help(object) : 내장함수, 객체의 도움말 출력

help() : 내장함수, help 함수 실행.

divmod(x,y) : 내장함수, 몫과 나머지 한번에 계산, 튜플로 리턴

type(object) : 내장함수, 자료형 확인

dir([object]) : 내장함수, 객체의 어트리뷰트(함수, 변수 등의 이름들)을 리턴한다. 그리고 객체의 클래스 속성 및 슈퍼

클래스의 속성까지 한꺼번에 표시함.

execfile(filename[,globals, [locals]]) : 내장함수, 인터프리터 안에서 파일을 실행

eval(source[,globals, [locals]]) : 내장함수, 문자열로 된 파이썬 식 실행

exec code([in globals, [locals]]) : 내장함수, 문자열로 된 문 실행

compile(string, filename, kind) : 내장함수, 문자열을 컴파일하여 파이썬 코드를 리턴한다.

raw\_input([prompt]) : 내장함수, 키보드로부터 문자열을 읽어 들임

input([prompt]) : 내장함수, 입력된 문자열을 파이썬 식으로 처리해서 넘겨준다.

pprint.pprint(object, ...) : pprint 모듈, 복잡한 자료를 출력할 때 씀

len(object) : 내장함수, 문자열의 길이 출력

sys.getrefcount(object) : sys 모듈, 레퍼런스 카운트 값을 확인

id(object) : 내장함수, 객체의 주소를 식별

range([start,] stop [,step]) : 내장함수, start 부터 stop 까지 step 만큼의 범위로 리스트 생성

sys.maxint : sys모듈, 최대 정수 값을 가진 상수.

enumerate(iterable) : 내장함수, (인덱스, 요소 값)의 튜플 자료로 값을 넘겨줌.

complex(real, [imag]) : 내장함수, 복소수를 만드는 함수. imag 없이 real 이 문자열일 경우 복소수형으로 변환될 수 있으면 변환.

<complex>.real : 복소수의 실수 부분을 취함.

<complex>.imag : 복소수의 허수 부분을 취함.

<complex>.conjugate() : 복소수의 켤레 복소수를 취함

decimal.Decimal(object) : decimal 모듈, 오차 없이 정확한 수를 취함. 단, object 에는 정수를 그대로 쓸 수 있지만

실수는 문자열로 표현해야 한다.

abs(number) : 내장함수, 수의 절대값

int(x, [base]) : 내장함수, base 진법의 수를 10진 정수형으로 변환. x 가 문자열일 경우 파이썬 정수형으로 변환될 수

있으면 정수로 변환.

long(x, [base]) : 내장함수, base 진법의 수를 10진 long 형으로 변환. x 가 문자열일 겨우 파이썬 롱형으로 변환될 수 있으면 롱형으로 변환.

float(x) : 내장함수, 수를 실수형으로 변환. x 가 문자열일 경우 파이썬 실수형으로 변환될 수 있으면 실수형으로 변환.

pow(x, y, [z]) : 내장함수, (x \*\* y) % z 로 리턴

vars([object]) : 내장함수, object 없이 쓸 경우 지역 변수의 사전을 넘김. object 가 들어가면 그 객체(모듈, 클래스,

클래스 인스터스)의 사전을 넘김.

locals() : 내장함수, 지역 변수의 사전을 넘김.

string.Template(string) : string 모듈, 단순한 문자열 대치 기능. string 에는 '$키' 의 형식으로 입력하고,

(object).substitute(dict) 을 사용 하여 키에 대응 하는 값을 대치시킴.

str.upper() : 대문자로 변환.

str.lower() : 소문자로 변환.

str.swapcase() : 대소문자 전환.

str.capitalize() : 첫 문자만 대문자로 변환.

str.title() : 단어의 첫 문자를 대문자로 변환.

str.count(sub[, start[,end]]) : start 부터 end 까지 sub 에 있는 문자열이 발생한 횟수를 리턴.

str.find(sub[, start[,end]]]) : start 부터 end 까지 sub 에 있는 문자열의 옵셋을 리턴. 단, 최초로 발견된 문자열

의 옵셋만. 찾는 문자열이 없으면 -1 리턴.

str.rfind(sub[, start[,end]]) : find와 같지만 문자열의 뒤쪽부터 탐색.

str.index(sub[, start[,end]]) : find와 같지만 찾는 문자열이 없을 경우 예외를 일으킨다.

str.rindex(sub[, start[,end]]) : index와 같지만 문자열의 뒤쪽부터 탐색.

str.startswith(prefix[, start[,end]]) : start 부터 prefix로 시작하는 문자열인지 아닌지를 참, 거짓으로 리턴.

str.endswith(suffix[, start[,end]]) : start 부터 subfix로 끝나는 문자열인지 아닌지를 참, 거짓으로 리턴.

str.strip([chars]) : chars 의 문자를 좌우에서 하나씩 없애나감.

str.rstrip([chars]) : chars 의 문자를 오른쪽에서 하나씩 없애나감.

str.lstrip([chars]) : chars 의 문자를 왼쪽에서 하나씩 없애나감.

str.replace(old, new[, count]) : old 의 문자를 new의 문자로 count 만큼 바꾼다.

str.split([sep [, maxsplit]]) : sep 으로 문자열을 maxsplit 만큼 분리.

str.join(sequence) : sequence 를 문자열로 결합.

str.splitlines([keepends]) : 라인 단위로 분리. keepends 가 True 이면 /n 을 유지. 거짓이면 제거.

