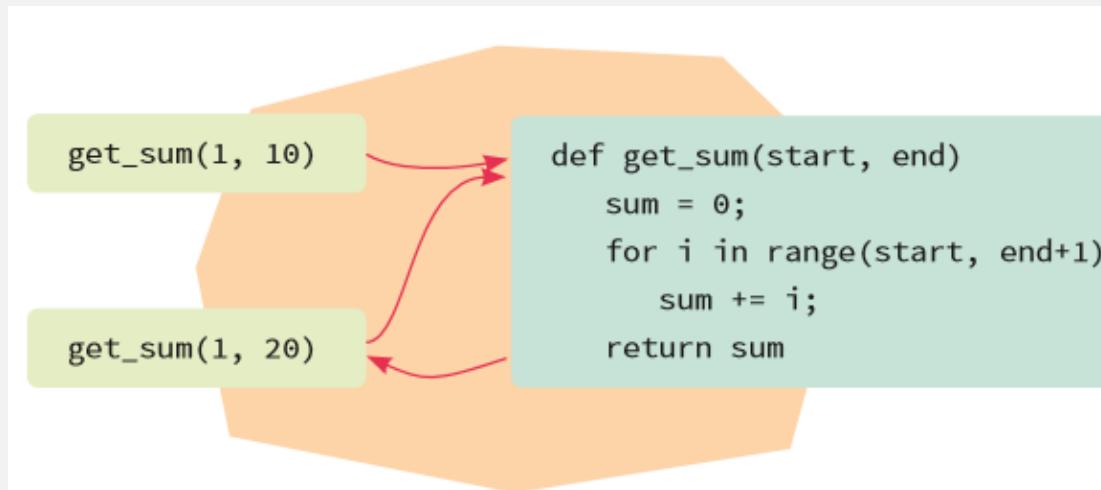
# Why functions ?



A boy sitting at a desk with his hand on his chin, looking thoughtful. A speech bubble above him contains the Korean text: "비슷한 코드인데 하나로 합칠 수 있을까?" (Is it possible to combine similar code into one?).

```
sum = 0;  
for i in range(1, 11)  
    sum += i;
```

```
sum = 0;  
for i in range(1, 21)  
    sum += i;
```



Two green boxes on the left show calls to the function `get_sum(1, 10)` and `get_sum(1, 20)`. A large orange hexagon contains the function definition:

```
def get_sum(start, end)  
    sum = 0;  
    for i in range(start, end+1)  
        sum += i;  
    return sum
```

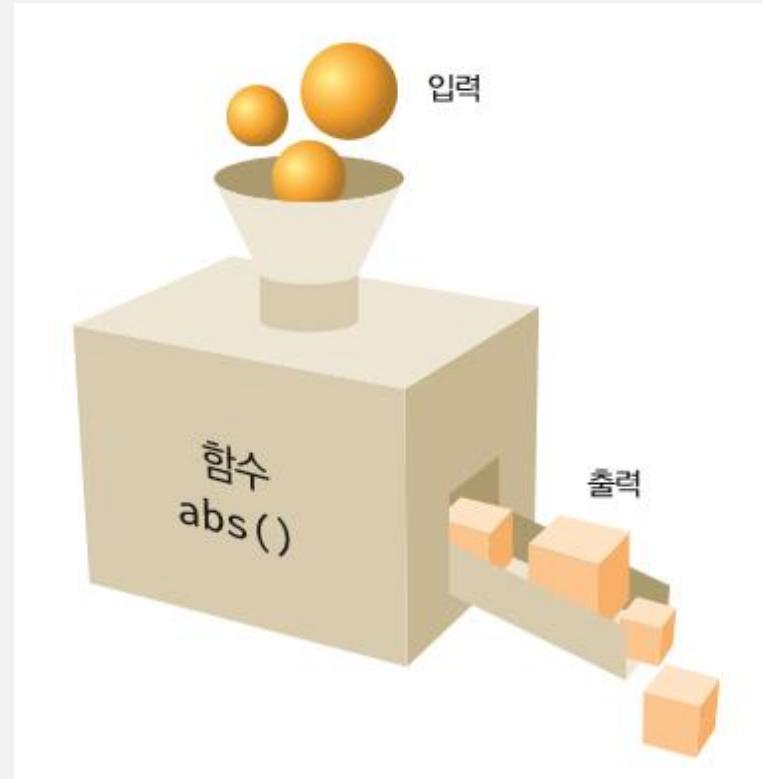
Red arrows point from the two `get_sum` calls to the function definition, indicating that the function can be reused for different ranges.



A woman with glasses and a blue jacket pointing upwards, with a speech bubble containing the Korean text: "함수를 사용하면 됩니다." (It's better to use a function).

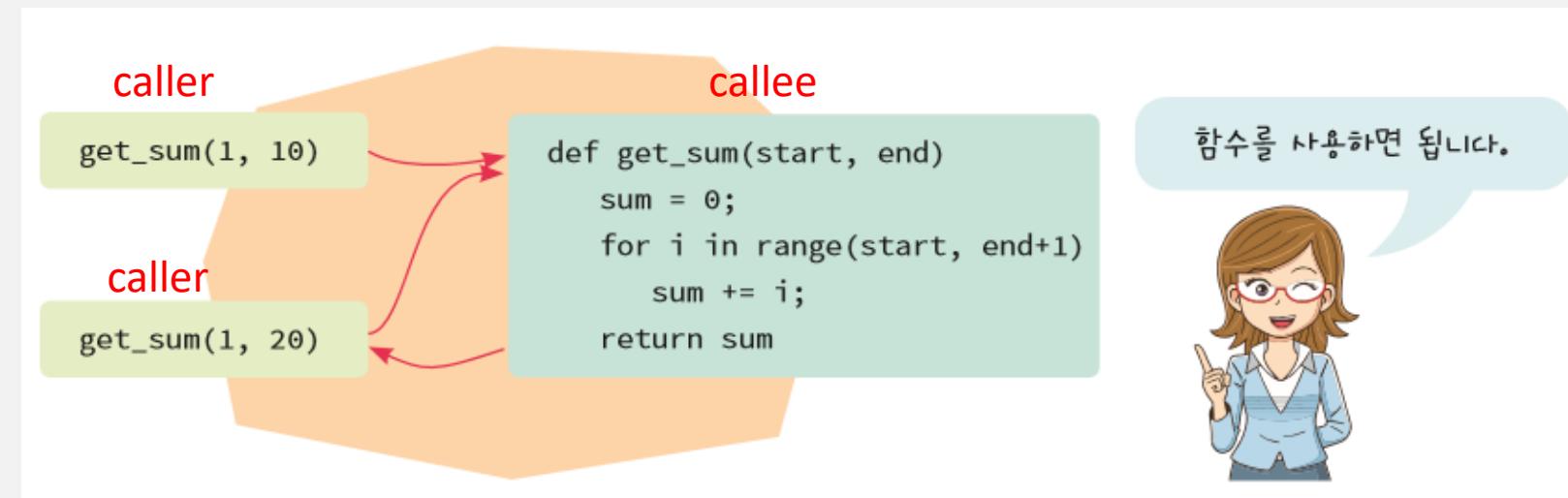
# What is a function ? – Definition (1/2)

- 특정 작업을 수행하는 명령어들의 모음에 이름을 붙인 것
- 함수는 작업에 필요한 데이터(arguments)를 전달받을 수 있으며, 작업이 완료된 후에는 작업의 결과를 호출자(caller)에게 반환할 수 있음



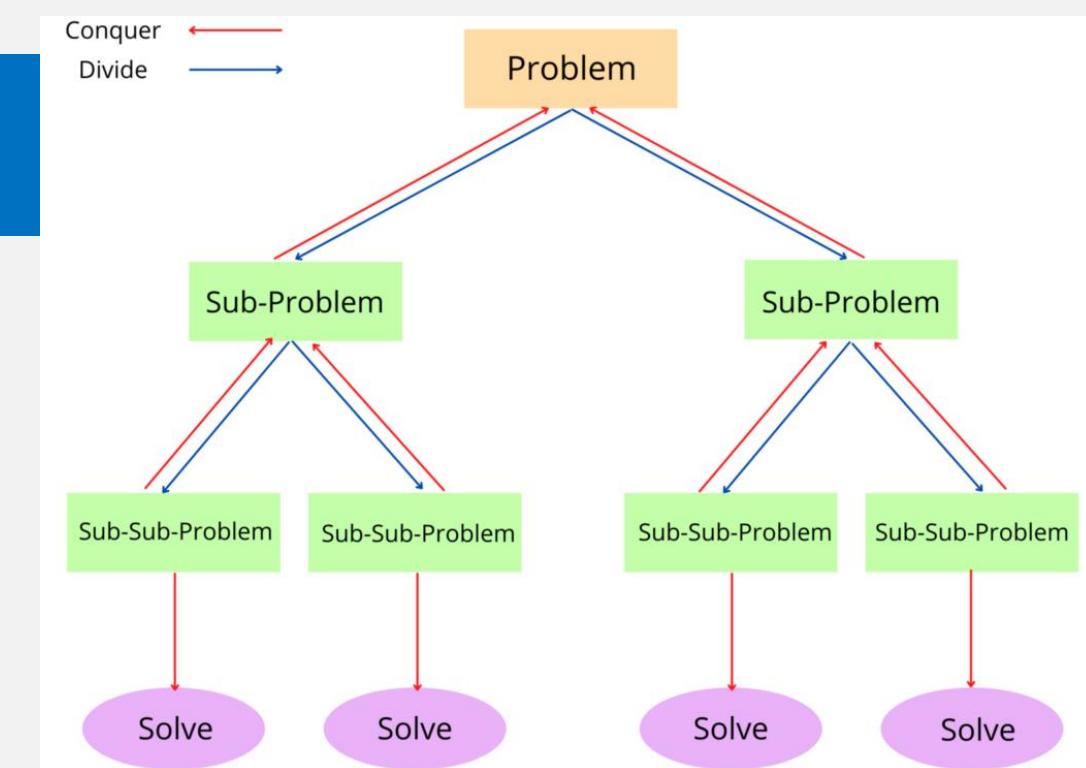
# What is a function ? – Definition & Necessity (2/2)

- 어떤 일을 수행하는 code block
- 더 큰 program을 제작하는데 사용할 수 있는 작은 조각
- 레고 블록을 이용해 무엇인가를 만드는 개념
- def keyword 사용하여 함수를 생성 (define)
- 함수의 이름을 사용해 함수를 호출(call, invocation)하여 사용

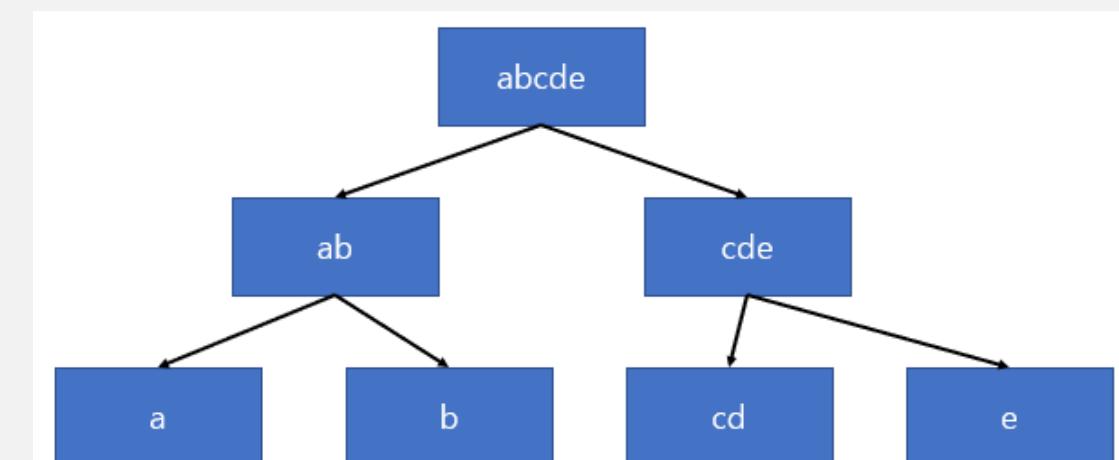


# Divide-and-conquer (& combine) (1/3)

- Divide-and-conquer (& combine)
  - Program을 작성하기 쉽고 관리하기 편리한 작은 조각으로 구성 (fragmentation)
  - Divide-and-conquer (& combine)
  - refinement

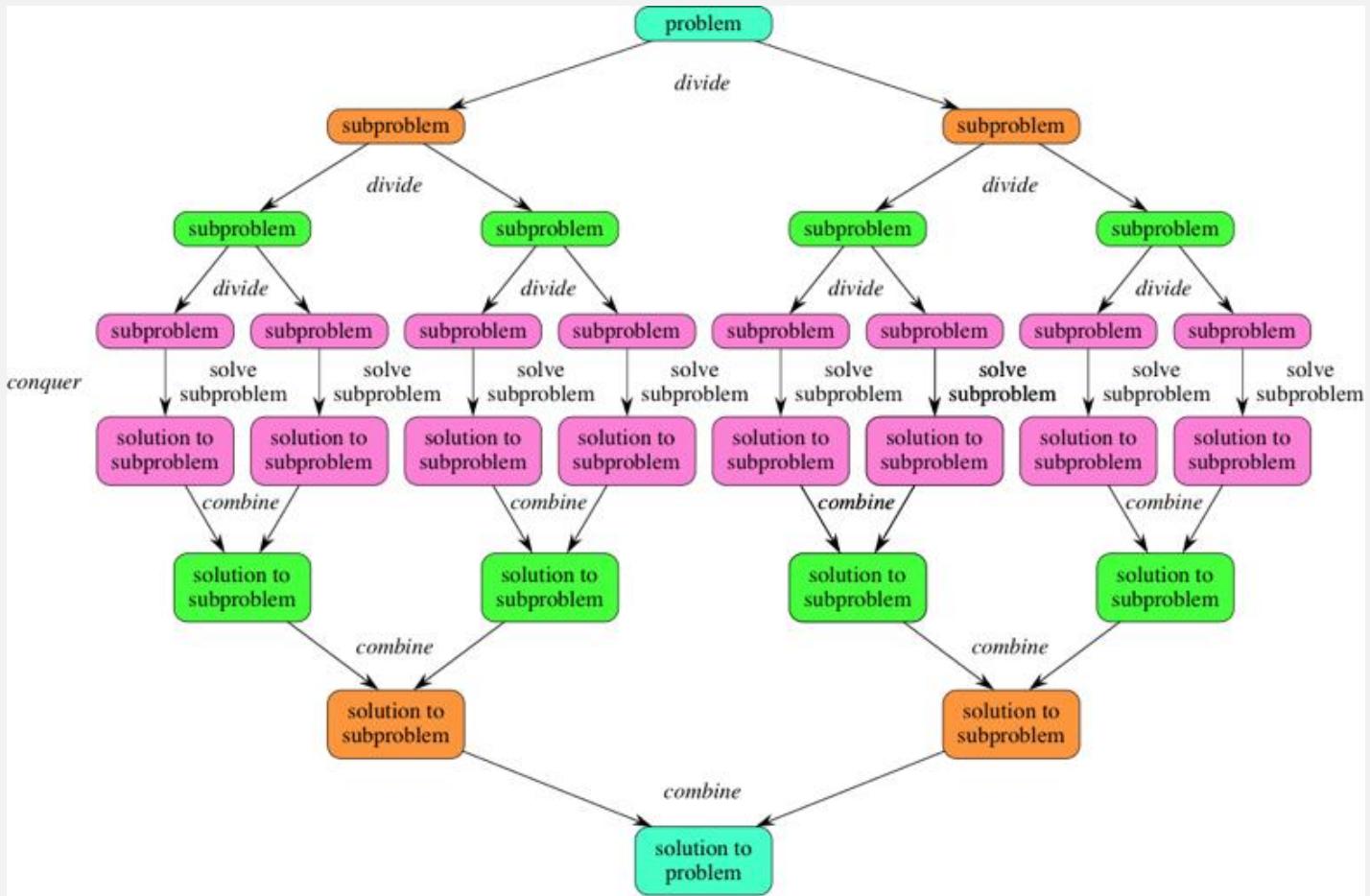


<https://www.baeldung.com/cs/divide-and-conquer-vs-dynamic-programming>



# Divide-and-conquer (& combine) (2/3)

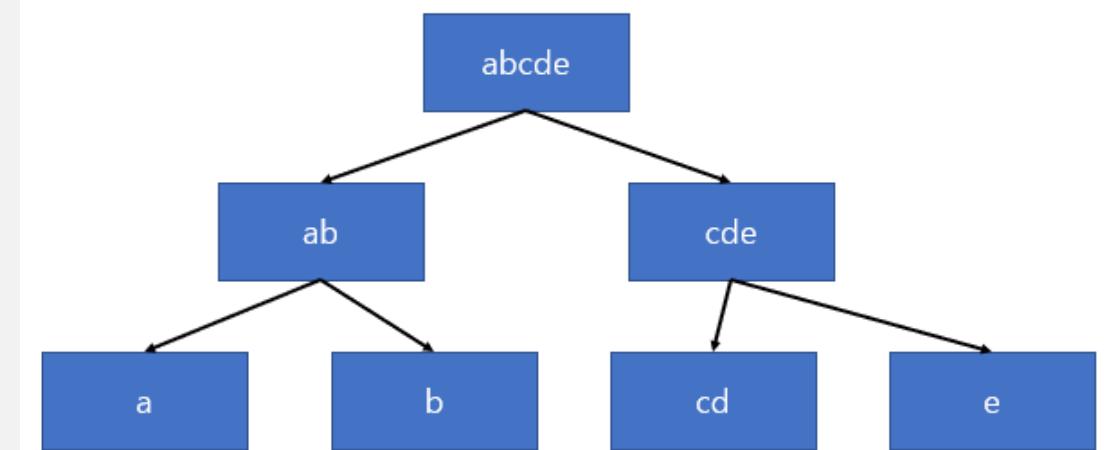
- Divide-and-conquer (& combine)
    - Program을 작성하기 쉽고 관리하기 편리한 작은 조각으로 구성  
(fragmentation)
    - Divide-and-conquer (& combine)
    - refinement



<https://ko.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms>

# Divide-and-conquer (& combine) (3/3)

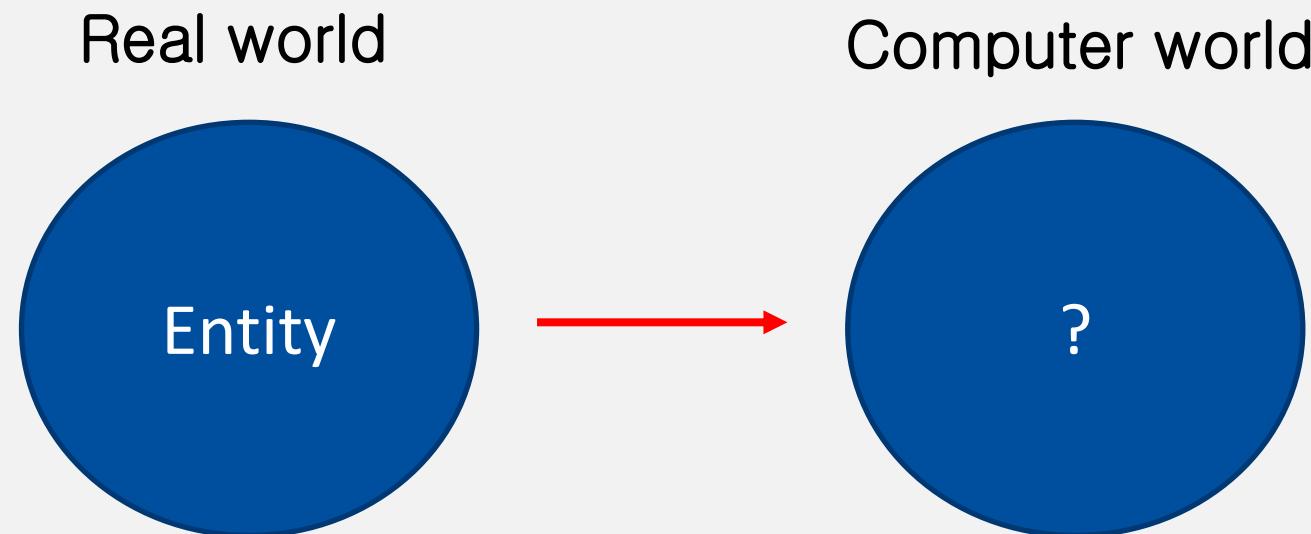
- Program을 divide(refinement)하는 2가지 방안
  - **Module**
    - program의 여러 부분들을 담은 각기 분리된 file
  - **Object**
    - Information hiding / encapsulation
- \* Function
  - 어떤 일을 수행하는 코드 덩어리
- 함수 정의
  - def 함수명(argument\_1, .., argument\_n):



<http://blog.naver.com/PostView.nhn?blogId=qbxlvnf11&logNo=221222565505&parentCategoryNo=&categoryNo=&viewDate=&isShowPopularPosts=false&from=postView>

# Programming Paradigm (1/3)

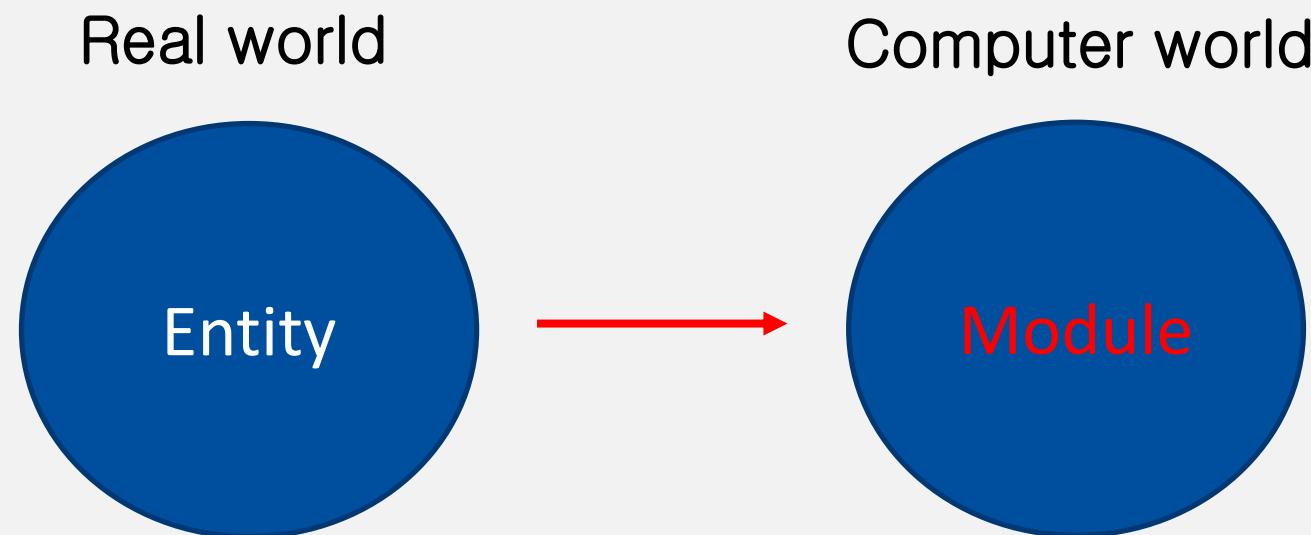
- Mapping / Modeling / 추상화 (抽象化, abstraction)



