스마트 모빌리티 프로그래밍

Ch 5. 함수 (Function)

영남대학교 정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- 파이썬 함수 개요
- 코드블록, 네임스페이스 (name space)와 유효범위 (scope)
- 파이썬 프로그램의 변수 (variable)
- 함수의 인수 (argument) 전달 형식
- 함수에서 인수로 전달된 값의 변경
- 파이썬 내장 함수 (embedded function)
- 람다함수 (lambda function)
- 1급함수 (first class function)
- 재귀함수 (recursive function)
- 제네레이터 함수 (generator function)
- 사용자 정의 함수 설계 및 구현

파이썬 함수 개요

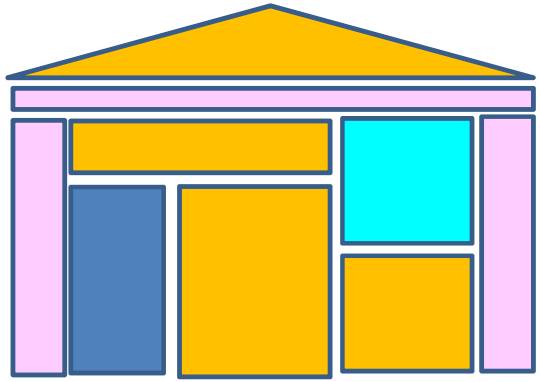
모듈화 프로그래밍의 개념

◆ 모듈(module)
- 독립되어 있는 프로그램의 일부분

◆ 모듈화 프로그래밍
- 모듈 개념을 사용하는 프로그래밍 기법
- 프로그램에 포함되는 기능들을 모듈 별로 나누어 설계 및 구현
- 일부 모듈은 기존에 이미 개발되어 있는 모듈이나 라이브러리를 사용

◆ 모듈화 프로그래밍의 장점
- 각 모듈들은 독자적으로 전문회사에 의해 개발 가능
- 다른 모듈과 독립적으로 변경 가능
- 유지 보수가 쉬워진다.
- 모듈의 재사용 가능

규모가 큰 컴퓨터 프로그램은 다수의 모듈(컴포넌트)로 구성되며, 각 모듈은 전문가 및 전문회사에서 개발됩니다.
컴퓨터 프로그램의 모듈들은 다른 프로그램에도 재사용될 수 있습니다.



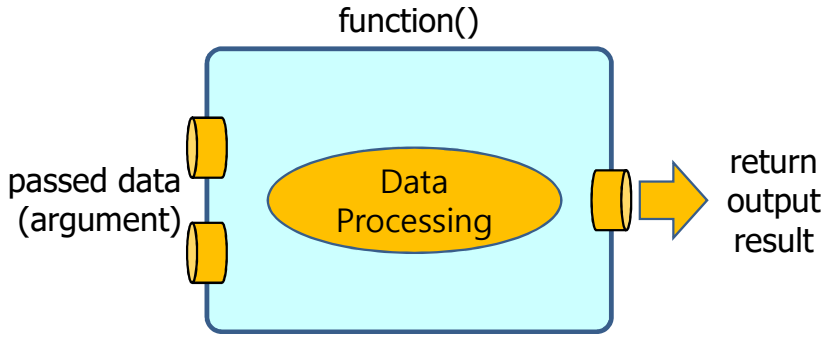
함수의 개념

◆ 함수(function)

- 특정한 작업을 수행하는 독립적인 모듈
- 함수는 입력을 받으며 출력을 생성한다.

◆ 함수 호출(function call)

- 함수를 호출하여 사용하는 것



함수 사용의 장점과 단점

◆ 함수 사용의 장점

- 함수를 사용하면 코드가 중복되는 것을 막을 수 있다.
- 한번 작성된 함수 (모듈)는 여러 번 재사용할 수 있다.
- 함수를 사용하면 전체 프로그램을 모듈로 나눌 수 있어서 개발 과정이 쉬워지고 보다 체계적이 되면서 유지보수도 쉬워진다.

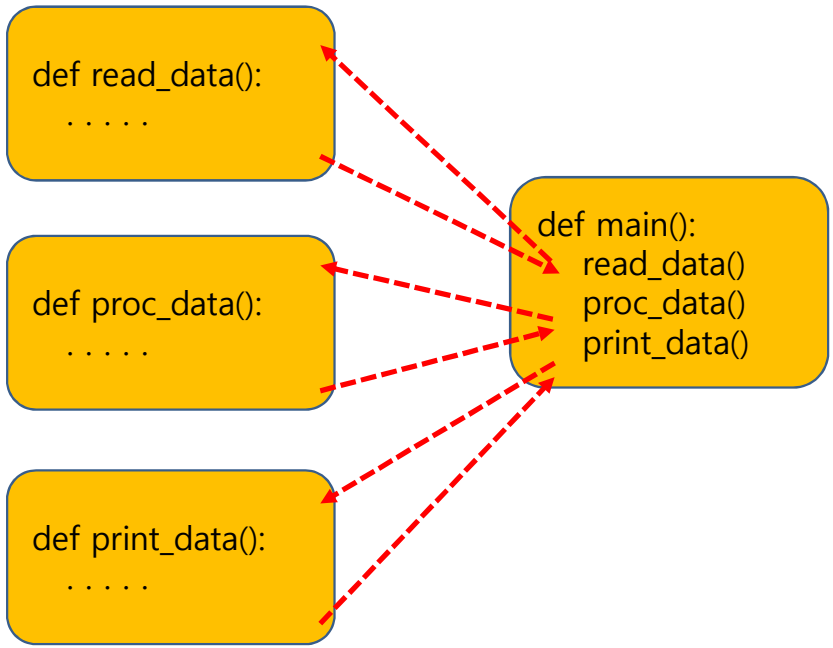
◆ 함수 사용의 단점

- 함수 호출을 할 때 마다 운영체제가 함수에서 사용되는 지역 변수들의 준비와 인수 (argument) 전달 등을 처리해야 하므로 부담이 발생한다.
- 따라서 너무 작은 단위의 함수를 구성하여 자주 호출하는 경우 성능에 문제가 발생할 수도 있다.

함수들의 연결

◆ 함수들의 연결

- 프로그램은 여러 개의 함수들로 이루어진다.
- 함수 호출을 통하여 서로 서로 연결된다.
- 함수들의 실행을 종합적으로 제어 및 관리하는 함수가 main()이다.



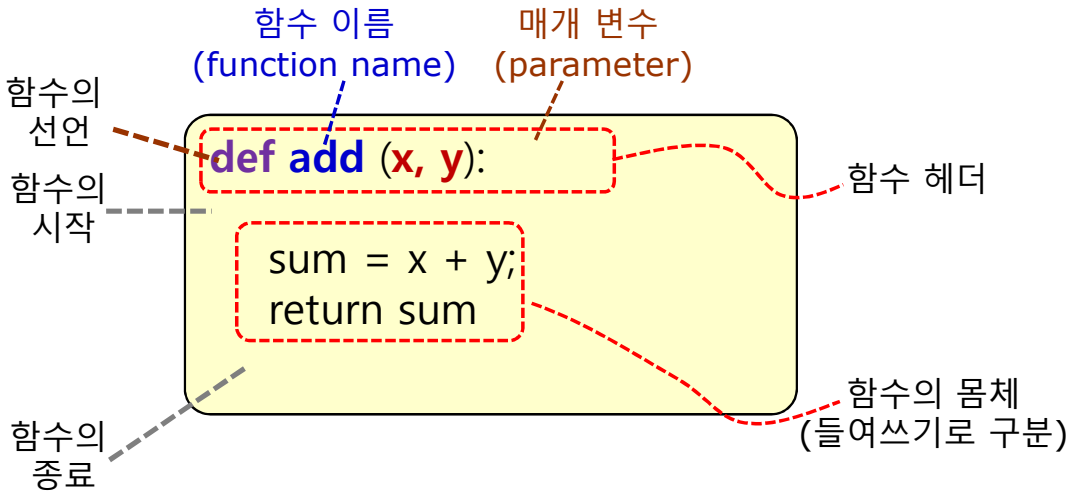
함수의 종류

◆ 사용자 정의함수 vs. Library 함수

함수 (function)	개발 및 제공	예
라이브러리 함수 (library function)	• 프로그래밍언어에서 기본적인 함수로 제공	• input() • print() • time() • rand()
사용자 정의 함수 (user defined function)	• 사용자가 직접 설계 및 구현 • 필요에 따라 필요한 함수를 구현	• mtrx_add() • mtrx_subtract() • mtrx_multiply()

함수의 구조

◆ 함수의 구조


함수 이름
(function name)
매개 변수
(parameter)
함수의
선언
함수의
시작
def add (x, y):
sum = x + y;
return sum
함수 헤더
함수의 몸체
(들여쓰기로 구분)
함수의
종료

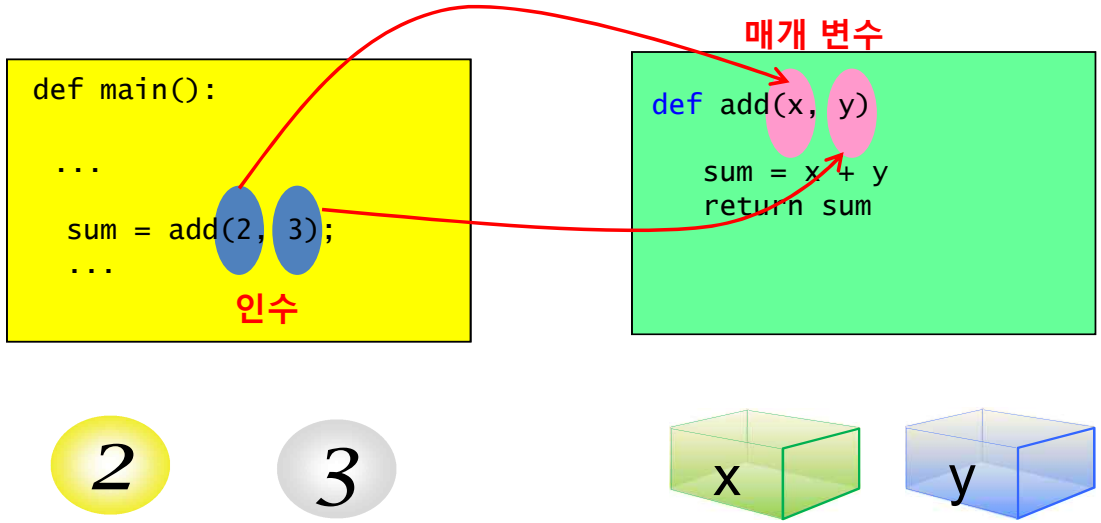
매개 변수 (parameter), 인수 (argument)

◆ 매개 변수(parameter)

- 형식 인수, 형식 매개 변수라고도 함
- 함수 원형 (function prototype) 선언에서 형식 인수 (형식 매개변수) 데이터 유형 지정

◆ 인수(argument)

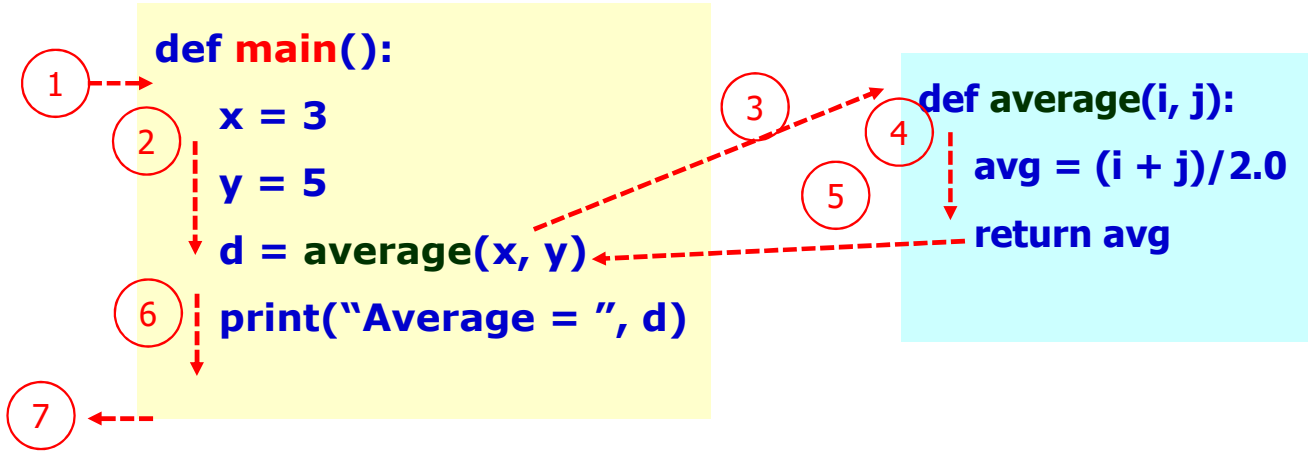
- 실인수, 실매개 변수라고도 함
- 프로그램 실행단계에서 실제 함수 호출에 전달되는 데이터



함수 호출과 결과값의 반환

◆ 함수 호출(function call)

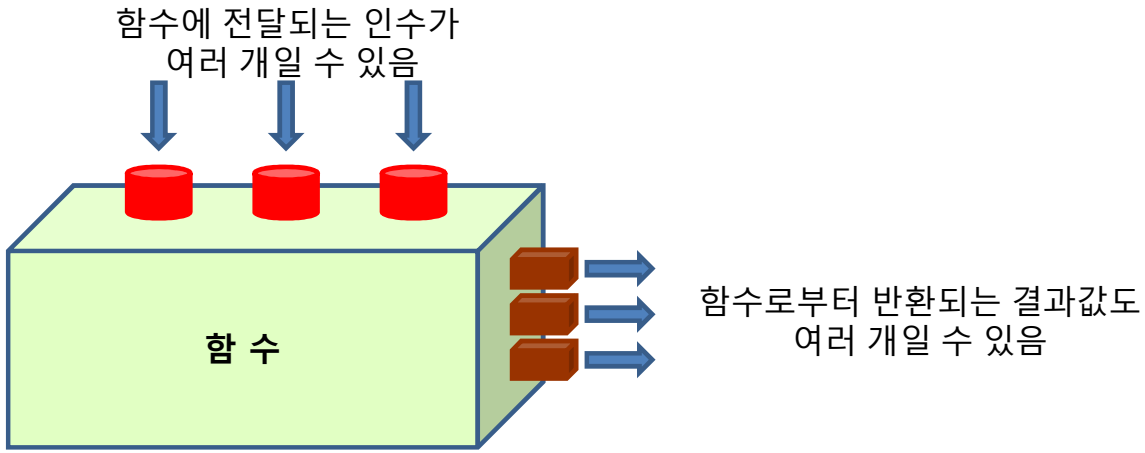
- 함수를 사용하기 위하여 함수의 이름을 적어주는 것
- 함수안의 문장들이 순차적으로 실행된다.
- 문장의 실행이 끝나면 호출한 위치로 되돌아 간다.
- 결과값을 전달할 수 있다.



