

스마트 모빌리티 프로그래밍

Ch 6. 파이썬 모듈/패키지, 사용자 정의 모듈/패키지



영남대학교 정보통신공학과
교수 김 영 탁

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

Outline

- ◆ 파이썬 모듈과 패키지 개요
- ◆ `sys`, `os`, `os.path`, `shutil` 모듈
- ◆ `array`, `time`, `calendar`, `bisect`, `copy` 모듈
- ◆ `math`, `random` 모듈
- ◆ `sortedcontainer`, `queue` 모듈
- ◆ `regular expression (re)` 모듈
- ◆ 사용자 정의 모듈 (user-defined module)
- ◆ 사용자 정의 패키지 (user-defined package)
- ◆ 사용자 정의 모듈/패키지와 `__name__` 변수



파이썬 모듈과 패키지

파이썬 모듈과 패키지 (1)

◆ 모듈 (Module)

- 파이썬 프로그램 실행 (processing)의 기본 단위
- 파이썬 프로그램 소스 파일 (*.py) : 변수, 함수, 클래스를 포함
- 파이썬 바이트 코드 (*.pyc)
- 파이썬 C-확장 동적 설치 라이브러리 (DLL) 파일 (*.pyd)

◆ 패키지 (Package)

- 모듈과 서브 패키지들을 계층적으로 구성
- 파이썬 표준 라이브러리는 C 기반의 내장 모듈, 다양한 응용 모듈 및 패키지로 구성되어 있음

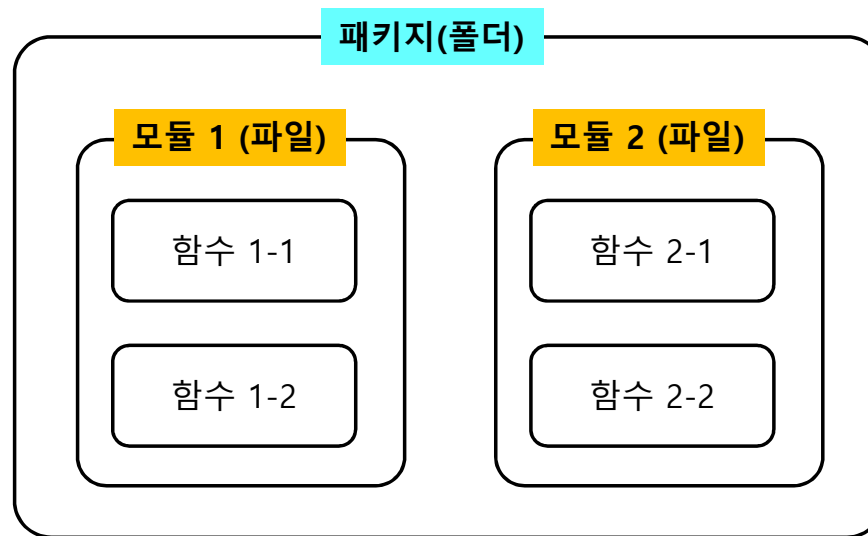
◆ 사용자 모듈 및 패키지

- 사용자가 직접 모듈과 패키지를 구성하여 편리하게 사용할 수 있게 함

파이썬 모듈과 패키지 (2)

◆ 패키지 (package), 모듈 (module)

- 패키지와 서브 패키지는 디렉토리 (폴더) 단위로 관리
- 모듈은 파일 단위로 관리되며, 내부에 클래스와 함수(메소드)를 포함



모듈과 패키지의 설치 (1)

◆ import

- import 명령어는 모듈과 패키지를 메모리에 설치
- 이미 설치되었던 모듈을 (수정한 후) 다시 설치하기 위해서는 `imp.reload()` 함수를 사용

◆ import 명령어 사용 양식

- `import <module_name_list>`
- `import <module_name> as <new_name>`
- `from <module_name> import <name_list>`
- `from <module_name> import *`

◆ del

- 설치된 모듈을 삭제하기 위해서는 `del` 명령어를 사용

파이썬 표준 모듈 (1)

기능 분류	파이썬 표준 모듈
Data Types	array, types, copy, enum, datetime, calendar, collections, heapq, bisect, weakref
Numeric and mathematical	numbers, math, cmath, decimal, fractions, random, statistics
Binary data service	struct, codecs
Text processing	re (regular expression), readline, string
File and directory access	pathlib, os.path, fileinput, stat, filecmp, tempfile
Data persistence	pickle, copyreg, shelve, marshal, dbm, sqlite3
Data compression and archiving	zlib, gzip, bz2, lzma, zipfile, tarfile
Generic operating system services	os, io, time, argparse, getopt, logging, getpass, curses, platform, errno
Concurrent execution	threading, multiprocessing, concurrent, subprocess, sched, queue, _thread, _dummy_thread, dummy_threading
Networking and inter-process communication	asyncio, socket, ssl, select, selector, asyncore, asynchat, signal, mmap
Internet data handling	email, json, mailcap, mailbox, mimetypes, base64, binhex, binascii, quopri, uu
Internet protocols and support	webbrowser, cgi, http, bottle



파이썬 표준 모듈 (2)

기능 분류	파이썬 표준 모듈
Multimedia services	audioop, aifc, sunau, wave, chunk, colorsys, imghdr, sndhdr, ossaudiodev, pyaudio
Internationalization	gettext, locale
Program framework	turtle, cmd, shlex
Graphical User Interface (GUI)	tkinter, tkinter.ttk, tkinter.tix, tkinter.scrolledtext
Python runtime services	sys, sysconfig, builtins, __main__, warnings, dataclasses, abc, atexit, traceback, __future__, gc (garbage collector), inspect, site
Python language services	parser, token, keyword, tokenize
MS windows specific services	Msilib (Microsoft installer library), Msvcrt (MS VC++ runtime), winreg (Windows registry access), winsound
Linux specific services	posix, pwd (print working directory), grp, tty, pty, fcntl, pipes, resource, syslog

파이썬 확장 패키지 (extension packages)

◆ Python extension packages

Python Extension Package	설명 및 관련 methods
NumPy SciPy	과학 계산을 위한 라이브러리로서 다차원 배열을 처리하는데 필요한 여러 유용한 기능을 제공 ndarray, universal function
Matplotlib	파이썬에서 데이터를 차트나 그래프 등으로 데이터 시각화(Data Visualization) 시켜주는 라이브러리 패키지 라인 플롯, 바 차트, 파이차트, 히스토그램, Box Plot, Scatter Plot 등을 비롯하여 다양한 차트와 플롯 스타일 등을 지원 Matplotlib.org 갤러리 웹페이지에서 다양한 샘플 차트를 볼 수 있음
Pillow	PIL (Python Imaging Library)-friendly image processing
OpenCV	영상처리 기능 제공
pandas	데이터 분석(Data Analysis) 기능 제공
TensorFlow	인공지능, 딥러닝 분야에 많이 사용되는 확장패키지
keras	파이썬으로 구현된 딥러닝 라이브러리, TensorFlow위에서 딥러닝 모델을 쉽게 구성할 수 있게 함

파이선 표준 모듈

- **sys, os, os.path, shutil, array** 모듈

sys 모듈의 사용 예 – builtin_module_names, platform, version, prefix, exec_prefix, executable

◆ sys 모듈의 사용 예

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import sys
>>> sys.builtin_module_names
(' _abc', ' _ast', ' _bisect', ' _blake2', ' _codecs', ' _codecs_cn', ' _codecs_hk', ' _codecs_iso2022', ' _codecs_jp', ' _codecs_kr', ' _codecs_tw', ' _collections', ' _contextvars', ' _csv', ' _datetime', ' _functools', ' _heapq', ' _imp', ' _io', ' _json', ' _locale', ' _lsp', ' _operator', ' _pickle', ' _random', ' _sha1', ' _sha256', ' _sha3', ' _sha512', ' _signal', ' _sre', ' _stat', ' _statistics', ' _string', ' _struct', ' _symtable', ' _thread', ' _tracemalloc', ' _warnings', ' _weakref', ' _winapi', ' _xxsubinterpreters', ' array', ' atexit', ' audioop', ' binascii', ' builtins', ' cmath', ' errno', ' faulthandler', ' gc', ' itertools', ' marshal', ' math', ' mmap', ' msvcrt', ' nt', ' parser', ' sys', ' time', ' winreg', ' xxsubtype', ' zlib')
>>> sys.platform
'win32'
>>> sys.version
'3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)]'
>>> sys.prefix
'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32'
>>> sys.exec_prefix
'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32'
>>> sys.executable
'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32\\pythonw.exe'
>>>
```

sys 모듈의 사용 예 – path, float_info

◆ path, float_info

```
>>> sys.path
['', 'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32\\Lib\\i
dlelib', 'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python37-32', '
C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32\\python38.zip
', 'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32\\DLLs', '
C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32\\lib', 'C:\\U
sers\\Administrator\\AppData\\Local\\Programs\\Python\\Python38-32', 'C:\\Users\\Admi
nistrator\\AppData\\Local\\Programs\\Python\\Python38-32\\lib\\site-packages']
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.22507
38585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.2204
46049250313e-16, radix=2, rounds=1)
>>> sys.float_info.dig
15
>>>
```

os 모듈 - 메소드

◆ os 모듈 메소드

os module method	설명
os.access(path, mode)	지정된 경로 (path)를 지정된 모드로 접근할 수 있는지 확인 mode = os.F_OK, os.R_OK, os.W_OK, os.X_OK
os.chdir(path)	지정된 경로의 디렉토리로 이동
os.getcwd()	현재 방문중인 current working directory의 경로를 반환
os.listdir(path='.')	지정된 경로의 디렉토리에 있는 폴더와 파일들의 이름들을 반환
os.mkdir(path)	지정된 경로에 디렉토리를 생성
os.remove(path)	지정된 경로의 파일을 삭제
os.rename(src, dst)	src 이름의 파일 또는 폴더를 dst 이름으로 변경
os.rmdir(path)	지정된 경로의 디렉토리를 삭제
os.walk(top, topdown=True)	top으로 지정된 디렉토리에 포함된 파일과 디렉토리를 탐색
os.startfile(path[, operation])	지정된 경로의 파일에 대하여 지정된 조작 (operation: open, print, edit)을 수행 만약 operation이 지정되어 있지 않으면 'open' 실행
os.system(command)	지정된 명령어 (e.g., calc, mspaint)를 수행

os 모듈 – getcwd(), chdir(), walk()

```
# os module - getcwd(), chdir(), walk()

import os
cwd_str = os.getcwd()
print("os.getcwd() = ", cwd_str)
os.chdir("C:/MyPyPackage")
cwd_str = os.getcwd()
print("after os.chdir('C:/MyPyPackage'), os.getcwd() = ", cwd_str)
print("os.listdir() : \n", os.listdir())
os.chdir("./mySorting")
print("os.chdir('./mySorting')")
cwd_str = os.getcwd()
print("after os.chdir('./mySorting'), os.getcwd() = ", cwd_str)

print("os.walk() : ")
for t in os.walk('.'):
    print(t)
```

```
os.getcwd() = C:\MyPyPackage\TextBook - 2019\ch 7 Python Module, Package\7.2.2 os, os.path
after os.chdir('C:/MyPyPackage'), os.getcwd() = C:\MyPyPackage
os.listdir() :
['desktop.ini', 'myAlgorithms', 'myArray', 'myFibonacci', 'myGraph', 'mySorting', 'myTree'
, 'Spyder', 'Test_myPyPackage', 'TextBook - 2019', '__init__.py', '__pycache__']
os.chdir('./mySorting')
after os.chdir('./mySorting'), os.getcwd() = C:\MyPyPackage\mySorting
os.walk() :
('.', ['__pycache__'], ['mergeSort.py', 'quickSort.py', 'selectionSort.py', '__init__.py'])
('.\__pycache__', [], ['mergeSort.cpython-37.pyc', 'quickSort.cpython-37.pyc', 'selectionS
ort.cpython-37.pyc', '__init__.cpython-37.pyc'])
```



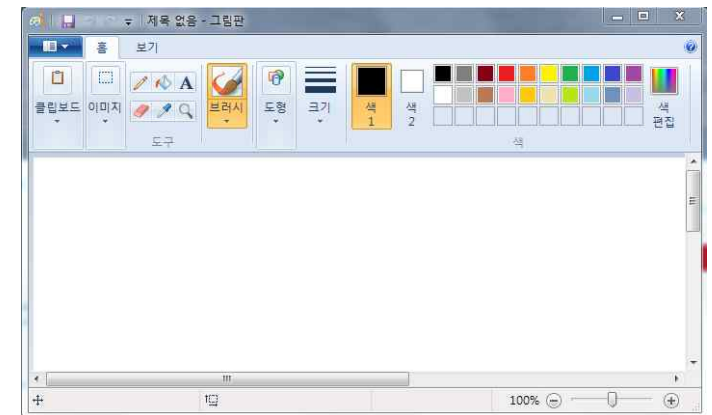
os 모듈 – system()

◆ system() 함수 사용 예