# Programming Paradigm (2/3)

- **Structure Programming (SP)**

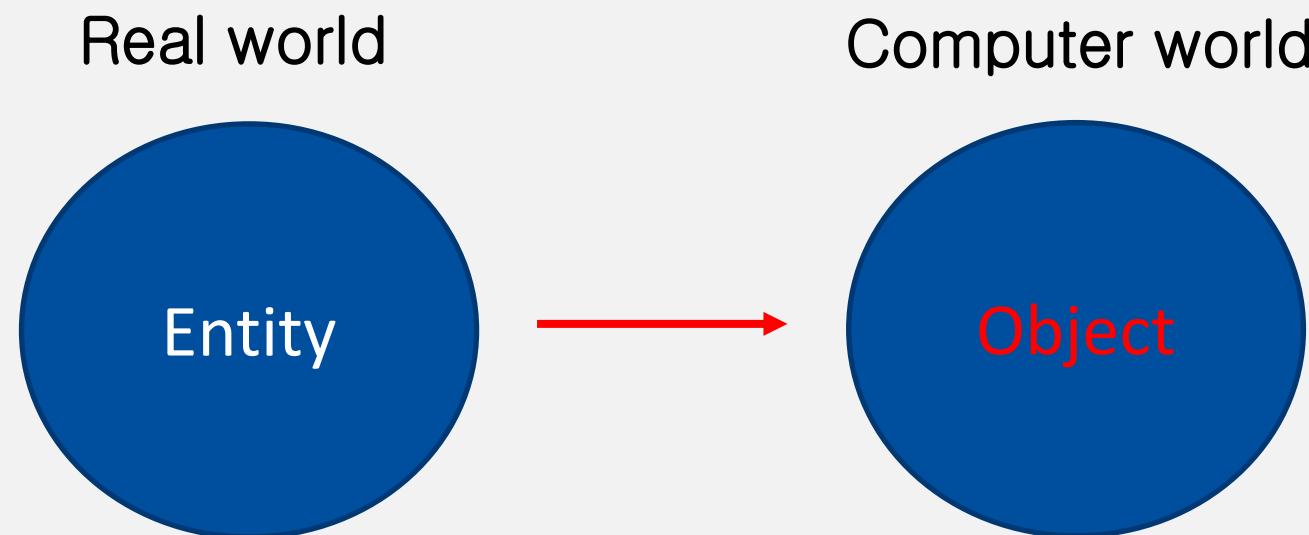
- Module
  - Structure 기반
  - Readability (가독성 : 可讀性)



# Programming Paradigm (3/3)

- **Object-Oriented Programming (OOP)**

- Object
  - Encapsulation (from information hiding)
  - Reusability (재사용성 : 再使用性)



# 함수의 종류

- **System defined function (SDF)**
  - The Python Language provides **pre-defined functions** to make programming easy. These pre-defined functions are known as system defined functions
  - Also called as **library functions, standard functions, pre-defined functions and built-in function**  
ex) `print()`, `input()`, `sqrt()`, `len()`, `sort()`, `insert()`, `append()`, `abs()`, ...
- **User defined function (UDF)**
  - A user-defined function is a function provided by the user of a program  
ex) `get_sum()`

# 함수 사용의 장점

- Program 안에서 중복된 코드 제거
- 복잡한 programming 작업을 더 간단한 작업들로 분해가능 (refinement)
- 함수는 한번 만들어지면 다른 program에서도 재사용(reuse)될 수 있음
- 함수를 사용하면 가독성(可讀性 : readability)이 증대되고
- 유지 관리도 쉬워짐

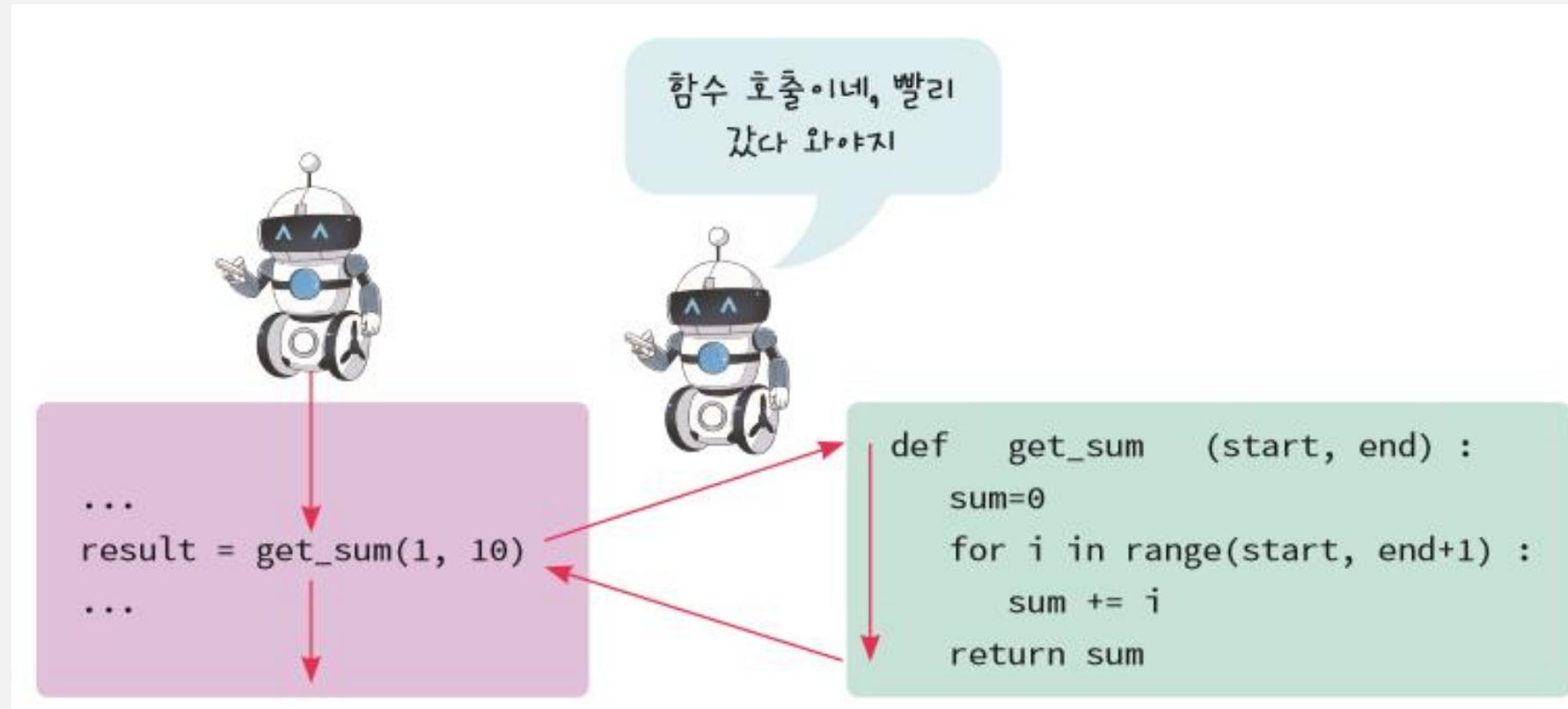
# 함수의 이름 (naming convention)

- 함수의 목적(주요 행위)을 설명하는 **동사** 또는 **동사+명사**를 사용
  - 함수의 이름만 보아도 어떤 역할을 하는지 직관적으로 이해할 수 있게 naming (可讀性 증대)

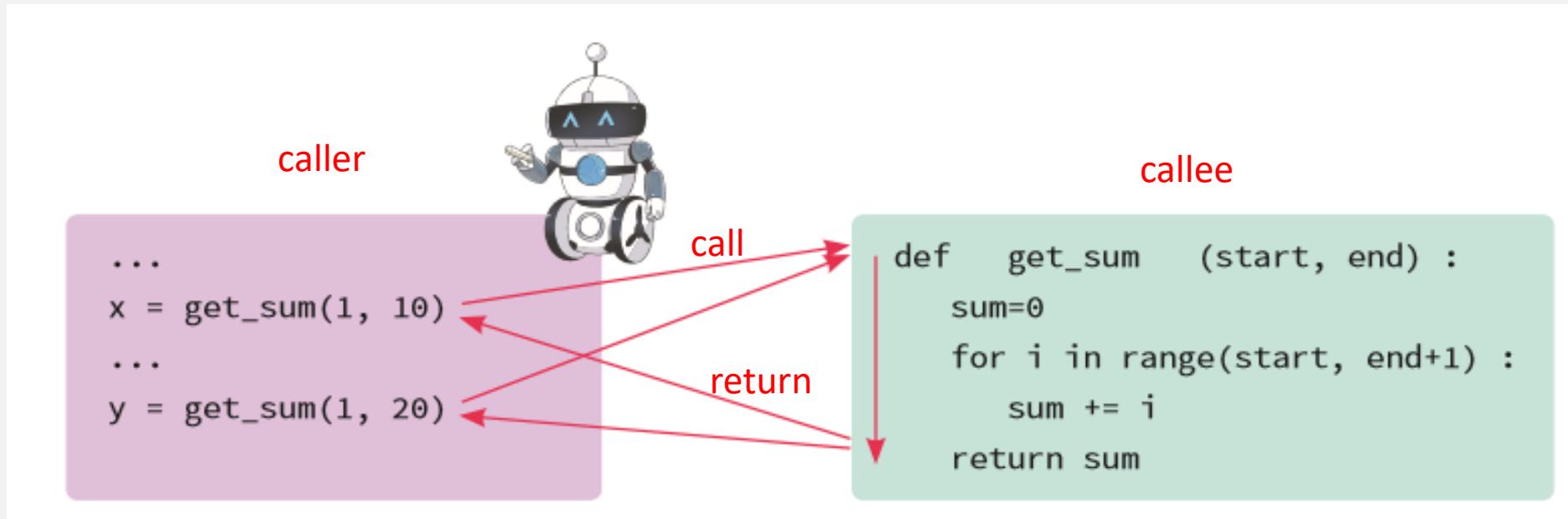
|                              |                                   |
|------------------------------|-----------------------------------|
| <i>square(side)</i>          | // 정수를 제곱하는 함수.<br>// 정사각형 면적 구하기 |
| <i>compute_average(list)</i> | // 평균을 구하는 함수                     |
| <i>set_cursor_type(c)</i>    | // 커서의 타입을 설정하는 함수                |
| <i>paint_the_wall(color)</i> | // 벽에 색칠하기                        |

# 함수 호출

- 함수를 사용하려면 함수를 호출(call, invocation)하여야 함



# 함수는 여러 번 호출할 수 있음 – 재사용(reuse)



# 예제

```
def get_sum(start, end) :  
    sum=0  
    for i in range(start, end+1) :  
        sum += i  
    return sum  
  
# main  
print( get_sum(1, 10))  
print( get_sum(1, 20))
```

55  
210

```
1 # 함수선언 #함수를 선언하는 부분
2 def get_sum(start, end) :
3     print("g-1) In get_sum({}, {})".format(start, end))
4
5     sum = 0
6     for num in range(start, end + 1) :
7         sum = sum + num
8
9     print("g-2) sum = {}".format(sum))
10    return sum
11
12 #main #함수가 끝나는 부분을 Main이라고 일반적으로 한다.
13 print("n1) main starts")
14
15 begin_val = int(input("n> 시작값을 입력하시오 : "))
16 end_val = int(input("n> 종료값을 입력하시오 : "))
17
18 print("n2) 시작값 = ", begin_val)
19 print("    종료값 = ", end_val)
20
21 print("n3) {} 부터 {} 까지 합은 {} 입니다."
22     .format(begin_val, end_val, get_sum(begin_val, end_val)))
23
```

1) main starts  
=> 시작값을 입력하시오 : 1  
=> 종료값을 입력하시오 : 100  
  
2) 시작값 = 1  
종료값 = 100  
  
g-1) In get\_sum(1, 100)  
g-2) sum = 5050 Function 먼저 실행  
  
3) 1 부터 100 까지 합은 5050 입니다.

```

1 D = True
2 #D = False
3
4 # 함수선언
5 def in_values():
6     if D:
7         print("Wn i-1) In_value()")
8
9     begin_val = int(input("Wn >> 시작값을 입력하시오 : "))
10    end_val = int(input(" >> 종료값을 입력하시오 : "))
11
12    if D:
13        print("Wn i-2) begin_val : {}, end_val : {}"
14            .format(begin_val, end_val))
15
16    return begin_val, end_val
17
18 def get_sum(start, end):
19    if D:
20        print("Wn g-1) In get_sum({}, {})".format(start, end))
21
22    sum = 0
23    for num in range(start, end + 1):
24        sum = sum + num
25
26    if D:
27        print("g-2) sum = {}".format(sum))
28
29    return sum
30
31 #main
32 if D:
33     print("Wn 1) main starts")
34
35 in_v, out_v = in_values()
36
37 print("Wn >> 시작값 = {}, 종료값 : {}".format(in_v, out_v))
38
39 ret_sum = get_sum(in_v, out_v)
40 print("Wn >> 합 : ", ret_sum)
41

```

```

>> 시작값을 입력하시오 : 1
>> 종료값을 입력하시오 : 100
>> 시작값 = 1, 종료값 : 100
>> 합 : 5050
```

```

1) main starts

i-1) In\_value()

```

>> 시작값을 입력하시오 : 1
>> 종료값을 입력하시오 : 100

```

i-2) begin\_val : 1, end\_val : 100

>> 시작값 = 1, 종료값 : 100

```

g-1) In get_sum(1, 100)
g-2) sum = 5050

```

>> 합 : 5050

```

1 # Define functions
2 # 학기초 학생들이 해야할 일들
3
4 # 수강신청
5 Capacity = 200 #기술사 정원 상수는 대문자 변수
6
7 def regist_class(subject1, subject2):
8     print("Wn-1) {}, {} 수강신청".format(subject1, subject2))
9     print("r-2) 실제 이 부분에서는 수강신청업무를 진행합니다.")
10    return("OK")
11
12 #기술사 신청
13 def dormitory_apply(s_id, s_name):
14     print("d-1) {}, {} 기숙사 신청".format(s_id, s_name))
15     print("d-2) 실제 이 부분에서는 기숙사신청 절차를 실행합니다. ")
16     current_number_of_applicants = 199
17
18     if Capacity > current_number_of_applicants :
19         msg = "성공적으로 기숙사 신청을 완료했습니다."
20     else:
21         msg = "기술사 정원을 넘어섰습니다."
22
23     return(msg)
24
25 # 신청내역 확인
26 def check_up(subject1, subject2, student_id, student_name):
27     print("c-1) 학생의 수강신청 내역, 기숙사 신청정보 출력")
28     print("c-2) 과목 1 :", subject1)
29     print("    과목 2 :", subject2)
30     print("c-3) 기숙사 신청학생 학번 :", student_id)
31     print("    기숙사 신청학생 이름 :", student_name)
32
33 # 기숙사비 정산
34 def how_much(num_of_month):
35     print ("h-1) 20xx.3 ~ 6 기숙사비 정산")
36     room = 300000 * num_of_month # room charge
37     meal = 5000 * 2 * 30 * num_of_month
38     sum = room + meal
39
40     print("h-2) room charge =", room)
41     print("    식대 =", meal)
42     print("    총 기숙사비 =", sum)
43     return sum
44
45     print("h-3) end function") # 이 문장은 실행되지 않습니다.
46

```

```

47 # main
48 print("1) main")
49 print("2) 수강신청")
50
51 sub1 = "Python"
52 sub2 = "Java"
53 result_of_regist = regist_class(sub1, sub2)
54
55 if result_of_regist == "OK" :
56     print("3) 성공적으로 수강신청을 완료했습니다.")
57 else:
58     print("3) 학부사무실로 연락하기 바랍니다.")
59
60 print("Wn4) 기숙사 신청")
61 stu_id = "20501234"
62 name = "김국민"
63 res_of_dorm_apply = dormitory_apply(stu_id, name)
64 print("5)", res_of_dorm_apply)
65
66 print("Wn6) 전체 신청상황을 출력합니다.")
67 check_up(sub1, sub2, stu_id, name)
68
69 print("Wn7) 기숙사비 정산")
70 num_of_month = 4
71 total = how_much(num_of_month)
72
73 print("Wn8) 기숙사비 :", total)

```

1) main  
2) 수강신청

r-1) Python, Java 수강신청  
r-2) 실제 이 부분에서는 수강신청업무를 진행합니다.  
3) 성공적으로 수강신청을 완료했습니다.

4) 기숙사 신청  
d-1) 20501234, 김국민 기숙사 신청  
d-2) 실제 이 부분에서는 기숙사신청 절차를 실행합니다.  
5) 성공적으로 기숙사 신청을 완료했습니다.

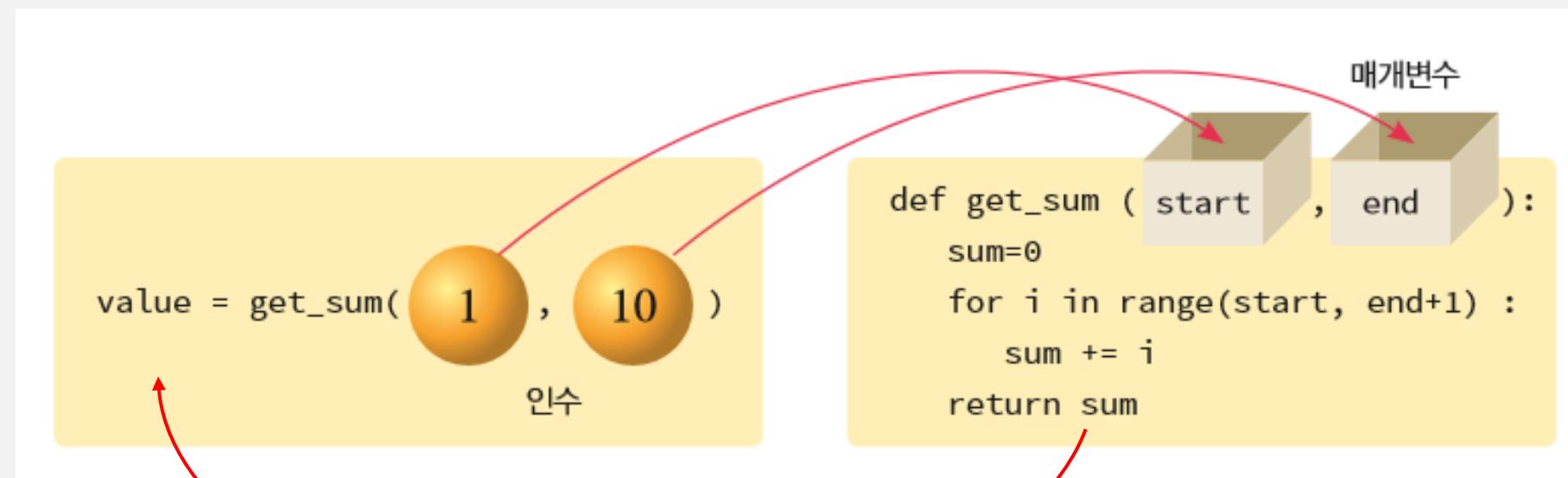
6) 전체 신청상황을 출력합니다.  
c-1) 학생의 수강신청 내역, 기숙사 신청정보 출력  
c-2) 과목 1 : Python  
 과목 2 : Java  
c-3) 기숙사 신청학생 학번 : 20501234  
 기숙사 신청학생 이름 : 김국민

7) 기숙사비 정산  
h-1) 20xx.3 ~ 6 기숙사비 정산  
h-2) room charge = 1200000  
 식대 = 1200000  
 총 기숙사비 = 2400000

8) 기숙사비 : 2400000

# Argument passing (1/6)

- 함수사용 할 때
  - 인자(argument) 를 함수에 전달 가능
    - 매개변수(parameter)
  - 함수실행 결과값 리턴 가능



# Argument passing (1/6)

- Caller
  - Actual arguments (parameters)
- Callee
  - Formal parameters

## Function definition

```
def fun_name(arg1,arg2) → Formal parameters  
    statement(s)
```

## Function call

```
fun_name(a,b) → Actual parameters/arguments
```

[https://www.google.com/imgres?imgurl=https%3A%2F%2Fmiro.medium.com%2Fv2%2Fresize%3Afit%3A1392%2F1\\*DhAUznkbAffBUvMqF8rFuw.png&tbnid=GmxRWMcGIIEZsM&vet=12ahUKEwj2j5nevPeBAXVza\\_UHHe3vB24QMygCegQIARBQ..&imgrrefurl=https%3A%2F%2Flevelup.gitconnected.com%2F5-types-of-arguments-in-python-function-definition-e0e2a2cafd29&docid=xP9TptWdaU8qiM&w=696&h=170&q=formal%20parameter%20and%20actual%20parameter%20python&ved=2ahUKEwj2j5nevPeBAXVza\\_UHHe3vB24QMygCegQIARBQ](https://www.google.com/imgres?imgurl=https%3A%2F%2Fmiro.medium.com%2Fv2%2Fresize%3Afit%3A1392%2F1*DhAUznkbAffBUvMqF8rFuw.png&tbnid=GmxRWMcGIIEZsM&vet=12ahUKEwj2j5nevPeBAXVza_UHHe3vB24QMygCegQIARBQ..&imgrrefurl=https%3A%2F%2Flevelup.gitconnected.com%2F5-types-of-arguments-in-python-function-definition-e0e2a2cafd29&docid=xP9TptWdaU8qiM&w=696&h=170&q=formal%20parameter%20and%20actual%20parameter%20python&ved=2ahUKEwj2j5nevPeBAXVza_UHHe3vB24QMygCegQIARBQ)

# Argument passing - Variable length arguments (2/6)

- 가변 매개변수 함수 (1/3)
  - Python에서는 매개변수를 원하는 만큼 받을 수 있는 함수를 만들 수 있다.
  - **가변 매개변수는 하나만 사용 할 수 있으며, 가변 매개변수 뒤에는 일반 매개변수가 올 수 없다.**

```
def <함수 이름>(<매개 변수>, <매개 변수>, ... , *<가변 매개변수>):  
    <코드>
```

```
variable_parameter.py
File Edit Format Run Options Window Help
1 def get_sum(*args) :
2     print("g-1) type of args :", type(args))
3     print("g-2) args =", args)
4
5     ret = 0
6     for each in args :
7         ret += each
8
9     print("g-3) sum =", ret)
10    return ret
11
12
13 #main
14 print("1) main")
15 print("Wn2) 1 ~ 10 까지의 합:", get_sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
16 print("Wn3) 10, 20.5, 30.1 합:", get_sum(10, 20.5, 30.1))
17 print("Wn4) 10.1, 20.4, 30.6 합 :", get_sum(10.1, 20.4, 30.6))
18 print("5) End..")
19
```

```
1) main
g-1) type of args : <class 'tuple'>
g-2) args = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
g-3) sum = 55

2) 1 ~ 10 까지의 합: 55
g-1) type of args : <class 'tuple'>
g-2) args = (10, 20.5, 30.1)
g-3) sum = 60.6

3) 10, 20.5, 30.1 합: 60.6
g-1) type of args : <class 'tuple'>
g-2) args = (10.1, 20.4, 30.6)
g-3) sum = 61.1

4) 10.1, 20.4, 30.6 합 : 61.1
5) End..
```

# Argument passing - Variable length arguments (3/6)

- 가변 매개변수 함수 (2/3)

```
def <함수 이름>(<매개 변수>, <매개 변수>, ... , *<가변 매개변수>):  
    <코드>
```

The screenshot shows a Python code editor with a file named `variable_parameter_2.py`. The code defines a function `get_sum` that takes two required parameters `p1` and `p2`, followed by a variable number of arguments via the `*args` parameter. It prints each argument and calculates their sum. The `main` function demonstrates this with three calls: one with integers 1-5, one with floats 10-20.5, and one with floating-point numbers 1.1-4.1.