str.rsplit([sep [,maxsplit]]) : 오른쪽 부터 split 기능 처리.

str.center(width[, fillchar]) : 전체 width 에서 가운데에 정렬 시킨다. 이때 좌우 부분을 fillchar 로 채운다.

str.ljust(width[, fillchar]) : 전체 width 에서 왼쪽에 정렬 시킨다. 이때 우측 부분을 fillchar 로 채운다.

str.rjust(width[, fillchar]) : 전체 width 에서 오른쪽에 정렬 시킨다. 이때 좌측 부분을 fillchar 로 채운다.

str.expandtabs([tabsize]) : 문자열 내의 탭(\t)을 tabsize 만큼 공배으로 변경. 기본 값은 8.

str.isdigit() : 문자열이 숫자로만 구성 됐는지 참, 거짓으로 리턴

str.isalpha() : 문자열이 문자로만 구성 됐는지 참, 거짓으로 리턴

str.isalnum() : 문자열이 숫자 혹은 문자로 구성 됐는지 참, 거짓으로 리턴

str.islower() : 문자열이 소문자로만 구성 됐는지 참, 거짓으로 리턴

str.isupper() : 문자열이 대문자로만 구성 됐는지 참, 거짓으로 리턴

str.istitle() : 문자열이 제목 문자열인지 참, 거짓으로 리턴

str.zfill(width) : 문자열 크기를 width 만큼 늘리고 빈 자리를 0으로 채운다.

string.digits : string 모듈, 10진법의 수들 출력.

string.octdigits : string 모듈, 8진법의 수들 출력.

string.hexdigits : string 모듈, 16진법의 수들 출력.

string.letters : string 모듈, 영문자들 출력.

string.lowcase : string 모듈, 영 소문자 출력.

string.uppercase : stirng 모듈, 영 대문자 출력.

string.punctuation : string 모듈, punctuation 문자 출력.

string.printable : stirng 모듈, 인쇄 가능한 문자들 모두 출력.

string.whitespace : string 모듈, 공백 문자 출력.

str(object) : 내장함수, 객체를 문자열로 변환.

unicode(string [, encoding[, errors]]) : 내장함수, 문자열을 유니 코드 문자열로 변환.

str.encode([encoding[,errors]]) : 내장함수, 문자열을 원하는 코딩으로 변환.

ord(c) : 내장함수, 문자의 코드 값을 구함.

chr(i) : 내장함수, 코드 값을 문자로 변환.

unichr(i) : 내장함수, 유니코드 값을 유니코드 문자로 변환.

object.\_\_doc\_\_ : 내장 멤버, 객체의 문서 문자열 출력.

L.append(object) : 자료를 리스트 끝에 추가

L.insert(index, object) : 자료를 지정된 위치에 삽입

L.index(value, [start, [stop]]) : 요소 검색.

L.count(value) : 요소 개수 출력

L.sort(key=None, reverse=False) : 리스트를 정렬하며 리턴값은 없음. key 는 비교할 인수를 통과시킬 함수를 받음.

reverse 는 Ture 일때 역순 정렬.

L.reverse() : 자료 순서 역순으로 정렬.(주의 : 리턴값 없음)

L.remove(value) : 지정 자료 값을 앞에서부터 한 개 삭제.

L.extend(iterable) : 리스트를 추가.

L.pop([index]) : 리스트의 지정된 값 하나를 읽어 내고 삭제.

cmp(x,y) : 내장함수, x,y 의 값을 비교하여 x>y : 1, x==y : 0, x<y : -1 을 리턴.

sorted(iterable[, key][, reverse]) : 내장함수, sort와 같은 기능이지만 새로운 리스트로 리턴함.

reversed(sequence) : 내장함수, 시퀀스 자료형을 역순으로 참조.

sys.argv : sys 모듈, 명령행에 쓰여진 인수들을 읽음.

getopt.getopt(args, options[, long\_options]) : getopt 모듈, 명령행에 쓰여진 인수들에서 옵션을 분리함. 리턴 값은

[(option, value)] 형식의 리스트와 나머지 인수들의 리스트이다.

array.array(typecode [, initializer]) : array 모듈, typecode 자료형의 새 배열을 리턴, initializer 는 리스트

또는 문자열이어야 함.

glob.glob(pathname) : glob 모듈, pathname 에 해당하는 경로의 리스트를 리턴함.

os.path.isfile(path) : os.path 모듈, path 가 일반 파일이면 참값 리턴.

os.path.isdir(path) : os.path 모듈, path 가 디렉토리이면 참값 리턴.

os.path.islink(path) : os.path 모듈, path 가 심볼릭 링크이면 참값 리턴.

os.path.ismount(path) : os.path 모듈, path 가 마운트 포인트이면 참값 리턴(unix)

os.path.exists(path) : os.path 모듈, path 가 존재하면 참값 리턴.

os.path.getsize(filename) : os.path 모듈, 파일의 크기 리턴.

os.path.getatime(filename) : os.path 모듈, 파일의 최근 접근 시간 리턴, 리턴 값은 기준시간(1970년 1월 1일 자정)

부터 현재 까지 경과한 초.

os.path.getmtime(filename) : os.path 모듈, 파일의 수정 시간 리턴.

time.ctime(seconds) : time 모듈, 초를 문자열 시간으로 변환.

time.time() : time 모듈, 현재 시간을 소수점 형태의 초로 리턴함.

list([iterable]) : 내장함수, 리스트로 변환.

