스마트 모빌리티 프로그래밍

Ch 4. 파이썬 고급 자료형과 연산

영남대학교 정보통신공학과

교수 김 영 탁

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

Outline

- 문자열 (string)
- bytes, bytearray, memoryview
- 리스트(list)
- 튜플(tuple)
- 딕셔너리(dict)
- 집합(set), 동결집합(frozenset)

파이썬 기본 자료형

◆ Python build-in data type

자료형		데이터/원소의 변경 가능성 (mutable)	원소의 열거 가능성 (iterable)	사용 예시
부울(boolean)	bool	no	no	True, False
숫자(numeric)	int	no	no	정수, a = 100
	float	no	yes	실수, x = 123.4567
	complex	no	yes	복소수, c = 15 +23j
시퀀스(sequence)	str	no	yes	문자열
	bytes	no	yes	바이츠
	bytearray	yes	yes	바이트 배열
	memoryview	no/yes	yes	메모리뷰
	list	yes	yes	리스트, L = [0, 1, 2 , 3, 4]
	tuple	no	yes	튜플, T = (Kim, 29, Korea, Male)
	range	no	yes	for i in range(0, 10, 1):
매핑(mapping)	dict	yes	yes	{ 0 : “Zero”, 1 : “One”, 2 : “Two”}
집합(set)	set	yes	yes	digits = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
	frozenset	no	yes	집합의 초기 설정 후, 추가적인 원소 변경 불가

문자열 - str (string)

컴퓨터에서 사용되는 문자 자료형 (1)

◆ ASCII (American Standard Code for Information Interchange)

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
0x30	0 (0x30)	1 (0x31)	2	3	4	5	6	7	8	9 (0x39)	: (0x3A)	;	<	=	>	?
0x40	@	A (0x41)	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z (0x5A)	[(0x5B)	\	]	^	_
0x60	`	a (0x61)	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z (0x7A)	{ (0x7B)	|	}	~	DEL

컴퓨터에서 사용되는 문자 자료형 (2) - unicode

◆ Unicode

- https://home.unicode.org/
- **유니코드**(영어: Unicode)는 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준
- 유니코드의 목적은 현존하는 문자 인코딩 방법들을 모두 유니코드로 통합, 교체하려는 것임
- 유니코드에서 한글 코드 범위는 AC00 ~ D7AF 임 (https://jjeong.tistory.com/696)

U+AC00 to U+AD00

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
UTF8: 234, 176, 128;	UNICODE: AC0	가	각	갂	갃	간	갅	갆	갇	갈	갉	갊	갋	갌	갍	갎	갏
UTF8: 234, 176, 144;	UNICODE: AC1	감	갑	값	갓	갔	강	갖	갗	갘	같	갚	갛	개	객	갞	갟
UTF8: 234, 176, 160;	UNICODE: AC2	갠	갡	갢	갣	갤	갥	갦	갧	갨	갩	갪	갫	갬	갭	갮	갯
UTF8: 234, 176, 176;	UNICODE: AC3	갰	갱	갲	갳	갴	갵	갶	갷	갸	갹	갺	갻	갼	갽	갾	갿
UTF8: 234, 177, 128;	UNICODE: AC4	걀	걁	걂	걃	걄	걅	걆	걇	걈	걉	걊	걋	걌	걍	걎	걏
UTF8: 234, 177, 144;	UNICODE: AC5	걐	걑	걒	걓	걔	걕	걖	걗	걘	걙	걚	걛	걜	걝	걞	걟
UTF8: 234, 177, 160;	UNICODE: AC6	걠	걡	걢	걣	거	걥	걦	걧	걨	걩	걪	걫	걬	걭	걮	걯
UTF8: 234, 177, 176;	UNICODE: AC7	걱	걲	걳	건	건	걶	걷	걸	걹	걺	걻	걼	걽	걾	걿	검
UTF8: 234, 178, 128;	UNICODE: AC8	겁	겂	것	겄	것	겆	겇	겈	겉	겊	겋	게	겍	겎	겏	겐
UTF8: 234, 178, 144;	UNICODE: AC9	겑	겒	겓	겔	겕	겖	겗	겘	겙	겚	겛	겜	겝	겞	겟	겠
UTF8: 234, 178, 160;	UNICODE: ACA	겡	겢	겣	겤	겥	겦	겧	겨	격	겪	겫	견	겭	겮	겯	결
UTF8: 234, 178, 176;	UNICODE: ACB	겸	겺	겻	겼	경	곀	곁	계	곃	계	곅	곆	곇	곈	곉	곊
UTF8: 234, 179, 128;	UNICODE: ACC	곋	곌	곍	곎	곏	곐	곑	곒	곓	곔	곕	곖	곗	곘	곙	곚
UTF8: 234, 179, 144;	UNICODE: ACD	곛	곜	곝	곞	곟	고	곡	곢	곣	곤	곥	곦	곧	골	곩	곪
UTF8: 234, 179, 160;	UNICODE: ACE	곫	곬	곭	곮	곯	곰	곱	곲	곳	공	곴	과	곶	곷	곸	곹
UTF8: 234, 179, 176;	UNICODE: ACF	곺	곻	과	곽	곾	곿	관	괁	괂	괃	괄	괅	괆	괇	괈	괉
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