```
variable_parameter_2.py
File Edit Format Run Options Window Help
1 def get_sum(p1, p2, *args) :
2
3     print("g-1)", p1, p2, args)
4     ret = p1 + p2
5
6     for each in args :
7         ret += each
8
9     print("g-2) sum =", ret)
10    return ret
11
12 def main() :
13     print("m-1) 1 ~ 5 까지의 합 : ", get_sum(1, 2, 3, 4, 5))
14     print("m-2) 10, 20.5 합 : ", get_sum(10, 20.5))
15     print("m-3) 1.1, 2.1, 3.1, 4.1 합 : ", get_sum(1.1, 2.1, 3.1, 4.1))
16
17 # main
18 print("n1) main")
19 main()
20
21 print("n2) End..")
```

Output window:

```
1) main
g-1) 1 2 (3, 4, 5)
g-2) sum = 15
m-1) 1 ~ 5 까지의 합 : 15
g-1) 10 20.5 ()
g-2) sum = 30.5
m-2) 10, 20.5 합 : 30.5
g-1) 1.1 2.1 (3.1, 4.1)
g-2) sum = 10.4
m-3) 1.1, 2.1, 3.1, 4.1 합 : 10.4
2) End..
```

Page Number: 24

# Argument passing - Variable length arguments (3/6)

- 가변 매개변수 함수 (3/3)

```
def <함수 이름>(<매개 변수>, <매개 변수>, ... , *<가변 매개변수>):  
    <코드>
```

– 가변 매개변수는 하나만 사용 할 수 있으며, 가변 매개변수 뒤에는 일반 매개변수가 올 수 없음

variable\_parameter\_3.py

```
File Edit Format Run Options Window Help  
1 def get_sum(p1, *args, p2) : # error.  
2     # 가변 매개변수 뒤에는 일반 매개변수가 올 수 없음  
3  
4     print("Wng-1)", p1, p2, args)  
5     ret = p1 + p2  
6  
7     for each in args :  
8         ret += each  
9  
10    print("g-2) sum =", ret)  
11    return ret  
12  
13 def main() :  
14     print("m-1) 1 ~ 5 까지의 합 : ", get_sum(1, 2, 3, 4, 5))  
15     print("m-2) 10, 20.5 합 :", get_sum(10, 20.5))  
16     print("m-3) 1.1, 2.1, 3.1, 4.1 합 :", get_sum(1.1, 2.1, 3.1, 4.1))  
17  
18 # main  
19 print("Wn1) main")  
20 main()  
21  
22 print("Wn2) End..")
```

```
1) main  
Traceback (most recent call last):  
  File "C:/소사/강의예제/variable_parameter_3.py", line 20, in <module>  
    main()  
  File "C:/소사/강의예제/variable_parameter_3.py", line 14, in main  
    print("m-1) 1 ~ 5 까지의 합 : ", get_sum(1, 2, 3, 4, 5))  
TypeError: get_sum() missing 1 required keyword-only argument: 'p2'
```

# Argument passing - Default argument (4/6)

- 기본 매개변수 (1/3)
  - Python에서는 함수의 매개변수가 기본값을 가질 수 있음
  - 인자가 전달되지 않을 경우 default 값을 정의하는 매개변수
  - 이것을 **기본 매개변수 (디폴트 인수(default argument))**라고 함

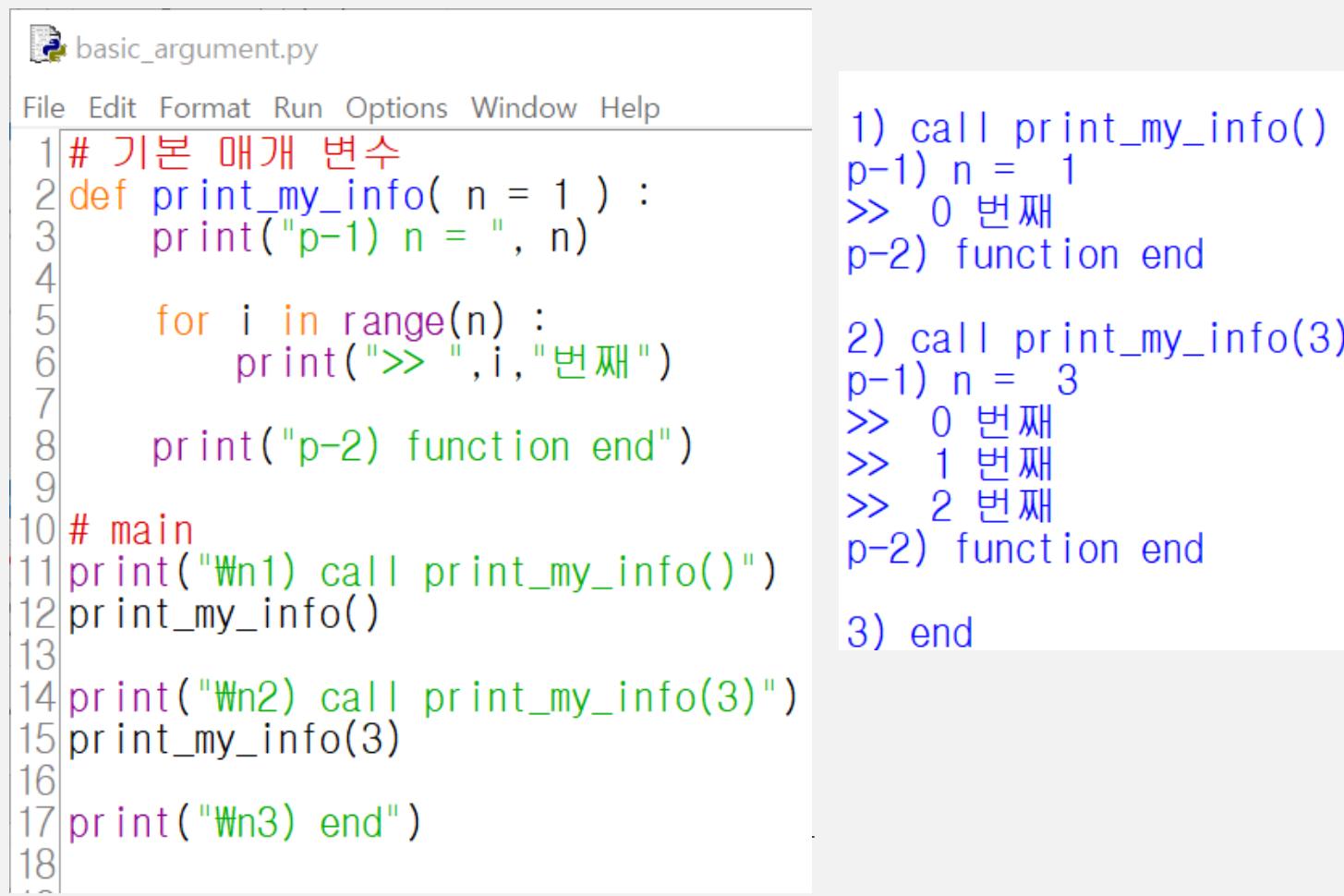
```
def greet(name, msg="별일 없죠?"):  
    print("안녕 ", name + ', ' + msg)
```

```
greet("영희")
```

안녕 영희, 별일 없죠?

# Argument passing - Default argument (5/6)

- 기본 매개변수 (2/3)
  - 인자가 전달되지 않을 경우 default 값을 정의하는 매개변수



```
basic_argument.py
File Edit Format Run Options Window Help
1 # 기본 매개 변수
2 def print_my_info( n = 1 ) :
3     print("p-1) n = ", n)
4
5     for i in range(n) :
6         print("=> ",i,"번째")
7
8     print("p-2) function end")
9
10 # main
11 print("\n1) call print_my_info()")
12 print_my_info()
13
14 print("\n2) call print_my_info(3)")
15 print_my_info(3)
16
17 print("\n3) end")
```

1) call print\_my\_info()  
p-1) n = 1  
>> 0 번째  
p-2) function end

2) call print\_my\_info(3)  
p-1) n = 3  
>> 0 번째  
>> 1 번째  
>> 2 번째  
p-2) function end

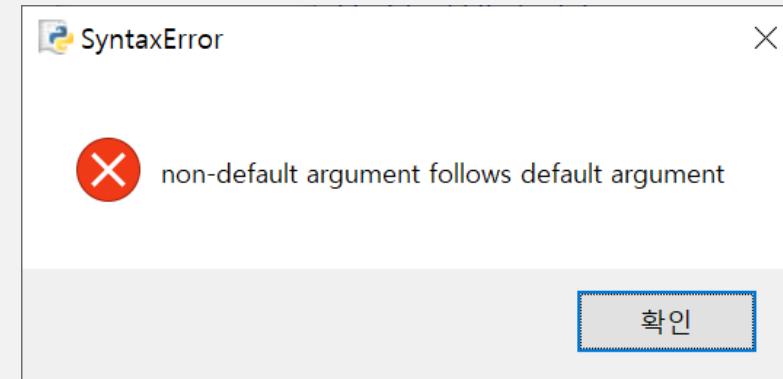
3) end

# Argument passing - Default argument (6/6)

- 기본 매개변수 (3/3)
  - 인자가 전달되지 않을 경우 default 값을 정의하는 매개변수
  - 기본 매개변수 뒤에는 일반 매개변수가 올 수 없다.

basic\_argument\_1.py

```
File Edit Format Run Options Window Help
1 # 기본 매개 변수
2 def print_my_info(n = 1, other_arg) :
3     print("p-1) n = ", n)
4
5     for i in range(n) :
6         print(">> ", i, "번째")
7
8     print("p-2) function end")
9
10 # main
11 print("n1) call print_my_info()")
12 print_my_info()
13
14 print("n2) call print_my_info(3)")
15 print_my_info(3)
16
17 print("n3) call print_my_info(3, park")
18 print_my_info(3, "park")
19
20 print("n4) end")
21
```



# Calling a function

- 함수 호출(call)
  - 호출자(caller)
  - 피호출자(callee)

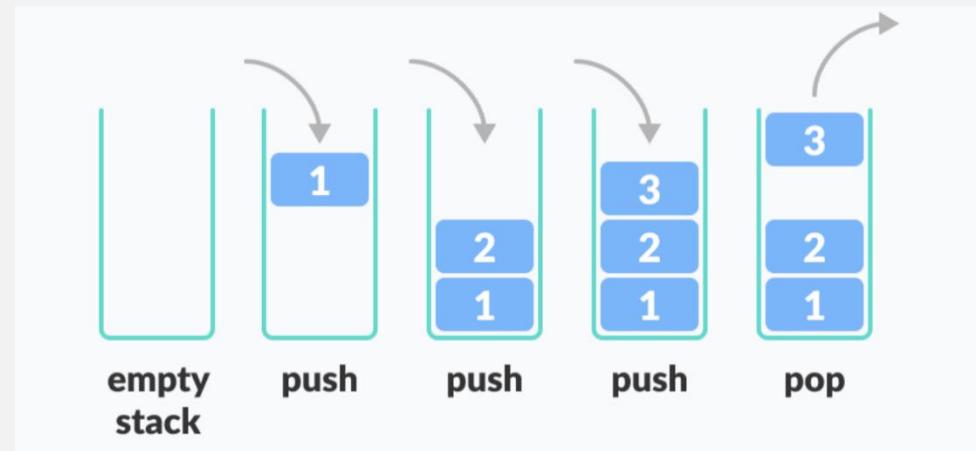
5.1example2.py

```
File Edit Format Run Options Window Help
1 # Define a function
2 def f1():
3     print ("f1 starts.")
4     f2()
5     print ("f1 ends.")
6
7 def f2():
8     print ("f2 starts.")
9     f3()
10    print ("f2 ends.")
11
12 def f3():
13     print ("f3 starts.\n")
14     print ("f3 ends.\n")
15
16 def main():
17     print("main starts.")
18     f1()
19     print("main ends.\n")
20
21 # main program
22 print("1) Program start.\n")
23 main()
24 print("2) Program ends.\n")
25
```

1) Program start.  
main starts.  
f1 starts.  
f2 starts.  
f3 starts.  
  
f3 ends.  
f2 ends.  
f1 ends.  
main ends.  
  
2) Program ends.  
````

# Calling a function

- 함수는 호출될 때, 독자적 memory 공간이 할당됨
  - Stack으로 할당
  - 함수내부에서 정의된 변수(local variable)는 이 memory 공간을 사용함
    - 인자(매개변수 : parameter) 포함
- 함수가 종료될 때, 할당되었던 memory 공간은 소멸됨
  - 따라서 이 memory 공간을 사용하고 있던 모든 변수 및 인자도 동시에 소멸됨



```

1 def FtoC(input_fahrenheit):
2
3     print("Wnf-1) 입력된 화씨온도 =", input_fahrenheit)
4
5     temp_c = (5.0 * (input_fahrenheit - 32.0)) / 9.0;
6     print("f-2) 섭씨온도 =", temp_c)
7
8
9     print("f-3) id(input_fahrenheit) =", id(input_fahrenheit))
10    print("f-4) id(fahrenheit) =", id(fahrenheit))
11    print("f-5) id(temp_c) =", id(temp_c))
12
13    return temp_c;
14
15 #main
16 print("1) main")
17 fahrenheit = float(input("Wn> 화씨온도를 입력하시오: "))
18
19 print("2) fahrenheit =", fahrenheit)
20 print("3) id(fahrenheit) =", id(fahrenheit))
21
22 # FtoC() 함수를 호출한다.
23 print("Wn4) 변경된 화씨온도 :", FtoC(fahrenheit))
24
25 #error
26 print("5) id(temp_c) =", id(temp_c))
27

```

1) main

>> 화씨온도를 입력하시오: 32  
2) fahrenheit = 32.0  
3) id(fahrenheit) = 2468768190704

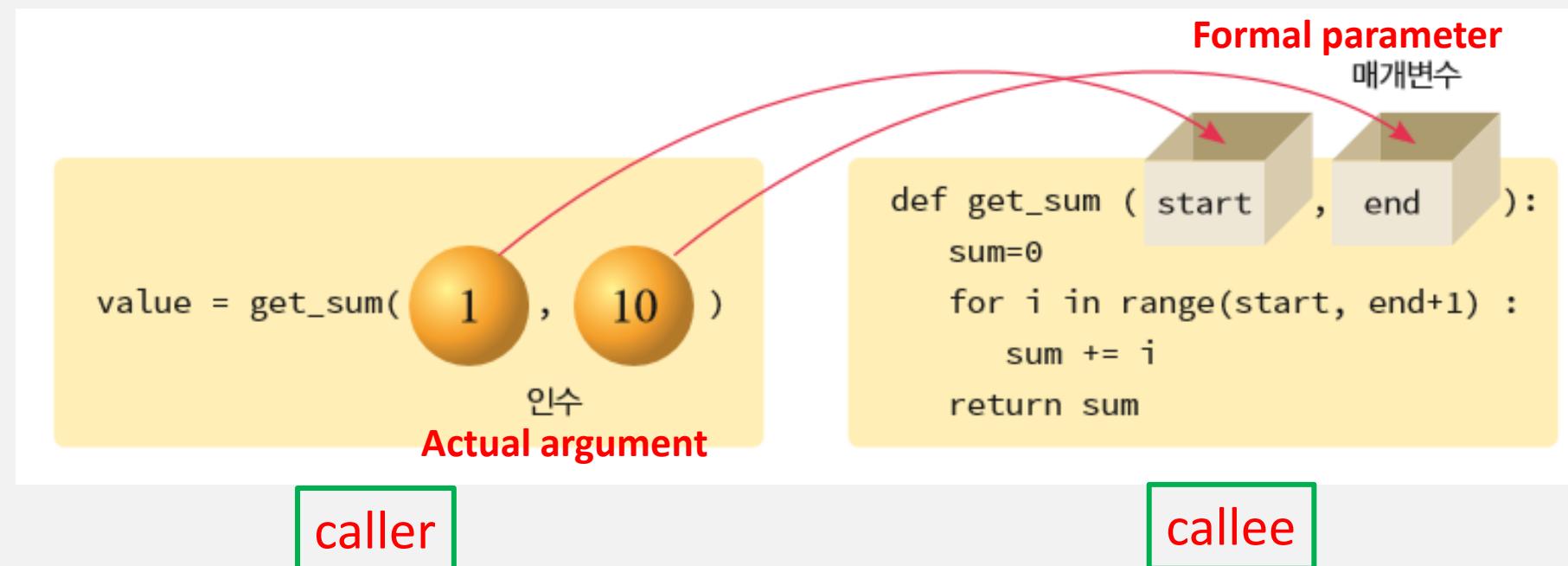
f-1) 입력된 화씨온도 = 32.0  
f-2) 섭씨온도 = 0.0  
f-3) id(input\_fahrenheit) = 2468768190704  
f-4) id(fahrenheit) = 2468768190704  
f-5) id(temp\_c) = 2468732817872

4) 변경된 화씨온도 : 0.0

Traceback (most recent call last):  
File "C:\W소사\강의예제\ftoc.py", line 26, in <module>  
print("5) id(temp\_c) =", id(temp\_c))  
NameError: name 'temp\_c' is not defined

# Passing arguments to a function

- **Actual argument (parameter)**
  - 인수(argument)는 호출 program(caller)에 의하여 함수에 실제로 전달되는 값들
- **Formal parameter**
  - 매개 변수(parameter)는 callee에서 이 값을 전달받는 변수



# Passing arguments to a function

- 인자(argument) 넘기기
  - Actual argument : arguments in caller, ex) pName, pNum
  - Formal parameter : arguments in callee, ex) name, num

5.1example5.py

```
File Edit Format Run Options Window Help
1 # Define a function
2 def printPerson(name, num):
3
4     print("Wnp-1) formal parameters : {}, {}".format(pName, pNum))
5     name = "Soo"
6     num = 1774
7
8     print("p-2) In the func.: {}, {}".format(name, num))
9
10
11 # main program
12 print("1) main")
13
14 pName = "Park"
15 pNum = 4559
16
17 print("2) Before call, actual arguments : {}, {}".format(pName, pNum))
18 printPerson(pName, pNum)
19
20 print("Wn3) After call, actual arguments : ", pName, pNum)
21 #print("3) After call, name, num :", name, num)
22
```

- 1) main
- 2) Before call, actual arguments : Park. 4559  
p-1) formal parameters : Park, 4559  
p-2) In the func.: Soo. 1774
- 3) After call, actual arguments : Park 4559

# Passing arguments to a function

- 함수를 호출할 때, 변수를 전달하는 2가지 방법
  - 값에 의한 호출(**call-by-value**)
    - Caller의 actual arguments의 값을 callee의 formal parameter로 copy하여 전달
    - Callee의 formal parameter는 local 변수
    - Callee의 formal parameter 값의 변경이 caller의 actual argument 값에 반영되지 않음
    - Primitive data type
  - 참조에 의한 호출(**call-by-reference**)
    - Caller의 actual arguments의 address를 callee의 formal parameter로 전달
    - Callee의 formal parameter 값의 변경이 caller의 actual argument 값에 반영됨
    - mutable data : list, dictionary, set 등  
user defined class

```
1 def modify1(m, nai):
2     print("r-1) m =", m, ", id(m) =", id(m))
3     print("r-2) nai =", nai, ", id(nai) =", id(nai))
4     m += " To You !"
5     nai = nai + 2
6
7     print("Wnr-3) new m =", m, ", id(m) =", id(m))
8     print("r-4) new nai =", nai, ", id(nai) =", id(nai))
9
10
11 #main
12 msg = "Happy Birthday"
13 age = 20
14
15 print("1) msg =", msg, ", id(msg) =", id(msg))
16 print("2) age =", age, ", id(age) =", id(age))
17 print()
18
19 modify1(msg,age)
20 print("Wn3) msg =", msg, ", id(msg) =", id(msg))
21 print("4) age =", age, ", id(age) =", id(age))
22
```

- Call by value
  - Caller의 actual argument address와 callee의 formal parameter address가 서로 다름

1) msg = Happy Birthday , id(msg) = 2571982355504  
2) age = 20 , id(age) = 2571942128528

r-1) m = Happy Birthday , id(m) = 2571982355504  
r-2) nai = 20 , id(nai) = 2571942128528

r-3) new m = Happy Birthday To You !! .id(m) = 2571982372912  
r-4) new nai = 22 , id(nai) = 2571942128592

3) msg = Happy Birthday , id(msg) = 2571982355504  
4) age = 20 , id(age) = 2571942128528

```
1 def modify1(m, nai):
2     print("r-1) m =", m, ", id(m) =", id(m))
3     print("r-2) nai =", nai, ", id(nai) =", id(nai))
4     m[0] = " Happy Birthday To You !"
5     nai[0] = 22
6
7     print("Wnr-3) new m =", m, ", id(m) =", id(m))
8     print("r-4) new nai =", nai, ", id(nai) =", id(nai))
9
10 #main
11 msg = ["Happy Birthday"]
12 age = [20]
13
14 print("1) msg =", msg, ", id(msg) =", id(msg))
15 print("2) age =", age, ", id(age) =", id(age))
16 print()
17
18 modify1(msg, age)
19 print("Wn3) msg =", msg, ", id(msg) =", id(msg))
20 print("4) age =", age, ", id(age) =", id(age))
```

- Call by reference
  - Caller의 actual argument address와 callee의 formal parameter address가 서로 같음