```
# os module - getcwd(), chdir(), walk()

import os
commands = ['calc', 'mspaint']

for i in range(len(commands)):
    print("performing os.system({})".format(commands[i]))
    os.system(commands[i])
    input("hit any key to continue")
```



os.path 모듈

◆ os.path 모듈

os.path module method	설 명
os.path.exists()	파일이 존재하는지 확인
os.path.isfile()	파일인지 확인
os.path.isdir()	디렉토리인지 확인
os.path.getsize()	파일 또는 디렉토리의 크기를 확인
os.path.join()	운영체제 (OS)의 지정 형식에 따라 경로를 결합

File and Directory Handling with functions of os.path (1)

```
import os
import os.path
import shutil # shell utility for file copy

test_dir = "C:/tmp/pyTest"
file_1 = "C:/tmp/pyTest/test_file_1.txt"
file_2 = "C:/tmp/pyTest/test_file_2.txt"
if os.path.exists(test_dir) == False:
    os.mkdir(test_dir)

if os.path.exists(file_1) == True:
    print("File already exists, so delete it now.")
    os.remove(file_1)
else:
    print("{} is not existing.".format(file_1))
    print("Creating {}".format(file_1))
    with open(file_1, 'w') as f1:
        f1.write("Test file 1.")
    file_size = os.path.getsize(file_1)
    print("File size of {} = {}".format(file_1, file_size))
    f1.close()
    print("Contents in File {}".format(file_1))
    with open(file_1, 'r') as f1:
        for line in f1:
            print(line)
    f1.close
```

```
C:/tmp/pyTest/test_file_1.txt is not existing.
Creating C:/tmp/pyTest/test_file_1.txt
File size of C:/tmp/pyTest/test_file_1.txt = 12
Contents in File C:/tmp/pyTest/test_file_1.txt
Test file 1.
```

```
Copy from C:/tmp/pyTest/test_file_1.txt to C:/tmp/pyTest/test_file_2.txt
File size of C:/tmp/pyTest/test_file_2.txt = 43
Contents in File C:/tmp/pyTest/test_file_2.txt
Test file 1..... Appended lines in file 2.
```

```
Files in C:/tmp/pyTest :
C:/tmp/pyTest\myFact.py
C:/tmp/pyTest\test_file_1.txt
C:/tmp/pyTest\test_file_2.txt
```

```
Delete both test files and directory
```



File and Directory Handling with functions of os.path (2)

```
print("\nCopy from {} to {}".format(file_1, file_2))
shutil.copy(file_1, file_2)
with open(file_2, 'a') as f2:
    f2.write("..... Appended lines in file 2.")
file_size = os.path.getsize(file_2)
print("File size of {} = {}".format(file_2, file_size))
print("Contents in File {}".format(file_2))
with open(file_2, 'r') as f2:
    for line in f2:
        print(line)
f2.close

print("\nFiles in {} :".format(test_dir))
for dirName, subDirList, fnames in os.walk(test_dir):
    for fname in fnames:
        print(os.path.join(dirName, fname))

print("\nDelete both test files and directory")
os.remove(file_1)
os.remove(file_2)
```

```
C:/tmp/pyTest/test_file_1.txt is not existing.
Creating C:/tmp/pyTest/test_file_1.txt
File size of C:/tmp/pyTest/test_file_1.txt = 12
Contents in File C:/tmp/pyTest/test_file_1.txt
Test file 1.
```

```
Copy from C:/tmp/pyTest/test_file_1.txt to C:/tmp/pyTest/test_file_2.txt
File size of C:/tmp/pyTest/test_file_2.txt = 43
Contents in File C:/tmp/pyTest/test_file_2.txt
Test file 1..... Appended lines in file 2.
```

```
Files in C:/tmp/pyTest :
C:/tmp/pyTest/myFact.py
C:/tmp/pyTest/test_file_1.txt
C:/tmp/pyTest/test_file_2.txt
```

```
Delete both test files and directory
```



shutil 모듈

◆ shutil 모듈

shutil 함수	설 명
shutil.copyfile(src, dst)	파일 src의 내용을 파일 dst로 복사. dst는 파일 이름으로 한정됨
shutil.copy(src, dst)	파일 src를 dst 파일 또는 디렉토리에 복사
shutil.copytree(src, dst)	src 디렉토리 산하에 있는 모든 파일들과 디렉토리들을 dst로 복사

array 모듈 (1)

◆ array type code

배열 자료형코드	C 프로그램 자료형	파이썬 자료형	원소 크기 (바이트)
'b'	signed cha	int	1
'B'	unsigned char	int	1
'u'	wchar_t	unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	4
'I'	unsigned int	int	4
'l'	signed long	int	4
'L'	unsigned long	int	4
'q'	signed double long	int	8
'Q'	unsigned double long	int	8
'f'	float	float	4
'd'	double	double	8

array 모듈 (2)

◆ array 클래스 메소드

배열 클래스 메소드	설 명
array.append(x)	x를 배열에 첨가
array.byteswap()	배열의 바이트들의 위치를 교환 (swap)
array.count(x)	배열에서 x의 발생 빈도를 파악
array.extend(iterable)	iterable에 있는 각 항목을 배열에 삽입
array.frombytes(s)	바이트 (s)에 있는 바이트들을 배열에 삽입
array.fromfile(f, n)	파일 f에 있는 항목 n개를 배열에 삽입
array.fromlist(list)	리스트 list에 있는 항목을 배열에 삽입
array.fromstring(s)	문자열 s에 있는 바이트들을 배열에 삽입
array.fromunicode(s)	유니코드 s에 있는 바이트들을 배열에 삽입
array.index(x)	x가 위치한 곳의 인덱스를 반환
array.insert(i, x)	x를 인덱스 i의 위치에 삽입
array.pop([i])	array.pop() : 배열의 마지막 원소를 반환하며, 그 원소를 삭제 array.pop(i) : i-번째 원소를 반환하며, 그 원소를 삭제
array.remove(x)	배열의 첫 부분에서 x 개의 원소를 삭제
array.reverse()	배열의 원소들의 순서를 거꾸로 변경
array.tobytes()	배열의 원소들을 바이트로 반환
array.tofile(f)	배열의 원소들을 파일 f에 바이트로 저장
array.tolist()	배열 원소들을 리스트로 반환
array.tounicode()	배열 원소들을 유니코드 문자열로 반환 (array.typecode = 'u')



array 모듈 (3)

```
# Python standard module - array
```

```
from array import *
A = array('i')
print("type(A) = ", type(A))
print("A.typecode = ", A.typecode)
print("A.itemsize = ", A.itemsize)
A.append(1)
print("After A.append(1),")
print(" A = ", A)
A.extend([2, 3, 4, 5])
print("After A.extend([2, 3, 4, 5]),")
print(" A = ", A)
```

```
for i in range(len(A)):
    print("A[{0:2d}] = {0:2d}".format(i, A[i]))
```

```
type(A) = <class 'array.array'>
A.typecode = i
A.itemsize = 4
After A.append(1),
    A = array('i', [1])
After A.extend([2, 3, 4, 5]),
    A = array('i', [1, 2, 3, 4, 5])
A[ 0] = 1
A[ 1] = 2
A[ 2] = 3
A[ 3] = 4
A[ 4] = 5
```

array 모듈 (4)

```
# Python standard module - array
```

```
from array import *
A = array('i', range(5))
print("A = ", A)
print("A+A = ", A+A)
print("A*3 = ", A*3)
print("A.pop() = ", A.pop())
print("After A.pop(), A = ", A)
```

```
B = array('d', [1.0, 2.0, 3.14])
print("B = ", B)
```

```
C = array('b', [49, 50, 51, 52])
print("C = ", C)
C_B = C.tobytes()
print("C_B = ", C_B)
```

```
D = array('i', b'1234')
print("D = ", D)
D_B = D.tobytes()
print("D_B = ", D_B)
D_L = D.tolist()
print("D_L = ", D_L)
```

```
A = array('i', [0, 1, 2, 3, 4])
A+A = array('i', [0, 1, 2, 3, 4, 0, 1, 2, 3, 4])
A*3 = array('i', [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4])
A.pop() = 4
After A.pop(), A = array('i', [0, 1, 2, 3])
B = array('d', [1.0, 2.0, 3.14])
C = array('b', [49, 50, 51, 52])
C_B = b'1234'
D = array('i', [875770417])
D_B = b'1234'
D_L = [875770417]
```

파이선 표준 모듈
- time, calendar, bisect, copy 모듈

time 모듈 (1)

◆ time 모듈의 메소드

메소드	설 명
time.time()	기준시점인 (Jan. 1, 1970, 00:00:00 sec)으로부터 경과된 초단위 시간을 float형 실수로 반환
time.asctime([t])	gmtime() 또는 localtime()에 의해 생성된 시간을 나타내는 구조체정보를 문자열로 변환
time.ctime([secs])	초단위의 경과 시간을 문자열로 변환 time.ctime()과 time.asctime(time.localtime())는 현재 시간을 문자열로 변환
time.gmtime([secs])	UTC의 기준시점으로 부터의 경과시간을 반환
time.localtime([secs])	UTC의 기준시점으로 부터의 현지 경과시간을 반환
time.mktime(t)	localtime()의 반대 기능을 수행
time.sleep(secs)	secs로 설정된 초단위 시간 동안 실행을 중지
time.time_ns()	time() 함수와 동일하나 나노 초 (nano-second) 단위의 시간을 반환
time.process_time()	현재 프로세스의 시스템 및 사용자 CPU 시간을 반환 sleep()에 의하여 중단되었던 시간은 포함하지 않음
time.process_time_ns()	process_time()과 동일하나 나노 초 (nano-second) 단위의 경과 시간을 반환
time.perf_counter()	performance counter의 값을 반환하며, 짧은 경과시간을 매우 정밀하게 측정할 수 있게 함
time.perf_counter_ns()	perf_counter()와 동일하나 나노 초 (nano-second) 단위로 측정할 수 있도록 함

time 모듈 (2)

◆ time.struct_time 속성

index	attributes	description
0	tm_year	year
1	tm_mon	month, [1, 12]
2	tm_mday	day, [1, 31]
3	tm_hour	hour, [0, 23]
4	tm_min	minute, [0, 59]
5	tm_sec	second, [0, 59]
6	tm_wday	week day, 0: Monday, 1: Tuesday,
7	tm_yday	year day, [1, 366]
8	tm_isdst	0, 1, -1, daylight saving

time 모듈 (3)

◆ time 모듈 사용 예

```
# using time module (1)
```

```
import time
```

```
t1 = time.time()
pc_ns_1 = time.perf_counter_ns()
print("time.time() : {} [sec]".format(t1))
print("time.perf_counter_ns() : {} [nsec]".format(pc_ns_1))
print("time.gmtime() :\n", time.gmtime())
```

```
print("sleeping 10 secs .....")
time.sleep(10)
```

```
t2 = time.time()
pc_ns_2 = time.perf_counter_ns()
print("time.time() : {} [sec]".format(t2))
print("elapsed time = {} [sec]".format(t2 - t1))
print("time.perf_counter_ns() : {} [nsec]".format(pc_ns_2))
print("elapsed perf_counter_ns : {} [nsec]".format(pc_ns_2 - pc_ns_1))
print("time.process_time() : {} [sec]".format(time.process_time()))
print("time.process_time_ns() : {} [ns]".format(time.process_time_ns()))
```