tuple([iterable]) : 내장함수, 튜플로 변환.

os.path.abspath(path) : os.path 모듈, 상대 경로를 절대 경로로 전환.(주의 : 파일 자체의 절대 경로가 아니라 실행 파일이 위치한 폴더의 절대 경로로 전환.)

os.path.split(p) : os.path 모듈, 경로를 (head, tail)로 분리. (디렉토리명, 파일명)으로 리턴.

os.path.join(a, ...) : os.path 모듈, 디렉토리와 파일명을 결합.

os.path.normpath(path) : os.path 모듈, 파일명을 정규화한다.(예 : A/./B → A/b)

os.path.splitext(p) : os.path 모듈, 파일명으로부터 확장자를 분리함.

os.linesep : os 모듈, 파일의 라인 분리 문자.

os.sep : os 모듈, 경로명에서 각 요소들을 분리하는 문자.

os.pathsep : os 모듈, 경로명과 경로명을 구분해 주는 문자.

os.curdir : os 모듈, 현재 디렉토리를 나타내는 문자.

os.pardir : os 모듈, 부모 디렉토리를 나타내는 문자.

urlparse.urlparse(urlstring[, default\_scheme[, allow\_fragments]]) : urlparse 모듈, url 을 (addressing

scheme, network location, path, parameters, query, fragment identifier) 로 분리.bluekyu

urlparse.urlunparse(parts) : urlparse 모듈, 튜플로 분리된 성분들을 하나의 url로 변환.

urlparse.urljoin(base, url[, allow\_fragments]) : urlparse 모듈, 기본 url 과 상대 url 을 연결하여 절대 url 을

생성.

sys.getsizeof(object[, default]) : sys 모듈, 객체의 크기를 바이트로 리턴.

dict([arg]) : 내장함수, 사전 객체를 생성함.

zip([iterable, ...]) : 내장함수, 자료를 순서대로 묶어 줌. \*를 사용하면 zip 으로 묶인 객체를 다시 풀어줌. 짧은 인수

를 기준으로 나머지는 버림.

D.keys() : 사전에서 키들을 리스트로 리턴.

D.values() : 값들을 리스트로 리턴.

D.items() : 사전의 키, 값을 리스트 내에 (key, value)의 쌍으로 리턴.

key in D : 멤버쉽 테스트. D가 key를 가지고 있으면 True 리턴.

D.clear() : 사전 D의 모든 아이템 삭제.

D.copy() : 사전 복사(Shallow Copy)

D.get(key[, default]) : 값이 존재하면 D[key], 아니면 초기값 리턴.

D.setdefault(key[, default]) : get과 같으나 값이 존재하지 않을 때 값을 설정(D[key] = default) 하고 초기값 리턴.

D.update([other]) : 키, 값의 쌍으로 되어 있는 자료형을 추가 해줌.

D.popitem() : (키, 값) 튜플을 리턴하고 사전에서 항목을 제거.

D.pop(key[, default]) : key 항목의 값을 리턴하고 사전에서 제거.

globals() : 전역 영역의 심볼 테이블을 리턴.

locals() : 지역 영역의 심볼 테이블을 리턴.

s.issubset(t) : s가 t의 부분 집합? (같은 표현 : s <= t)

s.issuperset(t) : s가 t의 포함 집합? (같은 표현 : s >= t)

s.union(t) : s 와 t 의 합집합 (같은 표현 : s | t)

s.intersection(t) : s 와 t 의 교집합 (같은 표현 : s & t)

s.difference(t) : s 와 t 의 차집합 (같은 포현 : s - t)

s.symmetric\_difference(t) : s 와 t 의 배타집합 (같은 표현 : s ^ t)

s.copy() : s 를 얕은 복사.

s.update(t) : s와 t의 합집합을 s에 저장 (같은 표현 : s |= t)

s.intersection\_update(t) : s와 t의 교집합을 s에 저장 (같은 표현 : s &= t)

s.difference\_update(t) : s와 t의 차집합을 s에 저장 (같은 표현 : s -= t)

s.symmetric\_difference\_update(t) : s와 t의 배차집합을 s에 저장 (같은 표현 : s ^= t)

s.add(x) : 원소 x를 s에 추가

s.remove(x) : 원소 x를 s에서 제거, 없으면 KeyError 예외 발생

s.discard(x) : 원소 x가 있다면 s에서 제거.

s.pop() : s에서 임의의 원소를 하나 리턴하고 집합에서는 제거, 빈 집합이면 KeyError 에외 발생.

s.clear() : 집합 s의 모든 원소 삭제.

copy.copy(x) : copy 모듈, x를 얕은 복사 함.

copy.deepcopy(x) : copy 모듈, x를 깊은 복사 함.

round(x[, n]) : 내장 함수, x를 10^(-n) 자리까지 반올림.

math.floor(x) : math 모듈, x 보다 작거나 같은 수 중에서 가장 큰 정수형의 실수.

math.ceil(x) : math 모듈, x 보다 크거나 같은 수 중에서 가장 작은 정수형의 실수.

repr(object) : 내장 함수, srt 보다 좀 더 형식적인 문자열로 변환.(`object` 와 같음), eval 로 역표현이 가능.

hex(x) : 내장 함수, 10진수에서 16진수로 변환.

oct(x) : 내장 함수, 10진수에서 8진수로 변환.

open(filename[, mode[, bufsize]]) : 내장 함수, 파일 객체를 얻음.

file.read([size]) : 파일 객체에서 자료를 읽는다.