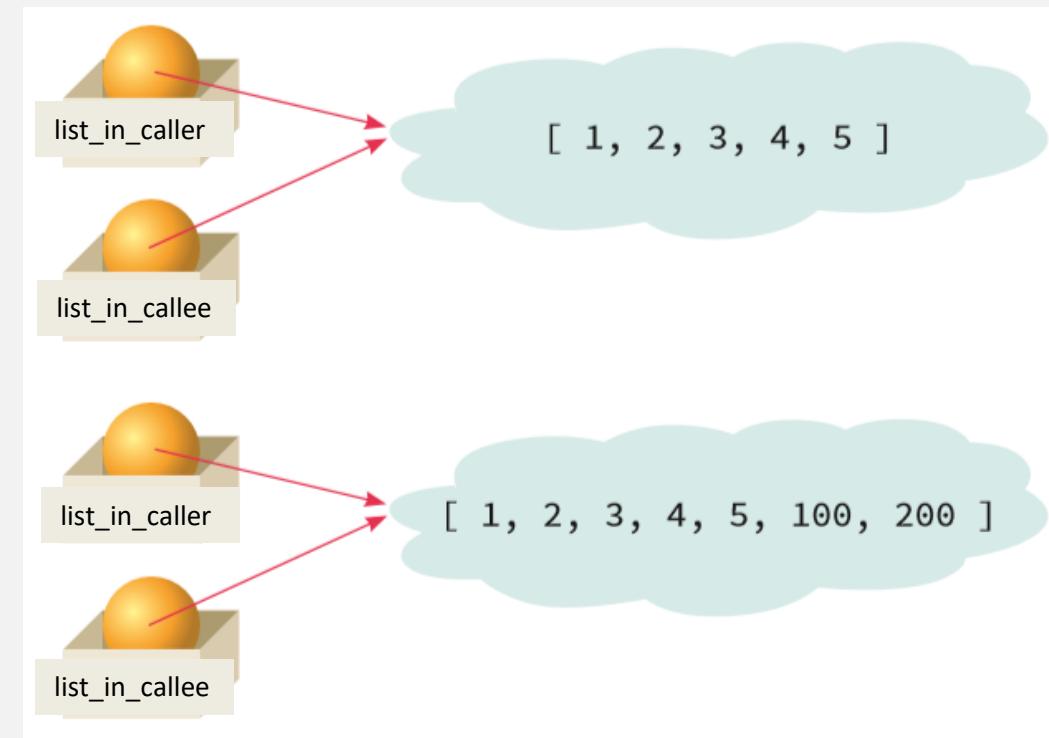
```
1) msg = ['Happy Birthday'] , id(msg) = 1919882023232
2) age = [20] , id(age) = 1919877136320
r-1) m = ['Happy Birthday'] , id(m) = 1919882023232
r-2) nai = [20] , id(nai) = 1919877136320
r-3) new m = [' Happy Birthday To You !!'] , id(m) = 1919882023232
r-4) new nai = [22] , id(nai) = 1919877136320
3) msg = [' Happy Birthday To You !!'] , id(msg) = 1919882023232
4) age = [22] , id(age) = 1919877136320
```

# Call by reference 예제 2)

call\_by\_ref\_2.py

```
File Edit Format Run Options Window Help
1 def modify2(list_in_callee):
2     print("Wnr-1) list_in_callee = ", list_in_callee)
3     list_in_callee += [100,200]
4     print("r-2) list_in_callee = ", list_in_callee)
5
6 #main
7 list_in_caller = [1,2,3,4,5]
8 print("Wn1) list_in_caller = ", list_in_caller)
9
10 modify2(list_in_caller)
11 print("Wn2) list_in_caller = ", list_in_caller)
12
```

```
1) list_in_caller = [1, 2, 3, 4, 5]
r-1) list_in_callee = [1, 2, 3, 4, 5]
r-2) list_in_callee = [1, 2, 3, 4, 5, 100, 200]
2) list_in_caller = [1, 2, 3, 4, 5, 100, 200]
```



# Passing arguments to a function

5.1passing-function.py

File Edit Format Run Options Window Help

```
1 def exp(g,m):  
2     print("ne-1) In exp. ","return g(", m,") --> f1 or f2(",m,")")  
3     return(g(m))  
4  
5 def f1(x):  
6     print("f1-1) In f1(", x, ")")  
7     print("f1-2) x*x = ", x*x)  
8     return x*x  
9  
10 def f2(x):  
11     print("f2-1) In f2(",x, ")")  
12     print("f2-2) x*x*x = ", x*x*x)  
13     return x*x*x  
14  
15  
16 # main  
17 n = 5  
18 print("1) In main, n =", n)  
19 print("2) call exp(f1," ,n, ")")  
20 print("n3) 결과 :", n, " , ", exp(f1, n))  
21  
22 print("n4) call exp(f2," ,n, ")")  
23 print("n5) 결과 ", n, " , ", exp(f2,n))  
24
```

- “함수” 자체도 함수의 인자가 될 수 있음

1) In main, n = 5  
2) call exp(f1, 5 )

e-1) In exp. return g( 5 ) --> f1 or f2( 5 )  
f1-1) In f1( 5 )  
f1-2) x\*x = 25

3) 결과 : 5 , 25

4) call exp(f2, 5 )

e-1) In exp. return g( 5 ) --> f1 or f2( 5 )  
f2-1) In f2( 5 )  
f2-2) x\*x\*x = 125

5) 결과 5 , 125  
````

# Returning a value

- 반환값(return value)
  - Caller로 반환되는 Callee에서 실행한 결과값

```
value = get_sum( 1 , 10 )
```

```
def get_sum ( start , end ):  
    sum=0  
    for i in range(start, end+1) :  
        sum += i  
    return sum
```

# Returning a value

- 함수 결과값 리턴
  - return (결과값)

5.1example7.py

File Edit Format Run Options Window Help

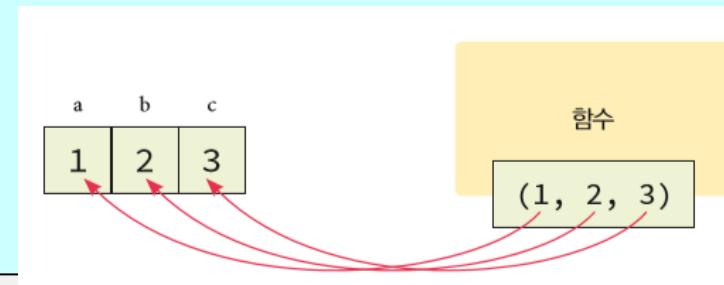
```
1 # Function to calculate tax and return the total
2 def calculateTax(price, tax_rate):
3     total = price + (price * tax_rate)
4     return total          # return the total
5
6
7 # main
8 my_price = float(input ("1) Enter a price: "))
9
10 # call the function and store the result in totalPrice
11 totalPrice = calculateTax(my_price, 0.06)
12 print ("2) Price = ", my_price)
13 print ("3) Total price = ", totalPrice)
14
```

1) Enter a price: 100  
2) Price = 100.0  
3) Total price = 106.0

# 여러 개의 값 반환하기

```
def sub():
    return 1, 2, 3

a, b, c = sub()
print(a, b, c)
```



1 2 3

```

1 # Functions to calculate tax and return the total
2 D = 0
3
4 def input_values():
5     my_price = float(input (">> 상품의 가격을 입력하세요 : "))
6     tax_rate_percent = float(input (">> 적용 세율을 %단위로 입력하세요 : "))
7     my_tax_rate = tax_rate_percent / 100
8
9     if D:
10         print("i-1) my_price : {}, my_tax_rate :{}".format(my_price, my_tax_rate))
11
12     return my_price, my_tax_rate
13
14 def calculateTax(product_price, state_tax_rate):
15     total = product_price + (product_price * state_tax_rate)
16
17     return total
18
19 # 출처 : https://skogkatt.tistory.com/87
20 def what_day_is_today():
21     import datetime
22
23     now = datetime.datetime.now()
24     t =[ '월', '화', '수', '목', '금', '토', '일' ]
25     r = datetime.datetime.today().weekday()
26
27     day = str(now.year) + '년' + \
28           str(now.month) + '월' + \
29           str(now.day) + '일' + \
30           str(t[r]) + '요일' + \
31           str(now.hour) + '시' + \
32           str(now.minute) + '분' + \
33           str(now.second) + '초'
34
35     if D:
36         print("##1) day : ", day)
37
38     return day
39

```

```

40 def print_receipt(product_price, state_tax_rate, total):
41
42     date_of_sales = what_day_is_today()
43
44     print("WnWn====")
45     print("                영 수 증")
46     print("-----")
47     print(" * 구매해 주셔서 감사합니다.")
48     print(" 판매시간 : ", date_of_sales)
49     print(" 세전 상품가격 : ", product_price, "원")
50     print(" 세율 : ", state_tax_rate * 100, "%")
51     print(" 세후 상품가격 : ", total, "원")
52     print("====")
53
54 # main
55 # 상품의 가격과 세금요율을 입력받음
56 price, tax_rate = input_values()
57
58 # 세금을 포함한 상품가격
59 totalPrice = calculateTax(price, tax_rate)
60
61 #영수증 출력
62 print_receipt(price, tax_rate, totalPrice)

```

```

>> 상품의 가격을 입력하세요 : 100
>> 적용 세율을 %단위로 입력하세요 : 8

=====
영 수 증
-----
* 구매해 주셔서 감사합니다.
판매시간 : 2023년 10월 30일 월요일 21시 14분 23초
세전 상품가격 : 100.0 원
세율 : 8.0 %
세후 상품가격 : 108.0 원
=====
```

# Keyword argument

- 인수들이 위치가 아니고 keyword에 의하여 함수로 전달되는 방식

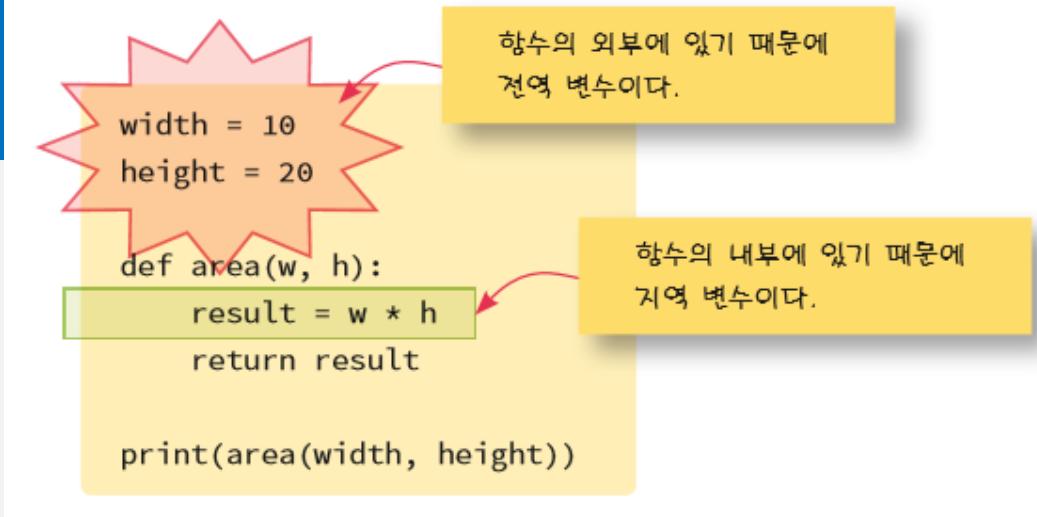
The screenshot shows a code editor window with the file `keyword_argument.py`. The code defines a function `calc` and demonstrates its use with various keyword arguments. The output on the right shows the results of running the code.

```
keyword_argument.py
File Edit Format Run Options Window Help
1 def calc(x, y, z):
2     print("Wnc-1) In calc(), x,y,z =", x,y,z)
3     return x+y+z
4
5 #main
6 value = calc(10, 20, 30)
7 print("1) value =", value)
8
9 value = calc(x=10, y=20, z=30)
10 print("2) value =", value)
11
12 value = calc(z=30, y=20, x=10)
13 print("3) value =", value)
14
15
16 value = calc(y=100, z=200, x=300)
17 print("4) value =", value)
18
```

c-1) In calc(), x,y,z = 10 20 30  
1) value = 60  
  
c-1) In calc(), x,y,z = 10 20 30  
2) value = 60  
  
c-1) In calc(), x,y,z = 10 20 30  
3) value = 60  
  
c-1) In calc(), x,y,z = 300 100 200  
4) value = 600

# Variable scope

- 범위(scope)에 따른 변수(variable) 구분
  - 지역(Local) 변수
    - 함수나 Class 내부에서만 사용
    - Formal parameter는 지역변수
    - 예) result, w, h
  - 전역(Global) 변수: program 파일 안에서 사용  
예) width, height
  - 빌트인(Built-in) 변수: Python에서 특별 정의하여 사용
- Python에서 변수 이름 찾는 순서
  - 지역 → 전역 → 빌트인
  - 그래도 존재하지 않으면 error 발생



# Variable scope

- Python이 전역변수를 다루는 방식은 타 언어와 **아주 다른** 접근 방식을 활용
  - 다른 설정이 없으면 함수 안에서 선언된 변수는 무조건 지역변수
  - 전역변수는 함수 내에서 “read-only”로 사용
- Local/Global 변수 접근 에러
  - 지역 변수를 해당 함수(클래스) 외부에서 사용하려는 경우
  - 전역 변수의 값을 함수(클래스)에서 변경하려는 경우
    - 예외: global로 선언해주면 가능

# Variable scope (local / global variable)

```
width = 10  
height = 20  
  
def area(w, h):  
    result = w * h  
    return result  
  
print(area(width, height))
```

함수의 외부에 있기 때문에  
전역 변수이다.

함수의 내부에 있기 때문에  
지역 변수이다.

# Local variable

- 변수 s는 함수 sub() 안에서만 사용
- 함수의 호출이 종료되면 s는 사라짐
- 지역 변수를 해당 함수(클래스) 외부에서 사용하려는 경우 error 발생

local\_variable\_1.py

File Edit Format Run Options Window Help

```
1 def sub():
2     print("s-1) In sub(), 변수 s는 local variable")
3
4     s = "바나나가 좋음!" # s : local variable
5     print("s-2) s =", s)
6
7 # main
8 print("\n1) main, call sub()")
9 sub()
10
11 print("\n2) main에서 local variable s 접근 시도 : error !")
12 print(s) # sub()의 지역변수 s 접근 시도. error !
13
```

```
1) main, call sub()
s-1) In sub(), 변수 s는 local variable
s-2) s = 바나나가 좋음!

2) main에서 local variable s 접근 시도 : error !
Traceback (most recent call last):
  File "C:\소사\강의\제작\local_variable_1.py", line 12, in <module>
    print(s) # sub()의 지역변수 s 접근 시도. error !
  NameError: name 's' is not defined
```

# Global variable

- 전역변수 s를 함수 sub()에서 “read-only”로 사용

global\_variable\_1.py

File Edit Format Run Options Window Help

```
1 def sub():
2     print("Wn1) in sub(), global variable s 접근 : OK !")
3     print("s2) s = ", s, ", id(s) = ", id(s))
4
5 # main
6 print("Wn1) in main, global variable s 선언")
7
8 s = "사과가 좋음 !" # s : global variable
9 print("2) s = ", s, ", id(s) = ", id(s))
10
11 sub()
12 print("Wn3) end.")
```

1) in main, global variable s 선언  
2) s = 사과가 좋음 ! , id(s) = 2443694283776  
s1) in sub(), global variable s 접근 : OK !  
s2) s = 사과가 좋음 ! , id(s) = 2443694283776  
3) end.

# 지역 변수와 전역 변수

```
def sub():
    s = "바나나가 좋음!"
    print("s-1) s =", s)
#main
s = "사과가 좋음!"
sub()
print("1) in main, s =", s)
```

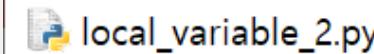
전역 변수를 사용하는 것이 아님!  
지역 변수 s를 새로이 정의하는 것임

```
s-1) s = 바나나가 좋음!
1) in main, s = 사과가 좋음!
```

- Python이 전역변수를 다루는 방식은 타 언어와 아주 다른 접근 방식을 활용
- 다른 설정이 없으면 함수 안에서 선언된 변수는 무조건 지역변수

# 지역 변수와 전역 변수

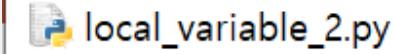
```
1) s = 사과가 좋음! , id(s) = 1500220271616
Traceback (most recent call last):
  File "C:\과소사\과소사-강의예제\local_variable_2.py", line 11, in <module>
    sub()
  File "C:\과소사\과소사-강의예제\local_variable_2.py", line 2, in sub
    print("s1) s = ", s, ", id(s) =", id(s))
UnboundLocalError: local variable 's' referenced before assignment
```



File Edit Format Run Options Window Help

```
1 def sub():
2     #print("s1) s = ", s, ", id(s) =", id(s))
3
4     s = "바나나가 좋음!"
5     print("s2) s = ", s, ", id(s) =", id(s))
6
7 #main
8 s = "사과가 좋음!"
9 print("1) s = ", s, ", id(s) =", id(s) )
10
11 sub()
12 print("Wn2) s = ", s, ", id(s) =", id(s) )
13
```

```
1) s = 사과가 좋음! , id(s) = 2356634631168
s2) s = 바나나가 좋음! , id(s) = 2356634402704
2) s = 사과가 좋음! , id(s) = 2356634631168
```



File Edit Format Run Options Window Help

```
1 def sub():
2     print("s1) s = ", s, ", id(s) =", id(s))
3
4     s = "바나나가 좋음!"
5     print("s2) s = ", s, ", id(s) =", id(s))
6
7 #main
8 s = "사과가 좋음!"
9 print("1) s = ", s, ", id(s) =", id(s) )
10
11 sub()
12 print("Wn2) s = ", s, ", id(s) =", id(s) )
13
```

```
1) s = 사과가 좋음! , id(s) = 2017504731616
Traceback (most recent call last):
  File "C:\소사\강의예제\global_variable_2.py", line 11, in <module>
    sub()
  File "C:\소사\강의예제\global_variable_2.py", line 2, in sub
    print("s1) s = ", s, ", id(s) =", id(s))
UnboundLocalError: cannot access local variable 's' where it is not associated with a value
```

# 전역 변수를 함수 안에서 사용하려면

```
def sub():
    global s

    print("s-1) s = ", s)
    s = "바나나가 좋음!"
    print("s-2) s = ", s)

# main
s = "사과가 좋음!"
sub()
print("1) s = ", s)
```

```
s-1) s = 사과가 좋음!
s-2) s = 바나나가 좋음!
1) s = 바나나가 좋음!
```

# 지역 변수와 전역 변수

local\_variable\_2.py

File Edit Format Run Options Window Help

```
1 def sub():
2     print("s1) s = ", s, ", id(s) =", id(s))
3
4     s = "바나나가 좋음!"
5     print("s2) s = ", s, ", id(s) =", id(s))
6
7 #main
8 s = "사과가 좋음!"
9 print("1) s = ", s, ", id(s) =", id(s) )
10
11 sub()
12 print("2) s = ", s, ", id(s) =", id(s) )
13
```

1) s = 사과가 좋음! ,id(s) = 2017504731616

Traceback (most recent call last):

File "C:\소사\강의\예제\global\_variable\_2.py", line 11, in <module>

sub()

File "C:\소사\강의\예제\global\_variable\_2.py", line 2, in sub

print("s1) s = ", s, ", id(s) =", id(s))

UnboundLocalError: cannot access local variable 's' where it is not associated with a value

local\_variable\_3.py

File Edit Format Run Options Window Help

```
1 def sub():
2     global s
3     print("1) s = ", s, ", id(s) =", id(s))
4
5     s = "바나나가 좋음!"
6     print("2) s = ", s, ", id(s) =", id(s))
7
8 #main
9 s = "사과가 좋음!"
10 print("3) s = ", s, ", id(s) =", id(s) )
11
12 sub()
13 print("4) s = ", s, ", id(s) =", id(s) )
14
```

1) s = 사과가 좋음! , id(s) = 2848019164160

s1) s = 사과가 좋음! , id(s) = 2848019164160

s2) s = 바나나가 좋음! , id(s) = 2848018939792

2) s = 바나나가 좋음! , id(s) = 2848018939792

# 예제

```
def sub(x, y):  
    global a  
  
    a = 7  
    x, y = y, x  
    b = 3  
    print(a, b, x, y)
```

```
a,b,x,y = 1,2,3,4  
sub(x, y)  
print(a, b, x, y)
```

```
7 3 4 3  
7 2 3 4
```

# Variable scope

- Local/global 변수 접근 에러
  - 함수(Class)에서 global 변수 (`my_price`) 접근 가능

```
5.1example8.py

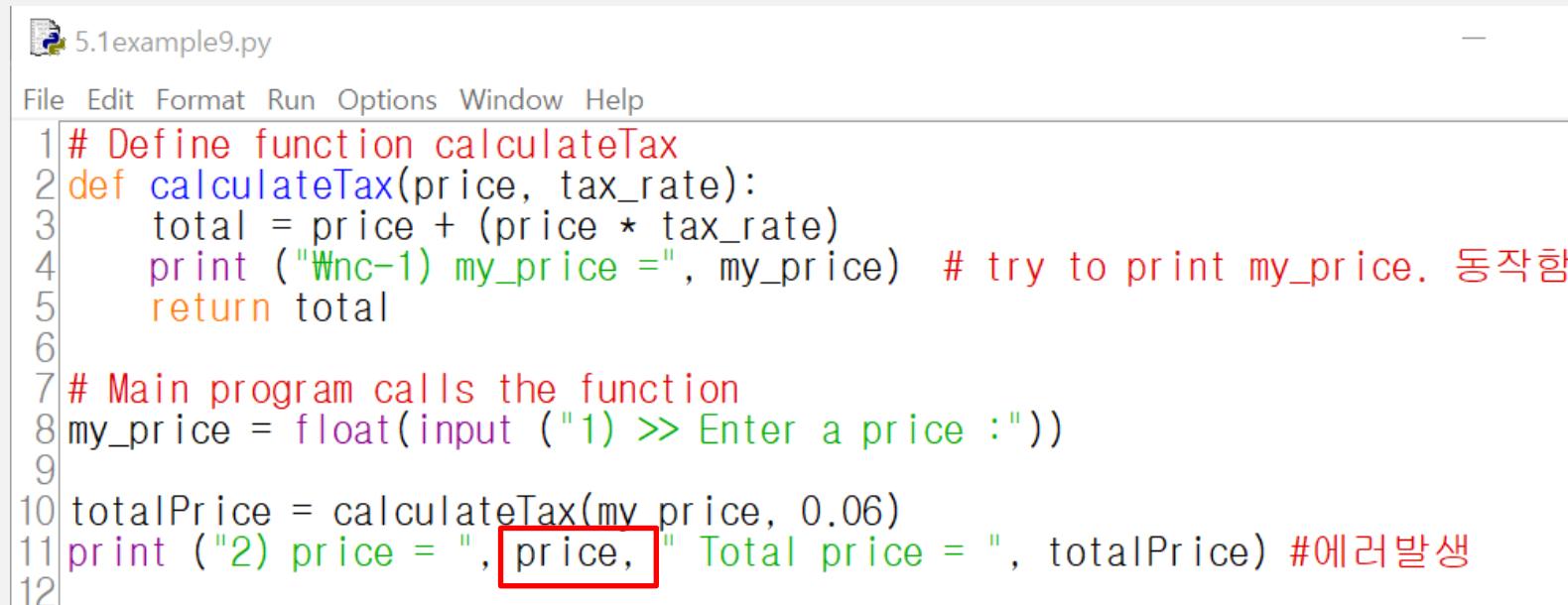
File Edit Format Run Options Window Help
def calculateTax(price, tax_rate):
    total = price + (price * tax_rate)
    print ("1) my_price =", my_price) # try to print my_price. 동작함
    return total