```
time.time() : 1610069653.9439895 [sec]
time.perf_counter_ns() : 371197900 [nsec]
time.gmtime() :
    time.struct_time(tm_year=2021, tm_mon=1, tm_mday=8, tm_hour=1, tm_min=34,
tm_sec=13, tm_wday=4, tm_yday=8, tm_isdst=0)
sleeping 10 secs .....
time.time() : 1610069663.9701953 [sec]
elapsed time = 10.026205778121948 [sec]
time.perf_counter_ns() : 10396829100 [nsec]
elapsed perf_counter_ns : 10025631200 [nsec]
time.process_time() : 0.109375 [sec]
time.process_time_ns() : 109375000 [ns]
time.asctime() : Fri Jan  8 10:34:23 2021
t = time.localtime() :
    time.struct_time(tm_year=2021, tm_mon=1, tm_mday=8, tm_hour=10, tm_min=34,
tm_sec=23, tm_wday=4, tm_yday=8, tm_isdst=0)
t.tm_year : 2021
t.tm_mon : 1
t.tm_mday : 8
```

```
# using time module (2)

ascii_t = time.asctime()
print("time.asctime() : ", ascii_t)
t = time.localtime()
print("t = time.localtime() :\n ", t)
print("t.tm_year : ", t.tm_year)
print("t.tm_mon : ", t.tm_mon)
print("t.tm_mday : ", t.tm_mday)
```

```
time.time() : 1610069653.9439895 [sec]
time.perf_counter_ns() : 371197900 [nsec]
time.gmtime() :
    time.struct_time(tm_year=2021, tm_mon=1, tm_mday=8, tm_hour=1, tm_min=34,
tm_sec=13, tm_wday=4, tm_yday=8, tm_isdst=0)
sleeping 10 secs .....
time.time() : 1610069663.9701953 [sec]
elapsed time = 10.026205778121948 [sec]
time.perf_counter_ns() : 10396829100 [nsec]
elapsed perf_counter_ns : 10025631200 [nsec]
time.process_time() : 0.109375 [sec]
time.process_time_ns() : 109375000 [ns]
time.asctime() : Fri Jan  8 10:34:23 2021
t = time.localtime() :
    time.struct_time(tm_year=2021, tm_mon=1, tm_mday=8, tm_hour=10, tm_min=34,
tm_sec=23, tm_wday=4, tm_yday=8, tm_isdst=0)
t.tm_year : 2021
t.tm_mon : 1
t.tm_mday : 8
```



calendar 모듈

```
# using calendar module
```

```
import calendar
weekday_names = ["Monday", "Tuesday", "Wednesday", \
                 "Thursday", "Friday", "Saturday", "Sunday"]
print("calendar : ", calendar)
for yr in range(2010, 2031):
    isLY = calendar.isleap(yr)
    print("calendar.isleap({0}) : {1}".format(yr, isLY ))
for dy in range(1, 20):
    wkD =calendar.weekday(2021, 1, dy)
    print("(2021, 1, {:2d}) : {}".format(dy, weekday_names[wkD]))

calendar.prcal(2021)
```

```
calendar.isleap(2010) : False (2021, 1, 1) : Friday
calendar.isleap(2011) : False (2021, 1, 2) : Saturday
calendar.isleap(2012) : True (2021, 1, 3) : Sunday
calendar.isleap(2013) : False (2021, 1, 4) : Monday
calendar.isleap(2014) : False (2021, 1, 5) : Tuesday
calendar.isleap(2015) : False (2021, 1, 6) : Wednesday
calendar.isleap(2016) : True (2021, 1, 7) : Thursday
calendar.isleap(2017) : False (2021, 1, 8) : Friday
calendar.isleap(2018) : False (2021, 1, 9) : Saturday
calendar.isleap(2019) : False (2021, 1, 10) : Sunday
calendar.isleap(2020) : True (2021, 1, 11) : Monday
calendar.isleap(2021) : False (2021, 1, 12) : Tuesday
calendar.isleap(2022) : False (2021, 1, 13) : Wednesday
calendar.isleap(2023) : False (2021, 1, 14) : Thursday
calendar.isleap(2024) : True (2021, 1, 15) : Friday
calendar.isleap(2025) : False (2021, 1, 16) : Saturday
calendar.isleap(2026) : False (2021, 1, 17) : Sunday
calendar.isleap(2027) : False (2021, 1, 18) : Monday
calendar.isleap(2028) : True (2021, 1, 19) : Tuesday
calendar.isleap(2029) : False
calendar.isleap(2030) : False
```

2021

January							February							March						
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
					1	2	3													
4	5	6	7	8	9	10	8	9	10	11	12	13	14	8	9	10	11	12	13	14
11	12	13	14	15	16	17	15	16	17	18	19	20	21	15	16	17	18	19	20	21
18	19	20	21	22	23	24	22	23	24	25	26	27	28	22	23	24	25	26	27	28
25	26	27	28	29	30	31								29	30	31				

April							May							June												
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su						
					1	2	3	4						1	2						1	2	3	4	5	6
5	6	7	8	9	10	11	3	4	5	6	7	8	9	7	8	9	10	11	12	13						
12	13	14	15	16	17	18	10	11	12	13	14	15	16	14	15	16	17	18	19	20						
19	20	21	22	23	24	25	17	18	19	20	21	22	23	21	22	23	24	25	26	27						
26	27	28	29	30			24	25	26	27	28	29	30	28	29	30										
							31																			

July							August							September										
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su				
					1	2	3	4						1						1	2	3	4	5
5	6	7	8	9	10	11	2	3	4	5	6	7	8	6	7	8	9	10	11	12				
12	13	14	15	16	17	18	9	10	11	12	13	14	15	13	14	15	16	17	18	19				
19	20	21	22	23	24	25	16	17	18	19	20	21	22	20	21	22	23	24	25	26				
26	27	28	29	30	31		23	24	25	26	27	28	29	27	28	29	30							
							30	31																

October							November							December													
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su							
					1	2	3						1	2	3	4	5						1	2	3	4	5
4	5	6	7	8	9	10	8	9	10	11	12	13	14	6	7	8	9	10	11	12							
11	12	13	14	15	16	17	15	16	17	18	19	20	21	13	14	15	16	17	18	19							
18	19	20	21	22	23	24	22	23	24	25	26	27	28	20	21	22	23	24	25	26							
25	26	27	28	29	30	31	29	30						27	28	29	30	31									



bisect 모듈

◆ bisect 모듈의 함수

bisect 모듈 함수	설 명
bisect.bisect_left(L, x)	정렬된 순서를 유지하도록 리스트 L에 x를 삽입할 위치를 찾아 반환. 만약 리스트에 x가 이미 있으면, x의 앞쪽 (왼쪽)을 반환
bisect.bisect_right(L, x)	정렬된 순서를 유지하도록 리스트 L에 x를 삽입할 위치를 찾아 반환. 만약 리스트에 x가 이미 있으면, x의 뒤쪽 (오른쪽)을 반환
bisect.bisect(L, x)	리스트 L로부터 x의 위치를 찾아 반환
bisect.insort_left(L, x)	리스트 L에 x를 정렬된 순서로 삽입
bisect.insort_right(L, x)	리스트 L에 x를 순서에 맞추어 삽입. x의 기존 항목 다음에 삽입
bisect.insort(L, x)	리스트 L에 x를 순서에 맞추어 삽입. 만약 리스트 L에 이미 x가 포함되어 있으면 기존 항목 다음에 x를 삽입. 삽입 후에도 정렬 상태 유지

bisect 모듈 (1)

```
# using bisect module
```

```
import bisect
A = []
B = [5,3,1,9,0,8,2,7,4,6]
print("B = ", B)
for x in B:
    pos = bisect.bisect(A, x)
    bisect.insort(A, x)
    print("bisect.insort(A, {})",\
          " at position {}".format(x, pos))
    print("A : ", A)

def binsearch(L, value):
    pos = bisect.bisect_left(L, value)
    if (pos == len(L)) or (L[pos] != value):
        return False
    else:
        return pos
for x in B:
    p = binsearch(A, x)
    print("{} is at {} in A".format(x, p))
```

```
B = [5, 3, 1, 9, 0, 8, 2, 7, 4, 6]
bisect.insort(A, 5) at position 0
A = [5]
bisect.insort(A, 3) at position 0
A = [3, 5]
bisect.insort(A, 1) at position 0
A = [1, 3, 5]
bisect.insort(A, 9) at position 3
A = [1, 3, 5, 9]
bisect.insort(A, 0) at position 0
A = [0, 1, 3, 5, 9]
bisect.insort(A, 8) at position 4
A = [0, 1, 3, 5, 8, 9]
bisect.insort(A, 2) at position 2
A = [0, 1, 2, 3, 5, 8, 9]
bisect.insort(A, 7) at position 5
A = [0, 1, 2, 3, 5, 7, 8, 9]
bisect.insort(A, 4) at position 4
A = [0, 1, 2, 3, 4, 5, 7, 8, 9]
bisect.insort(A, 6) at position 6
A = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
5 is at 5 in A
3 is at 3 in A
1 is at 1 in A
9 is at 9 in A
0 is at 0 in A
8 is at 8 in A
2 is at 2 in A
7 is at 7 in A
4 is at 4 in A
6 is at 6 in A
```



bisect 모듈 (2)

```
# bisect with breakpoints
import bisect

def grade(score, breakpoints, grades):
    i = bisect.bisect(breakpoints, score)
    return grades[i]

Breakpoints = [59.5, 69.5, 79.5, 89.5]
Scores = [55, 80, 75, 88, 90, 100]
Grades = 'FDCBA'

grades = [grade(score, Breakpoints, Grades) for score in Scores]
print("Breakpoints : ", Breakpoints)
print("Scores : ", Scores)
print("grade of each score : ", grades)
score_grade = list(zip(Scores, grades))
print("score_grade : ", score_grade)

Breakpoints : [59.5, 69.5, 79.5, 89.5]
Scores : [55, 80, 75, 88, 90, 100]
grade of each score : ['F', 'B', 'C', 'B', 'A', 'A']
score_grade : [(55, 'F'), (80, 'B'), (75, 'C'), (88, 'B'), (90, 'A'), (100, 'A')]
```



copy 모듈

```
# copy module
```

```
import copy
```

```
A = [1, 2]
```

```
B = [3, 4]
```

```
X = [A, B]
```

```
Y = copy.copy(X)
```

```
print("X = ", X)
```

```
print("Y = ", Y)
```

```
print("A is Y[0] : ", A is Y[0])
```

```
print("id(A)={}, id(B)={}, id(X)={}, id(Y)={}"\  
      .format(id(A), id(B), id(X), id(Y)))
```

```
print("id(X[0])={}, id(X[1])={}, id(Y[0])={}, id(Y[1])={}"\  
      .format(id(X[0]), id(X[1]), id(Y[0]), id(Y[1])))
```

```
A[0], B[1] = 10, 40 # update both X and Y
```

```
print("X = ", X)
```

```
print("Y = ", Y)
```

```
Z = copy.deepcopy(X)
```

```
print("Z = ", Z)
```

```
print("id(X[0])={}, id(X[1])={}, id(Z[0])={}, id(Z[1])={}"\  
      .format(id(X[0]), id(X[1]), id(Z[0]), id(Z[1])))
```