함수 실행 결과의 반환 (return)

◆ 반환 값(return value)

- 호출된 함수가 호출한 곳으로 작업의 결과값을 전달하는 것
- 반환 값을 여러 개 보낼 수 있음




```
return 0
return x
return x, y, z
```

파이썬 함수

◆ 파이썬 함수의 기본 구조

```
def function_name(<parameter list>):
      "description of function"
       statement_1
       statement_2
       sub_block_1:
            sub_block_statement_1_1
            sub_block_statement_1_2
       statement_3
       return ret_val

def add(x, y):
    "add x and y"
    sum = x + y
    return sum

def multiply(x, y):
    "multiply x with y"
    result = x * y
    return result
```

```
# Example usage of function

def printList(L):
    "print list L"
    L_len =  len(L)
    for i in range(L_len):
        print("{:3}".format(L[i]), end=' ')
    print()

# main()
L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print("list L = ", end=' ')
printList(L)
```

```
list L =   0   1   2   3   4   5   6   7   8   9
```

파이썬 함수 사용의 예 – find_min_max()

◆ Function find_min_max(List)

```
# Function call with list arguments and return tuples

def find_min_max(L):
    nMin = nMax = L[0]
    for e in L:
        if nMin > e:
            nMin = e
        if nMax < e:
            nMax = e
    return nMin, nMax

#
print('min, max = find_min_max([3, 1, 0, 5, 6, 9, 4, 2])')
min, max = find_min_max([3, 1, 0, 5, 6, 9, 4, 2])
print('min : ', min)
print('max : ', max)
#
print('min, max = find_min_max([3, 1, 0, 5, 6, 9, 4, 2, -1, 10])')
min, max = find_min_max([3, 1, 0, 5, 6, 9, 4, 2, -1, 10])
print('min : ', min)
print('max : ', max)
```

```
 RESTART: C:/YTK-Progs/2018 Book (Python)/ch 5  Function
rn tuple.py
min, max = minmax([3, 1, 0, 5, 6, 9, 4, 2])
min :  0
max :  9
min, max = minmax([3, 1, 0, 5, 6, 9, 4, 2, -1, 10])
min :  -1
max :  10
>>>
```

함수의 속성

◆ 함수의 속성

속성	설명
__doc__	함수를 설명하는 문서 또는 문자열
__name__	함수의 이름
__qualname__	전역 (global scope)에서 함수가 포함된 모듈의 경로 (path)을 나타내며 dot (.)을 포함하여 표시
__module__	함수가 선언되어 있는 모듈의 이름
__defaults__	매개 변수와 기본값 (default value) 의 튜플
__code__	함수를 컴파일하여 생성된 코드 객체
__globals__	함수가 포함된 모듈의 전역 이름 영역에 대한 딕셔너리
__closure__	함수의 변수 바인딩에 대한 셀 튜플

함수의 호출과 인수 전달

◆ Function definition and function call

```
# Function call with arguments

def add(x, y):
    return x + y

#
print('add(1, 2) : ', add(1, 2)) #addition of integers
print('add(1.0, 2.0) : ', add(1.0, 2.0)) #addition of float data
print('add("abc", "def") : ', add("abc", "def")) # addition of strings
print('add([1, 2, 3], [3, 4, 5]) : ', add([1, 2, 3], [3, 4, 5])) #addition of list
print('add((1, 2, 3), (3, 4, 5)) : ', add((1, 2, 3), (3, 4, 5))) #addition of tuples
print('add({1, 2, 3}, {3, 4, 5}) : ', add({1, 2, 3}, {3, 4, 5})) #addition of sets
```

```
add(1, 2) :  3
add(1.0, 2.0) :  3.0
add("abc", "def") :  abcdef
add([1, 2, 3], [3, 4, 5]) :  [1, 2, 3, 3, 4, 5]
add((1, 2, 3), (3, 4, 5)) :  (1, 2, 3, 3, 4, 5)
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 6  Function def, gen
asic Functions, Parameter, Argument\(2) function calls - add()
    print('add({1, 2, 3}, {3, 4, 5}) : ', add({1, 2, 3}, {3, 4
  File "C:\MyPyPackage\TextBook - 2019\ch 6  Function def, gen
asic Functions, Parameter, Argument\(2) function calls - add()
    return x + y
TypeError: unsupported operand type(s) for +: 'set' and 'set'
```

함수 실행 결과 반환 - return

◆ 함수 실행 결과의 반환 - return

```
# Python function that returns multiple data - circle_AreaCircum()

PI = 3.141592
def circle_AreaCircum(radius):
    ar = PI * radius * radius # calculation of area
    circum = 2.0 * PI * radius # calculation of circumference
    return ar, circum

r = 10.0
area, circumference = circle_AreaCircum(r)

print("Circle of radius ({}) :\n Area({}),\n Circumference({})"\
      .format(r, area, circumference))
```

```
Circle of radius (10.0) :
 Area(314.1592),
 Circumference(62.83184)
```

코드 블록

◆ 코드블록 정의

- 모듈
- 함수 (function)
- 클래스 (class)

```
#Block_0_Start
Header-line:
    #Block_1_Start
    Header-line:
        #Block_2_Start
        Header-line:
        . . . . .
        #Block_2_End
    #Block_1_End
#Block_0_End
```

```
# Example of code block - fibo()
def fibo(n):
    if n < 0:
        return None
    elif 0 <= n < 2:
        return n
    else:
        return fibo(n-2) + fibo(n-1)
# ----------
for n in range(10):
    fibo_n = fibo(n)
    print("fibo({:3}) = {:7}".format(n, fibo_n))
```

이름 공간 (Name Space)

◆ 이름공간 (네임스페이스, Name space)

- name: variable name, function name, class name, module name
- name space: dict of name and bound object

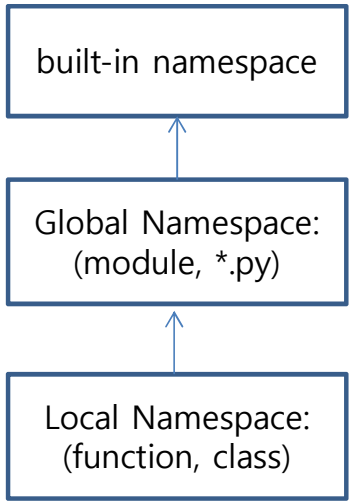
Name Space	설 명
지역 (local)	함수 또는 클래스 코드 블록의 내부에서 바인딩된 이름 locals() 함수로 확인 가능
전역 (global)	코드 블록이 속한 모듈의 최상위 (top-level) 네임 스페이스이며, 모듈 전체에서 사용가능한 이름으로 globals() 함수로 확인 가능
내장 (built-in)	파이썬에 내장되어 있는 객체: 내장 모듈 (__builtins__) 또는 dir(__builtins__)로 확인 가능

유효 범위 (Scope)

◆ 유효범위 (Scope)

- 이름이 어느 유효 범위에 속하는지 정의
- 유효 범위의 우선 규칙 (name scope rule)
  - 지역 이름 범위 (local namespace/scope)
  - 전역 이름 범위 (global namespace/scope)
  - 내장형 이름 범위 (built-in namespace/scope)

◆ 이름 범위의 상속 관계



이름 영역의 명시적 설정 – global, nonlocal

◆ 이름 영역 (name scope)의 명시적 설정 명령

Name scope 지정 선언	의미
global	지역 변수로 선언된 것을 사용하지 않고 전역 변수로 선언된 것을 사용
nonlocal	지역 변수로 선언된 것을 사용하지 않고, 그 코드 영역 바깥에서 선언한 변수를 사용

지역 변수(local variable)와 전역변수(global variable)

```
# Global vs. Local, Name space, scope

x = 1 # global variable
y = 3 # global variable
z = 5 # global variable
def func():
    x = 10 # local variable
    global y
    y = 30 # global variable
    z = 50 # local variable
    print("in func():: x = {0:2d}, y = {1:2d}, z = {2:2d}".format(x, y, z))
    def sub_func():
        nonlocal z
        z = 100
        print("in sub_func():: x = {0:2d}, y = {1:2d}, z = {2:2d}".format(x, y, z))
        print("in sub_func():: locals() : ", locals())
        print("in sub_func():: globals() : ", globals())
    sub_func()
    print("in func():: x = {0:2d}, y = {1:2d}, z = {2:2d}".format(x, y, z))
    print("in func():: locals() : ", locals())
    print("in func():: globals() : ", globals())

# -------------
func()
print("main:: x = {0:2d}, y = {1:2d}, z = {2:2d}".format(x, y, z))
```

```
in func():: x = 10, y = 30, z = 50
in sub_func():: x = 10, y = 30, z = 100
in sub_func():: locals() :  {'x': 10, 'z': 100}
in sub_func():: globals() :  {'__name__': '__main__', '__doc__': None
, '__package__': None, '__loader__': <class '_frozen_importlib.Builti
nImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__':
<module 'builtins' (built-in)>, '__file__': 'C:\\MyPyPackage\\TextBoo
k - 2019\\ch 6 Function def, generator, trigonometric\\6.5 Variables
- local, global\\local vs global variables - func(), inner().py', 'x'
: 1, 'y': 30, 'z': 5, 'func': <function func at 0x032E6DB0>}
in func():: x = 10, y = 30, z = 100
in func():: locals() :  {'sub_func': <function func.<locals>.sub_func
at 0x032E6DF8>, 'x': 10, 'z': 100}
in func():: globals() :  {'__name__': '__main__', '__doc__': None, '_
_package__': None, '__loader__': <class '_frozen_importlib.BuiltinImp
orter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <mo
dule 'builtins' (built-in)>, '__file__': 'C:\\MyPyPackage\\TextBook -
2019\\ch 6 Function def, generator, trigonometric\\6.5 Variables - lo
cal, global\\local vs global variables - func(), inner().py', 'x': 1,
'y': 30, 'z': 5, 'func': <function func at 0x032E6DB0>}
main:: x =  1, y = 30, z =  5
```

파이썬 함수 선언에서의
파라메타 (parameter) 형식 정의 및
함수호출에서의 인수 (argument) 전달

파이썬 함수의 인수 전달

형식지정 위치	Parameter/Argument Syntax	설명
함수 **선언** (**파라 메터 형식** 정의)	def func(arg)	보통 파라메타 (위치 파라메타) 선언
	def func(arg=**value**)	보통 파라메타 (위치 파라메타)에 기본값 설정
	def func(*varargs)	가변적 위치 파라메타 선언 (임의 개수의 위치 파라메터가 전달될 수 있음)
	def func(**kwargs)	가변적 키워드 파라메타 선언 (임의 개수의 키워드 파라메터가 전달될 수 있음)
	def func(arg, *other)	보통 파라메타와 가변적 위치 파라메타 사용 선언
	def func(*arg, **kwargs)	가변적 위치 파라메타와 가변적 키워드 파라메타 사용
	def func(*, name=value)	*표시 이후에는 키워드 파라메타만 사용하도록 선언하며, 기본값 설정
함수 **호출** (**인수** 정의)	func(arg1, arg2, , . . .)	보통 위치 인수 설정, 위치 인수로 전달
	func(arg=value)	키워드 인수, 이름으로 파라메타에 연계
	func(**iterable*)	열거 가능 *iterable* (e.g., string, list, tuple) 객체로 인수를 전달, iterable 객체의 각 개체들이 위치 인수로 연계됨
	func(**dict)	키워드:값으로 표현되는 항목들을 dict 자료형 인수로 전달되며, 인수에 포함된 키워드를 사용하여 키워드 파라메타에 전달됨

위치 인수 (Positional Argument)

```
# Python function with positional arguments

def func_posargs(x, y, z):
    print("passed arguments: x = {0}, y = {1}, z = {2}".format(x, y, z))
    return 3*x + 2*y + z

w = func_posargs(1, 2, 3) # positional arguments
print('w = func_posargs(1, 2, 3) ==> w : ', w)
```

```
passed arguments: x = 1, y = 2, z = 3
w = func_posargs(1, 2, 3) ==> w :  10
```

기본값 (default value)을 지정하는 위치인수

◆ 기본값이 설정된 위치인수

```
# Positional arguments with default values

def volume(width=1, length=1, height=1):
    print("passed arguments: width = {:3}, length = {:3}, height = {:3}"\
          .format(width, length, height))
    return width * length * height

print("volume() = ", volume())
print("volume() = ", volume(10))
print("volume() = ", volume(10, 20))
print("volume() = ", volume(10, 20, 30))
```

```
passed arguments: width =   1, length =   1, height =   1
volume() =  1
passed arguments: width =  10, length =   1, height =   1
volume() =  10
passed arguments: width =  10, length =  20, height =   1
volume() =  200
passed arguments: width =  10, length =  20, height =  30
volume() =  6000
```

키워드 인수 (Keyword argument)

◆ 키워드 인수

```
# Function call with positional arguments vs keyword arguments

def div(x, y):
    return x / y

#
print('div(1, 2) : ', div(1, 2)) # function call with positional arguments
print('div(2, 1) : ', div(2, 1)) # function call with positional arguments
print('div(x = 3, y = 4) : ', div(x = 3, y = 4)) # function call with keyword arguments
print('div(y = 4, x = 3) : ', div(y = 4, x = 3)) # function call with keyword arguments
```

```
div(1, 2) :  0.5
div(2, 1) :  2.0
div(x = 3, y = 4) :  0.75
div(y = 4, x = 3) :  0.75
```

dict 자료형으로 키워드 인수를 전달

◆ dict 자료형으로 키워드 인수 전달

```
# Passing keyword arguments using dict

def func_dict_args(x, y, z):
    print("passed arguments: x = {0}, y = {1}, z = {2}".format(x, y, z))
    return 3*x + 2*y + z

w = func_dict_args(**{'x':1, 'y':2, 'z':3})
  # passing keyword arguments using dict
print('w = func_dict_args(**{'x':1, 'y':2, 'z':3}) ==> w : ', w)
```

```
passed arguments: x = 1, y = 2, z = 3
w = func_dict_args(**{'x':1, 'y':2, 'z':3}) ==> w :  10
```

* 표시 다음에 키워드 인수만을 사용하는 함수

◆ 키워드 인수만을 사용하도록 설정

```
# Keyword-only arguments

def keywordOnlyFunc(x, *, y, z):
    print("passed arguments: x = {0}, y = {1}, z = {2}".format(x, y, z))
    return 3*x + 2*y + z

keywordOnlyFunc(1, y=2, z=3)
keywordOnlyFunc(1, **{'z':3, 'y':2})
keywordOnlyFunc(1, 2, 3)
```

```
passed arguments: x = 1, y = 2, z = 3
passed arguments: x = 1, y = 2, z = 3
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 6  Function def, generator, trigon
ometric\6.2 Function Arguments\(5) keyWordOnly Arguments - keywordOnlyfunc()
.py", line 10, in <module>
    keywordOnlyFunc(1, 2, 3)
TypeError: keywordOnlyFunc() takes 1 positional argument but 3 were given
```

가변적 위치 인수 (variable positional argument)

◆ 가변적 개수의 위치 인수 사용

```
# Variable positional arguments

def varPosArgsFunc(*args):
    print(type(args))
    print("args = ", args)
    s = 0
    for x in args:
      s += x
    return s

varPosArgsFunc(1)
varPosArgsFunc(1, 2)
varPosArgsFunc(1, 2, 3)
```

```
<class 'tuple'>
args =  (1,)
<class 'tuple'>
args =  (1, 2)
<class 'tuple'>
args =  (1, 2, 3)
```