file.write(str) : 파일 객체에서 자료를 쓴다.

file.close() : 파일 객체의 사용을 종료한다.

file.readline([size]) : 한 번에 한 줄씩 읽는다.

file.readlines([sizehint]) : 파일 전체를 라인 단위로 끊어서 리스트에 저장한다.

file.xreadlines() : 파일 전체를 한꺼번에 읽지는 않고, 필요할 때만 읽는다.

file.writelines(sequence) : 리스트 안에 있는 문자열을 연속해서 출력함.

file.seek(offset[, whence]) : 파일의 위치 이동.(whence 가 없으면 처음에서 offset 번째로, 1 이면 현재에서 offset

번째로, 2 이면 마지막에서 offset 번째로)

file.tell() : 현재의 파일 포인터 위치를 돌려줌.

file.flush() : 버퍼가 다 채워지지 않았어도 내부 버퍼의 내용을 파일에 보낸다.

file.fileno() : 파일 객체의 파일 기술자(File Descriptor)(정수)를 리턴한다.

file.isatty() : 파일 객체가 tty와 같은 장치이면 1 아니면 0을 리턴

file.truncate([size]) : 파일 크기를 지정된 �로 잘라 버림. 인수를 주지 않으면 현재 위치에서 자름.

file.closed : 파일이 close 되었으면 1 아니면 0

file.mode : 파일이 오픈된 모드

file.name : open() 할때 사용된 파일 이름

file.softspace : 1이면 print 문을 사용할 때 값 출력 사이에 자동적으로 스페이스가 출력됨. 0이면 자동으로 삽입 안됨.

xrange([start], stop[, step]) : 내장 함수, range() 와 비슷하지만 xrange 객체를 리턴 하기 때문에 튜플을 리턴 할

때 생기는 메모리를 줄일 수 있다.

sys.stdin : sys 모듈, 입력을 위한 객체 저장.

StringIO.StringIO([buffer]) : StringIO 모듈, 문자열을 파일 객체처럼 입출력한다.

StringIO.getvalue() : StringIO 모듈, StringIO 클래스에 저장된 내부 문자열을 가져옴.

pickle.dump(obj, file[, protocol]) : pickle 모듈, 객체를 파일로 출력. 프로토콜이 1 이상이면 이진 파일로 작성.

pickle.load(file) : pickle 모듈, 객체를 파일에서 읽어 들임.

pickle.dumps(obj[, protocol]) : pickle 모듈, 문자열로 객체를 파일로 출력. 프로토콜이 1 이상이면 이진 파일로 작

성.

pickle.loads(string) : pickle 모듈, 문자열에서 객체를 읽어 들임.

map(function, iterable, ...) : 내장 함수, 함수의 인수에 주어진 시퀀스 인수들을 사상시킴. 함수로 None 을 넣으면 인

수를 묶어주고, 긴 인수를 기준으로 짧은 부분은 None으로 채움.

### operator 모듈에 다양한 연산 함수가 정의 되어 있음. 자세한 것은 라이브러리 레퍼런스 3.8 참고 ###

filter(function, iterable) : 내장 함수, 주어진 시퀀스의 인수들을 함수에 대해서 필터링함. 함수로 None 을 넣으면 진

리값을 판별하는 데 사용함.

reduce(function, iterable[, initializer]) : 내장 함수, 두 개의 인수를 갖는 함수에 대해 시퀀스의 자료를 순서대로

넘긴다.

f.\_\_doc\_\_ : 함수 객체 속성, 문서 문자열.

f.func\_doc : 함수 객체 속성, 문서 문자열.

f.\_\_name\_\_ : 함수 객체 속성, 함수의 이름.

f.func\_name : 함수 객체 속성, 함수의 이름

f.func\_defaults : 함수 객체 속성, 함수의 기본 인수 값들.

f.func\_code : 함수 객체 속성, 함수의 코드 객체.

f.func\_globals : 함수 객체 속성, 함수의 전역 영역을 나타내는 사전.

f.func\_code.co\_name : 함수 코드 객체 속성, 함수의 이름.

f.func\_code.co\_argcount : 함수 코드 객체 속성, 필수적인 인수의 개수

f.func\_code.co\_nlocals : 함수 코드 객체 속성, 전체 지역 변수의 수.

f.func\_code.co\_varnames : 함수 코드 객체 속성, 지역 변수의 이름들.

f.func\_code.co\_code : 함수 코드 객체 속성, 코드 객체의 바이트 코드 명령어

f.func\_code.co\_names : 함수 코드 객체 속성, 바이트 코드가 사용하는 이름들.

f.func\_code.co\_filename : 함수 코드 객체 속성, 코드 객체를 포함하는 파일 이름.

f.func\_code.co\_flags : 함수 코드 객체 속성, 코드 객체가 가변인수와 키워드 인수를 갖는지 검사.

vars([object]) : 내장 함수, 인수 없이 쓰면 지역 공간의 사전 리턴. 인수를 쓰면 인수의 심볼 테이블을 리턴.

\_\_dict\_\_ : 속성, 정의 되어 있는 속성과 값의 사전을 리턴.

\_\_import\_\_(name[, globals[, locals[, fromlist[, level]]]]) : 내장 함수, 문자열로 표현된 모듈을 가져옴.

reload(module) : 내장 함수, 해당 모듈만을 다시 적재함.

\_\_name\_\_ : 속성, 모듈의 이름.

sys.path : sys 모듈, 모듈의 검색 경로를 저장하고 있는 리스트.