```
A[0], B[1] = 15, 45 # update only at X
```

```
print("X = ", X)
```

```
print("Z = ", Z)
```

```
X = [[1, 2], [3, 4]]  
Y = [[1, 2], [3, 4]]  
A is Y[0] : True  
id(A)=14655208, id(B)=14654632, id(X)=14655368, id(Y)=14659816  
id(X[0])=14655208, id(X[1])=14654632, id(Y[0])=14655208, id(Y[1])=14654632  
X = [[10, 2], [3, 40]]  
Y = [[10, 2], [3, 40]]  
Z = [[10, 2], [3, 40]]  
id(X[0])=14655208, id(X[1])=14654632, id(Z[0])=44427784, id(Z[1])=14654728  
X = [[15, 2], [3, 45]]  
Z = [[10, 2], [3, 40]]
```



파이선 표준 모듈 - math, random 모듈

math 모듈

분류	math 모듈 함수	설 명
상수	math.pi	원주율
	math.e	자연상수
통계	math.max()	시퀀스 자료형의 최대값
	math.min()	시퀀스 자료형의 최솟값
절대값 근사값	math.abs(x)	절대값 (정수형, integer)
	math.fabs()	절대값 (실수형, float)
	math.round()	근사값 (approximation)
	math.round(x, n)	근사값 (소숫점 n자리에서 반올림)
	math.ceil(x)	x보다 큰 정수 중에서 가장 작은 정수
	math.floor(x)	x보다 작은 정수 중에서 가장 큰 정수
지수, 로그	math.pow(a, b)	지수 (a^b)
	math.exp(x)	지수 e^x
	math.log(x, base)	밑수 base의 로그
	math.log10(x)	밑수 10의 로그
팩토리얼	math.factorial(x)	팩토리얼 (factorial)
제곱근	math.sqrt(x)	제곱근 (square root)
삼각함수	math.sin(x)	sin(x), x는 라디안 (radian) 단위
	math.cos(x)	cos(x), x는 라디안 (radian) 단위
	math.tan(x)	tan(x), x는 라디안 (radian) 단위
	math.asin(x)	arc sine, x는 라디안 (radian) 단위
	math.acos(x)	arc cosine, x는 라디안 (radian) 단위
각도	math.degrees(x)	라디안 단위의 x를 도 (degree) 단위로 변환
	math.radians(x)	도 (degree) 단위의 x를 라디안 단위로 변환



math 모듈 – math.pi, math.e, max(), min()

◆ math 모듈의 기본 상수 및 숫자 데이터 통계

```
# math module - constants, min(), max()
```

```
from math import *
```

```
print("pi = ", pi) # math.pi
```

```
print("e = ", e) # math.e
```

```
L = [-0.1, 85.7, 75, -5.5, 8.7]
```

```
print("L = ", L)
```

```
print("min(L) = ", min(L))
```

```
print("max(L) = ", max(L))
```

```
pi = 3.141592653589793
e = 2.718281828459045
L = [-0.1, 85.7, 75, -5.5, 8.7]
min(L) = -5.5
max(L) = 85.7
```

math 모듈 – abs(), fabs(), round(), ceil(), floor()

◆ math 모듈 – abs(), fabs(), round(), ceil(), floor()

```
# math module - abs(), fabs(), round(), round(x, n), ceil(), floor()

from math import *

for n in range(-10, 11):
    x = n / 10
    print("x ({:5.2f}) : abs(x) ({:5.2f}), fabs({:5.2f}),\
        round(x)({:3}), ceil(x)({:3}), floor(x)({:3})"\
        .format(x, abs(x), fabs(x), round(x), ceil(x), floor(x)))
```

```
x (-1.00) : abs(x) ( 1.00), fabs( 1.00), round(x) (-1), ceil(x) (-1), floor(x) (-1)
x (-0.90) : abs(x) ( 0.90), fabs( 0.90), round(x) (-1), ceil(x) ( 0), floor(x) (-1)
x (-0.80) : abs(x) ( 0.80), fabs( 0.80), round(x) (-1), ceil(x) ( 0), floor(x) (-1)
x (-0.70) : abs(x) ( 0.70), fabs( 0.70), round(x) (-1), ceil(x) ( 0), floor(x) (-1)
x (-0.60) : abs(x) ( 0.60), fabs( 0.60), round(x) (-1), ceil(x) ( 0), floor(x) (-1)
x (-0.50) : abs(x) ( 0.50), fabs( 0.50), round(x) ( 0), ceil(x) ( 0), floor(x) (-1)
x (-0.40) : abs(x) ( 0.40), fabs( 0.40), round(x) ( 0), ceil(x) ( 0), floor(x) (-1)
x (-0.30) : abs(x) ( 0.30), fabs( 0.30), round(x) ( 0), ceil(x) ( 0), floor(x) (-1)
x (-0.20) : abs(x) ( 0.20), fabs( 0.20), round(x) ( 0), ceil(x) ( 0), floor(x) (-1)
x (-0.10) : abs(x) ( 0.10), fabs( 0.10), round(x) ( 0), ceil(x) ( 0), floor(x) (-1)
x ( 0.00) : abs(x) ( 0.00), fabs( 0.00), round(x) ( 0), ceil(x) ( 0), floor(x) ( 0)
x ( 0.10) : abs(x) ( 0.10), fabs( 0.10), round(x) ( 0), ceil(x) ( 1), floor(x) ( 0)
x ( 0.20) : abs(x) ( 0.20), fabs( 0.20), round(x) ( 0), ceil(x) ( 1), floor(x) ( 0)
x ( 0.30) : abs(x) ( 0.30), fabs( 0.30), round(x) ( 0), ceil(x) ( 1), floor(x) ( 0)
x ( 0.40) : abs(x) ( 0.40), fabs( 0.40), round(x) ( 0), ceil(x) ( 1), floor(x) ( 0)
x ( 0.50) : abs(x) ( 0.50), fabs( 0.50), round(x) ( 0), ceil(x) ( 1), floor(x) ( 0)
x ( 0.60) : abs(x) ( 0.60), fabs( 0.60), round(x) ( 1), ceil(x) ( 1), floor(x) ( 0)
x ( 0.70) : abs(x) ( 0.70), fabs( 0.70), round(x) ( 1), ceil(x) ( 1), floor(x) ( 0)
x ( 0.80) : abs(x) ( 0.80), fabs( 0.80), round(x) ( 1), ceil(x) ( 1), floor(x) ( 0)
x ( 0.90) : abs(x) ( 0.90), fabs( 0.90), round(x) ( 1), ceil(x) ( 1), floor(x) ( 0)
x ( 1.00) : abs(x) ( 1.00), fabs( 1.00), round(x) ( 1), ceil(x) ( 1), floor(x) ( 1)
```

math 모듈 – pow, exp, log, log10

◆ pow(), exp(), log(), log10()

```
# math module - pow(), exp(), log(), log10()

from math import *

for n in range(10):
    print("n={0:2} : pow (10,{0:2}) = ({1:5.0}), exp({0:2}) = ({2:8.2f}),\
          log(exp({0:2}) = ({3:4}), log10(pow(10,{0:2})) = ({4:4})"\
          .format(n, pow(10, n), exp(n), log(exp(n)), log10(pow(10, n))))
```

```
n= 0 : pow (10, 0) = (1e+00), exp( 0) = ( 1.00), log(exp( 0) = ( 0.0), log10(pow(10, 0)) = ( 0.0)
n= 1 : pow (10, 1) = (1e+01), exp( 1) = ( 2.72), log(exp( 1) = ( 1.0), log10(pow(10, 1)) = ( 1.0)
n= 2 : pow (10, 2) = (1e+02), exp( 2) = ( 7.39), log(exp( 2) = ( 2.0), log10(pow(10, 2)) = ( 2.0)
n= 3 : pow (10, 3) = (1e+03), exp( 3) = (20.09), log(exp( 3) = ( 3.0), log10(pow(10, 3)) = ( 3.0)
n= 4 : pow (10, 4) = (1e+04), exp( 4) = (54.60), log(exp( 4) = ( 4.0), log10(pow(10, 4)) = ( 4.0)
n= 5 : pow (10, 5) = (1e+05), exp( 5) = (148.41), log(exp( 5) = ( 5.0), log10(pow(10, 5)) = ( 5.0)
n= 6 : pow (10, 6) = (1e+06), exp( 6) = (403.43), log(exp( 6) = ( 6.0), log10(pow(10, 6)) = ( 6.0)
n= 7 : pow (10, 7) = (1e+07), exp( 7) = (1096.63), log(exp( 7) = ( 7.0), log10(pow(10, 7)) = ( 7.0)
n= 8 : pow (10, 8) = (1e+08), exp( 8) = (2980.96), log(exp( 8) = ( 8.0), log10(pow(10, 8)) = ( 8.0)
n= 9 : pow (10, 9) = (1e+09), exp( 9) = (8103.08), log(exp( 9) = ( 9.0), log10(pow(10, 9)) = ( 9.0)
```

math 모듈 – sqrt(), pow()

◆ sqrt(), pow()

```
# math module - sqrt(), pow()

from math import *

for n in range(10):
    pow_n_2 = pow(n, 2)
    sqrt_pow_n_2 = sqrt(pow_n_2)
    print("n={0:2} : pow ({0:2}, 2) = {1:5}, sqrt(pow({0:2}, 2)) = {2:6.2f}" \
          .format(n, pow(n, 2), sqrt(pow(n, 2))))
```

```
n= 0 : pow ( 0, 2) = 0.0, sqrt(pow( 0, 2)) = 0.00
n= 1 : pow ( 1, 2) = 1.0, sqrt(pow( 1, 2)) = 1.00
n= 2 : pow ( 2, 2) = 4.0, sqrt(pow( 2, 2)) = 2.00
n= 3 : pow ( 3, 2) = 9.0, sqrt(pow( 3, 2)) = 3.00
n= 4 : pow ( 4, 2) = 16.0, sqrt(pow( 4, 2)) = 4.00
n= 5 : pow ( 5, 2) = 25.0, sqrt(pow( 5, 2)) = 5.00
n= 6 : pow ( 6, 2) = 36.0, sqrt(pow( 6, 2)) = 6.00
n= 7 : pow ( 7, 2) = 49.0, sqrt(pow( 7, 2)) = 7.00
n= 8 : pow ( 8, 2) = 64.0, sqrt(pow( 8, 2)) = 8.00
n= 9 : pow ( 9, 2) = 81.0, sqrt(pow( 9, 2)) = 9.00
```



math 모듈 – radians(), sin(), cos()

◆ radians(), sin(), cos()

Example usages of Python math module

```
import math
PI = math.pi
print("{:>10s} {:>10s} {:>10s} {:>10s}".format("d(degree)", "x(radian)", "sin(x)", "cos(x)"))
for d in range(0, 361, 30):
    x = math.radians(d) # convert degree into radian; x = d * PI / 180
    sin_x = math.sin(x)
    cos_x = math.cos(x)
    print("{:10.2f} {:10.2f} {:10.2f} {:10.2f}".format(d, x, sin_x, cos_x))
```

d(degree)	x(radian)	sin(x)	cos(x)
0.00	0.00	0.00	1.00
30.00	0.52	0.50	0.87
60.00	1.05	0.87	0.50
90.00	1.57	1.00	0.00
120.00	2.09	0.87	-0.50
150.00	2.62	0.50	-0.87
180.00	3.14	0.00	-1.00
210.00	3.67	-0.50	-0.87
240.00	4.19	-0.87	-0.50
270.00	4.71	-1.00	-0.00
300.00	5.24	-0.87	0.50
330.00	5.76	-0.50	0.87
360.00	6.28	-0.00	1.00



random 모듈

◆ random 모듈 method

분류	난수 모듈의 메소드	설 명
seed 설정	random.seed(a=None, version=2)	난수 발생기를 초기화시킴. 만약 a=None으로 설정되어 있으면 현재 시스템 시간을 seed로 사용함
range 구간내의 난수 생성	random.randrange(stop) random.randrange(start, stop[, step])	(start, stop, step) 범위의 난수를 생성 start가 설정되어 있지 않으면 0의 기본값 사용 step이 설정되어 있지 않으면 1의 기본값 사용
	random.randint(a, b)	(a, b+1) 범위의 정수형 난수 생성
	random.random()	[0.0, 1.0) 범위에서 균일 확률 분포의 난수를 생성
	random.uniform(a, b)	[a, b] or [b, a] 범위에서 균일 확률 분포의 난수를 생성
	random.gauss(mu, sigma) random.normalvariate(mu, sigma)	평균이 mu이며 표준편차가 sigma인 정규분포 (Gaussian, Normal distribution) 난수를 생성
	random.lognormvariate(mu, sigma)	평균이 mu이며 표준편차가 sigma인 Log-Gaussian 분포의 난수를 생성
샘플 선정	random.choice(seq)	주어진 시퀀스 seq에서 임의로 선정하여 반환
	random.sample(population, k)	주어진 시퀀스 population으로부터 k 개를 임의로 선정하여 반환
뒤섞기	random.shuffle(x[, random])	주어진 시퀀스 x를 뒤섞어 줌 random은 [0.0, 1.0] 구간의 난수를 사용하여, 기본값으로 random()을 사용



random 모듈 – random(), randrange(), uniform(), gauss()