가변적 위치 인수의 함수 호출에 다양한 자료형 인수를 전달

◆ 가변적 위치 인수를 사용하는 함수 호출

```
# Variable Positional arguments

def varPosArgsFunc(*args):
    print("args = <{}>, type= {}, len={}".format(args, type(args), len(args)))
    num_arg = len(args)
    for a in range(num_arg):
        t_arg = type(args[a])
        print("args[{}]: type = {}".format(a,t_arg))
        if t_arg == int:
            print("arg ({}) : ".format(t_arg), args[a])
        elif t_arg == float:
            print("arg ({}) : ".format(t_arg), args[a])
        elif (t_arg == list) or (t_arg == tuple):
            print("arg ({}) : ".format(t_arg), end='')
            len_list = len(args[a])
            for n in range(len_list):
                print(args[a][n], end=', ')
            print()
        else:
            print("Currently argument type {} cannot be processed !!".format(t_arg))
a, b = 1, 2.5
L = [1, 2, 3, 4, 5]
T = ('one', 'two', 'three')
print("L : ", L)
print("T : ", T)
varPosArgsFunc(a, b, L, T) # invoke function with variable positional arguments
```

```
L :  [1, 2, 3, 4, 5]
T :  ('one', 'two', 'three')
args = <(1, 2.5, [1, 2, 3, 4, 5], ('one', 'two'
, 'three'))>, type= <class 'tuple'>, len=4
args[0]: type = <class 'int'>
arg (<class 'int'>) :  1
args[1]: type = <class 'float'>
arg (<class 'float'>) :  2.5
args[2]: type = <class 'list'>
arg (<class 'list'>) : 1, 2, 3, 4, 5,
args[3]: type = <class 'tuple'>
arg (<class 'tuple'>) : one, two, three,
```

위치인수와 가변적 위치인수

◆ 위치인수와 가변적 위치인수의 혼합 사용

```
# Positional arguments and variable positional arguments

def varPosArgsFunc(a, *args):
    print("a = ", a, "args = ", args)
    print(type(args))
    s = 0
    for x in args:
      s += x
    return s

varPosArgsFunc(1)
varPosArgsFunc(1, 2)
varPosArgsFunc(1, 2, 3)
varPosArgsFunc(1, 2, 3, 4, 5)
```

```
a =  1 args =  ()
<class 'tuple'>
a =  1 args =  (2,)
<class 'tuple'>
a =  1 args =  (2, 3)
<class 'tuple'>
a =  1 args =  (2, 3, 4, 5)
<class 'tuple'>
```

가변적 키워드 인수 (variable keyword argument)

```
# Variable keyword arguments (1)

def varKwArgsFunc(**kwargs):
    print("kwargs = <{}>, type= {}, len={}".format(kwargs, type(kwargs),       len(kwargs)))
    num_arg = len(kwargs)
    print("num_arg : ", num_arg)
    for kw in kwargs:
        arg_type = type(kwargs[kw])
        #print("kwargs({}): type = {}".format(kw, type(kwargs[kw])))
        if arg_type == int:
            print("kwarg ({}) : {}".format(kw, kwargs[kw]))
        elif arg_type == float:
            print("kwarg ({}) : {}".format(kw, kwargs[kw]))
        elif arg_type == str:
            print("kwarg ({}) : {}".format(kw, kwargs[kw]))
        elif (arg_type == list) or (arg_type == tuple):
            print("kwarg ({}) : {}".format(kw, kwargs[kw]), end=' ')
            kwarg = kwargs[kw]
            len_list = len(kwarg)
            print("; elements of {} : ".format(type(kwarg)), end=' ')
            for n in range(len_list):
                print(kwarg[n], end=', ')
            print()
        else:
            print("Currently argument type {} cannot be processed!!".format(kw_arg))
```

```
L :  [1, 2, 3, 4, 5]
T :  ('one', 'two', 'three')
kwargs = <{'a': 1}>, type= <class 'dict'>, len=1
num_arg :  1
kwarg (a) : 1
kwargs = <{'a': 1, 'b': 2.5}>, type= <class 'dict'>, len=2
num_arg :  2
kwarg (a) : 1
kwarg (b) : 2.5
kwargs = <{'a': 1, 'b': 2.5, 'c': [1, 2, 3, 4, 5]}>, type= <class 'dict'>, len=3
num_arg :  3
kwarg (a) : 1
kwarg (b) : 2.5
kwarg (c) : [1, 2, 3, 4, 5] ; elements of <class 'list'> :  1, 2, 3, 4, 5,
kwargs = <{'a': 1, 'b': 2.5, 'c': [1, 2, 3, 4, 5], 'd': ('one', 'two', 'three')}>, type= <class 'dict'>, len=4
num_arg :  4
kwarg (a) : 1
kwarg (b) : 2.5
kwarg (c) : [1, 2, 3, 4, 5] ; elements of <class 'list'> :  1, 2, 3, 4, 5,
kwarg (d) : ('one', 'two', 'three') ; elements of <class 'tuple'> :  one, two, three,
```

```
# Variable keyword arguments (2)

L = [1, 2, 3, 4, 5]
T = ('one', 'two', 'three')
print("L : ", L)
print("T : ", T)
varKwArgsFunc(a=1)  # invoke function with variable keywords arguments
varKwArgsFunc(a=1, b=2.5)  # invoke function with variable keywords arguments
varKwArgsFunc(a=1, b=2.5, c=L)  # invoke function with variable keywords arguments
varKwArgsFunc(a=1, b=2.5, c=L, d=T)  # invoke function with variable keywords arguments
```

```
L :  [1, 2, 3, 4, 5]
T :  ('one', 'two', 'three')
kwargs = <{'a': 1}>, type= <class 'dict'>, len=1
num_arg :  1
kwarg (a) : 1
kwargs = <{'a': 1, 'b': 2.5}>, type= <class 'dict'>, len=2
num_arg :  2
kwarg (a) : 1
kwarg (b) : 2.5
kwargs = <{'a': 1, 'b': 2.5, 'c': [1, 2, 3, 4, 5]}>, type= <class 'dict'>, len=3
num_arg :  3
kwarg (a) : 1
kwarg (b) : 2.5
kwarg (c) : [1, 2, 3, 4, 5] ; elements of <class 'list'> :  1, 2, 3, 4, 5,
kwargs = <{'a': 1, 'b': 2.5, 'c': [1, 2, 3, 4, 5], 'd': ('one', 'two', 'three')}>, type= <class 'dict'>, len=4
num_arg :  4
kwarg (a) : 1
kwarg (b) : 2.5
kwarg (c) : [1, 2, 3, 4, 5] ; elements of <class 'list'> :  1, 2, 3, 4, 5,
kwarg (d) : ('one', 'two', 'three') ; elements of <class 'tuple'> :  one, two, three,
```

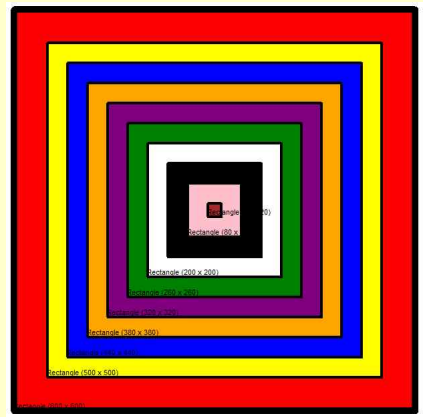
가변적 위치인수와 가변적 키워드 인수의 혼합

```
# Variable Positional arguments and variable Keyword arguments

def varPosVarKeywordFunc(*args, **kwargs):
    print("args = ", args)
    print("kwargs = ", kwargs)
    s = "var args: ( "
    for x in args:
        s += "{}, ".format(str(x))
    s += " ); var kwargs: { "
    for x in kwargs:
        s += "{}:{}, ".format(x, str(kwargs[x]))
    s += "}"
    return s

r1 = varPosVarKeywordFunc(1, A=1)
print('varPosVarKeywordFunc(1, A=1) ==> : \n', r1)

r2 = varPosVarKeywordFunc(1, 2, A=10, B=20)
print('varPosVarKeywordFunc(1, 2, A=10, B=20) ==> : \n', r2)

r3 = varPosVarKeywordFunc(1, 2, 3, A=10, B=20, C=30)
print('varPosVarKeywordFunc(1, 2, 3, A=10, B=20, C=30) ==> : \n', r3)
```

```
args =  (1,)
kwargs =  {'A': 1}
varPosVarKeywordFunc(1, A=1) ==> :
 var args: ( 1,  ); var kwargs: { A:1, }
args =  (1, 2)
kwargs =  {'A': 10, 'B': 20}
varPosVarKeywordFunc(1, 2, A=10, B=20) ==> :
 var args: ( 1, 2,  ); var kwargs: { A:10, B:20, }
args =  (1, 2, 3)
kwargs =  {'A': 10, 'B': 20, 'C': 30}
varPosVarKeywordFunc(1, 2, 3, A=1, B=2, C=30) ==> :
 var args: ( 1, 2, 3,  ); var kwargs: { A:10, B:20, C:30, }
```

파이썬 함수 사용의 예 – draw_rect()

```
# Function draw_rect() with arguments

import turtle
def draw_rect(pen, width, length, fill_color = "white", pen_color = "black", psize = 4):
    pen.pensize(psize)
    pen.color(pen_color, fill_color)
    pen.setheading(0)
    pen.up();  pen.setpos(-width//2, -length//2);   pen.down()

    pen.begin_fill()
    pen.setpos(width//2, -length//2)
    pen.setpos(width//2, length//2)
    pen.setpos(-width//2, length//2)
    pen.setpos(-width//2, -length//2)
    pen.end_fill()

#----------------------
t = turtle.Turtle()
t.ht() #hide turtle
colors = ["red", "yellow", "blue", "orange", "purple",\
              "green", "white", "black", "pink", "brown"]
c = 0
width, length = 600, 600
draw_rect(t, width, length, colors[c], "black", 10)
c += 1
t.write("Rectangle ({} x {})".format(width, length))
for i in range(250, 1, -30):
    width, length = i*2, i*2
    draw_rect(t, width, length, colors[c])
    c += 1
    t.write("Rectangle ({} x {})".format(width, length))
```



변경이 가능한 인수 (Mutable Arguments)

◆ 인수 값의 변경

```
# Mutable arguments

def funcA(x):
    print("at funcA(), before assignment :: x = {}, id = {}".format(x, id(x)))
    if isinstance(x, list): # list is mutable
        x[0] = 10 # 리스트 원소 변경
    if isinstance(x, dict): # dict is mutable
        x['color'] = 'red' # 딕셔너리 원소 변경
        x['price'] = 300 # 딕셔너리 원소 변경
    print("at funcA(), after assignment :: x = {}, id = {}".format(x, id(x)))

L = [1, 2, 3] # list
print("at main(), before function call :: L = {}, id = {}".format(L, id(L)))
funcA(L)
print("at main(), after function call :: L = {}, id = {}".format(L, id(L)))

D = {'color':'blue', 'price':100} # dictionary
print("at main(), before function call :: D = {}, id = {}".format(D, id(D)))
funcA(D)
print("at main(), after function call :: D = {}, id = {}".format(D, id(D)))
```

```
at main(), before function call :: L = [1, 2, 3], id = 66086920
at funcA(), before assignment :: x = [1, 2, 3], id = 66086920
at funcA(), after assignment :: x = [10, 2, 3], id = 66086920
at main(), after function call :: L = [10, 2, 3], id = 66086920
at main(), before function call :: D = {'color': 'blue', 'price': 100}, id = 63611680
at funcA(), before assignment :: x = {'color': 'blue', 'price': 100}, id = 63611680
at funcA(), after assignment :: x = {'color': 'red', 'price': 300}, id = 63611680
at main(), after function call :: D = {'color': 'red', 'price': 300}, id = 63611680
```

변경이 불가능한 인수 (Immutable Arguments)

◆ 인수값이 변경되지 않는 경우

```
# Immutable arguments

def funcB(x):
    print("at funcB(), before assignment :: x = {}, id = {}".format(x, id(x)))
    if isinstance(x, int): # int is immutable
        x = 10 # 새로운 정수형 변수 생성
    if isinstance(x, tuple):
        x = (10, 20, 30) # 새로운 튜플 생성
    if isinstance(x, list):
        x = [10, 20, 30] # 새로운 리스트 생
    print("at funcB(), after assignment :: x = {}, id = {}".format(x, id(x)))

a = 1 # int variable
print("at main(), before function call :: a = {}, id = {}".format(a, id(a)))
funcB(a)
print("at main(), after function call :: a = {}, id = {}".format(a, id(a)))
print()

T = (1, 2, 3) # tuple
print("at main(), before function call :: T = {}, id = {}".format(T, id(T)))
funcB(T)
print("at main(), after function call :: T = {}, id = {}".format(T, id(T)))
print()

L = [1, 2, 3] # list
id(L)
print("at main(), before function call :: L = {}, id = {}".format(L, id(L)))
funcB(L)
print("at main(), after function call :: L = {}, id = {}".format(L, id(L)))
```

```
at main(), before function call :: a = 1, id = 1533536176
at funcB(), before assignment :: x = 1, id = 1533536176
at funcB(), after assignment :: x = 10, id = 1533536320
at main(), after function call :: a = 1, id = 1533536176

at main(), before function call :: T = (1, 2, 3), id = 62198216
at funcB(), before assignment :: x = (1, 2, 3), id = 62198216
at funcB(), after assignment :: x = (10, 20, 30), id = 62198344
at main(), after function call :: T = (1, 2, 3), id = 62198216

at main(), before function call :: L = [1, 2, 3], id = 62130248
at funcB(), before assignment :: x = [1, 2, 3], id = 62130248
at funcB(), after assignment :: x = [10, 20, 30], id = 62130824
at main(), after function call :: L = [1, 2, 3], id = 62130248
```

파이썬 내장 함수 (embedded functions)

파이썬 내장 함수 (1)

분류	파이썬 내장 함수	설명
자료형 변수 생성/변환	bool(x)	x를 bool 형태로 변환 출력
	int(i_str)	문자열 i_str을 정수 데이터로 변환하여 반환
	float(f_str)	문자열 f_str을 부동 소수점으로 변환하여 반환
	complex(c_str)	문자열 c_str을 복소수로 변환하여 반환
	complex(r, i)	실수 r과 i를 전달받아 실수부 (real part)가 r, 허수부(imaginary part)가 i인 복소수 객체를 생성
	bin(x)	정수값 x를 전달받아 2진수 bit (binary digit) 문자열로 변환하여 반환
	hex(x)	정수값 x를 전달받아 16진수 bytes 문자열로 변환하여 반환
	oct(x)	정수 형태의 데이터 x를 전달받아 8진수 bytes 문자열로 변환하여 반환
	str(object)	객체 object를 문자열 형태로 변환하여 반환
	list(s)	반복 가능한 자료형 데이터 s를 전달받아 리스트로 만들어 반환
	tuples(s)	반복 가능한 자료형 데이터 s를 전달받아 tuple 형태로 변환하여 반환
수치 데이터 연산	abs(x)	x의 절대값 (absolute value) 복소수 (x + yj)인 경우 절대값 sqrt(x**2 + y**2)
	divmod(a, b)	a를 b로 나눈 몫과 나머지 (모듈로 계산)를 tuple로 출력 divmod(7, 3) => (2, 1)
	max(L)	반복 가능한 자료형 데이터 L을 전달받아 최댓값을 찾아 반환
	min(L)	반복 가능한 자료형 데이터 L을 전달받아 최솟값을 찾아 반환
	pow(x, y)	x를 y 제곱한 지수승 결과값을 반환
	round(n, r)	숫자 n을 소수점 r자리까지 반올림하여 반환