\_\_import\_\_(name[, globals[, locals[, fromlist[, level]]]]) : 내장 함수, 문자열로 표현된 모듈을 가져옴.

reload(module) : 내장 함수, 모듈을 재적함.

getattr(object, name[, default]) : 내장 함수, object에서 문자열로 주어진 name 속성을 얻어냄. 참조하려는 이름이

없다면 AttributeError 를 일으키거나 default 를 리턴.

setattr(object, name, value) : 내장 함수, object에 name의 속성으로 value 를 설정.

hasattr(object, name) : 내장 함수, object가 name 속성을 가지고 있는지 묻는다. 있다면 True 리턴.

delattr(object, name) : 내장 함수, object에서 name 속성을 없앤다.

staticmethod(function) : 내장 함수, 정적 메소드를 생성

classmethod(function) : 내장 함수, 클래스 메소드를 생성

object.\_\_init\_\_(self[, ...]) : 클래스 메소드, 생성자

object.\_\_del\_\_(self) : 클래스 메소드, 소멸자

object.\_\_add\_\_(self, other) : 클래스 메소드, +

object.\_\_sub\_\_(self, other) : 클래스 메소드, -

object.\_\_mul\_\_(self, other) : 클래스 메소드, \*

object.\_\_div\_\_(self, other) : 클래스 메소드, /

object.\_\_truediv\_\_(self, other) : 클래스 메소드, \_\_future\_\_ 모듈의 division 이 실행 되었을 때 / 연산자가 실행

됨

object.\_\_floordiv\_\_(self, other) : 클래스 메소드, //

object.\_\_mod\_\_(self, other) : 클래스 메소드, %

object.\_\_divmod\_\_(self, other) : 클래스 메소드, divmod()

object.\_\_pow\_\_(self, other[, modulo]) : 클래스 메소드, pow(), \*\*

object.\_\_lshift\_\_(self, other) : 클래스 메소드, <<

object.\_\_rshift\_\_(self, other) : 클래스 메소드, >>

object.\_\_and\_\_(self, other) : 클래스 메소드, &

object.\_\_xor\_\_(self, other) : 클래스 메소드, ^

object.\_\_or\_\_(self, other) : 클래스 메소드, |

###피연산자가 바뀐 경우에는 위 메소드에다가 r 을 앞에다가 붙이면 됨.###

###확장 산술 연산자(+=, -= 과 같은)는 위 메소드에다가 i 를 붙이면 됨. 단, divmod 는 확장 산술 연사자가 없음.###

object.\_\_neg\_\_(self) : 클래스 메소드, - (단항 연산자)

object.\_\_pos\_\_(self) : 클래스 메소드, + (단항 연산자)

object.\_\_abs\_\_(self) : 클래스 메소드, abs()

object.\_\_invert\_\_(self) : 클래스 메소드, ~ 비트 반전.

object.\_\_complex\_\_(self) : 클래스 메소드, complex(), 리턴 값은 반드시 복소수값

object.\_\_int\_\_(self) : 클래스 메소드, int(), 리턴 값은 반드시 정수형

object.\_\_long\_\_(self) : 클래스 메소드, long(), 리턴값은 반드시 Long 형

object.\_\_float\_\_(self) : 클래스 메소드, float(), 리턴값은 반드시 실수형

object.\_\_oct\_\_(self) : 클래스 메소드, oct(), 리턴값은 반드시 문자열

object.\_\_hex\_\_(self) : 클래스 메소드, hex(), 리턴값은 반드시 문자열

object.\_\_coerce\_\_(self, other) : 클래스 메소드, 반드시 일반적인 수 타입으로 변환될 수 있는 self, other 을 포함하

는 튜플을 리턴하거나 변환될 수 없을 경우 None 객체를 리턴해야 한다. 튜플의 값들이 내장된 연산이 적용될 수 있을 경우

다른 연산자 메소드보다 우선적으로 호출됨.(자세한 것은 파이썬 문서 참조.)

object.\_\_len\_\_(self) : 클래스 메소드, len(), 반드시 0 이상의 정수형을 리턴해야 함.

object.\_\_contains\_\_(self, item) : 클래스 메소드, in 연산자.

object.\_\_getitem\_\_(self, key) : 클래스 메소드, self[key]

object.\_\_setitem\_\_(self, key, value) : 클래스 메소드, self[key] = value

object.\_\_delitem\_\_(self, key) : 클래스 메소드, del self[key]

object.\_\_iter\_\_(self) : 클래스 메소드, 반복자를 요구할 때 호출됨.

###shelve 모듈을 이용하여 사전과 같은 방식으로 파일을 입,출력할 수 있다.###

object.\_\_repr\_\_(self) : 클래스 메소드, repr(), ` `, \_\_str\_\_ 메소드가 정의되어 있지 않다면 대신 호출됨. 리턴값은

반드시 문자열이어야 함.

object.\_\_str\_\_(self) : 클래스 메소드, str(), print, \_\_repr\_\_ 메소드가 정의되어 있지 않더라도 호출되지 않음. 리턴

값은 반드시 문자열이어야 함.

object.\_\_cmp\_\_(self, other) : 클래스 메소드, self < other 일 경우 음의 정수, self == other 일 경우 0, self >

other 일 경우 양의 정수 리턴. 메소드가 정의되어 있지 않다면 객체의 '주소'로 비교를 함.