```
# random module - random(), randrange(), uniform(), gauss()

import random
print("random.random() : ", random.random())
    #uniform distribution random value in [0.0, 1.0)
print("random.random() : ", random.random())
    #uniform distribution random value in [0.0, 1.0)
print("random.randrange(10) : ", random.randrange(10))
    # integer random value in range(start, stop, step)

print("random.uniform(1, 10) : ", random.uniform(1, 10))
    # uniform distribution random value in [1, 10]
print("random.uniform(10, 1) : ", random.uniform(10, 1))
    # uniform distribution random value in [10, 1]
print("random.gauss(0, 1) : ", random.gauss(0, 1))
    # gauss distribution random value with avg 0, sigma 1
print("random.gauss(0, 1) : ", random.gauss(0, 1))
    # gauss distribution random value with avg 0, sigma 1
```

```
random.random() : 0.03195189052355407
random.random() : 0.9110925583896621
random.randrange(10) : 0
random.uniform(1, 10) : 7.635759899909132
random.uniform(10, 1) : 1.380212916650418
random.gauss(0, 1) : 1.2026981623460038
random.gauss(0, 1) : 1.1423929815462734
```



random 모듈 – choice(), shuffle(), sample()

◆ random – choice(), shuffle(), sample()

```
# random module - choice, shuffle, sample
```

```
import random
print("random.choice('abcdefg') : ", random.choice('abcdefg'))
print("random.choice('abcdefg') : ", random.choice('abcdefg'))
items = [1, 2, 3, 4, 5]
random.shuffle(items)
print("items : ", items)

print("random.sample('abcdefg', 3) : ", random.sample('abcdefg', 3))
print("random.sample('abcdefg', 3) : ", random.sample('abcdefg', 3))
print("random.sample([1, 2, 3, 4, 5], 3) : ", random.sample([1, 2, 3, 4, 5], 3))
print("random.sample([1, 2, 3, 4, 5], 3) : ", random.sample([1, 2, 3, 4, 5], 3))
```

```
random.choice('abcdefg') : c
random.choice('abcdefg') : a
items : [1, 4, 2, 5, 3]
random.sample('abcdefg', 3) : ['f', 'b', 'd']
random.sample('abcdefg', 3) : ['d', 'f', 'c']
random.sample([1, 2, 3, 4, 5], 3) : [5, 4, 1]
random.sample([1, 2, 3, 4, 5], 3) : [3, 2, 1]
```



파이선 표준 모듈
– **sortedcontainers, queue** 모듈

SortedContainers 모듈

◆ SortedContainers 모듈

sortedcontainers 모듈 함수	설 명
SortedList()	정렬이 유지되는 리스트
SortedDict()	정렬이 유지되는 dict
SortedSet()	정렬이 유지되는 집합 (set)

sortedcontainers (1) – SortedList()

```
# Sorted containers
# for this program, sortedcontainers module must be installed !!
# if sortedcontainers module is not installed, install it firstly.
# > python -m pip install --upgrade sortedcontainers

from sortedcontainers import SortedList
L = ['z', 'y', 'x', 'i', 'j']
print("L : ", L)
sL = SortedList(L)
print("sL = SortedList(L) = ", end='')
print(sL)
A = ['c', 'b', 'a']
print("Adding elements in {} into SL ...".format(A))
for c in A:
    sL.add(c)
print("After additionally appending elements in {} :".format(A) )
print("Updated sL : ", sL)
```

```
L : ['z', 'y', 'x', 'i', 'j']
sL = SortedList(L) = SortedList(['i', 'j', 'x', 'y', 'z'])
Adding elements in ['c', 'b', 'a'] into SL ...
After additionally appending elements in ['c', 'b', 'a'] :
Updated sL : SortedList(['a', 'b', 'c', 'i', 'j', 'x', 'y', 'z'])
```



sortedcontainers (2) – SortedDict()

```
# sortedcontainers - SortedDict()
# for this program, sortedcontainers module must be installed !!
# if sortedcontainers module is not installed, install it firstly.
# > python -m pip install --upgrade sortedcontainers

from sortedcontainers import SortedDict

sorted_dict = SortedDict({'c': 3, 'a': 1, 'b': 2})
print("type(sorted_dict) = ", type(sorted_dict))
print("sorted_dict = ", sorted_dict)
x = sorted_dict.popitem(index=-1)
print("sorted_dict.popitem(index=-1) = ", x)
print("after sorted_dict.popitem(index=-1), sorted_dict = \n", sorted_dict)
y = sorted_dict.popitem()
print("sorted_dict.popitem() = ", y)
print("after sorted_dict.popitem(), sorted_dict = \n", sorted_dict)
```

```
type(sorted_dict) = <class 'sortedcontainers.sorteddict.SortedDict'>
sorted_dict = SortedDict({'a': 1, 'b': 2, 'c': 3})
sorted_dict.popitem(index=-1) = ('c', 3)
after sorted_dict.popitem(index=-1), sorted_dict =
SortedDict({'a': 1, 'b': 2})
sorted_dict.popitem() = ('b', 2)
after sorted_dict.popitem(), sorted_dict =
SortedDict({'a': 1})
```



sortedcontainers (3) – SortedSet()

```
# sortedcontainers - SortedSet()
# Note) if sortedcontainers module is not installed, install it firstly.
# > python -m pip install --upgrade sortedcontainers
```

```
from sortedcontainers import SortedSet
```

```
input_str = "gfedcba"
print("input_str = ", input_str)
sorted_set = SortedSet(input_str)

print("type(sorted_set) = ", type(sorted_set))
print("sorted_set = ", sorted_set)
print("sorted_set.bisect_left('c') = ", sorted_set.bisect_left('c'))
```

```
input_str = gfedcba
type(sorted_set) = <class 'sortedcontainers.sortedset.SortedSet'>
sorted_set = SortedSet(['a', 'b', 'c', 'd', 'e', 'f', 'g'])
sorted_set.bisect_left('c') = 2
```



queue module

◆ queue 모듈 클래스 및 예외 상황

큐 모듈 클래스 및 예외상황	설 명
class queue.Queue (maxsize=0)	FIFO 큐 클래스 생성자 만약 maxsize=0이면 큐 크기에 제한없음
class queue.LifoQueue (maxsize=0)	LIFO 큐 (Stack) 클래스의 생성자 만약 maxsize=0이면 큐 크기에 제한없음
class queue.PriorityQueue (maxsize=0)	우선순위 큐 (priority queue) 클래스의 생성자 만약 maxsize=0이면 큐 크기에 제한없음 (priority, item) 튜플로 삽입 및 추출 우선순위 (priority) 값이 작으면 우선 순위가 높음
exception queue.Empty	큐가 비어있을 때 get() 이나 get_nowait() 호출이 발생하면 queue.Empty 예외상황이 발생됨
exception queue.Full	큐가 가득차 있을 때 put() 또는 put_nowait() 호출이 발생하면 queue.Full 예외상황이 발생됨

queue module

◆ queue 메소드 (1)

큐 메소드	설 명
Queue.qsize()	큐의 크기를 반환
Queue.empty()	큐가 비어있으면 True, 비어있지 않으면 False
Queue.full()	큐가 가득 차 있으면 True, 가득 차 있지 않으면 False
Queue.put(item, block=True, timeout=None)	큐에 item을 삽입 i) block=True, timeout=None: 큐에 빈 공간이 생길 때 까지 대기 ii) block=True, timeout >0: timeout 까지 빈 공간이 생기는 것을 기다리며, 시간이 초과하면 예외상황을 발생 시킴 iii) block=False: 빈 공간이 없으면 기다리지 않고 즉시 예외 상황을 발생
Queue.put_nowait(item)	큐에 item을 삽입하며 만약 빈 공간이 없으면 기다리지 않고 즉시 예외 상황을 발생, Queue.put(item, False)와 동일
Queue.get(block=True, timeout=None)	큐로부터 item을 추출/삭제하여 반환 i) block=True, timeout=None: item이 생길 때 까지 대기 ii) block=True, timeout >0: timeout 까지 item이 생기는 것을 기다리며, 시간이 초과하면 예외 상황을 발생 시킴 iii) block=False: 추출할 item이 없는 경우 기다리지 않고 즉시 예외상황을 발생 시킴

queue module

◆ queue 메소드 (2)

큐 메소드	설 명
Queue.get_nowait()	Queue.get(False)와 동일
Queue.task_done()	이전에 큐에 넣어진 태스크가 완료되었음을 통보. get() 메소드로 가져온 각각의 태스크를 수행한 후에 task_done()을 사용하여 큐에 알려줌
Queue.join()	큐에 포함된 모든 항목이 처리될 때까지 대기

queue 모듈 (1) – FIFO queue

```
# usage of FIFO queue module
```

```
import queue
q1 = queue.Queue(3)
print("q1 : ", q1)
q1.put('a')
q1.put('b')
q1.put('c')
print("after inserting 3 items, q1.full() : ", q1.full())
try:
    q1.put('d', False)
except queue.Full:
    print("q1 is full now !")
finally:
    print("q1.qsize() : ", q1.qsize())

print("q1.get() : ", q1.get())
print("q1.get() : ", q1.get())
print("q1.get() : ", q1.get())
try:
    q1.get(False)
except queue.Empty:
    print("q1 is empty now !")
finally:
    print("q1.qsize() : ", q1.qsize())
```

```
q1 : <queue.Queue object at 0x029FEEB0>
after inserting 3 items, q1.full() : True
q1 is full now !
q1.qsize() : 3
q1.get() : a
q1.get() : b
q1.get() : c
q1 is empty now !
q1.qsize() : 0
```



queue 모듈 (2) – LIFO queue

```
# queue module - LifoQueue

import queue
lifo_q = queue.LifoQueue(5)
L = ['a', 'b', 'c', 'd', 'e']
print("lifo_q : ", lifo_q)
print("inserting elements in to lifo_Q : ", L)
for n in range(len(L)):
    lifo_q.put(L[n])

print("after inserting {} items lifo_q.full() : "\
      .format(len(L)), lifo_q.full())

M = []
for n in range(len(L)):
    x = lifo_q.get()
    M.append(x)
    #print("{} ".format(x), end="")
print("dequeued elements : ", M)
```

```
lifo_q : <queue.LifoQueue object at 0x029FEEB0>
inserting elements in to lifo_Q : ['a', 'b', 'c', 'd', 'e']
after inserting 5 items lifo_q.full() : True
dequeued elements : ['e', 'd', 'c', 'b', 'a']
```

queue 모듈 (3) – Priority queue

```
# queue module - PriorityQueue
```

```
import queue
SIZE = 5
priority_q = queue.PriorityQueue(SIZE)
print("priority_q : ", priority_q)
pri_elements = {'a':30, 'b':10, 'c':20, 'd':5, 'e':15} # {item:pri}
#keys = list(pri_elements.keys())
keys = ['a', 'b', 'c', 'd', 'e']
for n in range(len(pri_elements)):
    key = keys[n]
    pri = pri_elements[key]
    priority_q.put((pri, key))

print("after inserting 5 items, priority_q.full() : ", priority_q.full())
for n in range(SIZE):
    entry = priority_q.get()
    print("priority_q.get() : ", entry)
```