파이썬 내장 함수 (2)

분류	파이썬 내장 함수	설명
문자/문자열 연산	chr(i)	ASCII 코드 값 i에 해당하는 문자를 출력
	ord(c)	문자 c의 UNICODE 코드 값을 출력
반복 가능 자료형 연산	all(x)	반복 (iterate) 가능한 자료형 x의 모든 값이 True이면 True, 그렇지 않으면 False
	any(x)	반복 (iterate) 가능한 자료형 x의 값 중 하나라도 True이면 True, 그렇지 않으면 (모두가 False 이면) False
	enumerate()	순서가 있는 자료형 데이터를 전달받아 차례로 출력
	filter(a, b)	반복가능한 자료형 데이터 b를 함수 a에 대입하였을 때 결과가 True인 것만 출력
	len(s)	전달받은 s의 길이를 반환
	map(func, x)	함수 func와 반복 가능한 자료형 데이터 x를 전달받고 전달 받은 데이터 x의 각 요소를 함수 func에 적용하여 실행된 결과를 반환
	sorted(L)	인수 L을 정렬하여 리스트로 반환
	sum(L)	리스트 L에 포함된 모든 항목들을 합산하여 반환
	zip(i)	동일한 개수로 이루어진 자료형 (list, tuple)을 차례대로 짝으로 묶어 줌
객체	dir()	객체가 자체적으로 가지고 있는 변수나 함수를 표시
	id(object)	객체 object의 고유 주소 (identifier) 값을 출력
	isinstance(o, c)	객체 o가 클래스 c의 인스탄스인가를 판단하여 참이면 True, 거짓이면 False를 반환
	type(object)	전달된 객체 object의 자료형을 반환

파이썬 내장 함수 (3)

분류	파이썬 내장 함수	설명
파일/ 디렉토리 관리	open(f, m)	파일이름 f를 모드 m으로 열고 그 파일 객체를 반환 fin = open("InputData.txt", 'r')
	compile(s, f, n)	파일 f로부터 소스 s의 문자열을 읽어 컴파일하고 Python code object를 생성하며, 생성된 객체는 exec() 또는 eval()를 사용하여 실행
실행 가능 문자열	eval(expression)	실행 가능한 문자열을 전달받아 이를 실행한 후 결과값을 반환
	exec()	문자열로 된 Python 실행문을 전달받아 실행
입출력	input(prompt)	화면에 prompt를 출력하고, 표준입력장치(키보드)로부터 데이터를 입력
	print()	화면으로 지정된 양식에 따라 출력
함수	lambda()	def와 같은 함수 생성 기능을 제공하며, 이름이 없는 함수를 생성

반복가능 자료형의 연산을 위한 내장 함수 - map()

◆ map() function

```
# map() function

X = [1, 2, 3, 4, 5]
Sq_X = list(map(lambda x : x**2, X))
print("X    : ", X)
print("Sq_X : ", Sq_X)

Y = [1, 2, 3, 4, 5]
print("Y    : ", Y)
Pow_X_Y = list(map(pow, X, Y))
print("X^Y  : ", Pow_X_Y)
```

```
X    :  [1, 2, 3, 4, 5]
Sq_X :  [1, 4, 9, 16, 25]
Y    :  [1, 2, 3, 4, 5]
X^Y  :  [1, 4, 27, 256, 3125]
```

반복가능 자료형의 연산을 위한 내장 함수 - filter()

◆ filter()

```
# Python embedded function - filter()

X = list(range(-5, 6, 1))
print("X : ", X)
Negative_X = list(filter(lambda x : x<0, X))  # select only negative element
print("X with filtering(x < 0)  : ", Negative_X)

Positive_X = list(filter(lambda x : x>=0, X))  # select only positive element
print("X with filtering(x >= 0) : ", Positive_X)

Non_zero_X = list(filter(None, X)) # select non-zero element
print("X with filtering(None) : ", Non_zero_X)
```

```
X :  [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
X with filtering(x < 0)  :  [-5, -4, -3, -2, -1]
X with filtering(x >= 0) :  [0, 1, 2, 3, 4, 5]
X with filtering(None) :  [-5, -4, -3, -2, -1, 1, 2, 3, 4, 5]
```

반복가능 자료형의 연산을 위한 내장 함수 - zip()

◆ zip(), *zip()

```
# zip(), zip(*)

X = [1, 2, 3]
Y = ['a', 'b', 'c']
Z = [100, 200, 300]
print('X : ', X)
print('Y : ', Y)
print('Z : ', Z)

Zip_XY = list(zip(X, Y))
print('Zip_XY : ', Zip_XY)

A, B = zip(*Zip_XY)
print('A : ', A)
print('B : ', B)

D = list(zip(X, Y, Z))
print('D : ', D)

d1, d2, d3 = zip(*D)
print('d1 = ', d1)
print('d2 = ', d2)
print('d3 = ', d3)
```

```
X :  [1, 2, 3]
Y :  ['a', 'b', 'c']
Z :  [100, 200, 300]
Zip_XY :  [(1, 'a'), (2, 'b'), (3, 'c')]
A :  (1, 2, 3)
B :  ('a', 'b', 'c')
D :  [(1, 'a', 100), (2, 'b', 200), (3, 'c', 300)]
d1 =  (1, 2, 3)
d2 =  ('a', 'b', 'c')
d3 =  (100, 200, 300)
```

반복가능 자료형의 연산을 위한 내장 함수 – sorted()

```
# Python embedded function - sorted()

L = [5, 7, 3, 1, 9, 2, 4, 6, 8, 0] # list
print("L : ", L)
print("sorted(L) : ", sorted(L))
print("sorted(L, reverse=True) : ", sorted(L, reverse=True))

TL = [("Park", 26, 77.4), ("Kim", 25, 74.7), ("Lee", 23, 80.5), ("Ahn", 24, 82.2)]
print("TL : ", TL)
print("sorted(TL) : ", sorted(TL))
print("sorted(TL, reverse=True) : \n",  sorted(TL, reverse=True))
print("sorted(TL, key = lambda person : person[1]) : \n",\
      sorted(TL, key = lambda person : person[1]))
print("sorted(TL, key = lambda person : person[2]) : \n",\
      sorted(TL, reverse=True, key = lambda person : person[2]))
```

```
L :  [5, 7, 3, 1, 9, 2, 4, 6, 8, 0]
sorted(L) :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
sorted(L, reverse=True) :  [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
TL :  [('Park', 26, 77.4), ('Kim', 25, 74.7), ('Lee', 23, 80.5), ('Ahn', 24, 82.2)]
sorted(TL) :  [('Ahn', 24, 82.2), ('Kim', 25, 74.7), ('Lee', 23, 80.5), ('Park', 26, 77.4)]
sorted(TL, reverse=True) :
 [('Park', 26, 77.4), ('Lee', 23, 80.5), ('Kim', 25, 74.7), ('Ahn', 24, 82.2)]
sorted(TL, key = lambda person : person[1]) :
 [('Lee', 23, 80.5), ('Ahn', 24, 82.2), ('Kim', 25, 74.7), ('Park', 26, 77.4)]
sorted(TL, key = lambda person : person[2]) :
 [('Ahn', 24, 82.2), ('Lee', 23, 80.5), ('Park', 26, 77.4), ('Kim', 25, 74.7)]
```

파이썬 객체 (object) 관리에 관련된 내장 함수

◆ type(), dir()

```
# Python embedded function - type(), dir()

L = [0, 1, 2]
T = (0, 1, 2, 3, 4)
D = {'A':10, 'B':11, 'C': 12, 'D': 13}

print("L : ", L)
print("type(L) = ", type(L))
print("dir(L) = ", dir(L))

print("\nT : ", T)
print("type(T) = ", type(T))
print("dir(T) = ", dir(T))

print("\nD : ", D)
print("type(D) = ", type(D))
print("dir(D) = ", dir(D))
```

```
L :  [0, 1, 2]
type(L) =  <class 'list'>
dir(L) =  ['__add__', '__class__', '__contains__', '__delattr
__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '
__hash__', '__iadd__', '__imul__', '__init__', '__init_subcla
ss__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__
', '__reversed__', '__rmul__', '__setattr__', '__setitem__',
'__sizeof__', '__str__', '__subclasshook__', 'append', 'clear
', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remo
ve', 'reverse', 'sort']

T :  (0, 1, 2, 3, 4)
type(T) =  <class 'tuple'>
dir(T) =  ['__add__', '__class__', '__contains__', '__delattr
__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__getnewargs__', '__gt__'
, '__hash__', '__init__', '__init_subclass__', '__iter__', '_
_le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__set
attr__', '__sizeof__', '__str__', '__subclasshook__', 'count'
, 'index']

D :  {'A': 10, 'B': 11, 'C': 12, 'D': 13}
type(D) =  <class 'dict'>
dir(D) =  ['__class__', '__contains__', '__delattr__', '__del
item__', '__dir__', '__doc__', '__eq__', '__format__', '__ge_
_', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__iter__', '__le__', '__len
__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_e
x__', '__repr__', '__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', '
get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'updat
e', 'values']
```

파이썬 객체 (object) 관리에 관련된 내장 함수

◆ id(), isinstance()

```
# Python embedded function - id(), isinstance() (1)

def funcA(arg):
    print("funcA:: arg (type({}) = {}".format(type(arg), arg))
    if isinstance(arg, int):
        print("arg(id({})) is integer of bit_length ({})".format(id(arg), arg.bit_length()))
    elif isinstance(arg, list):
        print("arg(id({})) is list of len ({})".format(id(arg), len(arg)))
    elif isinstance(arg, tuple):
        print("arg(id({})) is tuple of len ({})".format(id(arg), len(arg)))
    elif isinstance(arg, dict):
        print("arg(id({})) is dict of len ({})".format(id(arg), len(arg)))
    else:
        print("arg(id({})) is unknown, yet".format(id(arg)))
    print()

d = 1234567
L = [0, 1, 2]
T = (0, 1, 2, 3, 4)
D = {'A':10, 'B':11, 'C': 12, 'D': 13}
```

```
# Python embedded function - id(), isinstance() (2)
# -------------------
print("d = {}, id(d) = {}".format(d, id(d)))
print("L = {}, id(L) = {}".format(L, id(L)))
print("T = {}, id(T) = {}".format(T, id(T)))
print("D = {}, id(D) = {}".format(D, id(D)))
funcA(d)
funcA(L)
funcA(T)
funcA(D)
```

```
d = 1234567, id(d) = 50512608
L = [0, 1, 2], id(L) = 35398200
T = (0, 1, 2, 3, 4), id(T) = 53045264
D = {'A': 10, 'B': 11, 'C': 12, 'D': 13}, id(D) = 50466272
funcA:: arg (type(<class 'int'>) = 1234567
arg(id(50512608)) is integer of bit_length (21)

funcA:: arg (type(<class 'list'>) = [0, 1, 2]
arg(id(35398200)) is list of len (3)

funcA:: arg (type(<class 'tuple'>) = (0, 1, 2, 3, 4)
arg(id(53045264)) is tuple of len (5)

funcA:: arg (type(<class 'dict'>) = {'A': 10, 'B': 11, 'C': 12, 'D': 13}
arg(id(50466272)) is dict of len (4)
```

내장함수(sum(), max(), min())를 사용한 배열의 통계 분석

```
# Application of Python embedded functions - array statistics
import math
def statistics(L):
    sum_L = sum(L)
    max_L = max(L)
    min_L = min(L)
    len_L = len(L)
    avg_L = sum_L / len_L
    sq_diff_sum = 0
    for x in L:
        sq_diff_sum += pow((x - avg_L), 2)
    var = sq_diff_sum / len_L
    std_dev = math.sqrt(var)
    print("L (size: {}) : {}".format(len_L, L))
    print("Statistics of L : Max ({}), Min ({}), Avg({:7.2f})"\
          .format(max_L, min_L, avg_L))
    print("  var ({:10.2f}), std_dev ({:10.2f})".format(var, std_dev))

#--------------------
A = list(range(10))
statistics(A)
B = list(range(-5, 5))
statistics(B)
C = list(range(0, -10, -1))
statistics(C)
```

```
L (size: 10) : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Statistics of L : Max (9), Min (0), Avg(   4.50)
  var (      8.25), std_dev (      2.87)
L (size: 10) : [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
Statistics of L : Max (4), Min (-5), Avg(  -0.50)
  var (      8.25), std_dev (      2.87)
L (size: 10) : [0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
Statistics of L : Max (0), Min (-9), Avg(  -4.50)
  var (      8.25), std_dev (      2.87)
```

람다 함수 (Lambda function)
1급 함수 (First class function)
함수 수식자 (Function Decorator)

람다 함수 (Lambda function)

◆ 람다 함수 (Lambda function)

- 함수의 이름을 설정하지 않고 함수의 기능만을 설정한 함수
- 주로 함수의 호출에서 인수로 전달되어, 해당 함수에서 처리해야 하는 데이터들에 대하여 람다 함수의 기능이 수행되도록 함

```
# Lambda function - double, triple

double = lambda x : x * 2
triple = lambda x : x + x + x

print("double : ", double)
print("double(5) = ", double(5))
print("double('ABC') = ", double('ABC'))

print("triple : ", triple)
print("triple(5) = ", triple(5))
print("triple('ABC') = ", triple('ABC'))
```

```
double :  <function <lambda> at 0x03260348>
double(5) =  10
double('ABC') =  ABCABC
triple :  <function <lambda> at 0x006D2108>
triple(5) =  15
triple('ABC') =  ABCABCABC
```

람다함수를 인수로 전달

◆ Lambda function as a parameter

```
# Lambda function - used as parameter

from types import * # types.LamdaType, types.FunctionType
def G(f, n=0):
    f_type = type(f)
    print("G(f)::f_type = ", f_type)
    if f_type is LambdaType or f_type is FunctionType:
        return [f(x) for x in range(n)]
    else:
        return None

# --------------
print("G(10) => ", G(10))
print("G(lambda x: x**2, 10) => ", G(lambda x: x**2, 10))
print("G(lambda x: x**3, 5) => ", G(lambda x: x**3, 5))
```

```
G(f)::f_type =  <class 'int'>
G(10) =>  None
G(f)::f_type =  <class 'function'>
G(lambda x: x**2, 10) =>  [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
G(f)::f_type =  <class 'function'>
G(lambda x: x**3, 5) =>  [0, 1, 8, 27, 64]
```