# main
my_price = float(input ("> Enter a price : "))
totalPrice = calculateTax(my_price, 0.06)    >> Enter a price : 100
print("1) price =", my_price)
print("2) Total price =", totalPrice)         c-1) my_price = 100.0
```

1) price = 100.0  
2) Total price = 106.0

# Variable scope

- Local/global 변수 접근 에러
  - 외부에서 함수(클래스)의 지역 변수 (price) 접근 에러

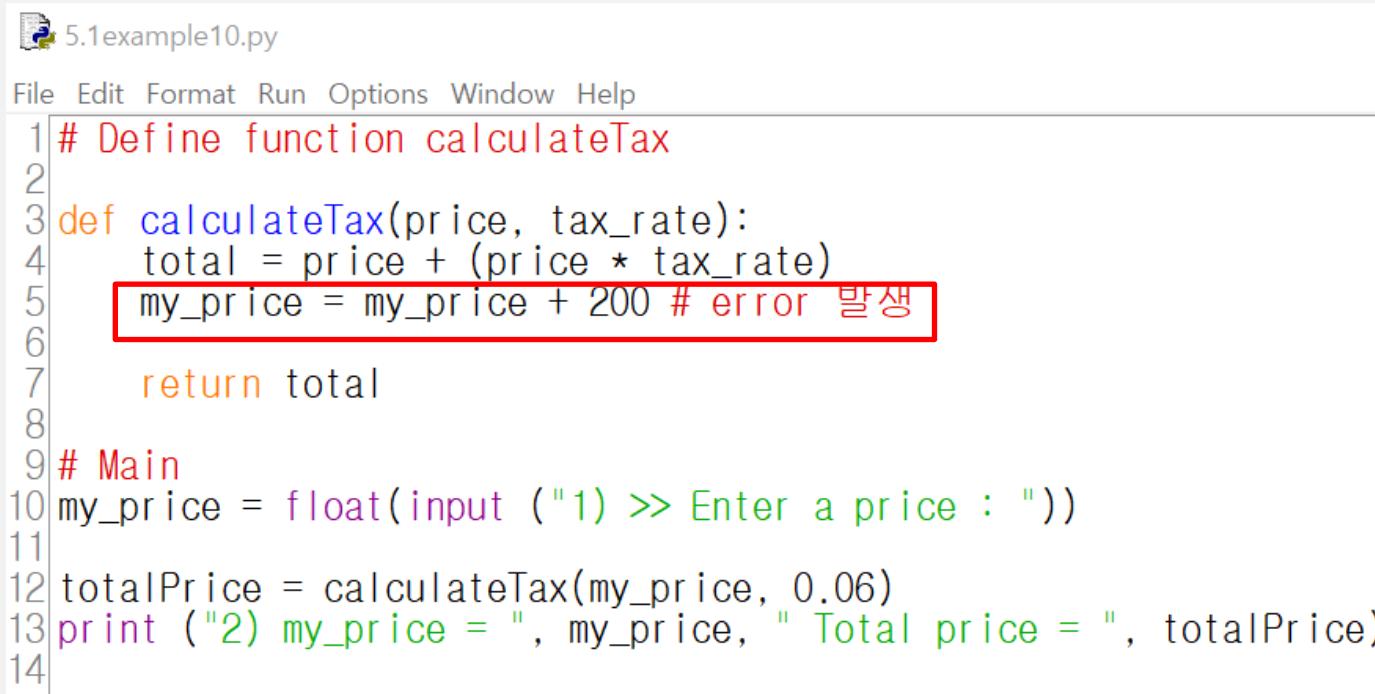


```
5.1example9.py
File Edit Format Run Options Window Help
1 # Define function calculateTax
2 def calculateTax(price, tax_rate):
3     total = price + (price * tax_rate)
4     print ("c-1) my_price =", my_price) # try to print my_price. 동작함
5     return total
6
7 # Main program calls the function
8 my_price = float(input ("1) >> Enter a price :"))
9
10 totalPrice = calculateTax(my_price, 0.06)
11 print ("2) price = ", price, " Total price = ", totalPrice) #에러발생
12
```

```
1) >> Enter a price :100
c-1) my_price = 100.0
Traceback (most recent call last):
  File "C:\소사\강의예제\5.1example9.py", line 11, in <module>
    print ("2) price = ", price, " Total price = ", totalPrice) #에러발생
NameError: name 'price' is not defined. Did you mean: 'print'?
```

# Variable scope

- Local/global 변수 접근 에러
  - 함수(클래스)에서 'global' 표시 없이 전역 변수(`my_price`) 값을 변경하려 하여 에러 발생

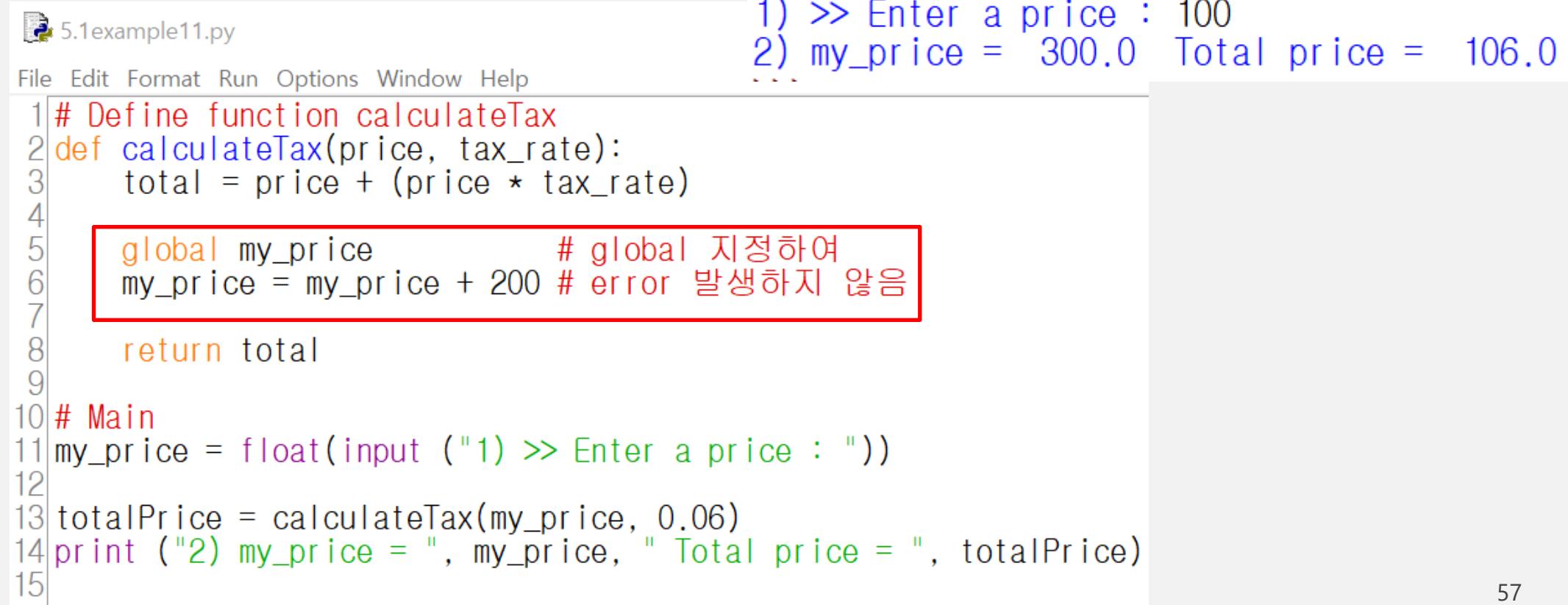


```
5.1example10.py
File Edit Format Run Options Window Help
1 # Define function calculateTax
2
3 def calculateTax(price, tax_rate):
4     total = price + (price * tax_rate)
5     my_price = my_price + 200 # error 발생
6
7     return total
8
9 # Main
10 my_price = float(input ("1) >> Enter a price : "))
11
12 totalPrice = calculateTax(my_price, 0.06)
13 print ("2) my_price = ", my_price, " Total price = ", totalPrice)
14
```

1) >> Enter a price : 100  
Traceback (most recent call last):  
 File "C:\소사\강의예제\W5.1example10.py", line 12, in <module>  
 totalPrice = calculateTax(my\_price, 0.06)  
 File "C:\소사\강의예제\W5.1example10.py", line 5, in calculateTax  
 my\_price = my\_price + 200 # error 발생  
UnboundLocalError: cannot access local variable 'my\_price' where it is not associated with a value

# Variable scope

- Local/global 변수 접근 에러
  - 함수(클래스)에서 ‘global’ 표시 하면 전역 변수 값을 변경 가능함
    - 하지만, 전역 변수를 함수(클래스)에서 변경하는 것은 좋지 않은 프로그래밍 습관임



```
5.1example11.py
File Edit Format Run Options Window Help
1 # Define function calculateTax
2 def calculateTax(price, tax_rate):
3     total = price + (price * tax_rate)
4
5     global my_price          # global 지정하여
6     my_price = my_price + 200 # error 발생하지 않음
7
8     return total
9
10 # Main
11 my_price = float(input ("1) >> Enter a price : "))
12
13 totalPrice = calculateTax(my_price, 0.06)
14 print ("2) my_price = ", my_price, " Total price = ", totalPrice)
15
```

1) >> Enter a price : 100  
2) my\_price = 300.0 Total price = 106.0

# Naming variables

- 동일한 이름의 전역변수와 지역변수 사용은 좋지 않은 습관
  - 프로그래밍 오류 발생의 원인
  - 아래 코드는 동작하나 좋지 않은 사례

The screenshot shows a Python code editor with a file named "5.1example12.py". The code defines a function "calculateTax" and uses it in a "Main" section. A terminal window to the right shows the execution of the code and its output.

```
5.1example12.py
File Edit Format Run Options Window Help
1 # Define function calculateTax
2 def calculateTax(price, tax_rate):
3     total = price + (price * tax_rate)
4     my_price = 200 #에러는 발생하지 않음. 왜?
5
6     return total
7
8 # Main
9 my_price = float(input ("1) >> Enter a price : "))
10
11 totalPrice = calculateTax(my_price, 0.06)
12 print ("2) my_price = ", my_price, " Total price = ", totalPrice)
13
```

1) >> Enter a price : 100  
2) my\_price = 100.0 Total price = 106.0

# Lab: 매개변수 = 지역변수

- 다음 program의 실행 결과는 어떻게 될까?
  - 동일한 이름의 전역변수와 지역변수 사용은 좋지 않은 습관에 따른 abnormal cases

```
# 함수가 정의된다.  
def sub( mylist ):  
    # 리스트가 함수로 전달된다.  
    mylist = [1, 2, 3, 4] # 새로운 리스트가 매개변수로 할당된다.  
    print ("함수 내부에서의 mylist: ", mylist)  
    return  
  
# main에서 sub() 함수를 호출한다.  
mylist = [10, 20, 30, 40];  
sub( mylist );  
print ("함수 외부에서의 mylist: ", mylist)
```

```
함수 내부에서의 mylist: [1, 2, 3, 4]  
함수 외부에서의 mylist: [10, 20, 30, 40]
```

local\_global\_variable\_0.py

```

File Edit Format Run Options Window Help
1 def sub( mylist ):
2
3     print ("Wns-1) sub 함수로 전달된 mylist: ", mylist,
4         ", id(mylist) =", id(mylist))
5
6     mylist = [1, 2, 3, 4] # 매개변수 mylist와 동일한 이름의
7         # 새로운 리스트 생성
8     #mylist = mylist + [1, 2, 3, 4]
9     #mylist += [1, 2, 3, 4]
10
11    print ("s-2) sub 함수에서 mylist를 새로이 정의 후: ", mylist,
12        ", id(mylist) =", id(mylist))
13    return
14
15 #main
16 mylist = [10, 20, 30, 40];
17 print ("Wn1) main에서 정의된 mylist: ", mylist,
18     ", id(mylist) =", id(mylist))
20
21 sub(mylist);
22 print ("Wn2) 리턴된 후 함수 외부(main)에서의 mylist: ", mylist,
23     ", id(mylist) =", id(mylist))
24

```

local\_global\_variable\_0\_1.py

```

File Edit Format Run Options Window Help
1 def sub( mylist ):
2
3     print ("Wns-1) sub 함수로 전달된 mylist: ", mylist,
4         ", id(mylist) =", id(mylist))
5
6     mylist[:] = [1, 2, 3, 4] # 새로운 리스트가 아닌 매개변수로
7         # 넘겨 받은 mylist를 변경
8
9     print ("s-2) sub 함수에서 mylist를 새로이 정의 후: ", mylist,
10        ", id(mylist) =", id(mylist))
11    return
12
13
14 #main
15 mylist = [10, 20, 30, 40];
16 print ("Wn1) main에서 정의된 mylist: ", mylist,
17     ", id(mylist) =", id(mylist))
18
19 sub(mylist);
20 print ("Wn2) 리턴된 후 함수 외부(main)에서의 mylist: ", mylist,
21     ", id(mylist) =", id(mylist))
22

```

Ln: 20 Col: 24

Ln: 24 Col: 0

1) main에서 정의된 mylist: [10, 20, 30, 40] , id(mylist) = 2317512264320  
 s-1) sub 함수로 전달된 mylist: [10, 20, 30, 40] , id(mylist) = 2317512264320  
 s-2) sub 함수에서 mylist를 새로이 정의 후: [1, 2, 3, 4] , id(mylist) = 2317512493056  
 2) 리턴된 후 함수 외부(main)에서의 mylist: [10, 20, 30, 40] , id(mylist) = 2317512264320

1) main에서 정의된 mylist: [10, 20, 30, 40] , id(mylist) = 2502004558336  
 s-1) sub 함수로 전달된 mylist: [10, 20, 30, 40] , id(mylist) = 2502004558336  
 s-2) sub 함수에서 mylist를 새로이 정의 후: [1, 2, 3, 4] , id(mylist) = 2502004558336  
 2) 리턴된 후 함수 외부(main)에서의 mylist: [1, 2, 3, 4] , id(mylist) = 2502004558336

```

1 def sub(mylist, mycomment):
2
3     print ("Wns1) mylist:", mylist, ", id(mylist) =", id(mylist))
4     print ("    mycomment:", mycomment, ", id(mycomment) =", id(mycomment))
5
6 #   global comment
7     print ("s2) comment :", comment, ", id(comment) = ", id(comment))
8
9     mylist += ["NEW", "LIST"]
10    print ("s3) mylist에 값 추가 후 :", mylist, ", id(mylist) =", id(mylist))
11
12 # 새로운 local 변수인 mylist를 선언하여 새로운 값을 할당
13 mylist = [1, 2, 3, 4]
14 mycomment = "Sub 입니다."
15
16 #   comment = "변경된 값"
17
18     print ("Wns4) sub 함수에서 mylist를 새로이 정의 후:", mylist, ", id(mylist) =", id(mylist))
19     print ("s5) sub 함수에서 mycomment를 새로이 정의 후:", mycomment,
20           ", id(comment) =", id(comment))
21
22     return mylist, mycomment
23
24
25 #main
26 print("1) main")
27
28 mylist = [10, 20, 30, 40]
29 comment = "Main 입니다."
30 print ("Wn2) mylist :", mylist, ", id(mylist) :", id(mylist))
31 print ("3) comment :", comment, ", id(comment) :", id(comment))
32
33 ret_list, ret_comment = sub(mylist, comment);
34
35 print("Wn4) 리턴된 값 ret_list :", ret_list, ", id(ret_list) =", id(ret_list))
36 print("5) mylist :", mylist, ", id(mylist) =", id(mylist))
37
38 print("Wn6) comment :", comment, ", id(comment) =", id(comment))
39 print("7) ret_comment :", ret_comment, ", id(ret_comment) =", id(ret_comment))
40

```

1) main  
 2) mylist : [10, 20, 30, 40] , id(mylist) : 2997114259456  
 3) comment : Main 입니다. , id(comment) : 2997113856480  
 s1) mylist: [10, 20, 30, 40] , id(mylist) = 2997114259456  
 mycomment: Main 입니다. , id(mycomment) = 2997113856480  
 s2) comment : Main 입니다. , id(comment) = 2997113856480  
 s3) mylist에 값 추가 후 : [10, 20, 30, 40, 'NEW', 'LIST'] , id(mylist) = 2997114259456  
 s4) sub 함수에서 mylist를 새로이 정의 후: [1, 2, 3, 4] , id(mylist) = 2997114030720  
 s5) sub 함수에서 mycomment를 새로이 정의 후: Sub 입니다. , id(comment) = 2997113852112  
 4) 리턴된 값 ret\_list : [1, 2, 3, 4] , id(ret\_list) = 2997114030720  
 5) mylist : [10, 20, 30, 40, 'NEW', 'LIST'] , id(mylist) = 2997114259456  
 6) comment : Main 입니다. , id(comment) = 2997113856480  
 7) ret\_comment : Sub 입니다. , id(ret\_comment) = 2997113852112

# Lab: 상수 (Constant)

- 파이를 전역 변수로 선언하고 이것을 이용하여서 원의 면적과 원의 둘레를 계산하는 함수를 작성해보자.

```
>> 원의 반지름을 입력하시오 : 10
```

```
원의 면적 : 314.159265358979
```

```
원의 둘레 : 62.8318530717958
```

# Solution



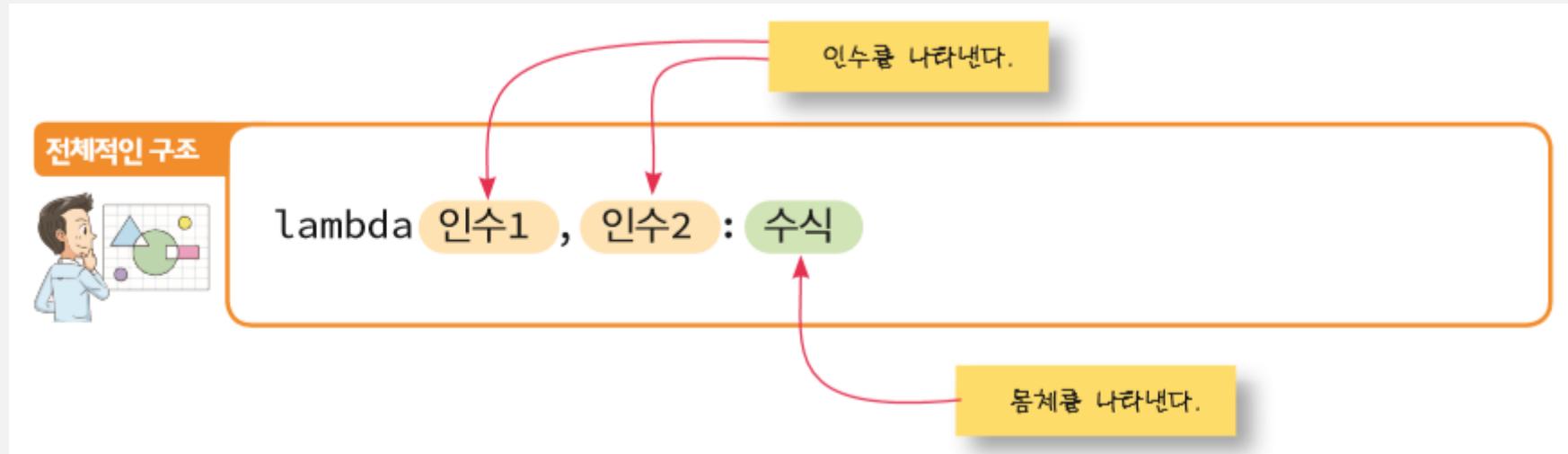
area\_of\_circle.py

```
File Edit Format Run Options Window Help
1 PI = 3.14159265358979    # 전역 상수
2
3 def circleArea(radius):
4     return PI*radius*radius
5
6 def circleCircumference(radius):
7     return 2*PI*radius
8
9 def main():
10    radius = float(input('Wn>> 원의 반지름을 입력하시오 : '))
11    print('원의 면적 : ', circleArea(radius))
12    print('원의 둘레 : ', circleCircumference(radius))
13
14 #main
15 main()
16
```

```
>> 원의 반지름을 입력하시오 : 10
원의 면적 : 314.159265358979
원의 둘레 : 62.8318530717958
```

# lambda function(무명 함수)

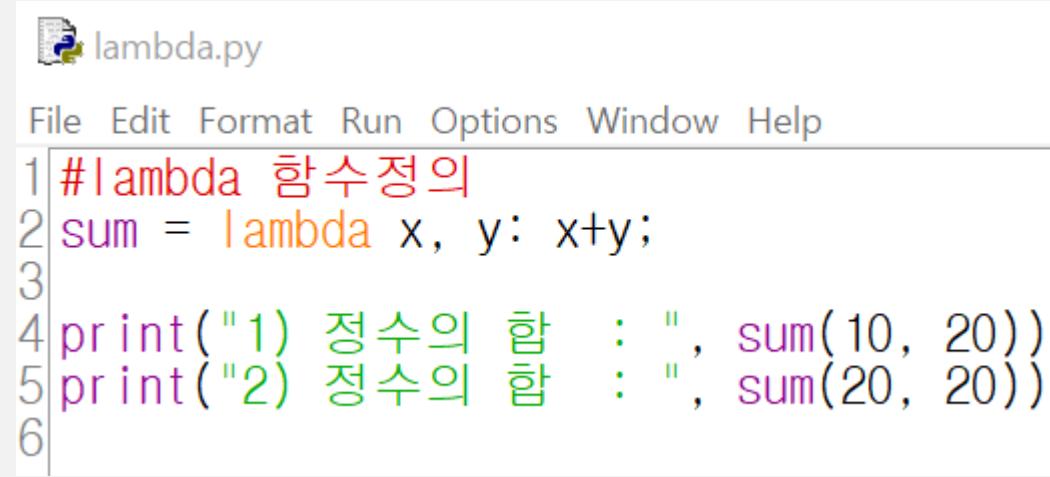
- 이름은 없고 몸체만 있는 함수
  - 이름없는 함수
  - Python에서 lambda function은 lambda keyword를 사용
  - 한 줄 함수
- lambda 인수: 아래의 예제에서 “인수1”, “인수2”



# lambda function(무명 함수)

- 주요 용도
  - Code 안에서 함수를 포함하는 곳 어디서든 활용 가능
  - GUI event를 처리하는 callback handler 등에서 사용
  - Jump table 등

# lambda function 예 1



```
lambda.py
File Edit Format Run Options Window Help
1 #lambda 함수정의
2 sum = lambda x, y: x+y;
3
4 print("1) 정수의 합 : ", sum(10, 20))
5 print("2) 정수의 합 : ", sum(20, 20))
6
```

```
1) 정수의 합 : 30
2) 정수의 합 : 40
``
```

# lambda function 예 2

lambda2.py

File Edit Format Run Options Window Help