U+D700 to U+D800

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
UTF8: 237, 156, 128;	UNICODE: D70	퀀	퀘	퀜	퀠	퀤	퀥	퀦	퀧	퀨	퀩	퀪	퀫	퀬	퀭	퀮	퀯
UTF8: 237, 156, 144;	UNICODE: D71	퀰	퀱	퀲	퀳	퀴	퀵	퀶	퀷	퀸	퀹	퀺	퀻	퀼	퀽	퀾	퀿
UTF8: 237, 156, 160;	UNICODE: D72	큀	큁	큂	큃	큄	큅	큆	큇	큈	큉	큊	큋	큌	큍	큎	큏
UTF8: 237, 156, 176;	UNICODE: D73	큑	큒	큓	큔	큕	큖	큗	큘	큙	큚	큛	큜	큝	큞	큟	큠
UTF8: 237, 157, 128;	UNICODE: D74	큡	큢	큣	큤	큥	큦	큧	큨	큩	큪	큫	크	큭	큮	큯	큰
UTF8: 237, 157, 144;	UNICODE: D75	큱	큲	큳	클	큵	큶	큷	클	큹	큺	큻	큼	큽	큾	큿	킀
UTF8: 237, 157, 160;	UNICODE: D76	킁	킂	킃	킄	킅	킆	킇	킈	킉	킊	킋	킌	킍	킎	킏	킐
UTF8: 237, 157, 176;	UNICODE: D77	킑	킒	킓	킔	킕	킖	킗	킘	킙	킚	킛	킜	킝	킞	킟	킠
UTF8: 237, 158, 128;	UNICODE: D78	킡	킢	킣	키	킥	킦	킧	킨	킩	킪	킫	킬	킭	킮	킯	킰
UTF8: 237, 158, 144;	UNICODE: D79	킱	킲	킳	킴	킵	킶	킷	킸	킹	킺	킻	킼	킽	킾	킿	타
UTF8: 237, 158, 160;	UNICODE: D7A	탁	탂	탃	탄	□	□	□	□	□	□	□	□	□	□	□	□
UTF8: 237, 158, 176;	UNICODE: D7B	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅘ	ㅙ	ㅚ	ㅛ	ㅜ	ㅝ	ㅞ	ㅟ
UTF8: 237, 159, 128;	UNICODE: D7C	ㅠ	ㅡ	ㅢ	ㅣ	ㅤ	ㅥ	ㅦ	□	□	□	□	ㄵ	ㄶ	ㄷ	ㄸ	ㄹ
UTF8: 237, 159, 144;	UNICODE: D7D	ㄺ	ㄻ	ㄼ	ㄽ	ㄾ	ㄿ	ㅀ	ㅁ	ㅂ	ㅃ	ㅄ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ
UTF8: 237, 159, 160;	UNICODE: D7E	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ	ㅏ	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅘ	ㅙ
UTF8: 237, 159, 176;	UNICODE: D7F	ㅚ	ㅛ	ㅜ	ㅝ	ㅞ	ㅟ	ㅠ	ㅡ	ㅢ	ㅣ	ㆍ	ㆎ	□	□	□	□
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