object.\_\_lt\_\_(self, other) : 클래스 메소드, self < other

object.\_\_le\_\_(self, other) : 클래스 메소드, self <= other

object.\_\_eq\_\_(self, other) : 클래스 메소드, self > other

object.\_\_ne\_\_(self, other) : 클래스 메소드, self >= other

object.\_\_gt\_\_(self, other) : 클래스 메소드, self == other

object.\_\_ge\_\_(self, other) : 클래스 메소드, self != other

### 위 비교함수가 cmp 보다 우선됨. ###

object.\_\_hash\_\_(self) : 클래스 메소드, 클래스 객체를 사전의 키로 이용할 때 필요. 이때 \_\_cmp\_\_ 나 \_\_eq\_\_ 메소드도

정의되야 함. 그리고 리턴값은 반드시 정수형이여야 함.

object.\_\_nonzero\_\_(self) : 클래스 메소드, bool(), 진리값(0, 1 포함)을 리턴해야 함. 메소드가 정의되어 있지 않으면

\_\_len\_\_ 메소드를 호출하고 이때 결과가 0이 아니면 True 를 리턴함. \_\_len\_\_도 정의되어 있지 않으면 True 를 리턴.

object.\_\_getattr\_\_(self, name) : 클래스 메소드, 정의되어 있지 않은 속성을 참조할 때 호출됨(위임 기법에서 사용됨).

name 은 문자열이며 호출한 속성의 이름을 가짐.

object.\_\_getattribute\_\_(self, name) : 클래스 메소드, \_\_getattr\_\_ 과 같으나 속성이 정의되어 있어도 호출됨. 그리

고 이 메소드를 사용하는 클래스는 object 클래스를 상속해야 함. 또, object.\_\_getattribute\_\_(self, name) 으로 사용

해서 값을 리턴해야 한다.

object.\_\_setattr\_\_(self, name, value) : 클래스 메소드, self.name = value 와 같이 속성에 치환(대입)이 일어날 때

호출됨. 그리고 치환(대입)은 일어나지 않으므로 이 메소드에서 다시 치환(대입) 시켜야함(단, 클래스 인스턴스의 사전

(\_\_dict\_\_)에 값을 저장시키는 방식이어야 함.)

object.\_\_delattr\_\_(self, name) : 클래스 메소드, del self.name 시에 호출됨. 그리고 삭제는 일어나지 않으므로 이

메소드에서 다시 삭제 시켜야함(단, 클래스 인스턴스의 사전 값을 없애는 방식이어야 함.)

object.\_\_call\_\_(self[, args...]) : 클래스 메소드, 함수처럼 호출할 수 있게 함.

callable(object) : 내장 함수, 객체가 호출 가능한지 알아봄.

\_\_slots\_\_ : 클래스 멤버 리스트. 클래스에 저장할 수 있는 속성 이름을 담고 있음. 그리고 이 클래스는 object 클래스를

상속해야 함. 단, 이 클래스에서는 \_\_dict\_\_ 는 사용되지 않음.

property([fget[, fset[, fdel[, doc]]]]) : 내장 함수, 멤버 값을 정의할 때 함수로 처리하도록 함. fget 은 값을 얻

을 때. fset 은 값을 쓸 때. fdel 은 값을 삭제할 때. doc 은 help 함수로 클래스를 호출할 때 보여지는 설명. 클래스는

object 를 상속해야함.

class.\_\_mro\_\_ : 클래스 멤버, 클래스의 탐색 순서를 튜플로 저장되는 멤버(오직, new-style 클래스에서만 작동)

class.mro() : 클래스 메소드, \_\_mro\_\_ 에 저장시키는 메소드. 오버라이딩될 수 있다(오직, new-style 클래스에서만 작동)

class cmd.Cmd([completekey[, stdin[, stdout]]]) : cmd 모듈의 클래스, 명령어 해석기의 단순한 프레임워크를 제공.

types : 모듈, 객체 타입의 이름을 정의한 모듈.

isinstance(object, classinfo) : 내장함수, object 가 classinfo 의 인스턴스이거나 classinfo 의 서브클래스일 경우

True 리턴.

issubclass(class, classinfo) : 내장함수, class 가 classinfo 의 서브클래스이면 True 리턴. 자기 자신도 자기 자신

의 서브클래스임.

class.\_\_bases\_\_ : 클래스 멤버, 클래스 객체의 슈퍼 클래스를 튜플로 가지는 멤버.

instance.\_\_class\_\_ : 인스턴스 멤버, 클래스 인스턴스가 속한 클래스를 가지는 멤버.

class.\_\_name\_\_ : 클래스 멤버, 클래스의 이름.

object.\_\_dict\_\_ : 클래스 멤버, 객체의 속성을 저장하는 사전. 여기서 키 값만을 빼내면 dir() 동일함(단, dir() 은 결

과가 정렬되어 있고, 슈퍼 클래스의 속성까지 표시함.)

super(type[, object-or-type]) : 내장함수, type 의 슈퍼 클래스에게 메소드 호출을 대신하는 객체를 리턴. type 의

\_\_mro\_\_ 속성을 사용하여 순서 결정.

super(type[, object-or-type]) : 내장함수, type 의 슈퍼 클래스에게 메소드 호출을 대신하는 객체를 리턴. type 의

\_\_mro\_\_ 속성을 사용하여 순서 결정.

im\_func : 사용자 정의 메소드 객체 속성, 함수 객체 속성.

im\_self : 사용자 정의 메소드 객체 속성, 클래스 인스턴스 객체 속성.

im\_class : 사용자 정의 메소드 객체 속성, 메소드가 정의된 클래스 속성.