```
1 # 예제 1
2 Function_List = [ lambda x : x**2,
3                   lambda y : y**3,
4                   lambda z : print("lambda {}**4 = {}".format(z, z**4)) ]
5
6 for f in Function_List:
7     print("1) f(2) =", f(2))
8     print("2) f(10) =", f(10), "\n")
9
10 # 예제 2
11 min = (lambda x,y : x if x < y else y )
12
13 min_value = min(100, 200)
14 print("3) min =", min_value)
15
```

1) f(2) = 4  
2) f(10) = 100

1) f(2) = 8  
2) f(10) = 1000

lambda) 2\*\*4 = 16  
1) f(2) = None  
lambda) 10\*\*4 = 10000  
2) f(10) = None

3) min = 100

# lambda function 예3

5.1example16-1.py

```
File Edit Format Run Options Window Help
1 def exp(g, m):
2     print("e-1) start")
3     print("e-2) g :",g)
4     print("e-3) m =", m)
5     print("e-4) g(m) =", g(m))
6
7     return (g(m))
8
9 print("1) >")
10 f1 = lambda x: x*x
11
12 print("2) >>")
13 f2 = lambda x: x*x*x
14
15 print("3) >>>\n")
16
17 n = 10
18 print("4) f1(10) :",f1(n))
19 print("5) f2(10) :",f2(n))
20
21 print("6) exp(f1, 10) =", exp(f1, n))
22 print("7) exp(f2, 10) =", exp(f2, n))
23
```

```
1) >
2) >>
3) >>>
4) f1(10) : 100
5) f2(10) : 1000
e-1) start
e-2) g :<function <lambda> at 0x000001EF81B04940>
e-3) m = 10
e-4) g(m) = 100
6) exp(f1, 10) = 100
e-1) start
e-2) g :<function <lambda> at 0x000001EF81B049D0>
e-3) m = 10
e-4) g(m) = 1000
7) exp(f2, 10) = 1000
```

# lambda function 예4

- 예: List 정렬
  - Lambda 함수 정의: Tuple 첫 번째, 두 번째 값으로 정렬

The screenshot shows a code editor window with a file named "5.1example20.py". The code is as follows:

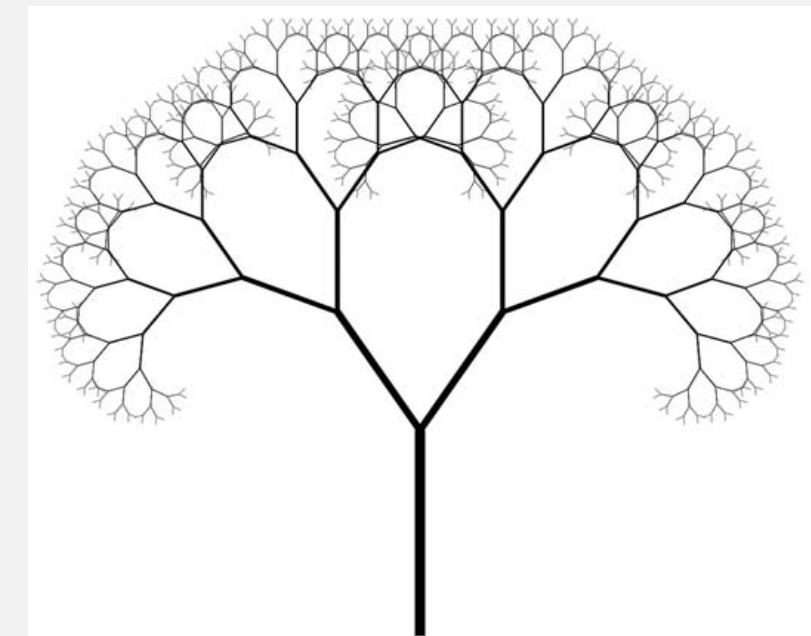
```
1 #tuple list 정렬
2 tuple_list = [ ("Fa",400), ("Re",200), ("Do",100), ("Mi",300) ]
3
4 print("1) Sorting 이전 :", tuple_list)
5
6 print("Wn2) x[0]로 sorting")
7 tuple_list.sort(key = lambda x: x[0])
8 print("3) After sorting :", tuple_list)
9
10 print("Wn4) x[1]로 sorting")
11 tuple_list.sort(key = lambda x: x[1])
12 print("5) After sorting :", tuple_list)
13
14 tuple_list.sort()
15 print("Wn6) sort() :", tuple_list)
16
```

To the right of the code, the output of each print statement is shown in blue:

- 1) Sorting 이전 : [('Fa', 400), ('Re', 200), ('Do', 100), ('Mi', 300)]
- 2) x[0]로 sorting
- 3) After sorting : [('Do', 100), ('Fa', 400), ('Mi', 300), ('Re', 200)]
- 4) x[1]로 sorting
- 5) After sorting : [('Do', 100), ('Re', 200), ('Mi', 300), ('Fa', 400)]
- 6) sort() : [('Do', 100), ('Fa', 400), ('Mi', 300), ('Re', 200)]

# Recursive Functions

- 재귀(再歸)함수(recursive function)
  - 자기 자신을 재호출하는 함수
  - 동일한 문제 해결 방식을 반복하는 상황에서 활용
  - 점화식(漸化式), 재귀식(再歸式, recurrence relation)
- Computer science 분야에서 흔하게 발생
  - 1부터 n까지 합산
    - 1부터 (n-1)까지 합산 + n
  - 피보나치 수열
    - $f(n) = f(n-1) + f(n-2)$ ,  $f(1)=f(2)=1$
  - 프랙탈(fractal)



# Recursive Functions

- 반복 패턴 + 종료 조건 (+ 시작 값들)
- 반복 함수 호출로 속도는 느림
  - 스택으로 부터 memory 할당 및 해제 반복  
Ex) 1부터 n까지 합산  
 $1부터 (n-1)까지 합산 + n$

5.1example13.py

File Edit Format Run Options Window Help

```
1 n = 100
2 sum=0
3
4 for i in range(1,n+1):
5     sum = sum + i
6
7 # main
8 print(">> n = {}, sum = {}".format(n, sum))
9
10
```

>> n = 100, sum = 5050  
...

5.1example14.py

File Edit Format Run Options Window Help

```
1 def sigma(n):
2     if n == 1:
3         return n
4     else:
5         return n + sigma(n-1)
6
7 # main
8 n=10
9 print(">> sigma({}) = {}".format(n, sigma(n)))
10
```

>> sigma(10) = 55  
...

# Recursive Functions

- 예1) 1부터 n까지 합산

$\text{sigma}(1) = 1$

$\text{sigma}(2) = \text{sigma}(1) + 2$

$\text{sigma}(3) = \text{sigma}(2) + 3$

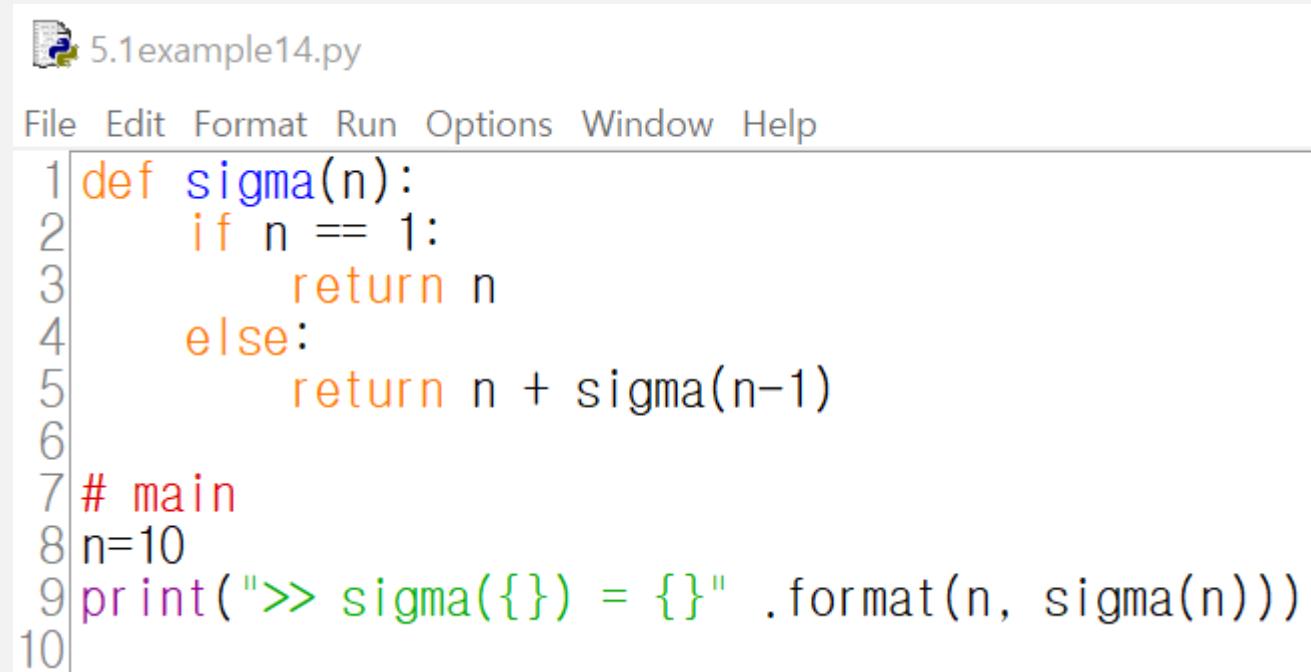
$\text{sigma}(4) = \text{sigma}(3) + 4$

.....

$\text{sigma}(n) = \text{sigma}(n-1) + n$

$\rightarrow \text{sigma}(1) = 1$

$\text{sigma}(n) = \text{sigma}(n-1) + n$



```
5.1example14.py
File Edit Format Run Options Window Help
1 def sigma(n):
2     if n == 1:
3         return n
4     else:
5         return n + sigma(n-1)
6
7 # main
8 n=10
9 print(">> sigma({}) = {}".format(n, sigma(n)))
10
```

# Recursive Functions

## – 예2) factorial

facto(0) = 1

facto(1) = 1 \* facto(0)

facto(2) = 2 \* facto(1)

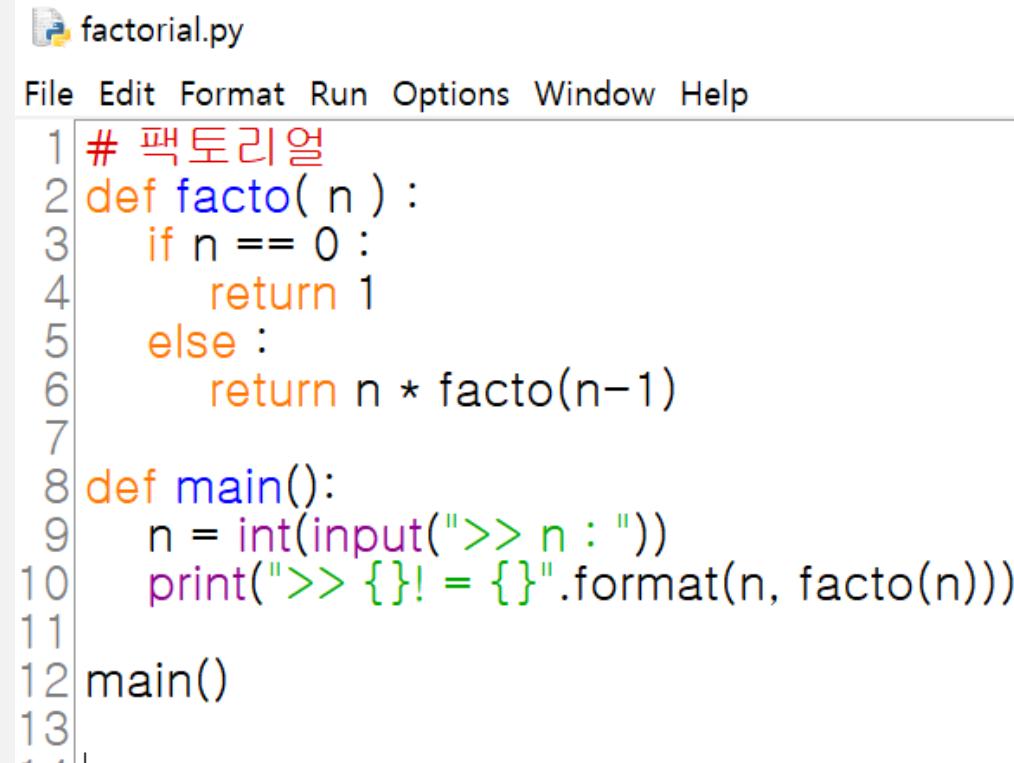
facto(3) = 3 \* facto(2)

.....

facto(n) = n \* facto(n-1)

→facto(0) = 1

facto(n) = n \* facto(n-1)



```
# 팩토리얼
def facto( n ) :
    if n == 0 :
        return 1
    else :
        return n * facto(n-1)

def main():
    n = int(input(">> n : "))
    print(">> {}! = {}".format(n, facto(n)))

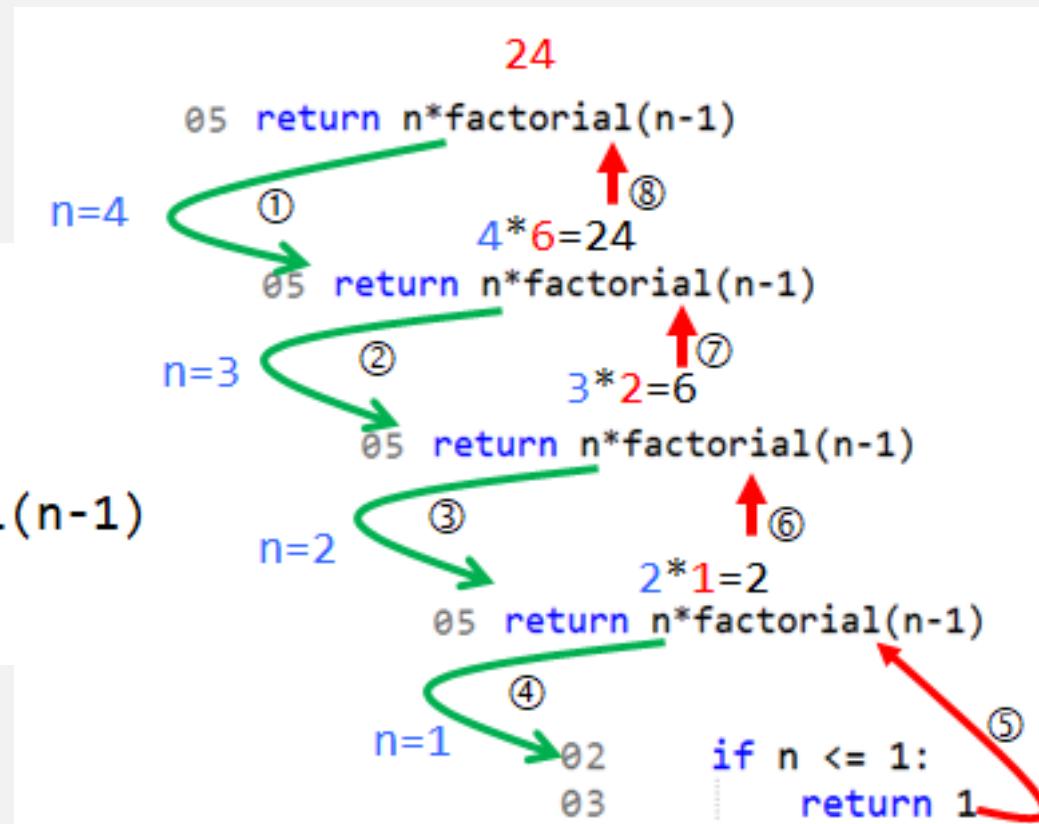
main()
```

```
>> n : 10
>> 10! = 3628800
```

# 팩토리얼 계산

- Line 07에서 factorial(4)로 함수 호출
- [그림]의 번호 순서대로 수행되어 24를 출력

```
01 def factorial(n):  
02     if n <= 1:  
03         return 1  
04     else:  
05         return n*factorial(n-1)  
06  
07 print(factorial(4))
```



# Recursive Functions

- 실습
  - n번째 피보나치(Fibonacci) 수를 구해보자.
  - 피보나치 수는 아래의 점화식으로 정의되는 수열이다.

$$F_n = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ F_{n-1} + F_{n-2}, & otherwise \end{cases}$$

# Recursive Functions

## – 예3) 피보나치(Fibonacci) 수열

- 0, 1, 1, 2, 3, 5, 8, 13, 21 .....

$$\text{fibo}(0) = 0$$

$$\text{fibo}(1) = 1$$

$$\text{fibo}(2) = \text{fibo}(1) + \text{fibo}(0) = 1$$

$$\text{fibo}(3) = \text{fibo}(2) + \text{fibo}(1) = 2$$

$$\text{fibo}(4) = \text{fibo}(3) + \text{fibo}(2) = 3$$

.....

$$\text{fibo}(n) = \text{fibo}(n-1) + \text{fibo}(n-2)$$

$$\rightarrow \text{fibo}(0) = 0$$

$$\text{fibo}(1) = 1$$

$$\text{fibo}(n) = \text{fibo}(n-1) + \text{fibo}(n-2)$$

fibonacci1.py

```
File Edit Format Run Options Window Help
1 def fibo ( n ) :
2     if n == 0 :
3         return 0
4     elif n == 1 :
5         return 1
6     else :
7         return fibo(n-1) + fibo(n-2)
8
9
10 n = int(input("=> n : "))
11 print("=> {}번째 피보나치 수 : {}".format(n, fibo(n)))
12
```

>> n : 10

>> 10번째 피보나치 수 : 55

# Recursive Functions

- 실습
  - n번째 피보나치 수와 피보나치 수열의 연산 횟수를 구해보자.

fibonacci2.py

```
File Edit Format Run Options Window Help
1 # 피보나치 수열의 연산 횟수를 구하는 함수
2 counter = 0
3
4 # 피보나치 수열을 구하는 함수
5 def fibo ( n ) :
6     global counter
7     counter += 1
8
9     if n == 0 :
10         return 0
11     elif n == 1 :
12         return 1
13     else :
14         return fibo(n-1) + fibo(n-2)
15
16 #main
17 n = int(input("=> n : "))
18 print("=> {}번 째 피보나치 수 : {}".format(n, fibo(n)))
19 print("=> 총 {}번 연산을 하였습니다.".format(counter))
20
```

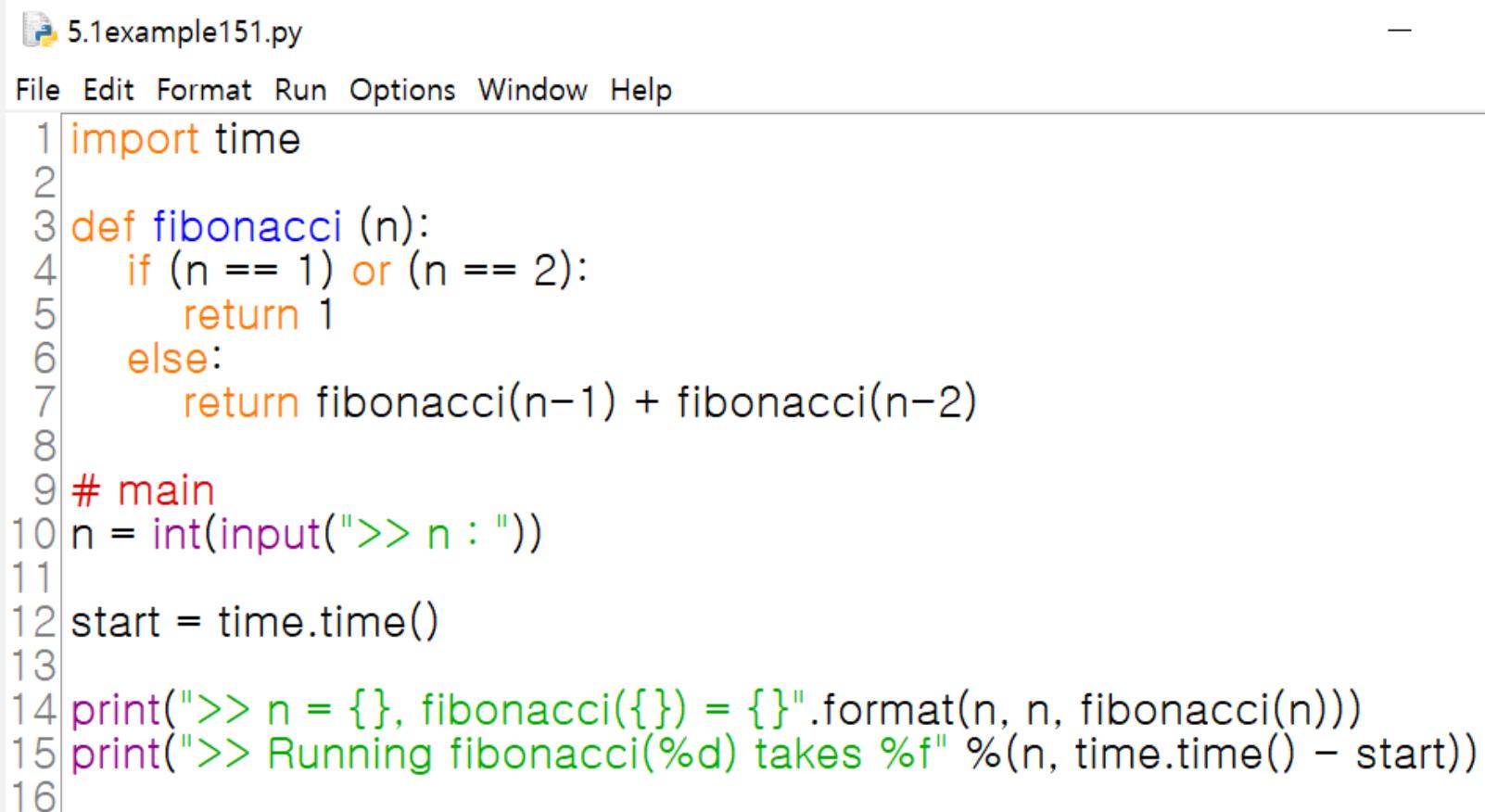
>> n : 10

>> 10번째 피보나치 수 : 55

>> 총 177번 연산을 하였습니다.

# 실습

- 피보나치 수열의 처리 시간을 구하시오