1급 함수 (first class function)

◆1급 함수

- 파이썬 함수를 변수 이름으로 설정할 수 있는 함수
- 파이썬 함수를 다른 함수에게 인수로 전달 할 수 있는 함수
- 다른 함수의 실행 결과로 반환되는 함수

```
# First class function - Greeting

def Greeting(name):
    return "Hello, " + name

sayHello = Greeting # binding first class function to a variable name
print("sayHello('Kim') ==> ", sayHello('Kim'))
print("dir(sayHello) : ", dir(sayHello))
```

```
sayHello('Kim') ==>  Hello, Kim
dir(sayHello) :  ['__annotations__', '__call__', '__class__', '__closur
e__', '__code__', '__defaults__', '__delattr__', '__dict__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__get__', '__getattribute
__', '__globals__', '__gt__', '__hash__', '__init__', '__init_subclass_
_', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__', '__
ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__rep
r__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']
```

1급함수를 함수호출의 인수로 전달

```
# Function as Object/Argument
#    to Another Function
import copy

def cubic(n):
    return n * n * n

def rFibo(n):
    if n <= 1:
        return n
    return rFibo(n-1) + rFibo(n-2)

def rFact(n):
    if n <= 1:
        return 1
    else:
        d = rFact(n-1)
        return n * d
```

```
def ApplyFunctionToAnotherFunction(L, func):
    """ Assume L is a list, func a function.
        Apply each element in L into function func,
        generates the result list, and return """
    L_result = []
    L_result.clear()

    for i in range(len(L)):
        d = L[i]
        res = func(d)
        L_result.insert(i, res)
    return L_result
```

◆ 실행 결과

```
################################
#  Application

L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print("L = ", L)

print("\nApply cubic() for each element of list L :")
L_res = ApplyFunctionToAnotherFunction(L, cubic)
print(L_res)

print("\nApply rFact() for each element of list L :")
L_res = ApplyFunctionToAnotherFunction(L, rFact)
print(L_res)

print("\nApply rFibo() for each element of list L :")
L_res = ApplyFunctionToAnotherFunction(L, rFibo)
print(L_res)
```

```
L =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Apply cubic() for each element of list L :
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]

Apply rFact() for each element of list L :
[1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]

Apply rFibo() for each element of list L :
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

함수의 실행 시간을 측정 – measureTime()

◆ 경과 시간 측정 함수

```
#First class function as argument to another function
import time
def measureTime(f):
    def wrapper(n): # n is argument
        t_before = time.time()
        ret = f(n)
        t_after = time.time()
        elapsed_time = t_after - t_before
        #print("Execution time : {0}".format(elapsed_time))
        return ret, elapsed_time
    return wrapper

def rFibo(n):
    if n <= 1:
        return n
    return rFibo(n-1) + rFibo(n-2)

measureTime_rFibo = measureTime(rFibo)
for n in range(1, 35, 1):
    fibo_n, t_elapsed = measureTime_rFibo(n)
    print("Fibo({:3d}) = {:8d}, took {:10.3f} milli-seconds"\
          .format(n, fibo_n, t_elapsed * 1000))
```

```
Fibo(  1) =        1, took      0.000 milli-seconds
Fibo(  2) =        1, took      0.000 milli-seconds
Fibo(  3) =        2, took      0.000 milli-seconds
Fibo(  4) =        3, took      0.000 milli-seconds
Fibo(  5) =        5, took      0.000 milli-seconds
Fibo(  6) =        8, took      0.000 milli-seconds
Fibo(  7) =       13, took      0.000 milli-seconds
Fibo(  8) =       21, took      0.000 milli-seconds
Fibo(  9) =       34, took      0.000 milli-seconds
Fibo( 10) =       55, took      0.000 milli-seconds
Fibo( 11) =       89, took      0.000 milli-seconds
Fibo( 12) =      144, took      0.000 milli-seconds
Fibo( 13) =      233, took      0.000 milli-seconds
Fibo( 14) =      377, took      0.000 milli-seconds
Fibo( 15) =      610, took     10.000 milli-seconds
Fibo( 16) =      987, took      0.000 milli-seconds
Fibo( 17) =     1597, took      0.000 milli-seconds
Fibo( 18) =     2584, took      0.000 milli-seconds
Fibo( 19) =     4181, took      0.000 milli-seconds
Fibo( 20) =     6765, took      0.000 milli-seconds
Fibo( 21) =    10946, took     10.000 milli-seconds
Fibo( 22) =    17711, took      0.000 milli-seconds
Fibo( 23) =    28657, took     10.000 milli-seconds
Fibo( 24) =    46368, took     20.000 milli-seconds
Fibo( 25) =    75025, took     40.000 milli-seconds
Fibo( 26) =   121393, took     50.000 milli-seconds
Fibo( 27) =   196418, took     90.000 milli-seconds
Fibo( 28) =   317811, took    140.000 milli-seconds
Fibo( 29) =   514229, took    240.000 milli-seconds
Fibo( 30) =   832040, took    370.000 milli-seconds
Fibo( 31) =  1346269, took    620.001 milli-seconds
Fibo( 32) =  2178309, took    990.001 milli-seconds
Fibo( 33) =  3524578, took   1590.002 milli-seconds
Fibo( 34) =  5702887, took   2580.004 milli-seconds
```

함수 수식자 (Function Decorator)

◆ 함수 수식자

- 함수의 실행 직전과 직후에 미리 설정되어 있는 기능을 수행하게 하는 기능 구현
- 다른 함수를 반환
- 함수 이름 앞에 '@' 를 붙여 함수 수식자로 설정
- 함수 수식자는 사용되기 이전에 미리 설정되어 있어야 함
- 함수 수식자는 1급 함수를 사용함

함수 수식자 (function decorator) 사용 예

◆ measureTime() as decorator

```
# decoration function for elapsed time measurement
import time

def measureTime(func):
    def wrapper(n):
        t_before = time.time()
        ret = func(n)
        t_after = time.time()
        elapsed_time = t_after - t_before
        print("measureTime::Execution time of func({:3d}) : {:.3f}".format(n, elapsed_time))
        return ret
    return wrapper

@measureTime
def rFibo(n):
    if n <= 1:
        return n
    return rFibo(n-1) + rFibo(n-2)

#----------------------------
for n in range(1, 11, 1):
    fibo_n = rFibo(n)
    print("Fibo({:3d}) = {:8d}".format(n, fibo_n))
```

```
measureTime::Execution time of func(  1) : 0.000
Fibo(  1) =        1
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  0) : 0.000
measureTime::Execution time of func(  2) : 0.012
Fibo(  2) =        1
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  0) : 0.000
measureTime::Execution time of func(  2) : 0.010
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  3) : 0.019
Fibo(  3) =        2
```

. . . .

```
measureTime::Execution time of func(  5) : 0.035
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  0) : 0.000
measureTime::Execution time of func(  2) : 0.005
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  3) : 0.010
measureTime::Execution time of func(  1) : 0.000
measureTime::Execution time of func(  0) : 0.000
measureTime::Execution time of func(  2) : 0.005
measureTime::Execution time of func(  4) : 0.021
measureTime::Execution time of func(  6) : 0.061
measureTime::Execution time of func(  8) : 0.167
measureTime::Execution time of func( 10) : 0.444
Fibo( 10) =       55
```

재귀함수
(Recursive function)

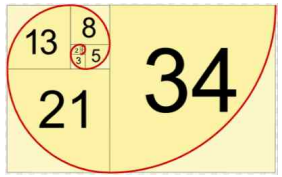
재귀함수 (Recursive Function)

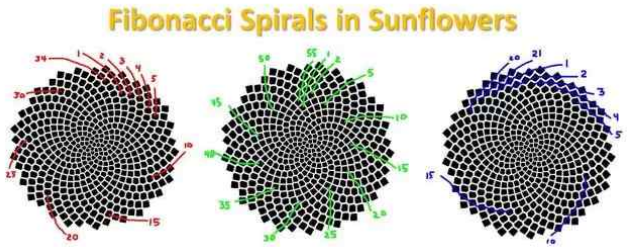
◆ 재귀함수 (recursive function)

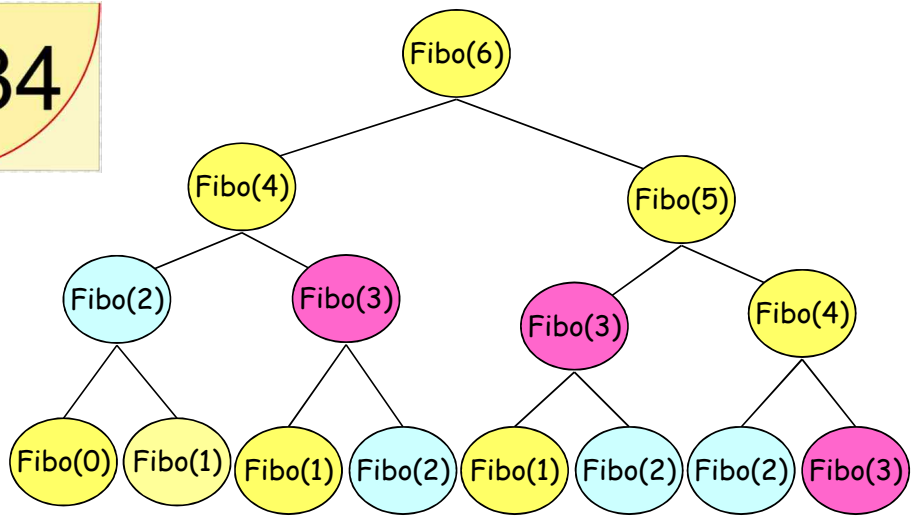
- 함수가 실행 중에 자신을 호출하는 함수 구조
- 분할 및 정복 (divide and conquer) 구조의 알고리즘 구현에 사용
- 대표적인 재귀함수 구조 : 피보나치 수열 (Fibonacci Series), 병합 정렬 (merge sort), 퀵 정렬 (quick sort)

◆ 피보나치 수열 (Fibonacci Series)

$F_0 = 0$
$F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$







재귀함수 구조의 피보나치 수열 계산

◆ Fibonacci Series

```
# Dynamic Programming of Fibonacci with memoization

memo = dict()
def dynFibo(n):
    if n in memo:
        return memo[n]
    elif n <= 1:
        memo[n] = n
        return n
    else:
        fibo_n = dynFibo(n-1) + dynFibo(n-2)
        memo[n] = fibo_n
        return fibo_n


N = int(input("inpt N for fibo : "))
for n in range(0,N+1,5):
    print("dynFibo({:3d}) : {:25}".format(n, dynFibo(n)))
```

```
inpt N for fibo : 100
dynFibo(  0) :                         0
dynFibo(  5) :                         5
dynFibo( 10) :                        55
dynFibo( 15) :                       610
dynFibo( 20) :                      6765
dynFibo( 25) :                     75025
dynFibo( 30) :                    832040
dynFibo( 35) :                   9227465
dynFibo( 40) :                 102334155
dynFibo( 45) :                1134903170
dynFibo( 50) :               12586269025
dynFibo( 55) :              139583862445
dynFibo( 60) :             1548008755920
dynFibo( 65) :            17167680177565
dynFibo( 70) :           190392490709135
dynFibo( 75) :          2111485077978050
dynFibo( 80) :         23416728348467685
dynFibo( 85) :        259695496911122585
dynFibo( 90) :       2880067194370816120
dynFibo( 95) :      31940434634990099905
dynFibo(100) :     354224848179261915075
```

재귀함수 구조 - 정수 (int)의 16진수 문자열 변환

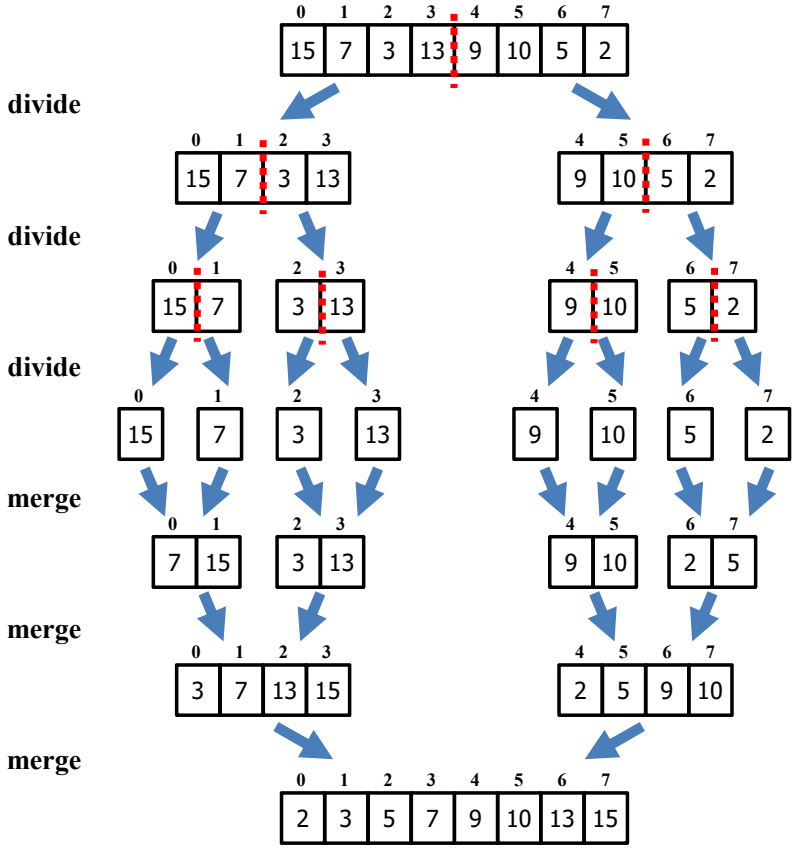
```
# Converting int into hexadecimal string

hexDeciChar = "0123456789ABCDEF"
def xtoa(hex_str, n, level):
    for i in range(level):
        print("  ", end='') # make indentation
    print("xtoa ({}, {:3}) :: ".format(hex_str, n))
    if n >= 16:
        hex_str = xtoa(hex_str, n//16, level+1)
    n = n % 16
    hex_str += str(hexDeciChar[n])
    for i in range(level):
        print("  ", end='')
    print("=> hex_str = ", hex_str)
    return hex_str

n = 0xABCD1234
level = 0
hexStr_n = "0x"
hexStr_n = xtoa(hexStr_n, n, level)
print("decimal ({}) => {}".format(n, hexStr_n))
```

```
xtoa (0x, 2882343476) ::
  xtoa (0x, 180146467) ::
    xtoa (0x, 11259154) ::
      xtoa (0x, 703697) ::
        xtoa (0x, 43981) ::
          xtoa (0x, 2748) ::
            xtoa (0x, 171) ::
              xtoa (0x,  10) ::
              => hex_str =  0xA
            => hex_str =  0xAB
          => hex_str =  0xABC
        => hex_str =  0xABCD
      => hex_str =  0xABCD1
    => hex_str =  0xABCD12
  => hex_str =  0xABCD123
=> hex_str =  0xABCD1234
decimal (2882343476) => 0xABCD1234
```

재귀함수 구조의 병합 정렬 (Merge Sort)

◆ 병합정렬 알고리즘

- 전체 알고리즘이 분할 (divide)과 병합(merge) 단계로 구성됨
- 분할 단계에서는 주어진 정렬 구간을 반복적으로 ½로 나누고, 최종적으로 2개씩의 원소가 남으면 이를 정렬
- 병합 단계에서는 정렬된 부분 구간들을 병합시키면서 정렬 기능을 수행
- 병합단계에서는 이미 부분 구간들이 정렬되어 있으므로 병합정렬이 신속하게 처리될 수 있음



재귀함수 구조의 Merge Sort 구현