```
priority_q : <queue.PriorityQueue object at 0x029FFEB0>
after inserting 5 items, priority_q.full() : True
priority_q.get() : (5, 'd')
priority_q.get() : (10, 'b')
priority_q.get() : (15, 'e')
priority_q.get() : (20, 'c')
priority_q.get() : (30, 'a')
```



정규 표현식 (regular expression, re) 모듈

정규 표현식 (regular expression) re 모듈

◆ 정규 표현식

- 복잡한 문자열을 처리하여 일정한 양식 (정규 표현식)에 따라 구성되어 있는 핵심 정보 (예 : 전화번호, E-mail 주소, 주민등록번호, 주소)를 추출할 수 있게 함
- 메타 문자 (meta character) 를 사용하여 문자열/문자 집합의 매칭, 숫자/숫자 집합의 매칭, 특수 기호/문자가 포함된 문자열의 매칭 등을 간결하게 표현
- 정규식을 사용한 문자열 검색(match, search, findall, finditer)을 쉽게 정의할 수 있음
- 예)
 - 주민등록번호(6자리의 숫자 - 7자리의 숫자) : `"(\d{6})[-]\d{7}"`
 - E-mail 주소 (user@host.com): `"([\w.-]+)@([\w.-]+)"`
- 참고자료
 - <https://wikidocs.net/1669>

정규식 (Regular Expression) 모듈

◆ Regular Expression에서 사용되는 메타 문자 (1)

메타 문자	의 미
.	줄 바꿈 (new line)을 제외한 모든 문자와 매칭 re.DOTALL과 함께 사용되면 줄 바꿈 포함
^ (caret)	문자열의 시작과 매칭. re.MULTILINE에서는 줄 바꿈 바로 뒤와 매칭
\$	문자열의 끝 또는 문자열의 끝에 있는 줄 바꿈 바로 앞과 매칭 re.MULTILINE에서는 줄 바꿈 바로 앞과 매칭
*	0번 이상 반복
+	1번 이상 반복
?	0 또는 1 반복. *?, +?, ??, {m, n}?는 최소 길이 문자열 매칭
{m}	m 길이 문자열 매칭
{m, n}	m부터 n회까지 최대 길이 매칭, {m, n}?는 최소 길이 문자열 매칭
[]	문자 집합 매칭, 예를 들어 [abc]는 'a', 'b', 'c'와 매칭 '.'는 범위 지정: [a-z], [0-9] 집합 내에서 특수 문자는 의미 없음: [(+*)]는 '(', '+', '*', ')'와 매칭 '^'는 그 다음의 문자를 제외한 문자, [^5]는 '5'이외의 문자, [^']는 '^'이외의 문자 [(){}] 또는 [](){}는 모두 괄호와 매칭
	A B는 'A', 'B'와 매칭
\	escape sequence 표시

정규식 (Regular Expression) 모듈

◆ Regular Expression에서 사용되는 메타 문자 (2)

메타 문자	의 미
(...)	괄호안의 정규식과 매칭하고, 그룹을 생성하며, \number로 그룹을 불러와 매칭 (?...)의 확장 표현은 자신의 그룹을 생성하지 않음 (?P<name>re)은 그룹 생성
(?aiLmsux)	각 문자는 정규식에서 re.A, re.I, re.L, re.M, re.S, re.U, re.X 등의 flag의 의미로 사용 (?i): 대소문자 구분하지 않음
(?:...)	괄호안의 정규식과 매칭하지만 자신의 그룹을 만들지 않음
(?P<name>...)	그룹에 의해 매칭되는 부분 문자열을 name 그룹 이름으로 정의
(?#...)	주석 (comment)으로 괄호의 내용을 무시
(?=...)	abc(?def)는 현재 위치에서 'def'가 앞에 없을 때에만 'abc'에 매칭
(?!...)	abc(?!def)는 현재 위치에서 'def'가 앞에 없을 때에만 'abc'와 매칭
(?<=...)	(?<=abc)def는 현재 위치에서 뒤에 'abc'가 있을 때 'def'와 매칭 괄호안의 정규식(abc)의 길이는 abc와 같이 고정길이
(?<!...)	(?<!abc)def는 현재 위치에서 뒤에 'abc'가 없을 때 'def'와 매칭
(?(id/name) yes no)	id 혹은 name 그룹이 있으면 yes로 매칭, 없으면 no로 매칭; no는 생략 가능, (<)?(\w+@(\w+)(?:(\.\w+)+)?(1)> \$)는 '<user@host.com>', 'user@host.com'와 매칭

정규식 (Regular Expression) 모듈

◆ Regular Expression에서의 제어문자

제어 문자	의미
\number	number는 매칭된 그룹 번호
\A	전체 문자열의 처음과 매칭
\b	단어의 시작 또는 끝에서 공백 문자열(empty string)에 매칭
\B	\b와 반대로 단어의 시작과 끝이 아닌 곳에서 공백 문자열에 매칭
\d	유니코드인 str은 유니코드 10진수 문자에 매칭. [0-9]를 포함한 모든 숫자. ASCII와 8비트 문자열 (bytes)에서 [0-9]와 동일
\D	\d의 반대, 즉 유니코드에서는 숫자 이외의 문자. ASCII에서 [^0-9]와 동일
\s	유니코드에서는 [\t\n\r\f\v]를 포함한 모든 공백 문자 ASCII에서는 [\t\n\r\f\v]와 동일
\S	\s의 반대로 유니코드에서는 모든 공백 문자이외의 문자
\w	유니코드에서 숫자, 밑줄 문자(_) 등의 모든 단어 문자 ASCII와 bytes에서 [a-zA-Z0-9_]와 동일
\W	\w의 반대. 유니코드에서는 모든 단어 문자 이외의 문자 ASCII에서 [^a-zA-Z0-9_]와 동일
\Z	전체 문자열의 끝에 매칭

정규식 (Regular Expression) 모듈

◆ 정규식 모듈의 기본 함수 (1)

regular expression 기본 함수	설 명
re.compile(pattern, flags=0)	정규식 표현인 pattern을 컴파일하여 정규식 객체 (regex)를 반환하며 match(), search() 등에 사용될 수 있게 함 rgex = re.compile(pattern); result = rgex.match(string)은 result = re.match(pattern, string)과 동일함 flag의 예 : re.A, re.I, re.L, re.M, re.S, re.X, re.U
re.search(pattern, string, flags=0)	주어진 문자열 (string)에서 정규식 'pattern'이 매칭되는 곳을 탐색하고, 매칭된 객체 (match object)를 반환; 만약 매칭되는 곳을 찾지 못하면 None을 반환
re.match(pattern, string, flags=0)	주어진 문자열 (string)에서 정규식 'pattern'이 한번 이상 발견되면 매칭된 객체 (match object)를 반환; 만약 매칭되는 곳을 찾지 못하면 None을 반환
re.fullmatch(pattern, string, flags=0)	주어진 문자열 (string) 전체가 정규식 'pattern'과 정확하게 매칭되면 그 매칭 객체를 반환
re.split(pattern, string, ma xsplitted=0, flags=0)	주어진 문자열 (string)을 정규식 'pattern'에 따라 문자열로 분할하고 리스트에 담아 반환

정규식 (Regular Expression) 모듈

◆ 정규식 모듈의 기본 함수 (2)

regular expression 기본 함수	설 명
re.findall(pattern, string, flags=0)	주어진 문자열에서 정규식 'pattern'과 매치되는 중복되지 않은 문자열을 리스트에 답아 반환
re.finditer(pattern, string, flags=0)	주어진 문자열에서 정규식 'pattern'과 매치되는 중복되지 않은 문자열의 반복자 (iterator)를 반환
re.sub(pattern, repl, string, count=0, flags=0)	주어진 문자열의 정규식 'pattern'에 매칭되는 문자열을 'repl'로 치환한 후 반환
re.subn(pattern, repl, string, count=0, flags=0)	sub() 메소드와 같이 동작하며; (new string, number of substitutions) 튜플을 반환
re.escape(string)	ASCII 문자, 숫자, 밑줄 문자를 제외한 문자에 대한 이스케이프 시퀀스 처리

정규식 (Regular Expression) 모듈

◆ match(), search()

```
# re (Regular Expression)
```

```
import re
```

```
# a decimal digit
```

```
m1 = re.match('[0-9]', "5678abcd")
```

```
print("m1 : ", m1)
```

```
print("m1.group() : ", m1.group())
```

```
# a string that starts with two decimal digits
```

```
m2 = re.match('\d\d', "56781234") # '[0-9][0-9]'
```

```
print("m2 : ", m2)
```

```
print("m2.group() : ", m2.group())
```

```
# search a string that starts with two decimal digits
```

```
m3 = re.search('\d\d', "abcd 1234")
```

```
print("m3 : ", m3)
```

```
if m3 != None:
```

```
    print("m3.group() : ", m3.group())
```

```
m1 : <re.Match object; span=(0, 1), match='5'>
m1.group() : 5
m2 : <re.Match object; span=(0, 2), match='56'>
m2.group() : 56
m3 : <re.Match object; span=(5, 7), match='12'>
m3.group() : 12
```



정규식 (Regular Expression) 모듈

◆ fullmatch(), findall(), finditer()

```
# regular expression - compile, fullmatch, findall, finditer
```

```
import re
p = re.compile(r'\d+') # one or more digit(s)
m1 = p.fullmatch("1234")
print("m1.group() : ", m1.group())
```

```
data_str = "123 abc 456 def 123 678"
m2 = p.findall(data_str)
print("m2 : ", m2)
```

```
it = p.finditer(data_str)
for m in it:
    print(m.span(), ":", m.group())
```

```
m1.group() : 1234
m2 : ['123', '456', '123', '678']
(0, 3) : 123
(8, 11) : 456
(16, 19) : 123
(20, 23) : 678
```



정규식 (Regular Expression) 모듈

◆ split(), sub(), subn()

```
# regular expression - split, sub, subn
```

```
import re
sample_str1 = "This is an example of Python usage."
p1 = re.compile(r'\W+') # 단어 문자 이외의 문자가 1개 이상
m1 = p1.split(sample_str1)
print("m1 : ", m1)
```

```
p2 = re.compile(r'(\W+)')
m2 = p2.split(sample_str1)
print("m2 : ", m2)
```

```
sample_str2 = "aaa 123 bbb 4567"
p3 = re.compile(r'\d+') # 1 or more digits
m3 = p3.sub('x', sample_str2)
print("m3 : ", m3)
```

```
m4 = p3.sub('x'*3, sample_str2) # substitute by "***"
print("m4 : ", m4)
```

```
m5 = p3.subn('x', sample_str2) # provide number of substitutions
print("m5 : ", m5)
```

```
m1 : ['This', 'is', 'an', 'example', 'of', 'Python', 'usage', '']
m2 : ['This is an example of Python usage.']
m3 : aaa x bbb x
m4 : aaa xxx bbb xxx
m5 : ('aaa x bbb x', 2)
```



정규식 (Regular Expression) 모듈

```
# re (Regular Expression) - searching E-mail address
import re
email_addr = "<user@host.com>"
m1 = re.search('([\w.-]+)', email_addr)

print("m1 : ", m1)
if m1 != None:
    print("m1.group() : ", m1.group())

m2 = re.search('([\w.-]+)@([\w.-]+)', email_addr)
print("m2.group() : ", m2.group())
print("m2.group(1) : ", m2.group(1)) # before '@'
print("m2.group(2) : ", m2.group(2)) # after '@'

m3 = re.search('<?([\w.-]+)>?', email_addr)
print("m3.group() : ", m3.group())

m4 = re.search('<?([\w.-]+)@([\w.-]+)>?', email_addr)
print("m4.group() : ", m4.group())
print("m4.groups() : ", m4.groups())
```

```
m1 : <re.Match object; span=(1, 5), match='user'>
m1.group() : user
m2.group() : user@host.com
m2.group(1) : user
m2.group(2) : host.com
m3.group() : <user
m4.group() : <user@host.com>
m4.groups() : ('<', 'user', 'host.com', '>')
```



정규식 (Regular Expression) 모듈

◆ compile(), search()

```
# re (Regular Expression) - compile(), search()
import re

address_book = \
    ["HS_Park 010-1234-4567 <hs.park@yu.ac.kr>",
     "SK_Kim 010-9800-2345 <sk.kim@google.com>",
     "HD_Yoon 010-2312-9876 <hd.yoon@samsung.co.kr>"]
```

```
p_NameTelNo = re.compile(r"(?P<name>\w+)\s+(?P<tel_no>(\d+)[-]\d+[-]\d+)"
for addr in address_book:
    m1 = p_NameTelNo.search(addr)
    print("name = ", m1.group("name"))
    print("tel_no = ", m1.group("tel_no"))
```

```
p_EmailAddr = re.compile(r"(?P<user_name>[\w.-]+)@(?P<host_addr>[\w.-]+)"
m2 = p_EmailAddr.search(addr)
print("user_name = ", m2.group("user_name"))
print("host_addr = ", m2.group("host_addr"))
print()
```

```
name = HS_Park
tel_no = 010-1234-4567
user_name = hs.park
host_addr = yu.ac.kr
```

```
name = SK_Kim
tel_no = 010-9800-2345
user_name = sk.kim
host_addr = google.com
```