문자열 (str) 관련 내장형 함수

◆ Str 자료형의 내장형 함수

str 자료형의 내장형 함수	설 명
len(object)	시퀀스 또는 콜렉션 자료형인 object의 원소 개수를 반환
chr(i)	문자 코드표에서 i-번째 문자 예) chr(0x41) = 'A', chr (0x61) = 'a' i의 범위: 0<= i <= 0x10FFFF
ord(c)	유니코드 문자 c의 코드표에서의 위치 예) ord('a') = 97 (0x61), ord('\u2020') = 8224
repr(object)	객체 object의 출력 문자열을 반환
ascii(object)	객체 object의 출력 가능한 ASCII 문자열을 반환
eval(expression)	인수로 전달된 스크립트 프로그램 expression을 검사하고 실행
exec(object [, globals [, locals]])	object로 전달된 파이썬 프로그램 코드를 실행 object는 문자열이거나 객체
input([prompt])	표준 출력장치에 prompt (안내문구)를 출력하고, 표준 입력장치로부터 한 줄을 읽어 문자열로 변환하여 반환

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

str 자료형 – len()

◆ str() – len()

```
# str - len()

str_abc = "ABCDEFG"
print("str_abc = ", str_abc)
print("len(str_abc) = ", len(str_abc))

L_str = list(str_abc)
print("L_str = ", L_str)
for i in range(len(L_str)):
    print("L_str[{:2}] = {}".format(i, L_str[i]))
```

```
str_abc =  ABCDEFG
len(str_abc) =  7
L_str =  ['A', 'B', 'C', 'D', 'E', 'F', 'G']
L_str[ 0] = A
L_str[ 1] = B
L_str[ 2] = C
L_str[ 3] = D
L_str[ 4] = E
L_str[ 5] = F
L_str[ 6] = G
```

str 자료형 – chr(), ord()

◆ str – chr(), ord()

```
# Handling str data type

L = [0x41, 0x42, 0x43, 0x44, 0x45]
for i in range(len(L)):
    print("chr({}) = {}".format(L[i], chr(L[i])))

STR = ['A', 'B', 'C', 'D', 'E']
for ch in range(len(STR)):
    print("ord({}) = {}".format(STR[ch], ord(STR[ch])))
```

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

```
chr(65) = A
chr(66) = B
chr(67) = C
chr(68) = D
chr(69) = E
ord(A) = 65
ord(B) = 66
ord(C) = 67
ord(D) = 68
ord(E) = 69
```

str 자료형 – repr(), ascii()

```
# str - repr(), ascii()

i = 100
print("i = ", i)
print("type(i) = ", type(i))
repr_i = repr(i)
print("repr(i) = ", repr_i)
print("type(repr_i) = ", type(repr_i))

ascii_i = ascii(i)
print("ascii(i) = ", ascii_i)
print("type(ascii_i) = ", type(ascii_i))

L = [1, 2, 3, 4]
print("L = ", L)
print("type(L) = ", type(L))
repr_L = repr(L)
print("repr(L) = ", repr_L)
print("type(repr_L) = ", type(repr_L))

ascii_L = ascii(L)
print("ascii(L) = ", ascii_L)
print("type(ascii_L) = ", type(ascii_L))
```

```
i =  100
type(i) =  <class 'int'>
repr(i) =  100
type(repr_i) =  <class 'str'>
ascii(i) =  100
type(ascii_i) =  <class 'str'>
L =  [1, 2, 3, 4]
type(L) =  <class 'list'>
repr(L) =  [1, 2, 3, 4]
type(repr_L) =  <class 'str'>
ascii(L) =  [1, 2, 3, 4]
type(ascii_L) =  <class 'str'>
```

str 자료형 – eval(), exec()

```
# Handling str data type - eval(), exec()

a = 10
print("a = {}".format(a))
print("eval('a+1') = {}".format(eval("a+1")))

sum = 0
command = "for i in range({}): sq = i*i; print('%2d %3d'%(i, sq))".format(a)
print("command = ", command)
exec(command)
```

```
a = 10
eval('a+1') = 11
command =  for i in range(10): sq = i*i; print('%2d %3d'%(i, sq))
 0   0
 1   1
 2   4
 3   9
 4  16
 5  25
 6  36
 7  49
 8  64
 9  81
```