```
# merge sort
def _merge(L_left, L_right):
    L_res = []
    i, j = 0, 0
    len_left, len_right = len(L_left), len(L_right)
    while i < len_left and j < len_right:
        if L_left[i] < L_right[j]:
            L_res.append(L_left[i])
            i += 1
        else:
            L_res.append(L_right[j])
            j += 1
    while (i < len_left):
        L_res.append(L_left[i])
        i += 1
    while (j < len_right):
        L_res.append(L_right[j])
        j += 1
    return L_res

def mergeSort(L): # merge_sort in increasing order
    if len(L) < 2:
        return L[:]
    else:
        middle = len(L) // 2
        L_left = mergeSort(L[:middle])
        L_right = mergeSort(L[middle:])
        return _merge(L_left, L_right)
```

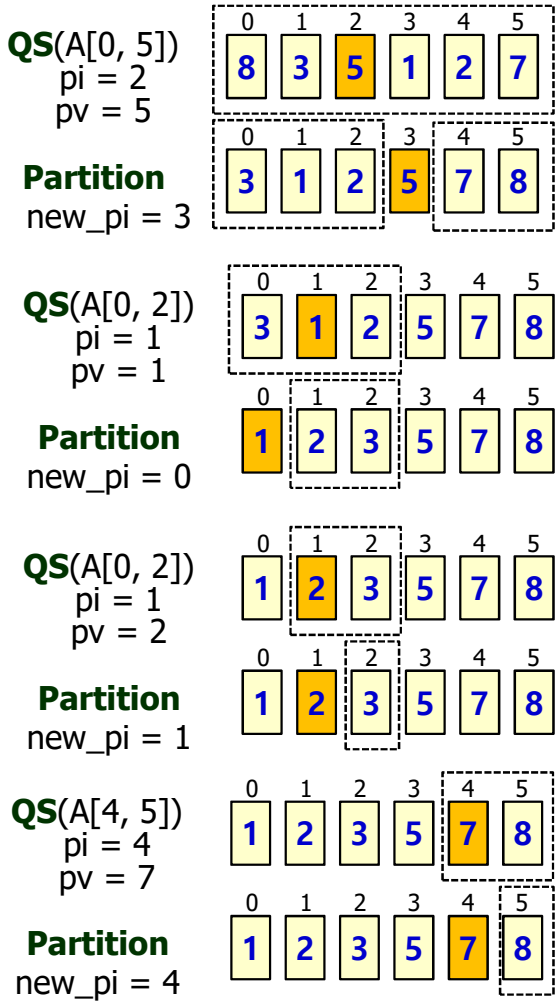
```
#---------------------------------
L = [3, 2, 1, 0, 9, 4, 7, 5, 8, 6]
print("L = ", L)
sorted_L = mergeSort(L)
print("sorted_L = ", sorted_L)
```

```
L =  [3, 2, 1, 0, 9, 4, 7, 5, 8, 6]
sorted_L =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

퀵정렬 (Quick Sorting)

◆ 분할 및 정복 (divide and conquer) 기법의 퀵 정렬

- 분할 및 정복 (divide and conquer) 방식의 알고리즘
- 탐색 구간의 중간 값을 pivot으로 선정하고, partition() 함수에서 이 pivot 보다 작은 원소들의 집합과 큰 원소들의 집합으로 분할 (pivot 원소의 위치는 변경될 수 있음)
- 분할된 각 구간에 대하여 quick_sort() 함수를 재귀함수 호출 (recursive function call)
- partition() 함수에서 pivot과의 비교 기능과 swapping 기능 수행
- partition() 함수를 사용한 분할 기능으로 탐색 구간을 ½ 정도씩으로 줄여감
- quick_sort() 함수의 재귀함수 호출에서 함수 호출의 오버헤드가 발생하며, 따라서 배열의 원소 개수가 작을 경우 선택정렬(selection sorting) 보다 늦은 성능을 가질 수 있음



재귀함수 구조의 Quick Sort 구현

```
# quickSort (1)
def _partition(arr, left, right, pi):
    pv = arr[pi]
    arr[pi], arr[right] = arr[right], arr[pi]
    npi = i = left  # npi : new pivot index
    while (i<=right-1):
        if (arr[i] <= pv):
            arr[npi], arr[i] = arr[i], arr[npi]
            npi += 1
        i += 1
    arr[npi], arr[right] = arr[right], arr[npi]
    return npi

def _quickSortLoop(arr, left, right):
    #print("quickSort", left, right)
    if (left >= right):
        return
    pi = (left + right)//2   # pivot index
    new_pi = _partition(arr, left, right, pi)
    #print("after partition : ", left, right, new_pi)
    if (left < new_pi -1):
        _quickSortLoop(arr, left, new_pi-1)
    if (new_pi + 1 < right):
        _quickSortLoop(arr, new_pi+1, right)

def quickSort(arr):
    left, right = 0, len(arr) -1 # indices of sorting region
    _quickSortLoop(arr, left, right)
```

```
# quickSort (2)

#----------------------------------
L = [3, 2, 1, 0, 9, 4, 7, 5, 8, 6]
print("L = ", L)
quickSort(L)
print("sorted L = ", L)
```

```
L =  [3, 2, 1, 0, 9, 4, 7, 5, 8, 6]
sorted L =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

제네레이터 함수, 제네레이터 수식
(Generator function, Generator expression)

제네레이터와 코루틴

◆ 제네레이터 (Generator)

- 제네레이터는 데이터를 순차적으로 생성하여 전달
- 제네레이터는 제네레이터 함수 (generator function) 또는 제네레이터 수식 (generator expression)으로 구현
- 제네레이터 함수는 return 대신 yield를 사용하여 (중간) 결과값을 전달하거나, 전달 받음

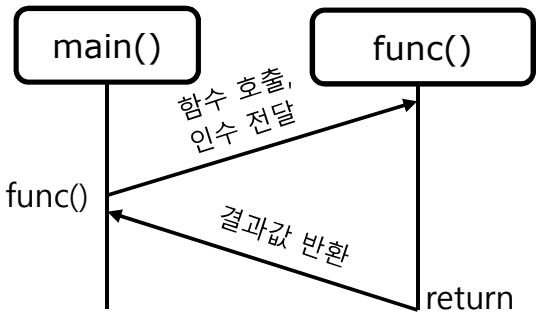
◆ 코루틴 (coroutine)

- 코루틴은 일반 함수와는 달리 여러 군데의 함수 진입점 (entry points)이 있으며, 실행 도중 잠시 정지하였다가 다시 실행을 재개 할 수 있음
- 코루틴은 대화형으로 실행할 수 있음
- 코루틴은 제네레이터 함수 또는 async를 사용하여 구현됨

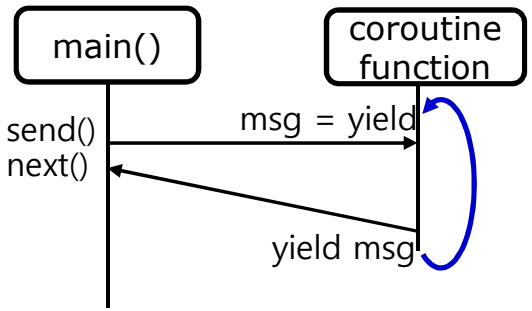
일반적인 서브루틴 호출 (subroutine call) vs. 코루틴 (coroutine)

◆ 일반적인 함수 호출 (subroutine)과 코루틴의 비교

- 서브루틴에서는 함수 호출에 대한 결과를 return으로 반환한 후, 함수가 종료됨
- 코루틴에서는 함수의 중간 결과를 yield로 전달한 후, 함수를 종료하지 않고 대기; 대입 연산자 오른쪽에 있는 yield를 사용하여 데이터를 전달 받음
- 코루틴을 호출하는 함수는 next()를 사용하여 데이터를 전달받으며, send()를 사용하여 데이터를 전달



(a) Subroutine



(b) Coroutine

제네레이터 함수 – myRange()

◆ Generator Function

```
# Generator function with yield

def myRange(n):
    i = 0
    while i < n:
        yield i #using yield instead of return
        i += 1

for i in myRange(5):
    print("returned value from myRange(n) : {:2d}".format(i))
```

```
returned value from myRange(n) :  0
returned value from myRange(n) :  1
returned value from myRange(n) :  2
returned value from myRange(n) :  3
returned value from myRange(n) :  4
```

제네레이터 함수 – myRange()

◆ myRange() with next()

```
# Generator function with yield and next

def myRange(n):
    i = 0
    while i < n:
        yield i #using yield instead of return
        i += 1
MAX_RANGE = 5
itr = myRange(MAX_RANGE)
for i in myRange(MAX_RANGE+1):
    d = next(itr)
    print("returned value from myRange(n) : {:2d}".format(d))
```

```
returned value from myRange(n) :  0
returned value from myRange(n) :  1
returned value from myRange(n) :  2
returned value from myRange(n) :  3
returned value from myRange(n) :  4
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 6
r function - myRange with yield and  next().
    d = next(itr)
StopIteration
```

제네레이터 함수 – yield를 사용한 데이터 전달

◆ 대입연산자 오른쪽의 yield를 사용한 데이터 수신

```
# Generator function that receives data from outside using (yield)

def echoMsg():
    print("Starting echoMsg() generator ....")
    try:
        while True:
            try:
                msg = yield # data received using (yield)
                print("Received message : ", msg)
            except Exception as e:
                print("Exception : ", e)
    finally:
        print("Generator function is closed now")

genMsg = echoMsg()
next(genMsg)

genMsg.send("First message")
genMsg.throw(TypeError, "Exception message !")
genMsg.send("Last message")
genMsg.close()
```

```
Starting echoMsg() generator ....
Received message :  First message
Exception :  Exception message !
Received message :  Last message
Generator function is closed now
```

사용자 정의 함수 구현

날짜 입력 함수 - inputDate()

```
# Getting date (year, month, day) in one line using split() and map()

def inputDate():
    input_date_str = input("input year month day : ")
    yr_mn_dy_strings = input_date_str.split(sep=' ')
    print("Input yr_mn_dy_strings : ", yr_mn_dy_strings)
    (year, month, day) = tuple(map(int, yr_mn_dy_strings))
    return year, month, day

# main()
while (1):
    yr, mn, dy = inputDate()
    print("Input date : year({}), month({}), day({})".format(yr, mn, dy))
    if yr == 0:
        break
```

```
input year month day : 2020 4 9
Input yr_mn_dy_strings :  ['2020', '4', '9']
Input date : year(2020), month(4), day(9)
input year month day : 1 1 1
Input yr_mn_dy_strings :  ['1', '1', '1']
Input date : year(1), month(1), day(1)
input year month day : 0 0 0
Input yr_mn_dy_strings :  ['0', '0', '0']
Input date : year(0), month(0), day(0)
```

윤년 (leap year) 여부 판단 – isLeapYear()

```
# isLeapYear()

def isLeapYear(yr):
    if (((yr%4 == 0) and (yr%100 != 0))\
        or (yr % 400) == 0):
        return True
    else:
        return False

# main()
for year in range(2000, 2026):
    if (isLeapYear(year)):
        print("Year {} is leap year".format(year))
    else:
        print("Year {} is not leap year".format(year))
```

```
Year 2000 is leap year
Year 2001 is not leap year
Year 2002 is not leap year
Year 2003 is not leap year
Year 2004 is leap year
Year 2005 is not leap year
Year 2006 is not leap year
Year 2007 is not leap year
Year 2008 is leap year
Year 2009 is not leap year
Year 2010 is not leap year
Year 2011 is not leap year
Year 2012 is leap year
Year 2013 is not leap year
Year 2014 is not leap year
Year 2015 is not leap year
Year 2016 is leap year
Year 2017 is not leap year
Year 2018 is not leap year
Year 2019 is not leap year
Year 2020 is leap year
Year 2021 is not leap year
Year 2022 is not leap year
Year 2023 is not leap year
Year 2024 is leap year
Year 2025 is not leap year
```

서기 1년 1월 1일 이후 누적된 날짜 수 계산 - getDaysFromJan01AD01()

```
# getDaysFromJan01AD01()
def inputDate():
    # refer previous function definition
def isLeapYear(yr):
    # refer previous function definition
def getDaysFromJan01AD01(yr, mn, dy):
    totalDays = 0
    daysInMonth = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    for y in range(1, yr):
        daysInYear = 366 if isLeapYear(y) else 365
        totalDays += daysInYear
    if isLeapYear(yr):
        daysInMonth[2] = 29
    for m in range(1, mn):
        totalDays += daysInMonth[m]
    totalDays += dy
    return totalDays
# main()
while(1):
    year, month, day = inputDate()
    if year == 0:
        break
    daysFromJan01AD01 = getDaysFromJan01AD01(year, month, day)
    print("Day (year({}), month({}), day({})) : Elapsed {} days from Jan01AD01"\
          .format(year, month, day, daysFromJan01AD01))
```

```
input year month day : 1 1 1
Input yr_mn_dy_strings :  ['1', '1', '1']
Day (year(1), month(1), day(1)) : Elapsed 1 days from Jan01AD01
input year month day : 2020 4 9
Input yr_mn_dy_strings :  ['2020', '4', '9']
Day (year(2020), month(4), day(9)) : Elapsed 737524 days from Jan01AD01
input year month day : 2020 12 25
Input yr_mn_dy_strings :  ['2020', '12', '25']
Day (year(2020), month(12), day(25)) : Elapsed 737784 days from Jan01AD01
input year month day : 0 0 0
Input yr_mn_dy_strings :  ['0', '0', '0']
```

지정된 날짜의 요일 계산 – getWeekDay()

```
# getWeekDay()

def inputDate():
    # refer previous function definition
def isLeapYear(yr):
    # refer previous function definition
def getDaysFromJan01AD01(yr, mn, dy):
    # refer previous function definition

WeekDayNames = ["Sunday", "Monday",\
 "Tuesday", "Wednesday", "Thursday",\
 "Friday", "Saturday"]
# main()
while(1):
    year, month, day = inputDate()
    if year == 0:
        break
    daysFromJan01AD01 = getDaysFromJan01AD01(year, month, day)
    weekDay = daysFromJan01AD01 % 7
    print("Day (year({}), month({}), day({})) : Elapsed {} days from Jan01AD01"\
          .format(year, month,day, daysFromJan01AD01))
    print("  week day = ", WeekDayNames[weekDay])
```