```
name = HD_Yoon
tel_no = 010-2312-9876
user_name = hd.yoon
host_addr = samsung.co.kr
```

사용자 정의 모듈 (User Defined Module)
사용자 정의 패키지 (User Defined Package)

사용자 정의 모듈 예 (1) - MyList, MySortings

◆ 사용자 정의 모듈의 예 - MyList

MyList.py (1)

```
import random
```

```
def genRandFloatList(size):
```

```
    L = []
```

```
    offset = - size / 200
```

```
    for i in range(size):
```

```
        d = (i / 100) + offset
```

```
        L.append(d)
```

```
    random.shuffle(L)
```

```
    return L
```

MyList.py (2)

```
def printFloatListSample(L, per_line = 10, sample_lines = 3):
```

```
    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:>10.2f} ".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:>10.2f} ".format(L[count])
```

```
                count += 1
```

```
            print(s)
```

```
            if count >= size:
```

```
                break
```



사용자 정의 모듈 예 (2) - MySortings

◆ 사용자 정의 모듈 예 (2) - MySortings

```
# User defined Module - MySortings.py (1)
def _merge(L_left, L_right):
    L_result = []
    i, j = 0, 0
    while i < len(L_left) and j < len(L_right):
        if L_left[i] < L_right[j]:
            L_result.append(L_left[i])
            i += 1
        else:
            L_result.append(L_right[j])
            j += 1
    while (i < len(L_left)):
        L_result.append(L_left[i])
        i += 1
    while (j < len(L_right)):
        L_result.append(L_right[j])
        j += 1
    return L_result
```

```
# User defined Module - MySortings.py (2)
def _mergeSort(L):
    if len(L) < 2:
        return L[:]
    else:
        middle = len(L) // 2
        L_left = _mergeSort(L[:middle])
        L_right = _mergeSort(L[middle:])
        L_result = _merge(L_left, L_right)
        return L_result

def mergeSort(L):
    if len(L) < 2:
        return L[:]
    else:
        L_result = _mergeSort(L)
    for i in range(len(L)):
        L[i] = L_result[i]
```



파이썬의 사용자 정의 모듈 (MyList, MySortings)의 활용

◆ 사용자 정의 모듈 (MyList, MySortings)의 사용

- 사용자 정의 모듈 MyList.py와 MySortings.py가 응용 프로그램과 같은 디렉토리/폴더에 위치

```
# Using user-defined modules MyList.py and MySorting.py
```

```
import MyList
import MySortings
```

```
def main():
```

```
    L = MyList.genRandFloatList(10000)
```

```
    print("generated random float list (before sorting) :")
```

```
    MyList.printFloatListSample(L, 10, 3)
```

```
    MySortings.mergeSort(L)
```

```
    print("after sorting :")
```

```
    MyList.printFloatListSample(L, 10, 3)
```

```
if __name__ == "__main__":
    main()
```

generated random float list (before sorting) :

-8.22	34.32	-6.80	-13.09	-25.63	-11.63	-0.70	-12.94	4.52	21.45
15.56	-26.40	49.18	18.40	19.16	-42.34	-26.03	-33.32	-18.41	3.31
-21.46	-44.29	3.22	6.01	-14.12	-47.83	-23.95	36.72	0.97	-29.00

.....

44.33	7.54	-30.84	2.32	-1.39	-44.97	20.13	-46.93	33.43	35.25
-42.51	23.54	18.33	37.61	-38.64	-31.21	-38.80	-18.26	-13.87	38.10
33.98	-49.82	22.58	1.46	-10.75	34.68	-21.31	-7.00	17.41	48.53

after sorting :

-50.00	-49.99	-49.98	-49.97	-49.96	-49.95	-49.94	-49.93	-49.92	-49.91
-49.90	-49.89	-49.88	-49.87	-49.86	-49.85	-49.84	-49.83	-49.82	-49.81
-49.80	-49.79	-49.78	-49.77	-49.76	-49.75	-49.74	-49.73	-49.72	-49.71

.....

49.70	49.71	49.72	49.73	49.74	49.75	49.76	49.77	49.78	49.79
49.80	49.81	49.82	49.83	49.84	49.85	49.86	49.87	49.88	49.89
49.90	49.91	49.92	49.93	49.94	49.95	49.96	49.97	49.98	49.99

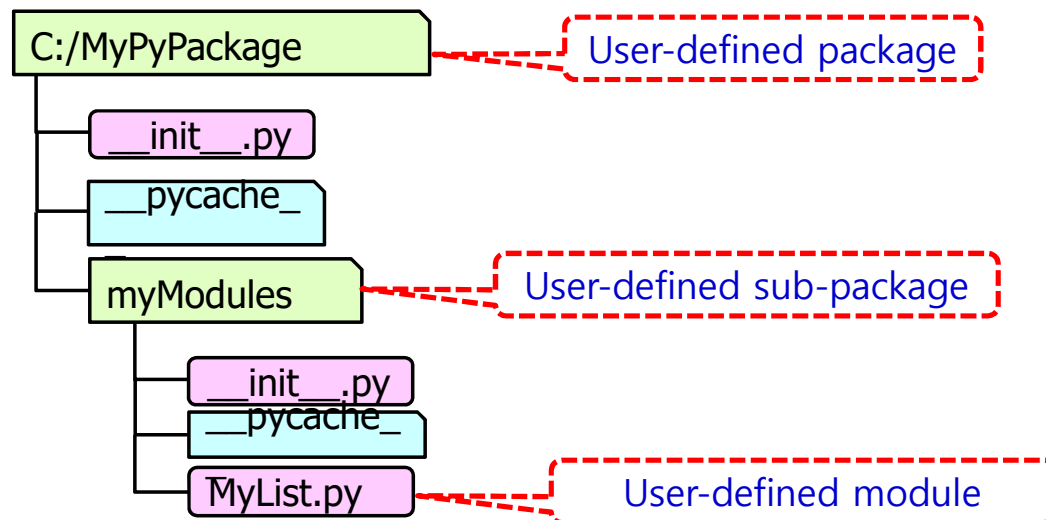


6.4.3 파이썬의 사용자 정의 패키지

◆ 사용자 정의 패키지

- 파이썬 프로그래밍 환경이 제공하는 패키지 및 모듈 이외에 사용자가 직접 구현하고 준비한 파이썬 프로그램 모듈들을 패키지로 통합하여 사용할 수 있게 함
- 사용자 정의 패키지를 배포하여 다른 사람이 사용할 수 있게 준비

◆ 사용자 정의 패키지 구현을 위한 폴더 준비



파이썬의 사용자 정의 패키지

◆ 사용자 정의 패키지 (MyPyPackage) 초기화 프로그램: __init__.py

```
#C:/MyPyPackage/__init__.py
__all__ = ['myModules']
print("C:/MyPyPackage/__init__.py -> importing myModules")
from . import myModules
```

◆ 사용자 정의 서브 패키지 (myModules) 초기화 프로그램 - __init__.py

```
# C:/MyPyPackage/myModules/__init__.py
__all__ = ['MyList']
print(" C:/MyPyPackage/myModules/__init__.py --> importing MyList, MySortings")
from . import MyList MySortings
```


파이썬의 사용자 정의 패키지

◆ MyPyPackage 패키지, myModules 서브 패키지와 dynFibo 모듈의 사용

- 사용자 정의 패키지를 별도의 디렉토리/폴더에 저장하고, 공용으로 사용하는 경우

```
# Usage of user-defined package and module
# C:/MyPyPackage/MyList.py, MySortings
```

```
import os, sys
sys.path.append("C:/MyPyPackage")
```

```
import MyList
import MySortings
```

```
L = MyList.genRandFloatList(10000)
print("generated random float list (before sorting) :")
MyList.printFloatListSample(L, 10, 3)
MySortings.mergeSort(L)
print("after sorting :")
MyList.printFloatListSample(L, 10, 3)
```

```
generated random float list (before sorting) :
-8.22      34.32      -6.80      -13.09      -25.63      -11.63      -0.70      -12.94      4.52      21.45
15.56      -26.40      49.18      18.40      19.16      -42.34      -26.03      -33.32      -18.41      3.31
-21.46      -44.29      3.22      6.01      -14.12      -47.83      -23.95      36.72      0.97      -29.00
. . . . .
44.33      7.54      -30.84      2.32      -1.39      -44.97      20.13      -46.93      33.43      35.25
-42.51      23.54      18.33      37.61      -38.64      -31.21      -38.80      -18.26      -13.87      38.10
33.98      -49.82      22.58      1.46      -10.75      34.68      -21.31      -7.00      17.41      48.53
after sorting :
-50.00      -49.99      -49.98      -49.97      -49.96      -49.95      -49.94      -49.93      -49.92      -49.91
-49.90      -49.89      -49.88      -49.87      -49.86      -49.85      -49.84      -49.83      -49.82      -49.81
-49.80      -49.79      -49.78      -49.77      -49.76      -49.75      -49.74      -49.73      -49.72      -49.71
. . . . .
49.70      49.71      49.72      49.73      49.74      49.75      49.76      49.77      49.78      49.79
49.80      49.81      49.82      49.83      49.84      49.85      49.86      49.87      49.88      49.89
49.90      49.91      49.92      49.93      49.94      49.95      49.96      49.97      49.98      49.99
```



사용자 정의 모듈/패키지와 응용 프로그램의
__name__ 변수와 __main__

__name__ 변수와 __main__

◆ 사용자 정의 모듈의 준비 : add.py

```
# add.py

def add(a, b):
    return a + b

print("add.py :: __name__ : ", __name__)
x, y = map(int, input("input x and y : ").split(' '))
sum = add(x, y)

print("{} + {} -> {}".format(x, y, sum))
```

◆ 이 파일을 F5를 사용하여 직접 실행시킬 경우

```
add.py :: __name__ : __main__
input x and y : 5 7
5 + 7 -> 12
```



사용자 정의 모듈/패키지와 응용 프로그램의 __name__ 변수와 __main__

- ◆ 앞에서 구현된 **add.py** 모듈의 **add()** 함수를
다른 파이썬 프로그램에서 **import** 시켜 사용하고자 하는 경우

```
# test_add.py

import add

print("test_add::__name__ : ", __name__)

if __name__ == "__main__":
    x, y = map(int, input("input x and y : ").split(' '))
    sum = add.add(x, y)

    print("{} + {} -> {}".format(x, y, sum))
```

```
add.py :: __name__ : add
input x and y : 5 7
5 + 7 -> 12
test_add::__name__ : __main__
input x and y : 10 20
10 + 20 -> 30
```



사용자 정의 모듈/패키지와 응용 프로그램의 __name__ 변수와 __main__

◆ 사용자 정의 모듈의 준비 (2) : add.py 모듈의 부분 수정

```
# add.py

def add(a, b):
    return a + b

if __name__ == "__main__": #이 파일이 다른 프로그램에 import되지 않고, main으로써 동작할 때에만 실행
    print("add.py :: __name__ : ", __name__)
    x, y = map(int, input("input x and y : ").split(' '))
    sum = add(x, y)

    print("{} + {} -> {}".format(x, y, sum))
```

◆ 이 수정된 add.py파일을 test_add.py에서 import시켜 실행시킬 경우

```
test_add::__name__ : __main__
input x and y : 5 7
5 + 7 -> 12
```



파이썬 프로그램의 `__name__` 변수

◆ `__name__` 변수는 파이썬 소스코드 파일이 어떤 기능으로 포함되어 있는지 구분

- import 되어 있는 경우, 그 모듈의 이름으로 설정
- 다른 파일에 import 되어 있지 않고, 직접 실행되고 있는 경우, `__main__`으로 설정

◆ `main()` 함수의 구현

- if `__name__ == "__main__"`: 으로 조건을 시험하여, `__name__`이 `__main__` 인 경우에 `main()` 함수가 실행되도록 구성

__main__을 사용한 프로그램 실행

◆ __main__ 사용 예제

```
# Usage of user-defined package and module
# C:/MyPyPackage/myModules/MyFibo.py

import os, sys
sys.path.append("C:/MyPyPackage/myModules")
print("sys.path : ", sys.path)

from MyFibo import *

def main():
    while True:
        n = int(input('n for dynFibo (-1 to quit) : '))
        if n == -1:
            break
        f_n = dynFibo(n)
        print("dynFibo({}) : {}".format(n, f_n))
if __name__ == "__main__":
    main()
```