str 자료형 – input()

```
input integer string : 12345
str_i = 12345, type(str_i) = <class 'str'>, type(i) = <class 'int'>, i = 12345
input hexadecimal string : 0x100
str_hex = 0x100, type(str_hex) = <class 'str'>, type(hx) = <class 'int'>, hx = 2
56, hx_str=0x100
input float string : 1.2345
str_float = 1.2345, type(str_float) = <class 'str'>, type(f) = <class 'float'>,
f = 1.2345
input complex string : 10.5-21.78j
str_cmplx = 10.5-21.78j, type(str_cmplx) = <class 'str'>, type(c) = <class 'comp
lex'>, c = (10.5-21.78j)
```

```
# str - input(), int(), float(), complex()

str_i = input("input integer string : ")
i = int(str_i)
print("str_i = {}, type(str_i) = {}, type(i) = {}, i = {}".format(str_i, type(str_i), type(i), i))
str_hex = input("input hexadecimal string : ")
hx = int(str_hex, 16) #convert hexadecimal string into integer
hx_str = hex(hx)
print("str_hex = {}, type(str_hex) = {}, type(hx) = {}, hx = {}, hx_str={}"\
      .format(str_hex, type(str_hex), type(hx), hx, hx_str))
str_float = input("input float string : ")
f = float(str_float)
print("str_float = {}, type(str_float) = {}, type(f) = {}, f = {}".format(str_float, type(str_float), type(f), f))
str_cmplx = input("input complex string : ")
c = complex(str_cmplx)
print("str_cmplx = {}, type(str_cmplx) = {}, type(c) = {}, c = {}".format(str_cmplx, type(str_cmplx), type(c), c))
```

문자열 관련 함수 (1)

문자열 str 함수	의미
str.lower()	영문 소문자로 변환
str.upper()	영문 대문자로 변환
str.swapcase()	대문자를 소문자로, 소문자를 대문자로 변환
str.capitalize()	대문자로 변환
str.title()	각 단어의 첫문자를 대문자로 변환
str.center(width [, fillchar])	지정된 구역의 중앙 정렬, fillchar로 채우기
str.ljust(width [, fillchar])	지정된 구역의 좌측 정렬, fillchar로 채우기
str.rjust(width [, fillchar])	지정된 구역의 우측 정렬, fillchar로 채우기
str.format(*args, **kwargs)	지정된 포맷으로 문자열 생성
str.count(sub[, start[, end]])	문자열에 포함된 서브문자열의 발생 회수
str.encode(endocding="utf-8", errors="strict")	지정된 문자표시 (코딩) 방식으로 인코딩
str.startswith(prefix[, start[, end]])	문자열이 지정된 접두어 (prefix)로 시작되는가를 확인
str.endswith(suffix[, start[, end]])	문자열이 지정된 접미사 (suffix)로 끝나는지 확인
str.find(sub[, start[, end]])	문자열에서 지정된 패턴 문자열을 찾기
str.rfind(sub[, start[, end]])	문자열에서 지정된 패턴 문자열을 역순으로 찾기
str.index(sub[, start[, end]])	문자열에서 지정된 패턴의 위치 (인덱스) 반환

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

문자열 관련 함수 (2)

문자열 str 함수	의미
str.isalnum(), str.isalpha() str.isdecimal(), str.isdigit() str.isidentifier(), str.islower(), str.isnumeric(), str.isspace(), str.isprintable(), str.istitle(), str.isupper()	문자열이 특정 양식 (10진수, 영문자, 소문자, 대문자, 출력 가능, title 등)인가를 확인하여 결과 반환
str.join(iterable)	문자열을 결합
str.strip([chars])	문자열에 포함된 공란(space, tab 등)을 삭제
str.lstrip([chars])	문자열이 시작되기 전에 포함된 공란(space, tab 등)을 삭제
str.rstrip([chars])	문자열이 끝난 후에 포함된 공란(space, tab 등)을 삭제
str.partition(sep)	문자열을 sep 문자 위치에서 분할
str.rpartition(sep)	문자열의 끝 부분으로부터 역순으로 맨 먼저 나타나는 sep 문자 위치에서 분할
str.replace(old, new[, count])	문자열의 특정 문자를 다른 문자로 변환
str.split(sep=None, maxsplit=-1)	문자열을 분할
str.rsplit(sep=None, maxsplit=-1)	문자열을 역순으로 분할
str.splitlines([keepends])	문자열을 줄 바꿈에 따라 분할
str.zfill(width)	문자열을 지정된 공간에 출력하며, 빈 공간을 0으로 채움
static str.maketrans(x[, y[, z]])	문자열 변환표 생성
str.translate(table)	문자열 변환표에 따라 변환