```
input year month day : 1 1 1
Day (year(1), month(1), day(1)) : Elapsed 1 days from Jan01AD01
  week day =  Monday
input year month day : 2020 1 1
Day (year(2020), month(1), day(1)) : Elapsed 737425 days from Jan01AD01
  week day =  Wednesday
input year month day : 2020 4 9
Day (year(2020), month(4), day(9)) : Elapsed 737524 days from Jan01AD01
  week day =  Thursday
input year month day : 2020 12 25
Day (year(2020), month(12), day(25)) : Elapsed 737784 days from Jan01AD01
  week day =  Friday
input year month day : 0 0 0
```

지정된 날짜의 달력 출력 – printMonth()

```
MonthNames = ["", "January", "February", "March", "April", "May", "June", "July",\
          "August", "September", "October", "November", "December"]
WeekDayNames = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",\
          "Friday", "Saturday"]
def printMonth(yr, mn):
    elapsedDay01 = getDaysFromJan01AD01(yr, mn, 1)
    weekDay01 = elapsedDay01 % 7
    daysInMonth = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    wkDayName = WeekDayNames[weekDay01]
    if isLeapYear(yr):
        daysInMonth[2] = 29
    #print("{}-{}-{} is {}".format(yr, mn, 1, wkDayName))
    print("\n{} of Year {}".format(MonthNames[mn], yr))
    print("============================")
    print(" SUN MON TUE WED THR FRI SAT")
    print("----------------------------")
    for i in range(weekDay01):
        print("    ", end='')
    wd = weekDay01
    for d in range(1, daysInMonth[mn]+1):
        print("{:4d}".format(d), end='')
        wd += 1
        if wd % 7 == 0:
            print() # insert new line
    print("\n============================\n")
```

```
input year month day : 1 1 1
Day (year(1), month(1), day(1)) : Elapsed 1 days from Jan01AD01
  week day =  Monday

January of Year 1
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
   1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30  31
============================

input year month day : 2020 4 9
Day (year(2020), month(4), day(9)) : Elapsed 737524 days from Jan01AD01
  week day =  Thursday

April of Year 2020
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
                   1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30
============================

input year month day : 2020 1 1
Day (year(2020), month(1), day(1)) : Elapsed 737425 days from Jan01AD01
  week day =  Wednesday

January of Year 2020
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
                   1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30  31
============================
```

날짜 입력과 달력 출력

```
# 지정된 날짜의 달력 출력

def inputDate():
    # refer previous function definition
def isLeapYear(yr):
    # refer previous function definition
def getDaysFromJan01AD01(yr, mn, dy):
    # refer previous function definition
MonthNames = ["", "January", "February", "March", "April", "May", "June", "July",\
            "August", "September", "October", "November", "December"]
WeekDayNames = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",\
            "Friday", "Saturday"]
def printMonth(yr, mn):
    # refer previous function definition
# main()
while(1):
    year, month, day = inputDate()
    if year == 0:
        break
    daysFromJan01AD01 = getDaysFromJan01AD01(year, month, day)
    weekDay = daysFromJan01AD01 % 7
    print("Day (year({}), month({}), day({})) : Elapsed {} days from Jan01AD01"\
          .format(year, month,day, daysFromJan01AD01))
    print("  week day = ", WeekDayNames[weekDay])
    printMonth(year, month)
```

```
input year month day : 1 1 1
Day (year(1), month(1), day(1)) : Elapsed 1 days from Jan01AD01
  week day =  Monday

January of Year 1
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
       1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30  31
============================

input year month day : 2020 4 9
Day (year(2020), month(4), day(9)) : Elapsed 737524 days from Jan01AD01
  week day =  Thursday

April of Year 2020
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
                   1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30
============================

input year month day : 2020 1 1
Day (year(2020), month(1), day(1)) : Elapsed 737425 days from Jan01AD01
  week day =  Wednesday

January of Year 2020
============================
 SUN MON TUE WED THR FRI SAT
----------------------------
                   1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30  31
============================
```

시간 입력 함수 - inputTime()

```
# input Time (hour, min, sec) in one line using split() and map()

def inputTime():
    input_time_str = input("input hour min sec : ")
    hr_mn_sec_strings = input_time_str.split(sep=' ')
    print("Input hr_mn_sec_strings : ", hr_mn_sec_strings)
    (hr, mn, sec) = tuple(map(int, hr_mn_sec_strings))
    return hr, mn, sec

# main()
while (1):
    hr, mn, sec = inputTime()
    print("Input time : hour({}), minute({}), second({})".format(hr, mn, sec))
    if hr == 25:
        break
```

```
input hour min sec : 12 15 10
Input hr_mn_sec_strings :  ['12', '15', '10']
Input time : hour(12), minute(15), second(10)
input hour min sec : 0 0 1
Input hr_mn_sec_strings :  ['0', '0', '1']
Input time : hour(0), minute(0), second(1)
input hour min sec : 25 1 1
Input hr_mn_sec_strings :  ['25', '1', '1']
Input time : hour(25), minute(1), second(1)
```

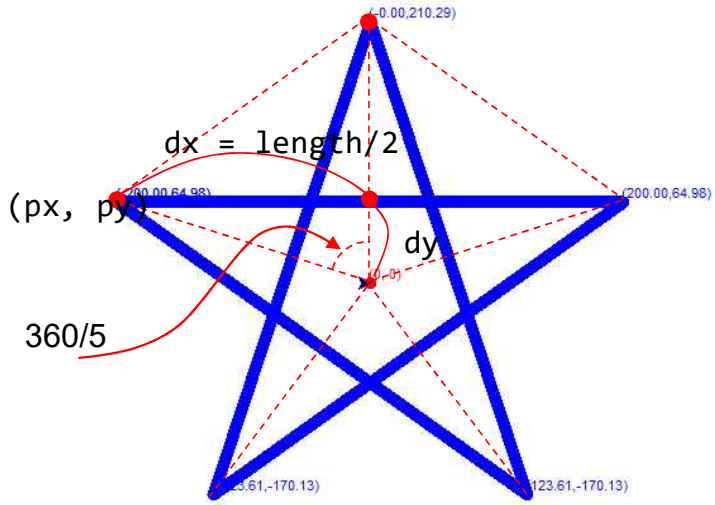
자정부터 경과된 초단위 시간 계산 - elapsedSeconds()

```
# elapsedSeconds()

def inputTime():
    # refer previous function definition

def elapsedSeconds(hr, mn, sec):
    elapsedSecs = 0
    if ((hr < 0 or hr > 23) or
        (mn < 0 or mn > 59) or
        (sec < 0 or sec > 59)):
        print("Error in time hour({}), min({}),\
          sec({})".format(hr, mn, sec))
        return 0
    for h in range(1, hr+1):
        elapsedSecs += 60 * 60
    for m in range(1, mn+1):
        elapsedSecs += 60
    elapsedSecs += sec
    return elapsedSecs

# main()
while (1):
    hr, mn, sec = inputTime()
    print("Input time : hour({}), minute({}), second({})".format(hr, mn, sec))
    if hr == 25:
        break
    elpSecs = elapsedSeconds(hr, mn, sec)
    print("Elapsed seconds from last midnight = ", elpSecs)
    secs_to_midnight = 24*60*60 - elpSecs
    print("Remaining seconds to next-midnight = ", secs_to_midnight)
```

```
input hour min sec : 0 0 0
Input time : hour(0), minute(0), second(0)
Elapsed seconds from last midnight =  0
Remaining seconds to next-midnight =  86400
input hour min sec : 0 0 1
Input time : hour(0), minute(0), second(1)
Elapsed seconds from last midnight =  1
Remaining seconds to next-midnight =  86399
input hour min sec : 0 1 0
Input time : hour(0), minute(1), second(0)
Elapsed seconds from last midnight =  60
Remaining seconds to next-midnight =  86340
input hour min sec : 1 0 0
Input time : hour(1), minute(0), second(0)
Elapsed seconds from last midnight =  3600
Remaining seconds to next-midnight =  82800
input hour min sec : 23 59 59
Input time : hour(23), minute(59), second(59)
Elapsed seconds from last midnight =  86399
Remaining seconds to next-midnight =  1
input hour min sec : 25 0 0
Input time : hour(25), minute(0), second(0)
```

draw_star()

◆ 별의 중심점이 주어질 때, 정확한 꼭지점 좌표 계산



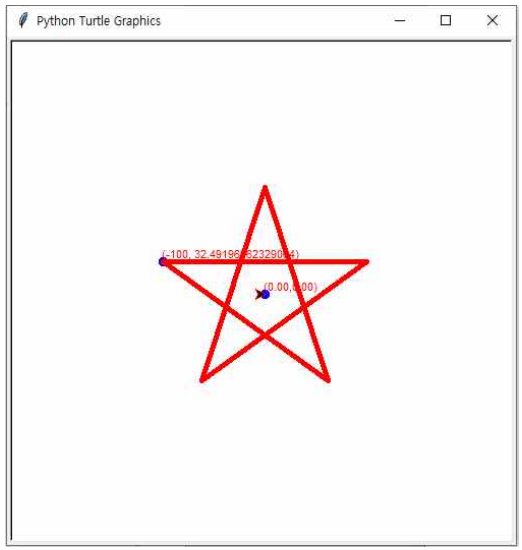
dx = length/2
px = px0 - dx
theta = 360 / 5
tan(theta) = (length/2) / dy
dy = (length/2) / tan(theta)
py = py0 + dy

```
import math
def getStartPosition_Star(length, px0, py0):
    px = px0 - length//2
    py = py0 + (length/2) / (math.tan(math.radians(360/5)))
    return px, py
```

draw_star()

```
# Simple Python Program to Draw a Star with given center position
import turtle
import math

def getStartPosition_Star(length, px0, py0):
    px = px0 - length//2
    py = py0 + (length/2) / (math.tan(math.radians(360/5)))
    return px, py

def draw_star(t, length, px0, py0, pen_color, pen_width):
    t.pencolor(pen_color)
    t.up(); t.home(); t.down()
    t.dot(10, "blue"); t.write(t.position())
    start_x, start_y =  getStartPosition_Star(length, px0, py0)
    t.width(pen_width)
    t.penup(); t.goto(start_x, start_y); t.dot(10, "blue")
    t.write((start_x, start_y)); t.pendown() #pen down to draw
    turn_angle = 2*360//5
    for i in range(5):
        t.forward(length)
        t.right(turn_angle)
    t.up(); t.home(); t.down()

# main()
turtle.setup(500, 500) #set width and height of canvas
t = turtle.Turtle()
t.shape('classic')

center_x, center_y = map(int, input("input center_x and center_y : ").split(' '))
pen_width, length = map(int, input("input pen_width and length of star : ").split(' '))
pen_color = input("pen_color of star (e.g., red, blue) = ")
draw_star(t, length, center_x, center_y, pen_color, pen_width)
```

```
input center_x and center_y : 0 0
input pen_width and length of star : 5 200
pen_color of star (e.g., red, blue) = red
```



리스트 출력 및 샘플 출력

```
# printList, printListSample (1)

import random

def printList(L, per_line = 10):
    size = len(L)
    count = 0
    while count < size:
        s = ""
        i = 0
        while (i < per_line) and (count < size):
            s += "{0:>7} ".format(L[count])
            count += 1
            i += 1
        print(s)
```

```
# printList, printListSample  (2)

def printListSample(L, per_line = 10, sample_lines = 2):
    count = 0
    size = len(L)
    for i in range(0, sample_lines):
        s = ""
        for j in range(0, per_line):
            if count > size:
                break
            s += "{0:>7} ".format(L[count])
            count += 1
        print(s)
        if count >= size:
            break
    if count < size:
        print('. . . . . .')
        if count < (size - per_line * sample_lines):
            count = size - per_line * sample_lines
        for i in range(0, sample_lines):
            s = ""
            for j in range(0, per_line):
                if count >= size:
                    break
                s += "{0:>7} ".format(L[count])
                count += 1
            print(s)
            if count >= size:
                break
```

리스트 출력 및 샘플 출력

```
# printList, printListSample (3)

# --------------
size = int(input("input size of list = "))

A = list(range(size))
random.shuffle(A)

print("List A = ")
printList(A, 10)

print("Sample of List A = ")
printListSample(A, 10, 2)
```

```
input size of list = 100
List A =
    32,     45,     10,     29,     28,     19,     84,      2,     88,     54,
    79,     94,     25,     33,      6,     17,     68,     78,     42,     41,
    26,      0,     58,     51,     50,     16,     72,     81,     12,     21,
    31,     95,     18,     86,     47,      9,     43,     52,     15,     98,
     8,     20,     55,     67,      5,     80,     75,     62,     11,     46,
    36,     90,     91,     48,     44,     96,     59,     87,     70,     14,
    22,     60,     61,     99,     23,     82,     24,     64,     65,     13,
    93,     97,     73,     63,      4,     53,     37,      1,      3,     76,
    89,     74,     40,     30,     56,      7,     85,     34,     71,     49,
    69,     57,     66,     83,     92,     35,     38,     27,     39,     77,
Sample of List A =
    32      45      10      29      28      19      84       2      88      54
    79      94      25      33       6      17      68      78      42      41
    ......
    89      74      40      30      56       7      85      34      71      49
    69      57      66      83      92      35      38      27      39      77
```

선택 정렬 및 성능 측정

```
#------ Selection Sorting
def selectionSort(L):
    size = len(L)
    for i in range(0, size-1):
        min_idx = i
        for j in range(i+1, size):
            if (L[min_idx] > L[j]):
                min_idx = j
        if (min_idx != i):
            L[min_idx], L[i] = L[i], L[min_idx]
```

```
Array Size (0 to quit) = 1000

Before Selection-Sort of A :
   278    937    525    765    594    534    173    263    810    234
   371    433    530    689    214    439    184    731    593    293
    ......
   529    621    166    718    702    489    509    472    798    519
   249    353    274    367    709    925    940    837    491    309
After Selection-Sort of A .....
     0      1      2      3      4      5      6      7      8      9
    10     11     12     13     14     15     16     17     18     19
    ......
   980    981    982    983    984    985    986    987    988    989
   990    991    992    993    994    995    996    997    998    999
Selection sorting took 0.019961833953857422 sec
```

```
# --------------
Selection_Sort_Limit = 50000
while True:
    size = int(input("Array Size (0 to quit) = "))
    if size == 0:
        break
    A = list(range(size))
    random.shuffle(A)

    if size <= Selection_Sort_Limit:
        print("\nBefore Selection-Sort of A :")
        printListSample(A, 10, 2)
        t1 = time.time()
        selectionSort(A)
        t2 = time.time()
        print("After Selection-Sort of A .....")
        printListSample(A, 10, 2)
        time_elapsed = t2 - t1
        print("Selection sorting took {} sec".format(time_elapsed))
```

정렬 알고리즘들의 성능 비교

```
# Comparisons of sorting algorithms (1)
import random, time
. . . . # printList(), printListSample()
. . . . # selectionSort(), mergeSort(), quicksort()
#-------  main()
Selection_Sort_Limit = 50000
while True:
    print("\nCompare sorting algorithms")
    size = int(input("Array Size (0 to quit) = "))
    if size == 0:
        break
    A = list(range(size))
    random.shuffle(A)

    if size <= Selection_Sort_Limit:
        print("\nBefore Selection-Sort of A :")
        printListSample(A, 10, 2)
        t1 = time.time()
        selectionSort(A)
        t2 = time.time()
        print("After Selection-Sort of A .....")
        printListSample(A, 10, 2)
        time_elapsed = t2 - t1
        print("Selection sorting took {} sec".format(time_elapsed))
```