```
sys.path : ['D:\\YTK-Lecture(수업)-20211030\\2021-2.5 예비사회인을 위한 파이썬 프로그래밍\\
파이썬 프로그램\\Ch 07 Module, Package', 'C:\\Users\\Owner\\AppData\\Local\\Programs\\Pytho
n\\Python310\\python310.zip', 'C:\\Users\\Owner\\AppData\\Local\\Programs\\Python\\Python3
10\\DLLs', 'C:\\Users\\Owner\\AppData\\Local\\Programs\\Python\\Python310\\lib', 'C:\\User
s\\Owner\\AppData\\Local\\Programs\\Python\\Python310', 'C:\\Users\\Owner\\AppData\\Local\\
Programs\\Python\\Python310\\lib\\site-packages', 'C:/MyPyPackage/myModules']
n for dynFibo (-1 to quit) : 100
dynFibo(100) : 354224848179261915075
n for dynFibo (-1 to quit) : 200
dynFibo(200) : 280571172992510140037611932413038677189525
n for dynFibo (-1 to quit) : 300
dynFibo(300) : 22232244629420445529739893461909967206666939096499764990979600
n for dynFibo (-1 to quit) : -1
```



Homework 6

Homework 6.1

6.1 사용자 정의 패키지/서브패키지 활용 (1)

- 사용자 정의 패키지/서브패키지 "C:/MyPyPackage/myModules/"를 생성하고, 중복되지 않는 정수형 난수의 리스트 생성, 샘플 출력 및 뒤섞기 기능을 제공하는 genRandList(L, size), printListSample(L, per_line, sample_lines) 및 shuffleList(L)를 포함하는 사용자 정의 모듈 "MyList.py"을 준비하라. 다음 응용 프로그램을 사용하여 정확한 동작을 확인할 것.
- (응용 프로그램 및 실행 예제)

```
# User-defined package/module
```

```
import sys
myPyPackage_dir = "C:/"
sys.path.append(myPyPackage_dir)

from MyPyPackage.myModules import MyList

L = []
n = 100

MyList.genRandList(L, n)
MyList.printListSample(L, 10, 5)
```

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



Homework 6.2

6.2 사용자 정의 패키지/서브패키지 활용 (2)

- 사용자 정의 패키지/서브패키지 "C:/MyPyPackage/myModules/" 에 선택 정렬 알고리즘을 구현한 selectionSort(L)을 포함하는 사용자 정의 "MySortings.py"을 구현하라.
- 앞에서 구현한 사용자 정의 모듈 "MyList.py"을 함께 사용하는 다음 응용 프로그램을 사용하여 정확한 동작을 확인하라.

```
# User-defined package/module

import sys
myPyPackage_dir = "C:/"
sys.path.append(myPyPackage_dir)

from MyPyPackage.myModules\
    import MyList, MySortings

L = []
n = 100

MyList.genRandList(L, n)
print("Before Sorting :")
MyList.printListSample(L, 10, 3)

MySortings.selectionSort(L)

print("\nAfter Sorting :")
MyList.printListSample(L, 10, 3)
```

```
C:/MyPyPackage/_init_.py -> importing myModules
C:/MyPyPackage/myModules/_init_.py() --> importing MyList module(s)
Before Sorting :
 43   11   48   14   34   38   97   66   86   85
 93   70   75   51   37   20   95   28   41    1
 62   36   15   18   92    5   46   27   44   12
. . . . .
 35   16   13   88   10   60   54   90   77   22
 80    3    8   96   30   19   94   42    9   58
  7   64   40   33   79   52   55    4   99    0

After Sorting :
  0    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
. . . . .
 70   71   72   73   74   75   76   77   78   79
 80   81   82   83   84   85   86   87   88   89
 90   91   92   93   94   95   96   97   98   99
```



Homework 6.3

6.3 사용자 정의 패키지/서브패키지 활용 (3)

- 앞의 Homework 6.1과 6.2에서 준비한 사용자 정의 모듈 MyList와 MySortings을 사용하여 중복되지 않는 50000개의 정수형 난수를 포함하는 리스트를 생성하라.
- 아울러, 파이썬 기본 제공 모듈인 array를 사용하여 정수형 난수 리스트에 포함된 원소들을 배열에 포함시켜라. 다음 응용 프로그램을 사용하여 리스트와 배열에 포함된 난수들을 선택 정렬시키고, 실행시간을 측정하여 비교하라.
- (응용 프로그램 및 실행 예제)

```
# Comparison of List and Array with user-defined modules (part 1)

import random, time, sys
from array import *
sys.path.append("C:/")
from MyPyPackage.myModules import MyList, MySortings

AR = array('i')
L = []
size = 50000
MyList.genRandList(L, size)

for x in L:
    AR.append(x)
```



```

# Comparison of List and Array with user-defined modules (part 2)
print("Array (size : {}) before sorting : ".format(size))
MyList.printListSample(AR, 10, 2)
t1 = time.time()
MySortings.selectionSort(AR)
t2 = time.time()
print("Array (size : {}) after sorting : ".format(size))
MyList.printListSample(AR, 10, 2)
print("Selection sorting for array of {} integers took {} sec"\
      .format(size, t2-t1))

print("\nList (size : {}) before sorting : ".format(size))
MyList.printListSample(L, 10, 2)
t1 = time.time()
MySortings.selectionSort(L)
t2 = time.time()
print("\nList (size : {}) after sorting : ".format(size))
MyList.printListSample(L, 10, 2)
print("Selection sorting for list of {} integers took {} sec"\
      .format(size, t2-t1))

```



```

C:/MyPyPackage/__init__.py -> importing myModules
C:/MyPyPackage/myModules/__init__.py() --> importing MyList module(s)
Array (size : 50000) before sorting :
 33634   4577   46380   35909   32284   12199   46026     428   25747   17179
 46288   20734   31512   42442   48634   1336    4881   12108   40029   2742
  . . . . .
  9713   25318   44017   12183    1106   13920   30248   20588   43720   22362
 34567   42345   42759   21282   29313   15243   32137   18054   14156   28176
Array (size : 50000) after sorting :
    0      1      2      3      4      5      6      7      8      9
   10     11     12     13     14     15     16     17     18     19
  . . . . .
 49980   49981   49982   49983   49984   49985   49986   49987   49988   49989
 49990   49991   49992   49993   49994   49995   49996   49997   49998   49999
Selection sorting for array of 50000 integers took 129.56686067581177 sec

List (size : 50000) before sorting :
 33634   4577   46380   35909   32284   12199   46026     428   25747   17179
 46288   20734   31512   42442   48634   1336    4881   12108   40029   2742
  . . . . .
  9713   25318   44017   12183    1106   13920   30248   20588   43720   22362
 34567   42345   42759   21282   29313   15243   32137   18054   14156   28176

List (size : 50000) after sorting :
    0      1      2      3      4      5      6      7      8      9
   10     11     12     13     14     15     16     17     18     19
  . . . . .
 49980   49981   49982   49983   49984   49985   49986   49987   49988   49989
 49990   49991   49992   49993   49994   49995   49996   49997   49998   49999
Selection sorting for list  of 50000 integers took 121.51226377487183 sec

```



Homework 6.4

6.4 사용자 정의 패키지/서브패키지 활용 (4)

- 앞에서 준비하였던 사용자 정의 모듈 "MySortings.py"에 병합정렬 (mergeSort) 기능을 수행하는 함수/메소드 mergeSort(L)를 추가하라.
- 다음 응용 프로그램을 사용하여 50000개의 중복되지 않는 정수형 난수에 대한 선택정렬과 병합정렬을 각각 실행시키고, 실행에 걸린 경과시간을 측정하여 비교하라.
- (응용 프로그램 및 실행 결과)

```
size of list (0 to terminate) = 50000
List (size : 50000) before merge sorting :
13012 25280 26021 9248 8402 26196 18900 17540 35911 31416
44426 3574 45109 32931 45258 15130 1246 15186 32564 24982
. . . . .
27340 33352 4503 44986 2042 44998 10587 17581 6394 19839
48053 20712 31274 16639 44966 13868 29870 23036 4130 7911

List (size : 50000) after merge sorting :
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
. . . . .
49980 49981 49982 49983 49984 49985 49986 49987 49988 49989
49990 49991 49992 49993 49994 49995 49996 49997 49998 49999
Merge sorting for list of 50000 integers took 0.2752869129180908 sec

List (size : 50000) before selection sorting :
42750 7691 40848 17361 23137 5854 20566 14039 4639 10681
45559 7123 28547 28677 5582 46822 43618 385 25119 49327
. . . . .
41996 27465 11965 14180 30710 12877 16230 3216 40579 40645
8690 16289 23829 28497 17398 3666 41150 9545 9824 9328

List (size : 50000) after selection sorting :
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
. . . . .
49980 49981 49982 49983 49984 49985 49986 49987 49988 49989
49990 49991 49992 49993 49994 49995 49996 49997 49998 49999
Selection sorting for list of 50000 integers took 120.65009355545044 sec
```



```

# Comparison of mergeSort and selectionSort with user-defined modules

import random, time, sys
sys.path.append("C:/")
from MyPyPackage.myModules import MyList, MySortings

while True:
    size = int(input("\nsize of list (0 to terminate) = "))
    L = []
    MyList.genRandList(L, size)
    print("List (size : {}) before merge sorting : ".format(size))
    MyList.printListSample(L, 10, 2)
    t1 = time.time()
    MySortings.mergeSort(L)
    t2 = time.time()
    print("\nList (size : {}) after merge sorting : ".format(size))
    MyList.printListSample(L, 10, 2)
    print("Merge sorting for list of {} integers took {} sec".format(size, t2-t1))

    MyList.shuffleList(L)
    print("\nList (size : {}) before selection sorting : ".format(size))
    MyList.printListSample(L, 10, 2)
    t1 = time.time()
    MySortings.selectionSort(L)
    t2 = time.time()
    print("\nList (size : {}) after selection sorting : ".format(size))
    MyList.printListSample(L, 10, 2)
    print("Selection sorting for list of {} integers took {} sec".format(size, t2-t1))

```

Homework 6.5

6.5 사용자 정의 패키지/서브패키지 활용 (5)

- 사용자 정의 모듈 "MyMatrix.py" 을 준비하고, 행렬의 덧셈, 뺄셈, 곱셈 연산을 수행하는 함수/메소드 mtrxAdd(A, B), mtrxSub(A, B), mtrxMul(A, B), printMtrx(M)를 포함시켜라. 이 사용자 정의 모듈을 사용자 정의 패키지 "MyPyPackage.myModules"에 포함시킬 것. 다음 응용 프로그램을 사용하여 정확한 동작을 확인할 것.
- (응용 프로그램 및 실행결과)

```
# Testing user-defined module MyMatrix

import random, time, sys
myPyPackage_dir = "C:/MyPyPackage"
sys.path.append(myPyPackage_dir)
from myModules import MyMatrix

A = [[1,2,3,4], [5,6,7,8], [9,10,0,1]]
B = [[1,0,0,0], [0,1,0,0], [0,0,1,1]]
C = [[1,0,0], [0,1,0], [0,0,1], [0,0,0]]
print("A = "); MyMatrix.printMtrx(A)
print("B = "); MyMatrix.printMtrx(B)
print("C = "); MyMatrix.printMtrx(C)
D = MyMatrix.mtrxAdd(A, B)
print("A + B = "); MyMatrix.printMtrx(D)

E = MyMatrix.mtrxSub(A, B)
print("A - B = "); MyMatrix.printMtrx(E)

F = MyMatrix.mtrxMul(A, C)
print("A * C = "); MyMatrix.printMtrx(F)
```

```
C:/MyPyPackage/myModules/__init__.py()
A =
1 2 3 4
5 6 7 8
9 10 0 1
B =
1 0 0 0
0 1 0 0
0 0 1 1
C =
1 0 0
0 1 0
0 0 1
0 0 0
A + B =
2 2 3 4
5 7 7 8
9 10 1 2
A - B =
0 2 3 4
5 5 7 8
9 10 -1 0
A * C =
1 2 3
5 6 7
9 10 0
```