\_\_doc\_\_ : 사용자 정의 메소드 객체 속성, 메소드 문서 문자열(im\_func.\_\_doc\_\_ 과 같음)

\_\_name\_\_ : 사용자 정의 메소드 객체 속성, 메소드 이름(im\_func.\_\_name\_\_ 과 같음)

\_\_module\_\_ : 사용자 정의 메소드 객체 속성, 메소드가 정의된 모듈의 이름.

sys.exc\_info() : sys 모듈, 발생한 예외의 종류, 에외의 값, traceback 정보 세 가지를 튜플로 리턴함.

\_\_debug\_\_ : 내장 상수, 디버그 모드의 여부를 가지는 상수. 파이썬이 -O 옵션을 가지고 실행 되면 이 값이 False 가 된다.

traceback.print\_exc([limit[, file]]) : traceback 모듈, 예외 정보를 출력하는 함수. print\_exception 함수의 속기

형임. 자세한 것은 traceback 모듈 참고.

class weakref.ref(object[, callback]) : weakref 모듈, 약함 참조 객체를 생성하는 함수. 객체가 존재하지 않으면

None 을 리턴함. callback 이 None 이 아닌 값으로 주어지면 참조하던 객체가 사라질 때 약한 참조 객체를 인자로 넘기면서 callback 을 호출함.

weakref.proxy(object[, callback]) : weakref 모듈, 프로식 객체를 생성함. weakref 객체에서와 같이 함수 형식을 사

용하지 않고 객체를 참조할 수 있음.

weakref.getweakrefcount(object) : weakref 모듈, 객체를 참조하고 있는 약한 참조 객체와 프록시 객체의 수를 리턴함.

weakref.getweakrefs(object) : weakref 모듈, 객체를 참조하고 있는 약한 참조 객체와 프로시 객체의 리스트를 리턴함.

class weakref.WeakValueDictionary([dict]) : weakref 모듈, 값으로 약한 참조 객체를 가지는 사전을 생성함. 약한 참조 객체가 참조하던 객체가 사라지면 자동으로 항목이 제거됨.

class weakref.WeakKeyDictionary([dict]) : weakref 모듈, 키로 약한 참조 객체를 가지는 사전을 생성함. 약한 참조

객체가 참조하던 객체가 사라지면 자동으로 항목이 제거됨.

iter(o[, sentinel]) : 내장함수, 반복자 객체를 리턴한다. 만약, 두번째 인자가 주어진다면 첫번째 인자는 호출될 수 있는 객체이어야 한다. 만약 리턴값이 두번째 인자와 같다면 StopIteration 이 발생한다.

iterator.next() : 클래스 메소드, 클래스로부터 next 메소드를 호출한다.

next(iterator[, default]) : 내장함수, 반복자로부터 다음 아이템을 반환한다. 디플트가 주어진다면 반복자가 종료될 때

리턴된다.

D.iterkeys() : 사전의 키에 대한 반복자를 리턴한다.

D.itervalues() : 사전의 값에 대한 반복자를 리턴한다.

D.iteritems() : 사전의 (키, 값) 튜플에 대한 반복자를 리턴한다.

itertools.chain(\*iterables) : itertools 모듈, 첫번째 객체부터 마지막 객체까지 요소들을 리턴하는 반복자를 생성한

다.

itertools.count([n]) : itertools 모듈, n 부터 시작하는 정수 값을 생성해 내는 반복자를 리턴한다. n이 주어지지 않으

면 기본값은 0 이다.

itertools.cycle(iterable) : itertools 모듈, 객체를 무한히 반복하는 반복자를 생성한다.

itertools.dropwhile(predicate, iterable) : itertools 모듈, predicate 가 참이 되는 데이터들은 버리다가 거짓이

되는 이후의 데이터들만 리턴하는 반복자를 생성한다.

itertools.takewhile(predicate, iterable) : itertools 모듈, predicate 가 참이 되는 데이터들을 취하다가 거짓이

되면 멈추는 반복자를 생성한다.

itertools.groupby(iterable[, key]) : itertools 모듈, 연속된 키들과 키들로부터 그룹화 시킨 그룹들을 리턴하는 반복

자를 생성한다. key 는 각 요소에 키 값을 취할 수 있는 함수이다. 일반적으로, 키들은 미리 정렬이 되어 있어야 한 그룹으로 묶일 수 있다.

operator.itemgetter(item[, args...]) : operator 모듈, item 에 있는 인덱스 값들을 얻는 호출 객체를 리턴함.

itertools.ifilter(predicate, iterable) : itertools 모듈, filter 와 같은 사용법이지만 반복자를 리턴.

itertools.imap(function, \*iterables) : itertools 모듈, map 과 같은 사용법이지만 반복자를 리턴.

itertools.izip(\*iterables) : itertools 모듈, zip과 같은 사용법이지만 반복자를 리턴.

itertools.starmap(function, iterable) : itertools 모듈, imap과 동일하나 반복자에서 얻어진 인수 튜플을 사용하여

사상 함수를 호출하는 것이 다름.

itertools.ifilterfalse(predicate, iterable) : itertools 모듈, ifilter 와 동일하나 조건이 거짓이 되는 자료를

리턴.

itertools.islice(iterable[, start], stop[, step]) : itertools 모듈, 슬라이싱 사용법과 같다.

itertools.repeat(object[, times]) : itertools 모듈, 객체를 지정된 횟수만큼 리턴하는 반복자를 생성. 횟수가 지정되