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

str 자료형 – lower(), upper(), capitalize(), title()

```
# str - upper(), lower(), capitalize(), title()

s1 = "a sample string"
s1_upper = s1.upper()
print("s1 = {}, s1.upper() => {}".format(s1, s1_upper))

s2 = "Another Sample String"
s2_lower = s2.lower()
print("s2 = {}, s2.lower() => {}".format(s2, s2_lower))

s3 = "test capitialize() function"
s3_capitalize = s3.capitalize()
print("s3 = {}, s3.capitalize() => {}".format(s3, s3_capitalize))

s4 = "test title() function"
s4_title = s4.title()
print("s4 = {}, s4.title() => {}".format(s4, s4_title))
```

```
s1 = a sample string, s1.upper() => A SAMPLE STRING
s2 = Another Sample String, s2.lower() => another sample string
s3 = test capitialize() function, s3.capitalize() => Test capitialize() function
s4 = test title() function, s4.title() => Test Title() Function
```

str 자료형 – center(), ljust(), rjust()

```
# str - center(), ljust(), rjust()

s = "a sample string"
s_len = len(s)
print("s = {}, len(s) = {}".format(s, s_len))
s_center = s.center(s_len+10)
print("s.center(s_len + 10)      => {}".format(s_center))

s_center_fill_underline = s.center(s_len+10, '_')
print("s.center(s_len + 10, '_') => {}".format(s_center_fill_underline))

s_ljust_fill_underline = s.ljust(s_len+10, '_')
print("s.ljust(s_len + 10, '_')  => {}".format(s_ljust_fill_underline))

s_rjust_fill_underline = s.rjust(s_len+10, '_')
print("s.rjust(s_len + 10, '_')  => {}".format(s_rjust_fill_underline))
```

```
s = a sample string, len(s) = 15
s.center(s_len + 10)      =>      a sample string
s.center(s_len + 10, '_') => _____a sample string_____
s.ljust(s_len + 10, '_')  => a sample string__________
s.rjust(s_len + 10, '_')  => __________a sample string
```

str 자료형 – format()

```
# str - format(), keyword arguments
Goods = ["Sedan", "Sport Car", "Family Car", "Off-road", "Jeep", "Toy Ferrari"]
Prices = [25000, 47123.45678, 35123.45678, 40000.1234, 45000.12345, 25]

for i in range(len(Goods)):
    s = "The price of {good:12} is {price:10.2f} dollars.".format(good=Goods[i], price=Prices[i])
    print(s)
```

```
The price of Sedan        is   25000.00 dollars.
The price of Sport Car    is   47123.46 dollars.
The price of Family Car   is   35123.46 dollars.
The price of Off-road     is   40000.12 dollars.
The price of Jeep         is   45000.12 dollars.
The price of Toy Ferrari  is      25.00 dollars.
```