```
5.1example151.py
File Edit Format Run Options Window Help
1 import time
2
3 def fibonacci (n):
4     if (n == 1) or (n == 2):
5         return 1
6     else:
7         return fibonacci(n-1) + fibonacci(n-2)
8
9 # main
10 n = int(input("=> n : "))
11
12 start = time.time()
13
14 print("=> n = {}, fibonacci({}) = {}".format(n, n, fibonacci(n)))
15 print("=> Running fibonacci({}) takes {:.6f} s".format(n, time.time() - start))
16
```

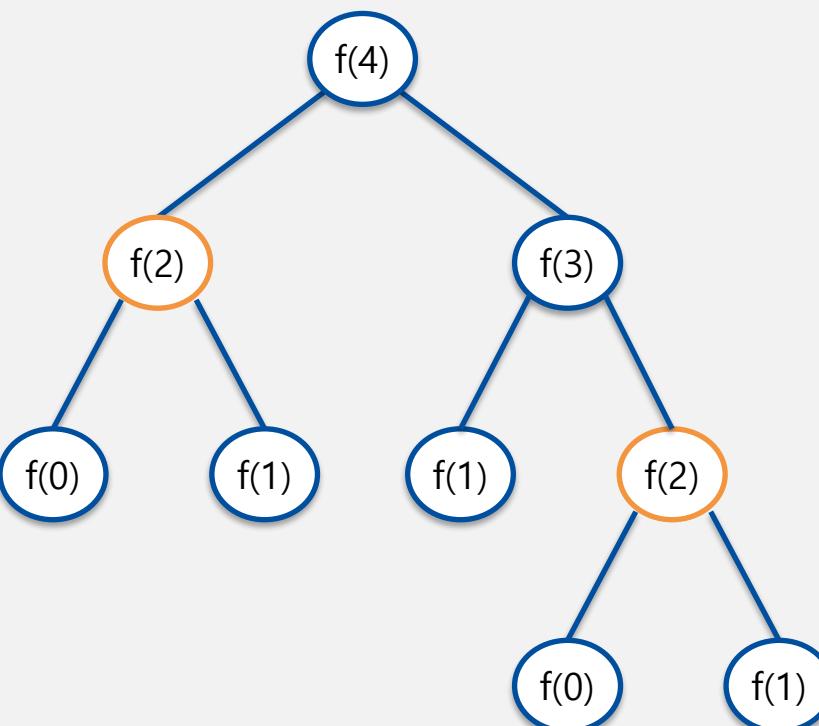
```
>> n : 20
>> n = 20, fibonacci(20) = 6765
>> Running fibonacci(20) takes 0.012928
```

## 숙제 – 파일명 : fibo-이름-학번-일시.py

- 피보나치 수열의 처리 시간이 오래 걸리는 이유는 무엇일까?
- 피보나치 수열을 재귀함수가 아닌 일반함수(for loop 사용)로 구현하고 처리 시간을 재귀함수였을 때와 비교하시오

# 실습

- 메모이제이션 (memoization)
  - 컴퓨터 program이 동일한 계산을 반복해야 할 때, 이전에 계산한 값을 memory에 저장함으로써 동일한 계산의 반복 수행을 제거하여 program 실행 속도를 빠르게 하는 기술



# 실습

- 메모이제이션 (memoization)

```
counter = 0 # 피보나치 수열의 연산 횟수
memo = {} # 피보나치 수열의 결과를 저장할 사전

# 피보나치 수열을 구하는 함수
def fibo ( n ) :
    global counter
    counter += 1
    if n in memo :
        return memo[n]
    if n == 0 :
        memo[n] = 0
    elif n == 1 :
        memo[n] = 1
    else :
        memo[n] = fibo(n-1) + fibo(n-2)
    return memo[n]

n = int(input("n >"))
print("{}번째 피보나치 수 : {}".format(n, fibo(n)))
print("총 {}번 연산을 하였습니다.".format(counter))
```

```

1 counter = 0 # 피보나치 수열의 연산 횟수
2 memo = {} # 피보나치 수열의 결과를 저장할 사전
3
4 # 피보나치 수열을 구하는 함수
5 def fibo (n) :
6
7     global counter
8
9     counter += 1
10
11    if n in memo :
12        print("f1) memo[{}] = {} 값 사용".format(n, memo[n]))
13        return memo[n]
14
15    if n == 0 :
16        memo[n] = 0
17        print("f2) memo[{}] = {} 저장".format(n, memo[n]))
18
19    elif n == 1 :
20        memo[n] = 1
21        print("f3) memo[{}] = {} 저장".format(n, memo[n]))
22
23    else :
24        memo[n] = fibo(n-1) + fibo(n-2)
25        print("f4) memo[{}] = {} 저장".format(n, memo[n]))
26
27    return memo[n]
28
29 # main
30 n = int(input("n> n = "))
31
32 print("n1) {}번째 피보나치 수 : {}".format(n, fibo(n)))
33 print("2) 총 {}번 연산을 하였습니다.".format(counter))
34 print("3) memo =", memo)
35

```

```

>> n = 10
f3) memo[1] = 1 저장
f2) memo[0] = 0 저장
f4) memo[2] = 1 저장

f1) memo[1] = 1 값 사용
f4) memo[3] = 2 저장

f1) memo[2] = 1 값 사용
f4) memo[4] = 3 저장

f1) memo[3] = 2 값 사용
f4) memo[5] = 5 저장

f1) memo[4] = 3 값 사용
f4) memo[6] = 8 저장

f1) memo[5] = 5 값 사용
f4) memo[7] = 13 저장

f1) memo[6] = 8 값 사용
f4) memo[8] = 21 저장

f1) memo[7] = 13 값 사용
f4) memo[9] = 34 저장

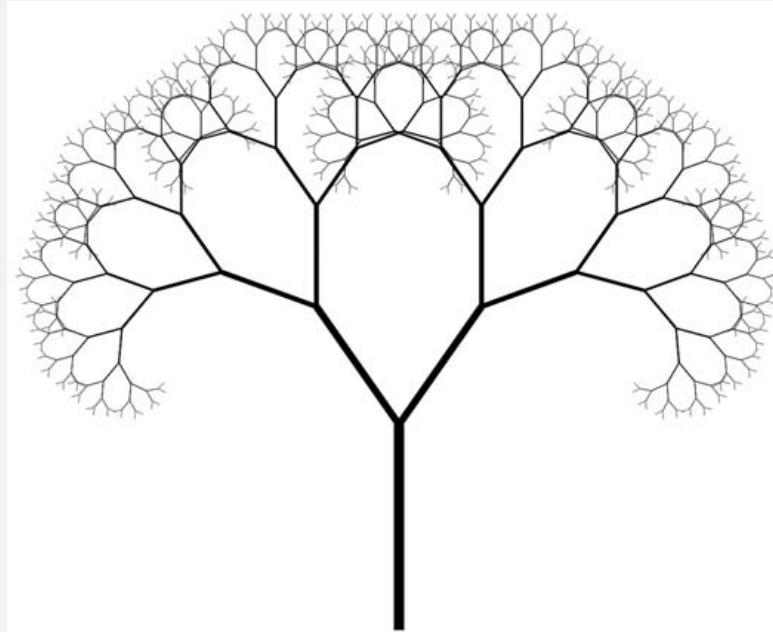
f1) memo[8] = 21 값 사용
f4) memo[10] = 55 저장

1) 10번째 피보나치 수 : 55
2) 총 19번 연산을 하였습니다.
3) memo = {1: 1, 0: 0, 2: 1, 3: 2, 4: 3, 5: 5, 6: 8, 7: 13, 8: 21, 9: 34, 10: 55}
```

```

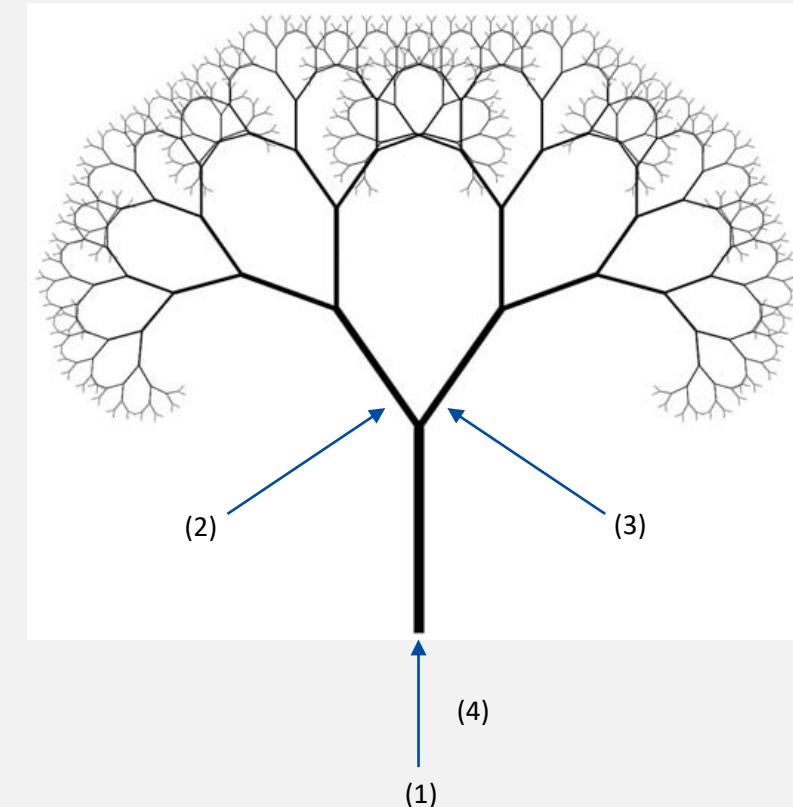
# 숙제 - 파일명 : tree-이름-학번-일시.py

- 재귀적 나무 그리기
  - 터틀 그래픽을 이용해서 다음 그림과 같은 재귀적 구조의 나무를 그리시오



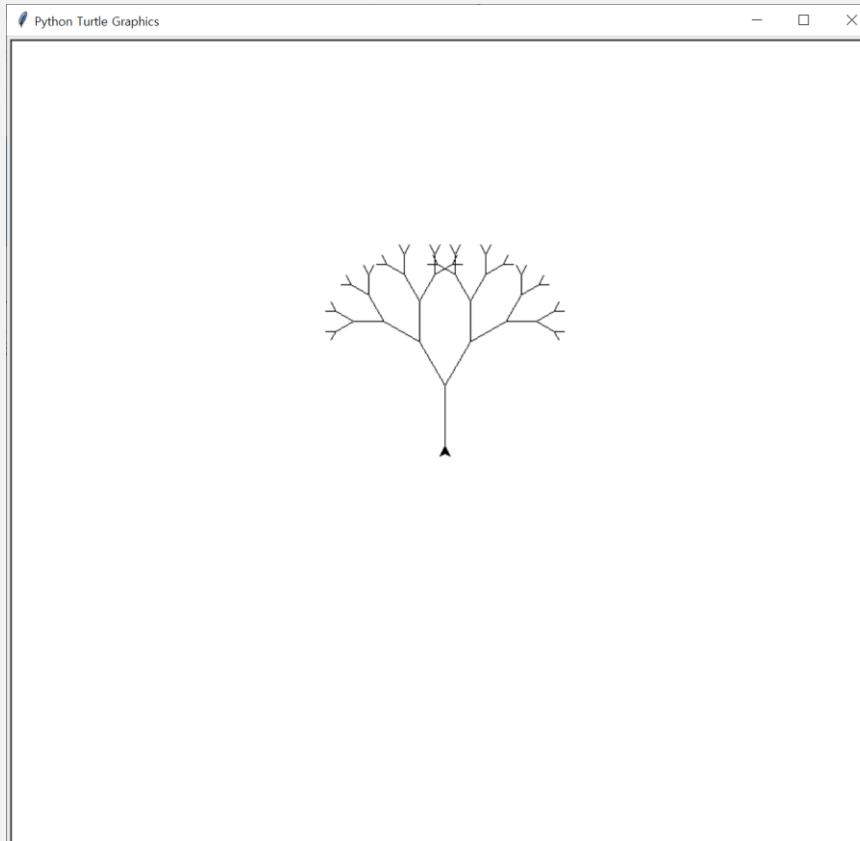
# 숙제

- 재귀적 나무 그리기
  - 재귀적 구조 찾기
    - 나무는 가운데 직선, 왼쪽 나무, 오른쪽 나무로 구성된다!
      - (1) 시작점에서 직선을 그린다
      - (2) 왼쪽 나무를 그린다
      - (3) 오른쪽 나무를 그린다
      - (4) 시작점으로 돌아온다



# 숙제

- 재귀적 나무 그리기
  - 재귀적 구조 찾기



5.1example152.py

File Edit Format Run Options Window Help

```
1 import turtle
2
3 s = turtle.Screen()
4 t = turtle.Turtle()
5
6 angle = 30
7
8 def drawTree(t, lineLength):
9
10     if (lineLength >0):
11
12         t.forward(lineLength)
13         t.left(angle)
14         drawTree(t, lineLength-10)
15
16         t.right(angle)
17         t.right(angle)
18         drawTree(t, lineLength-10)
19
20         t.left(angle)
21         t.backward(lineLength)
22
23
24 if __name__ == "__main__":
25     lineLength=60
26     t.left(90)
27     drawTree(t, lineLength)
28
```

# 내장 함수(built-in function)

- Python에서 기본적으로 제공하는 함수 <https://docs.python.org/3/library/functions.html>

| Built-in Functions   |  |   |   |
|--|--|---|---|
| A  | E  | L                                       | R   |
| abs()<br>aiter()<br>all()<br>any()<br>anext()<br>ascii()       | enumerate()<br>eval()<br>exec()                                    | len()<br>list()<br>locals()             | range()<br>repr()<br>reversed()<br>round()  |
| B  | F  | M                                       | S   |
| bin()<br>bool()<br>breakpoint()<br>bytearray()<br>bytes()      | filter()<br>float()<br>format()<br>frozenset()                     | map()<br>max()<br>memoryview()<br>min() | set()<br>setattr()<br>slice()<br>sorted()   |
| C  | G  | N                                       | staticmethod()<br>str()<br>sum()<br>super() |
| callable()<br>chr()<br>classmethod()<br>compile()<br>complex() | getattr()<br>globals()   | next()                                  | tuple()<br>type()                           |
| D  | H  | O                                       | T   |
| delattr()<br>dict()<br>dir()<br>divmod()                       | hasattr()<br>hash()<br>help()<br>hex()                             | object()<br>oct()<br>open()<br>ord()    | vars()                                      |
| I  | P  | V                                       | Z   |
|  | id()<br>input()<br>int()<br>isinstance()<br>issubclass()<br>iter() | pow()<br>print()<br>property()          | zip()                                       |
|  |  | -                                       | __import__()                                |

# 내장 함수(built-in function)

- `abs(x)`
  - 절대값을 반환 하는 함수
  - 복소수라면 제곱을 한 다음, 루트를 한 값을 리턴 한다.

```
# abs 는 절대 값을 반환 한다.  
print(abs(-3))  
# 복소수  $x + yj$  인 경우  $\sqrt{x^2 + y^2}$ 의 값을 반환 한다.  
print(abs(4 + 3j))
```

# 내장 함수(built-in function)

- max()
  - 인자 값 중 최대값을 반환 한다.
- min()
  - 인자 값 중 최소값을 반환 한다.

```
numbers = [91, 7, 1, 18, 29, 66, 89, 41, 96, 32]
```

```
print("MAX :", max(numbers))  
print("MIN :", min(numbers))
```

# 내장 함수(built-in function)

- float()
  - 문자열 혹은 정수를 실수로 바꾼다.

```
str_pi = "3.141592653589793"  
str_e = "2.718281828459045"
```

```
print(str_pi + str_e)
```

```
float_pi = float(str_pi)  
float_e = float(str_e)
```

```
print(float_pi + float_e)
```

# 내장 함수(built-in function)

- int()
  - 문자열 혹은 실수를 정수로 바꾼다.

```
str_han_birthday = "970203"
str_heo_birthday = "960913"

print(str_han_birthday + str_heo_birthday)

int_han_birthday = int(str_han_birthday)
int_heo_birthday = int(str_heo_birthday)

print(int_han_birthday + int_heo_birthday)
```

# 내장 함수(built-in function)

- enumerate(iterable, start = 0)
  - 시퀀스 객체를 입력 받아, enumerate 객체로 반환한다.
  - enumerate 객체는 (번호, 값) 들로 구성.

```
enumerate(iterable, start=0)
```

Return an `enumerate` object. `iterable` must be a sequence, an `iterator`, or some other object which supports iteration. The `__next__()` method of the iterator returned by `enumerate()` returns a tuple containing a count (from `start` which defaults to 0) and the values obtained from iterating over `iterable`.

```
>>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']
>>> list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
>>> list(enumerate(seasons, start=1))
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
```

Equivalent to:

```
def enumerate(sequence, start=0):
    n = start
    for elem in sequence:
        yield n, elem
        n += 1
```

# 내장 함수(built-in function)

- enumerate(iterable, start = 0)
  - 시퀀스 객체를 입력 받아, enumerate 객체로 반환한다.
  - enumerate 객체는 (번호, 값) 들로 구성.

enumerate.py

```
File Edit Format Run Options Window Help
1 actors = [ "Jack Nicholson", "Morgan Freeman", "Robert De Niro",
2           "Al Pacino", "Leonardo DiCaprio", "Tom Hanks", "Russell Crowe" ]
3
4 print("Wn1) list actors = ", actors)
5
6 enu_type = enumerate(actors)
7 print("Wn2) enu_type =", enu_type)
8
9 list_type = list(enu_type)
10 print("Wn3) list_type =", list_type)
11
12 for number, name in enumerate(actors, start = 1):
13     print(">> {}번째 Actor : {}".format(number, name))
```

1) list actors = ['Jack Nicholson', 'Morgan Freeman', 'Robert De Niro', 'Al Pacino', 'Leonardo DiCaprio', 'Tom Hanks', 'Russell Crowe']  
2) enu\_type = <enumerate object at 0x0000021F6E807140>  
3) list\_type = [(0, 'Jack Nicholson'), (1, 'Morgan Freeman'), (2, 'Robert De Niro'), (3, 'Al Pacino'), (4, 'Leonardo DiCaprio'), (5, 'Tom Hanks'), (6, 'Russell Crowe')]  
>> 1번째 Actor : Jack Nicholson  
>> 2번째 Actor : Morgan Freeman  
>> 3번째 Actor : Robert De Niro  
>> 4번째 Actor : Al Pacino  
>> 5번째 Actor : Leonardo DiCaprio  
>> 6번째 Actor : Tom Hanks  
>> 7번째 Actor : Russell Crowe

# 내장 함수(built-in function)

- sum()
  - 리스트 혹은 tuple의 합을 반환하는 함수

```
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
print(sum(number_list))
```

# 내장 함수(built-in function)

- sorted(iterable, [key], [reverse])
  - iterable 객체 안에 들어 있는 항목들로부터 정렬된 리스트를 생성하여 반환
  - key : 정렬의 기준이 되는 값
  - reverse : 정렬 결과를 뒤집을지 결정

```
number_list = [91, 7, 1, 18, 29, 66, 89, 41, 96, 32]
```

```
sorted_number_list = sorted(number_list)
```

```
print("number_list :", number_list)
```

```
print("sorted_number_list :", sorted_number_list)
```

# 내장 함수(built-in function)

- sorted(iterable, [key], [reverse])

```
student = [  
    ("Park", 20503253, 4.2),  
    ("Lee", 20503180, 3.7),  
    ("Song", 20503250, 4.5) ]  
  
print("Before sorted :", student)  
sort_by_id = sorted(student, key = lambda x : x[0])  
print("Sort by id :", sort_by_id)  
sort_by_grade = sorted(student, key = lambda x : x[2], reverse=True)  
print("Sort by grade :", sort_by_grade)
```

student에 있는 각 원소의 첫번째 값을 기준으로 정렬

# 내장 함수(built-in function)

- sorted(iterable, [key], [reverse])

```
# 사전을 정의 합니다.  
student = {  
    20503180 : 3.7,  
    20503250 : 4.5,  
    20503253 : 4.2  
}
```

```
# 학점을 기준으로 정렬해서 출력 합니다.  
for key, value in sorted(student.items(), key=lambda x:x[1], reverse=True) :  
    print(key, ":", value)
```

# 내장 함수(built-in function)

- sorted vs sort
- sorted(iterable, [key], [reverse])
  - Python 내장 함수
  - 입력값을 정렬하고 결과를 리스트로 리턴함
- list.sort()
  - 리스트 자료형의 함수
  - 리스트 자체를 정렬함. 결과 리턴 없음.