```
Compare sorting algorithms
Array Size (0 to quit) = 10000

Before Selection-Sort of A :
   9681    1030    8904    6101    6624    2400    2812    9511    5713    9464
   2866    9235     533    8541    1935    5945    7436    4347    5067    1140
     . . . . . .
    659     445    2112     669    3910    1184    9876    5957    2799    5624
   9320    8490    6076    1478    9735    8743     432    3136    4125      19
After Selection-Sort of A .....
      0       1       2       3       4       5       6       7       8       9
     10      11      12      13      14      15      16      17      18      19
     . . . . . .
   9980    9981    9982    9983    9984    9985    9986    9987    9988    9989
   9990    9991    9992    9993    9994    9995    9996    9997    9998    9999
Selection sorting took 1.6170401573181152 sec

Before mergeSort of A :
   4386    9366    4673    8855    9439    2702    9500    2980    8448    5662
   1309    8896     396    1920     771     971    6398    8498    2921    9784
     . . . . . .
   7602    1812    8623    3971     926    6720    9318    8750    8178    6208
    758    1807    2096    6473    7504    2072    1516    4507    2847    8252
After mergeSort of A :
      0       1       2       3       4       5       6       7       8       9
     10      11      12      13      14      15      16      17      18      19
     . . . . . .
   9980    9981    9982    9983    9984    9985    9986    9987    9988    9989
   9990    9991    9992    9993    9994    9995    9996    9997    9998    9999
Merge sorting took 0.016984224319458008 sec

Before quickSort of A :
   2676    9743    5286    3749    9096    6633     609    9017     743    7081
   4724    5581    5428    2063      73    4334    8369    2856    4660    2911
     . . . . . .
   9054    8061    6046      10    6680    2780    1080    1556    2800    2952
    812    2144    6642    8854    7066    6797    5809    7021    3167    8522
After quickSort of A :
      0       1       2       3       4       5       6       7       8       9
     10      11      12      13      14      15      16      17      18      19
     . . . . . .
   9980    9981    9982    9983    9984    9985    9986    9987    9988    9989
   9990    9991    9992    9993    9994    9995    9996    9997    9998    9999
Quick sorting took 0.016954421997070312 sec
```

```
# Comparisons of sorting algorithms (2)

    # testing MergeSorting
    random.shuffle(A)
    print("\nBefore mergeSort of A :")
    printListSample(A, 10, 2)
    t1 = time.time()
    sorted_A = mergeSort(A)
    t2 = time.time()
    print("After mergeSort of A :")
    printListSample(sorted_A, 10, 2)
    time_elapsed = t2 - t1
    print("Merge sorting took {} sec".format(time_elapsed))

    # testing Quick Sorting
    random.shuffle(A)
    print("\nBefore quickSort of A :")
    printListSample(A, 10, 2)
    t1 = time.time()
    quickSort(A)
    t2 = time.time()
    print("After quickSort of A :")
    printListSample(A, 10, 2)
    time_elapsed = t2 - t1
    print("Quick sorting took {} sec".format(time_elapsed))
```

```
Compare sorting algorithms
Array Size (0 to quit) = 1000000

Before mergeSort of A :
 942778   16075  672283  102085  104689  773448  613205   56337  324521  825732
 189165  385036  475565  577649  808743  834942  789529  463694  135749  680384
     . . . . . .
  86411  399959  732447  186199   65401   91534  676236  465938  916839  149711
 711609  346723  986314  599079  772986  583839  331609  150814  880780  693165
After mergeSort of A :
      0       1       2       3       4       5       6       7       8       9
     10      11      12      13      14      15      16      17      18      19
     . . . . . .
 999980  999981  999982  999983  999984  999985  999986  999987  999988  999989
 999990  999991  999992  999993  999994  999995  999996  999997  999998  999999
Merge sorting took 2.6160330772399902 sec

Before quickSort of A :
 234092    9262  133931  809194  503454  126870  997415  444944  234024  388926
 736943  893479  370376  962724  970999  115534  330936   96200  179595  357471
     . . . . . .
 344639  585623  221242  800698  525316   70798  864108  691330  334766  775671
  97823  849932   78672  711727    9593  437408   87200  415872  652017  373555
After quickSort of A :
      0       1       2       3       4       5       6       7       8       9
     10      11      12      13      14      15      16      17      18      19
     . . . . . .
 999980  999981  999982  999983  999984  999985  999986  999987  999988  999989
 999990  999991  999992  999993  999994  999995  999996  999997  999998  999999
Quick sorting took 2.837615966796875 sec
```

Homework 5

Homework 5.1

5.1 학생 정보 튜플 리스트의 통계 분석

- 학생 10명의 정보를 포함하는 튜플 (학생이름, 학과명, 학번, (연, 월, 일), 평균성적)들을 리스트에 포함시킨 후, 파이썬 내장함수 sum(), max(), min(), len()을 사용하여 학생들의 성적 분포 (최대, 최소, 평균)를 파악하여 출력하는 함수 statistics_student_GPA() 함수를 구현하라.

(실행 예시)

```
students =
('Kim', 'EE', 12345, (2000, 12, 25), 4.0)
('Lee', 'ME', 11234, (2019, 9, 1), 4.2)
('Park', 'ICE', 10234, (2019, 3, 1), 4.3)
('Hong', 'CE', 13123, (2021, 1, 1), 4.1)
('Yoon', 'ICE', 11321, (2001, 8, 15), 4.2)
('A', 'ICE', 12321, (2000, 7, 31), 4.2)
statistics_student_GPA ::
 - L_GPAs =  [4.0, 4.2, 4.3, 4.1, 4.2, 4.2]
 - num_students = 6
 - Statistics of GPAs : Max (4.3), Min (4.0), Avg (4.166666666666667)
```

Homework 5.2

5.2 정수형 난수 리스트의 정렬 및 경과시간 측정

- 중복되지 않는 정수형 (int) 난수를 지정된 개수 (예: 10,000 ~ 1,000,000) 만큼 생성하는 함수 genBigRandList(L, n)을 구현하라. 이 함수에는 리스트 L과 리스트에 포함될 중복되지 않는 난수의 개수 n이 전달된다.
- 주어진 리스트의 첫 부분 2줄 (한 줄에 10개씩)과 마지막 2줄을 출력하는 함수 printListSample(L, per_line, sample_lines)를 구현하라.
- 주어진 리스트의 원소들을 오름차순으로 병합정렬 구조로 정렬하는 함수 mergeSort(L)와 quicksort(L) 함수를 구현하라.
- 표준입력장치로 부터 1,000,000 이상 크기의 정수형 난수 개수를 입력 받은 후, genBigRandList() 함수를 사용하여 중복되지 않는 정수형 난수 리스트를 생성하고, 이를 printListSample() 함수를 사용하여 출력하라. 이 정수형 난수 리스트를 mergeSort() 함수를 사용하여 정렬하고, 정렬된 상태를 printListSample() 함수를 사용하여 출력하라.
- 정렬된 난수 리스트를 random.shuffle() 함수를 사용하여 뒤섞은 후, quicksort() 함수를 사용하여 정렬하고, 정렬된 상태를 printListSample() 함수를 사용하여 출력하라.
- mergeSort() 함수와 quicksort() 함수의 정렬에서 걸린 시간을 각각 time 모듈의 time() 메소드를 사용하여 측정하고, 출력하라.
- (실행 예시)

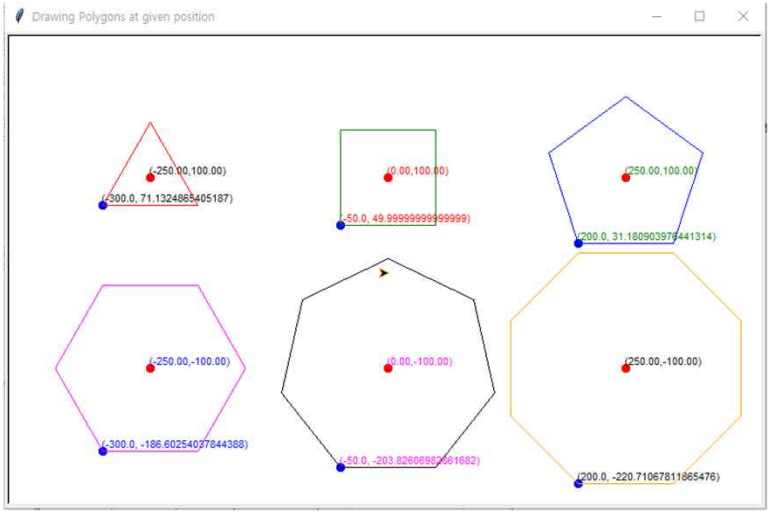
```
Compare sorting algorithms
Array Size (0 to quit) = 10000000

Before mergeSort of A :
6564306 7171990 4961746 4497550 7835062  889580 9798585 7500589 3179657 5998822
4742259 9603412 5693902 4054731 1742272 3080760 6922950 9538679 5012120 6219105
      ......
9161199 3935402 2561999 4521354  770715 2046448 1879239 3238242 4088621  918644
9002334  628597 9313044 2550788 3641686 7866920 5496259 8490096   73107 4455950
After mergeSort of A :
     0       1       2       3       4       5       6       7       8       9
    10      11      12      13      14      15      16      17      18      19
      ......
9999980 9999981 9999982 9999983 9999984 9999985 9999986 9999987 9999988 9999989
9999990 9999991 9999992 9999993 9999994 9999995 9999996 9999997 9999998 9999999
Merge sorting took 28.567585945129395 sec

Before quickSort of A :
8793703   24412 3012902 2630831 7829051 2951209 7989600 8540330 6165035 8614928
6744385 7039441 7487503 3399853 7238843 5073024  243629 2467283 3138825 5017262
      ......
3195303 8211201 9106643 2147543 7762804 8852461 2554143 4194318 1781083 8454095
1770480 1564410 3808485 2125494 5069869 3531328  205775 6498627  298181 2836630
After quickSort of A :
     0       1       2       3       4       5       6       7       8       9
    10      11      12      13      14      15      16      17      18      19
      ......
9999980 9999981 9999982 9999983 9999984 9999985 9999986 9999987 9999988 9999989
9999990 9999991 9999992 9999993 9999994 9999995 9999996 9999997 9999998 9999999
Quick sorting took 29.582677364349365 sec
```

Homework 5.3

5.3 튜플 리스트로 설정된 다각형 그리기

- 다각형 꼭지점 수 (num_vertices), 한 변의 길이 (side_length), 중심 좌표 (center_x, center_y), 색상 정보 (color)를 튜플 (num_vertices, side_length, center_x, center_y, color)로 정의하고, 이 튜플 들을 리스트 L_polygons로 선언한 후, 터틀 그래픽을 사용하여 지정된 중심 좌표에 다각형을 그리는 파이썬 프로그램을 작성하라.
- 다각형의 중심 좌표 위치에 크기 10인 붉은색 점 (dot)을 그리고, 중심 좌표값을 출력하며, turtle graphic window에 제목을 "Drawing polygons with turtle"로 설정할 것.
- 실행 예제:

```
L_polygons = [(3, 100, -250, 100, "red"), (4, 100, 0, 100, "green"), (5, 100, 250, 100, "blue"),\
              (6, 100, -250, -100, "magenta"), (7, 100, 0, -100, "black"), (8, 100, 250, -100, "orange")]
```



Homework 5.4

5.4 동적 프로그래밍 구조의 피보나치 수열 계산 - dynFibo()

- Fibonacci 수열을 신속하게 계산할 수 있도록 memo 기능을 포함하는 재귀함수 dynFibo(n)를 구성하고, 시작 (start), 끝 (end), 간격 (stride)의 정수를 입력 받은 후 해당 순서의 피보나치 수열을 출력하는 프로그램을 작성하라. 각 단계별 피보나치 수열 계산에서 이미 메모에 포함된 피보나치 수열 값은 별도로 계산하지 않고 메모의 내용을 사용하며, 이전에 계산된 적이 없은 피보나치 수열 값은 재귀함수를 사용하여 계산한 후, 이를 메모에 기록하여 두고, 계산된 결과값을 반환하도록 하라.
- 이 프로그램에서는 dynFibo(n)에서 걸린 경과 시간을 time() 함수를 사용하여 측정한 후, 계산된 피보나치 수열과 함께 출력하도록 하라. start = 50, end = 100, stride = 5로 설정하여 실행 결과를 출력하라.
- (실행 예시)

```
input start, stop, step of Fibonacci series : 50 100 5
dynFibo( 50) =               12586269025, took       0.00[milli_sec]
dynFibo( 55) =              139583862445, took       0.00[milli_sec]
dynFibo( 60) =             1548008755920, took       0.00[milli_sec]
dynFibo( 65) =            17167680177565, took       0.00[milli_sec]
dynFibo( 70) =           190392490709135, took       0.00[milli_sec]
dynFibo( 75) =          2111485077978050, took       0.00[milli_sec]
dynFibo( 80) =         23416728348467685, took       0.00[milli_sec]
dynFibo( 85) =        259695496911122585, took       0.00[milli_sec]
dynFibo( 90) =       2880067194370816120, took       0.00[milli_sec]
dynFibo( 95) =      31940434634990099905, took       0.00[milli_sec]
dynFibo(100) =     354224848179261915075, took       0.00[milli_sec]
```

Homework 5.5

5.5 행렬의 출력, 연산 프로그램

- 행렬을 표현하는 2차원 리스트를 행렬의 이름 (문자열)과 함께 전달받아 출력하는 함수 printMtrx(name, M)을 작성하라.
- 2차원 리스트 2개를 전달받아 행렬의 덧셈을 계산하여 결과를 반환하는 함수 addMtrx(A, B), 뺄셈을 계산하여 결과를 반환하는 함수 subMtrx(A, B), 그리고 곱셈을 계산하여 결과를 반환하는 함수 mulMtrx(A, B)를 작성하라.
- 다음 기능 시험 프로그램을 사용하여 결과를 출력하라.

(실행 예시)

```
# --------------------------------
A = [[1,2,3,4], [5,6,7,8], [9,0,1,2]]
B = [[1,0,0,0], [0,1,0,0], [0,0,1,0]]
C = [[1,0,0], [0,1,0], [0,0,1], [0,0,0]]

printMtrx("A", A)
printMtrx("B", B)
printMtrx("C", C)
D = addMtrx(A, B)
printMtrx("A + B", D)
E = subMtrx(A, B)
printMtrx("A - B", E)
F = mulMtrx(A, C)
printMtrx("A * C", F)
```

```
A =
  1  2  3  4
  5  6  7  8
  9  0  1  2
B =
  1  0  0  0
  0  1  0  0
  0  0  1  0
C =
  1  0  0
  0  1  0
  0  0  1
  0  0  0
A + B =
  2  2  3  4
  5  7  7  8
  9  0  2  2
A - B =
  0  2  3  4
  5  5  7  8
  9  0  0  2
A * C =
  1  2  3
  5  6  7
  9  0  1
```