지 않으면 무한히 반복.

itertools.tee(iterable[, n=2]) : itertools 모듈, n 개의 반복자를 생성.

os.listdir(path) : os 모듈, path 경로의 디렉토리의 전체 목록 리스트를 리턴.

dircache.listdir(path) : dircache 모듈, os.listdir() 로 부터 얻어낸 경로의 리스트를 리턴한다. 단, 경로의 변경이

없으면 디렉토리 구조를 다시 읽지 않는다.

dircache.annotate(head, list) : dircache 모듈, head/list 의 요소가 디렉토리일 경우 디렉토리 명에 '/' 을 추가한

다.

os.access(path, mode) : os 모듈, 파일의 허가권을 알아보는 함수. os.F\_OK는 존재 여부, os.R\_OK는 일기 권한,

os.W\_OK는 쓰기 권한, os.X\_OK는 실행 권한.

os.chmod(path, mode) : os 모듈, 파일의 허가권을 변경하는 함수.

os.rename(src, dst) : os 모듈, 파일이 이름 변경. 파일 이동은 파일 이름에 경로를 지정하여 이동 가능.

shutil.copyfile(src, dst) : shutil 모듈, 파일을 복사하는 함수.

os.remove(path) : os 모듈, 파일을 삭제함. 단, 디렉토리일 경우 OSError 발생.

os.link(src, dst) : os 모듈, 하드 링크를 생성.

os.symlink(src, dst) : os 모듈, 심볼릭 링크를 생성.

os.readlink(path) : os 모듈, 심볼릭 링크의 정보를 읽음.

os.utime(path, times) : os 모듈, 파일의 접근 시간과 수정 시간을 조절. times 가 None 이면 접근 시간과 수정 시간을 현재 시간으로 조정. times 값은 (atime, mtime) 형태의 튜플임.

os.stat(path) : os 모듈, 파일의 접근 시간과 수정 시간을 읽어 옮.

os.chown(path, uid, gid) : os 모듈, 파일 소유자를 바꿈.

os.stat(path) : os 모듈, 파일에 관한 자세한 내용들을 리턴. 리턴 값은 stat 모듈의 심볼들을 통해서 해석할 수 있음.

tempfile : tempfile 모듈, 임시 파일 이름을 생성시킬 수 있는 모듈.

os.chdir(path) : os 모듈, 작업하고 잇는 디렉토리 변경.

os.getcwd() : os 모듈, 작업하고 있는 디렉토리 확인.

os.mkdir(path[, mode]) : os 모듈, 디렉토리 생성. 기본 허가권은 0777(octal)임.

os.makedirs(path[, mode]) : os 모듈, 재귀적으로 디렉토리 생성.

os.rmdir(path) : os 모듈, 내용이 없는 디렉토리 삭제.

os.removedirs(path) : os 모듈, 재귀적으로 디렉토리를 삭제함.

shutil.rmtree(path[, ignore\_errors[, onerror]]) : shutil 모듈, 전체 디렉토리 구조를 모두 삭제함.

shutil.copytree(src, dst[, symlinks=False[, ignore=None]]) : shutil 모듈, 디렉토리 구조를 복사함.

max(iterable[, args...][, key]) : 내장 함수, iterable 한 개만 인자로 사용되면 그 중에서 가장 큰 값을 리턴, 여러

개의 인자가 들어오면 인자 중에서 가장 큰 값을 리턴. key에는 함수가 사용되고, 인자들을 함수에 대입하여 나온 리턴값들 중 최대를 리턴하는 인자를 리턴한다.

min(iterable[, args...][, key]) : 내장 함수, iterable 한 개만 인자로 사용되면 그 중에서 가장 작은 값을 리턴, 여

러 개의 인자가 들어오면 인자 중에서 가장 작은 값을 리턴. key에는 함수가 사용되고, 인자들을 함수에 대입하여 나온 리턴값들 중 최소를 리턴하는 인자를 리턴한다.

os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]]) : os 모듈, 디렉토리를 재귀적으로 탐색함.

top 은 루트 디렉토리, topdown 은 출력 순서, onerror 는 예외일 때 호출되는 함수, followlinks 는 False 일 때 상대

경로의 문제를 해결하지 않음. 리턴값은 (dirpath, dirnames, filenames) 의 튜플값.

os.path.expanduser(path) : os.path 모듈, 사용자의 홈 디렉토리 경로명을 확장해줌.

os.path.expandvars(path) : os.path 모듈, 쉘 변수를 확장해줌.

os.path.normcase(path) : os.path 모듈, 파일 이름을 정규화해줌.

os.path.basename(path) : os.path 모듈, 파일명만 추출.

os.path.dirname(path) : os.path 모듈, 디렉토리 경로 추출.

os.path.splitdrive(path) : os.path 모듈, 드라이브명 분리(윈도우용).

fnmatch.fnmatch(filename, pattern) : fnmatch 모듈, 파일 이름이 주어진 패턴(와일드카드)와 일치하는지 확인. 리턴값은 1, 0.

random.uniform(a, b) : random 모듈, a<=b 인 경우 a<=N<=b, b<a 인 경우 b<=N<=a 에서의 임의의 소수 N을 리턴.

slice([start,] stop [, step]) : 내장 함수, 슬라이싱할 때 사용되는 슬라이스 객체를 넘겨줌. 객체에 대해서 슬라이싱이

사용된다면 메소드(슬라이스 객체)와 같이 사용된다.