str 자료형 – encode()

```
# str - encode()

s_decimal = "0123456789"
print("s_decimal = ", s_decimal)
s_decimal_ascii = s_decimal.encode(encoding = "ascii")
print("s_decimal.encode(encoding = 'ascii') = ", s_decimal_ascii)
print("type(s_decimal_ascii) = ", type(s_decimal_ascii))
print("len(s_decimal_ascii) = ", len(s_decimal_ascii))
print()

s_alphaNum = "abcdefg0123456789"
print("s_alphaNum = ", s_alphaNum)
s_alphaNum_utf_16 = s_alphaNum.encode(encoding = "utf_16")
print("s_alphaNum.encode(encoding = 'utf_16') = ", s_alphaNum_utf_16)
print("type(s_alphaNum_utf_16) = ", type(s_alphaNum_utf_16))
print("len(s_alphaNum_utf_16) = ", len(s_alphaNum_utf_16))
print()

s_kr = "가나다라마바사"
print("s_kr = ", s_kr)
s_kr_cp949 = s_kr.encode(encoding = "cp949")
print("s_kr.encode(encoding = 'cp949')\
    = ", s_kr_cp949)
print("type(s_kr_cp949) = ", type(s_kr_cp949))
print("len(s_kr_cp949) = ", len(s_kr_cp949))
```

```
s_decimal =  0123456789
s_decimal.encode(encoding = 'ascii') =  b'0123456789'
type(s_decimal_ascii) =  <class 'bytes'>
len(s_decimal_ascii) =  10

s_alphaNum =  abcdefg0123456789
s_alphaNum.encode(encoding = 'utf_16') =  b'\xff\xfea\x00b\x00c\x00d\x00e\x00f\x
00g\x000\x001\x002\x003\x004\x005\x006\x007\x008\x009\x00'
type(s_alphaNum_utf_16) =  <class 'bytes'>
len(s_alphaNum_utf_16) =  36

s_kr =  가나다라마바사
s_kr.encode(encoding = 'cp949') =  b'\xb0\xa1\xb3\xaa\xb4\xd9\xb6\xf3\xb8\xb6\xb
9\xd9\xbb\xe7'
type(s_kr_cp949) =  <class 'bytes'>
len(s_kr_cp949) =  14
```

str 자료형 – startwith(), endwith()

```
# str - startswith, endswith

s = "Hello World! I am glad to meet you."
start_word = "Hello"
end_word = "."
print("s = ", s)
print("s starts with {} = {}".format(start_word, s.startswith(start_word)))
print("s ends with {} = {}".format(end_word, s.endswith(end_word)))
```

```
s =  Hello World! I am glad to meet you.
s starts with Hello = True
s ends with . = True
```

str 자료형 – find(), rfind(), index()

```
# str - find, rfind, index

s = "Hello World! I am glad to meet you."
kw_1 = "m"
kw_2 = "W"
print("s = ", s)
print("kw_1 = {}, kw_2 = {}".format(kw_1, kw_2))
print("s.find({}) = {}".format(kw_1, s.find(kw_1)))
print("s.rfind({}) = {}".format(kw_1, s.rfind(kw_1)))
print("s.find({}) = {}".format(kw_2, s.find(kw_2)))
print("s.index({}) = {}".format(kw_2, s.index(kw_2)))
```

```
s =  Hello World! I am glad to meet you.
kw_1 = m, kw_2 = W
s.find(m) = 16
s.rfind(m) = 26
s.find(W) = 6
s.index(W) = 6
```

str 자료형 – isalnum(), isdecimal(), isprintable()

```
# str - isalnum, isdecimal, isprintable

s_1 = "0123456789"
s_2 = "ABCDEFG"

print("s_1 = {}".format(s_1))
print("s_1.isalpha() = {}".format(s_1.isalpha()))
print("s_1.isdecimal() = {}".format(s_1.isdecimal()))
print("s_1.isnumeric() = {}".format(s_1.isnumeric()))
print("s_1.isalnum() = {}".format(s_1.isalnum()))
print("s_1.isprintable() = {}".format(s_1.isprintable()))
print()

print("s_2 = {}".format(s_2))
print("s_2.isalpha() = {}".format(s_2.isalpha()))
print("s_2.isdecimal() = {}".format(s_2.isdecimal()))
print("s_2.isnumeric() = {}".format(s_2.isnumeric()))
print("s_2.isalnum() = {}".format(s_2.isalnum()))
print("s_2.isprintable() = {}".format(s_2.isprintable()))
print("s_2.istitle() = {}".format(s_2.istitle()))
print("s_2.isupper() = {}".format(s_2.isupper()))
```

```
s_1 = 0123456789
s_1.isalpha() = False
s_1.isdecimal() = True
s_1.isnumeric() = True
s_1.isalnum() = True
s_1.isprintable() = True

s_2 = ABCDEFG
s_2.isalpha() = True
s_2.isdecimal() = False
s_2.isnumeric() = False
s_2.isalnum() = True
s_2.isprintable() = True
s_2.istitle() = False
s_2.isupper() = True
```