# 내장 함수

- **list.sort()**

- 리스트 자료형의 함수

- 예) **key=str.upper** 사용으로  
대소문자 구분 없이 정렬

```
5.1example18.py
File Edit Format Run Options Window Help
1 str_data = "The School of Computer Science in Kookmin University is the best college in Korea.".split()
2
3
4 print("1) str_data :", str_data)
5
6 str_data.sort()
7 print("2) sort 후, str_data : ", str_data)
8
9 str_data.sort(key=str.upper)
10 print("3) upper sort 후, str_data : ", str_data)
11
12 str_data.sort(key=str.upper, reverse=True)
13 print("4) upper reverse sort 후, str_data = ", str_data)
```

1) str\_data : ['The', 'School', 'of', 'Computer', 'Science', 'in', 'Kookmin', 'University', 'is', 'the', 'best', 'college', 'in', 'Korea.']}

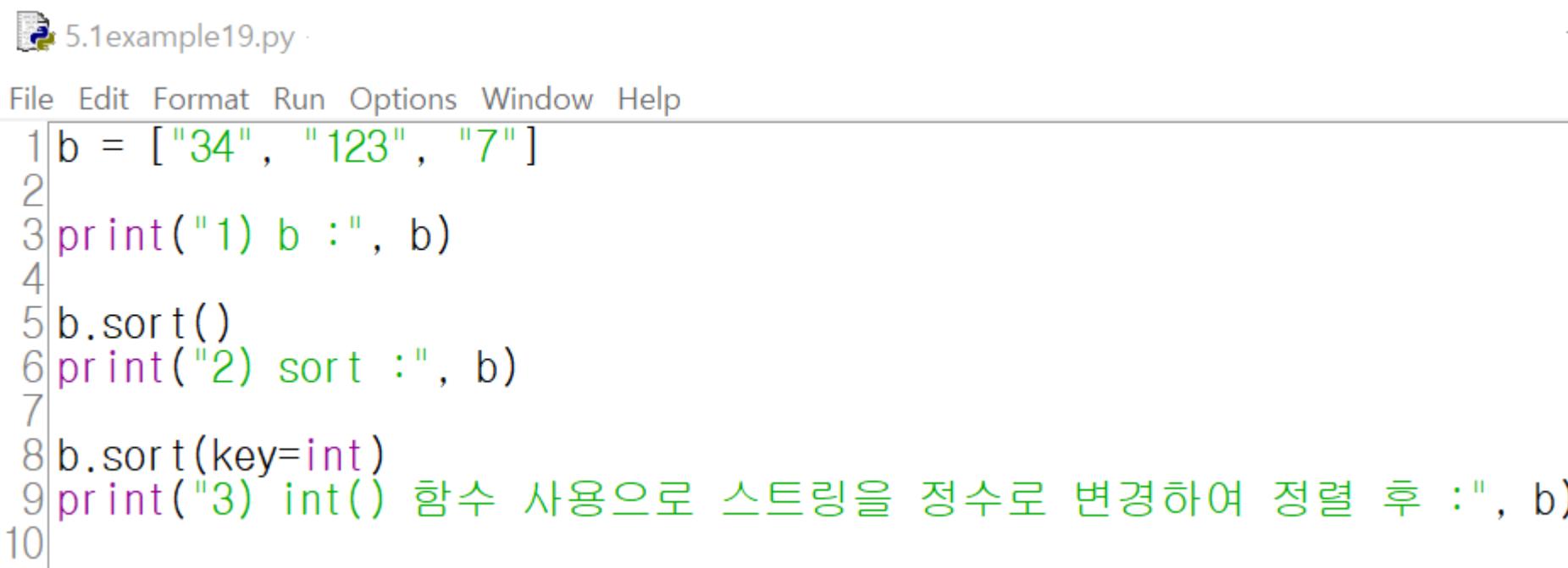
2) sort 후, str\_data : ['Computer', 'Kookmin', 'Korea.', 'School', 'Science', 'The', 'University', 'best', 'college', 'in', 'in', 'is', 'of', 'the']

3) upper sort 후, str\_data : ['best', 'college', 'Computer', 'in', 'in', 'is', 'Kookmin', 'Korea.', 'of', 'School', 'Science', 'The', 'the', 'University']

4) upper reverse sort 후, str\_data = ['University', 'The', 'the', 'Science', 'School', 'of', 'Korea.', 'Kookmin', 'is', 'in', 'in', 'Computer', 'college', 'best']

# 내장 함수(built-in function)

- list.sort()
    - 리스트 자료형의 함수
- 예) int() 함수 사용으로 스트링을 정수로 변경하여 정렬



```
5.1example19.py
File Edit Format Run Options Window Help
1 b = ["34", "123", "7"]
2
3 print("1) b :", b)
4
5 b.sort()
6 print("2) sort :", b)
7
8 b.sort(key=int)
9 print("3) int() 함수 사용으로 스트링을 정수로 변경하여 정렬 후 :", b)
10
```

```
1) b : ['34', '123', '7']
2) sort : ['123', '34', '7']
3) int() 함수 사용으로 스트링을 정수로 변경하여 정렬 후 : ['7', '34', '123']
```

# main

- 참고) [https://hashcode.co.kr/questions/3/if\\_\\_name\\_\\_-main\\_%EC%9D%80-%EC%99%9C%EC%93%B0%EB%82%98%EC%9A%94](https://hashcode.co.kr/questions/3/if__name__-main_%EC%9D%80-%EC%99%9C%EC%93%B0%EB%82%98%EC%9A%94) (2021/4/27 현재 )
- Script가 Python interpreter 명령어로 passing되어 실행되면( a.py같이) 다른 언어들과는 다르게 Python은 자동으로 실행되는 main 함수가 없음
- Python은 main 함수가 없는 대신 들여쓰기 하지 않은 모든 코드 (level 0 코드)를 실행
- 다만, 함수나 클래스는 정의되었지만, 실행되지는 않음
- \_\_name\_\_은 현재 모듈의 이름을 담고있는 내장 변수임
- a.py 같이 이 모듈이 직접 실행되는 경우에만, \_\_name\_\_ 은 “\_\_main\_\_”으로 설정됨



File Edit Format Run Options Window Help

```
1 def func():
2     print("5) 여기는 function func() in a.py")
3
4 print("6) top-level A.py")
5
6 if __name__ == "__main__":
7     print("7) a.py 직접 실행")
8 else:
9     print("8) a.py가 임포트되어 사용됨")
```

6) top-level A.py  
7) a.py 직접 실행

a.py를 실행 시



File Edit Format Run Options Window Help

```
1 import a
2
3 print("1) top-level in B.py")
4 print("2) a.py에 있는 a.func() 호출 ")
5 a.func()
6
7 if __name__ == "__main__":
8     print("3) b.py가 직접 실행")
9 else:
10    print("4) b.py가 임포트되어 사용됨")
```

6) top-level A.py  
8) a.py가 임포트되어 사용됨  
1) top-level in B.py  
2) a.py에 있는 a.func() 호출  
5) 여기는 function func() in a.py  
3) b.py가 직접 실행

b.py를 실행 시

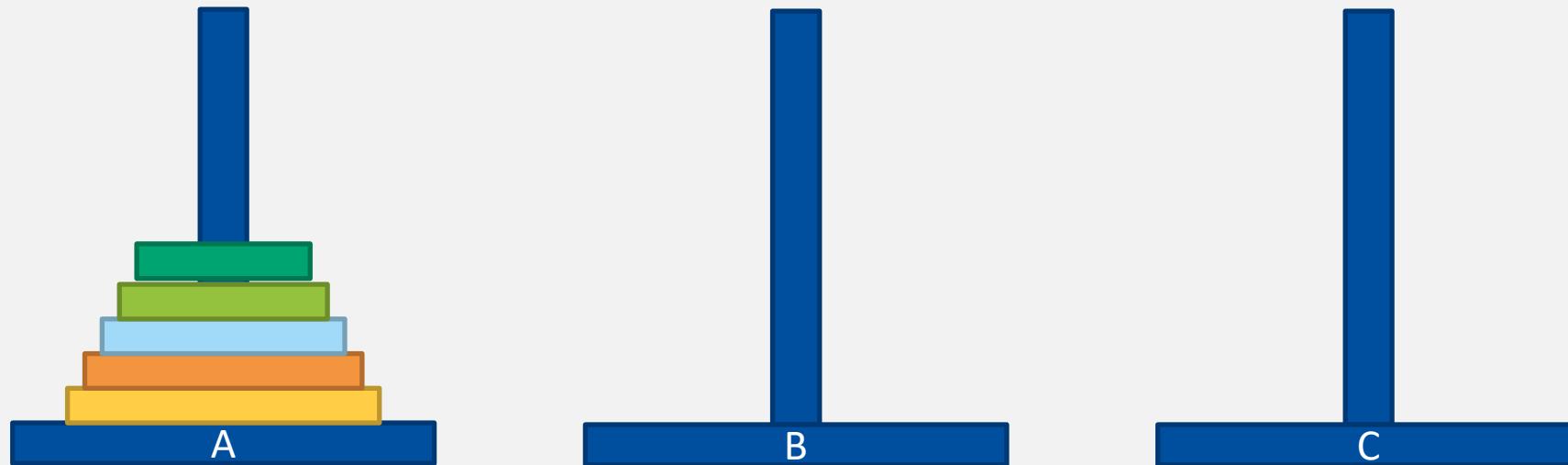
# 숙제 - 파일명 : hanoi-이름-학번-일시.py

- 하노이탑
  - 하노이탑을 재귀함수로 구현하시오.
  - 원반의 개수가  $n$ 개일 때, 몇 번 원반을 옮겨야 하는가?
  - 원반의 개수가  $n$ 개일 때, 어떻게 원반을 옮겨야 하는가?

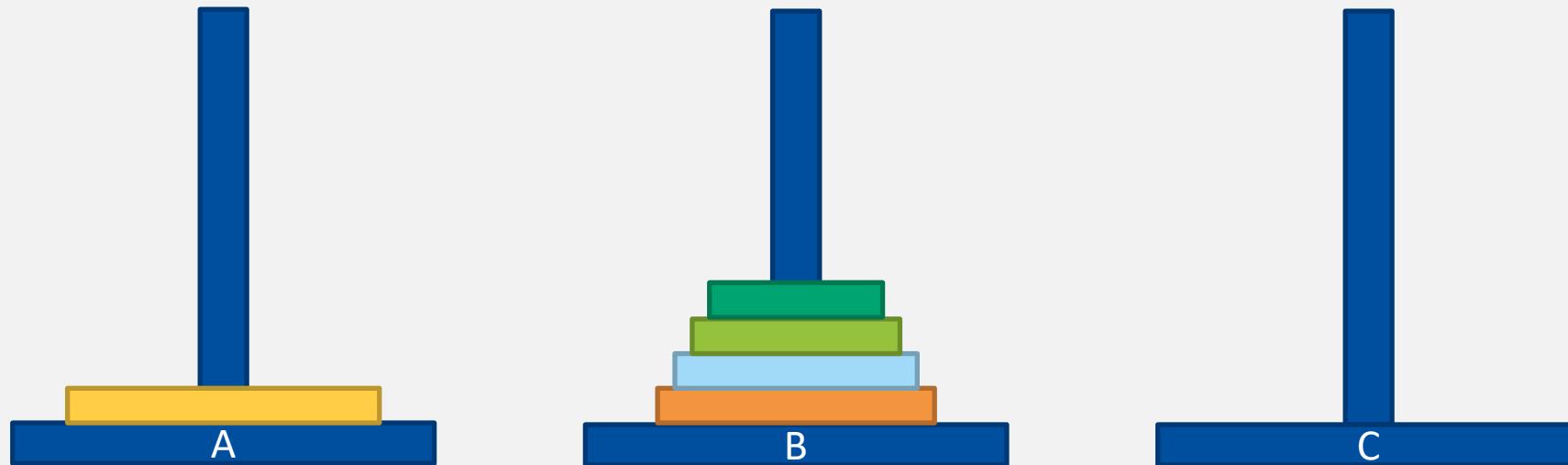
# 숙제

- 3 개의 장대가 있고 첫 번째 장대에는 반경이 서로 다른  $n$ 개의 원판이 쌓여 있다. 각 원판은 반경이 큰 순서대로 쌓여 있다. 이제 수도승들이 다음 규칙에 따라 첫 번째 장대에서 세 번째 장대로 옮기려 한다. 이 작업을 수행하는데 필요한 이동순서를 출력하는 program을 작성하라
1. 한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.
  2. 쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.(중간 과정 역시 그래야함)

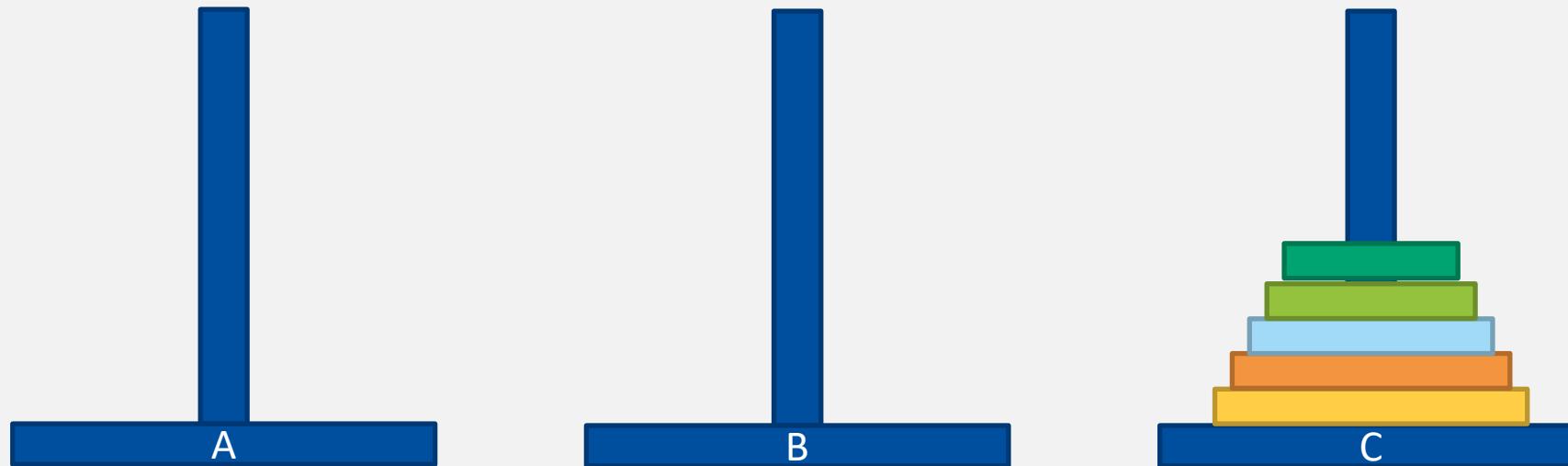
# 숙제



# 숙제



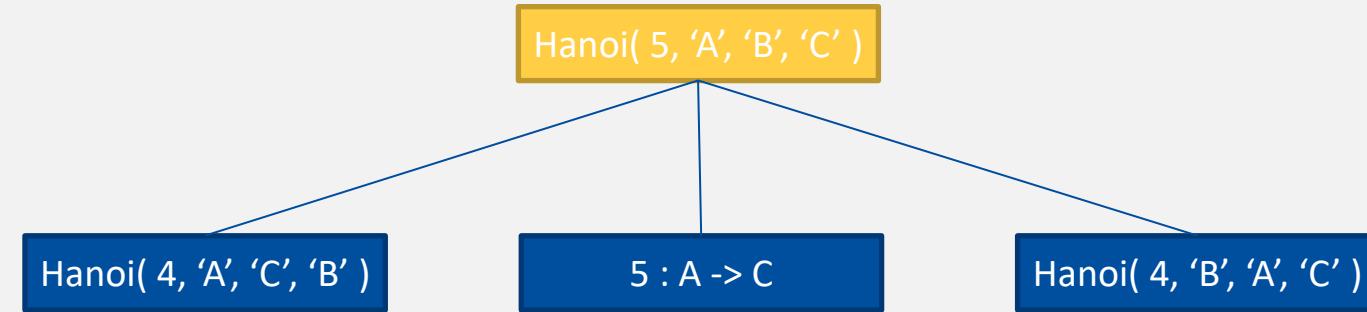
# 숙제



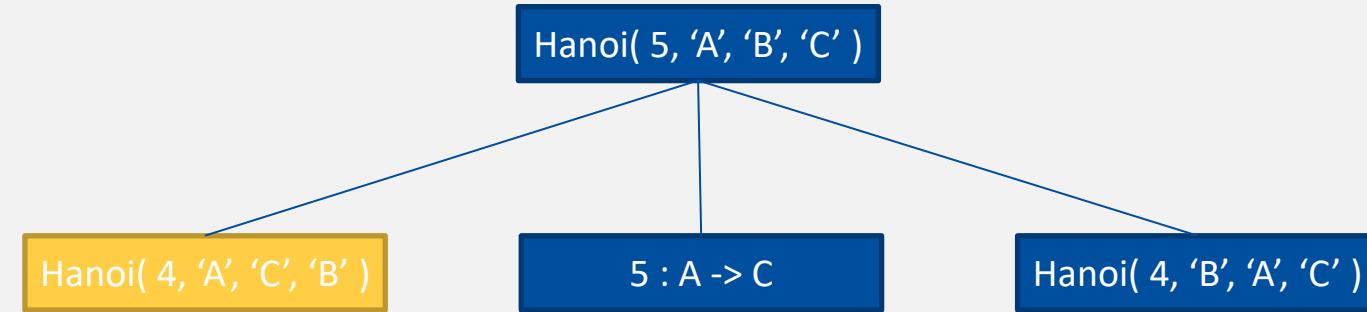
# 숙제

- 5 개의 원판을 A->C로 옮기는 방법
  1. 초록색 ~ 주황색 원판을 A -> B 로 옮긴다.
  2. 노란색 원판을 A -> C 로 옮긴다.
  3. 다시 초록색 ~ 주황색 원판을 B -> C 로 옮긴다.
- 4개의 원판을 A->B로 옮기는 방법은?
- 4개의 원판을 B->C로 옮기는 방법은?

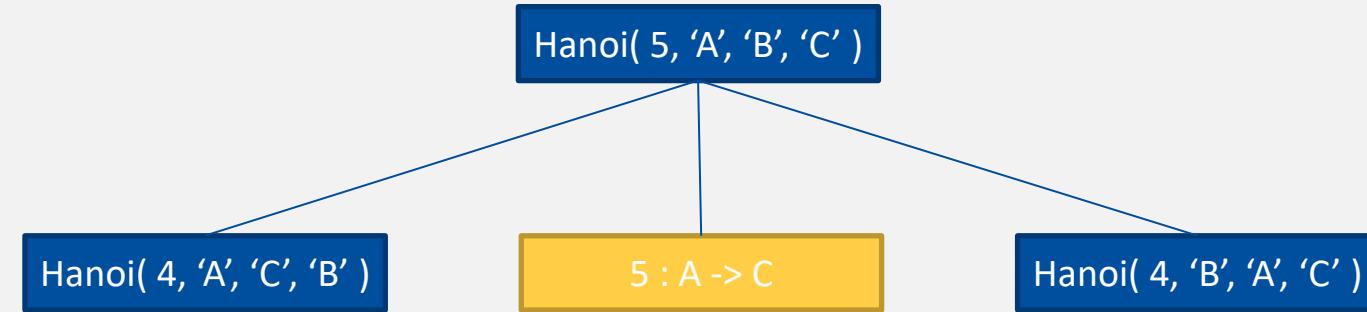
# 숙제 – Hanoi ( n, s, m, e )



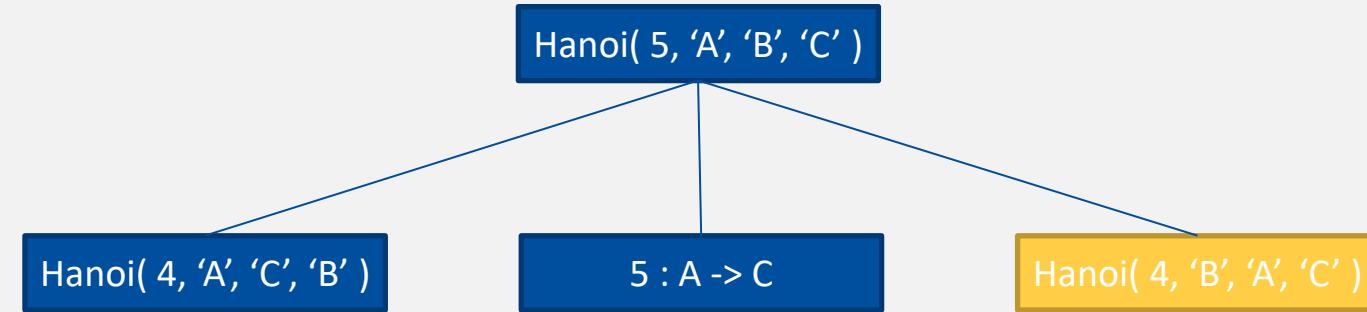
# 숙제 – Hanoi ( n, s, m, e )



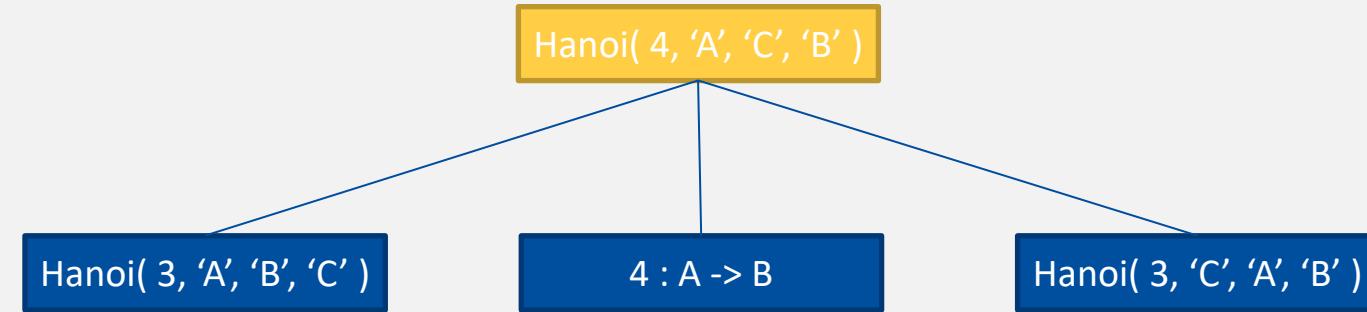
# 숙제 – Hanoi ( n, s, m, e )



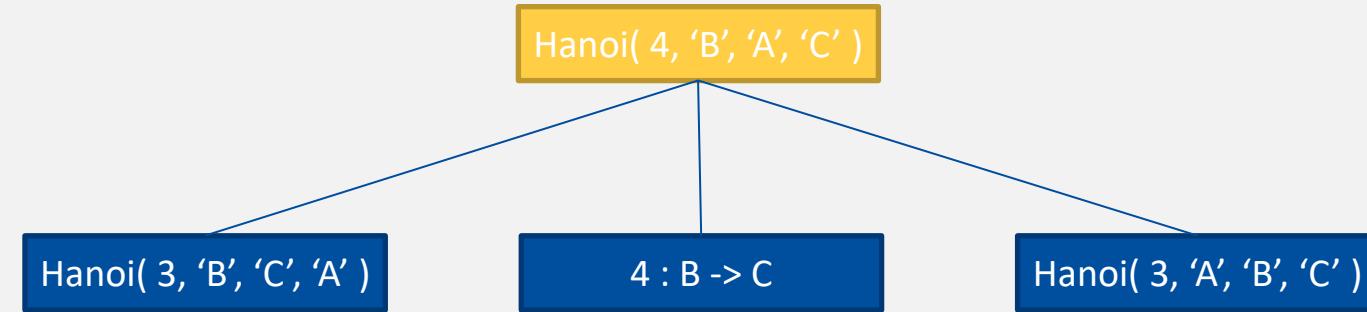
# 숙제 – Hanoi ( n, s, m, e )



# 숙제 – Hanoi ( n, s, m, e )



# 숙제 – Hanoi ( n, s, m, e )



# 숙제 – Hanoi ( n, s, m, e )

The screenshot shows a Python development environment with two windows. The left window is a code editor for `hanoi.py`, and the right window is a terminal window for Python 3.6.1.

**Code Editor (hanoi.py):**

```
hanoi.py - C:/Users/corea/Documents/hanoi.py (3.6.1)
File Edit Format Run Options Window Help
def hanoi( n, start, mid, end ) :
    # 원판이 1개인 경우 바로 옮길 수 있다.
    if n == 1 :
        print("%d : %c -> %c" %(n, start, end))
    else :
        # n-1개의 원판을 start에서 mid로 먼저 옮겨 준다.
        hanoi(n-1, start, end, mid)
        # 가장 아래에 있던 n번째 원판을 start에서 end로 옮겨준다.
        print("%d : %c -> %c" %(n, start, end))
        # mid에 있는 n-1개의 원판을 end로 옮겨준다.
        hanoi(n-1, mid, start, end)

if __name__ == '__main__':
    n = int(input("N : "))
    hanoi(n, 'A', 'B', 'C')
```

**Terminal (Python 3.6.1):**

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/corea/Documents/hanoi.py =====
5
1 : A -> C
2 : A -> B
1 : C -> B
3 : A -> C
1 : B -> A
2 : B -> C
1 : A -> C
4 : A -> B
1 : C -> B
2 : C -> A
1 : B -> A
3 : C -> B
1 : A -> C
2 : A -> B
1 : C -> B
5 : A -> C
1 : B -> A
2 : B -> C
1 : A -> C
3 : B -> A
1 : C -> B
2 : C -> A
1 : B -> A
4 : B -> C
1 : A -> C
2 : A -> B
1 : C -> B
3 : A -> C
1 : B -> A
2 : B -> C
1 : A -> C
>>>
```

## 숙제- 파일명 : day-이름-학번-일시.py

- 주어진 날짜로부터 x일 후의 날이 몇 일이고 무슨 요일인지 계산해주는 program을 작성하시오.
  - 예: 2023.10.19 의 1000일 후?

# Homework

- 아래의 프로그램을 다음 수업 시작 전까지 ecampus로 upload
- 파일명

fibo-이름-학번.py

tree-이름-학번.py

hanoi-이름-학번.py

day-이름-학번.py

\* 파일이 여러 개일 경우 zip으로 묶어서 제출