str 자료형 – join()

```
# str - join, lstrip, rstrip

A = ["Welcome", "to", "my", "world"]
B = ["My", "dear", "friends"]

print("A = ", A)
print("B = ", B)

s1 = "".join(A)
print("s1 = ", s1)
s2 = "".join(B)
print("s2 = ", s2)
s3 = s1 + s2
print("s3 = ", s3)

s4 = " ".join(A)
print("s4 = ", s4)
s5 = " ".join(B)
print("s5 = ", s5)
s6 = s4 + "! " + s5 + "!"
print("s6 = ", s6)
```

```
A =  ['Welcome', 'to', 'my', 'world']
B =  ['My', 'dear', 'friends']
s1 =  Welcometomyworld
s2 =  Mydearfriends
s3 =  WelcometomyworldMydearfriends
s4 =  Welcome to my world
s5 =  My dear friends
s6 =  Welcome to my world! My dear friends!
```

str 자료형 – strip, lstrip, rstrip

```
# str - strip, lstrip, rstrip

s = "     Welcome to my world!  My dear friends!     "
print("s = ", s)

s1 = s.strip()
print("s.strip()  = ", s1)

s2 = s.lstrip()
print("s.lstrip() = ", s2)

s3 = s.rstrip()
print("s.rstrip() = ", s3)
```

```
s =       Welcome to my world!  My dear friends!
s.strip()  =  Welcome to my world!  My dear friends!
s.lstrip() =  Welcome to my world!  My dear friends!
s.rstrip() =       Welcome to my world!  My dear friends!
```

str 자료형 – partition(), rpartition(), replace()

```
# str - partition, rpartition, replace

fruits = "apple banana grape orange"
print("fruits = ", fruits)

f1 = fruits.partition(' ')
print("fruits.partition(' ') = ", f1)

f2 = fruits.rpartition(' ')
print("fruits.rpartition(' ') = ", f2)

f3 = fruits.replace(' ', ', ')
print("fruits.replace(' ', ', ') = ", f3)
```

```
fruits =  apple banana graph orange
fruits.partition(' ') =  ('apple', ' ', 'banana graph orange')
fruits.rpartition(' ') =  ('apple banana graph', ' ', 'orange')
fruits.replace(' ', ', ') =  apple, banana, graph, orange
```

str 자료형 – split(), rsplit(), splitlines()

```
# str - split, rsplit, splitlines

animals = "lion tiger giraffe monkey rabbit dog cat"
print("animals = ", animals)

animals_split = animals.split(" ")
print("animals.split(' ') = ", animals_split)
print("type(animals_split) = ", type(animals_split))

animals_split_mxsplt_1 = animals.split(" ", maxsplit = 1)
print("animals.split(' ', maxsplit=1) = ", animals_split_mxsplt_1)

animals_rsplit = animals.rsplit(" ")
print("animals.rsplit(' ') = ", animals_rsplit)

animals_rsplit_mxsplt_1 = animals.rsplit(" ", maxsplit = 1)
print("animals.rsplit(' ', maxsplit = 1) = ", animals_rsplit_mxsplt_1)

s = "Welcome to my world! \n My dear friends! \n I am so glad to see you guys!"
s_splitlines = s.splitlines()
print("s = ", s)
print("s.splitlines() = ", s_splitlines)
```

```
animals =  lion tiger giraffe monkey rabbit dog cat
animals.split(' ') =  ['lion', 'tiger', 'giraffe', 'monkey', 'rabbit', 'dog', 'c
at']
type(animals_split) =  <class 'list'>
animals.split(' ', maxsplit=1) =  ['lion', 'tiger giraffe monkey rabbit dog cat'
]
animals.rsplit(' ') =  ['lion', 'tiger', 'giraffe', 'monkey', 'rabbit', 'dog', '
cat']
animals.rsplit(' ', maxsplit = 1) =  ['lion tiger giraffe monkey rabbit dog', 'c
at']
s =  Welcome to my world!
 My dear friends!
 I am so glad to see you guys!
s.splitlines() =  ['Welcome to my world! ', ' My dear friends! ', ' I am so glad
 to see you guys!']
```

str 자료형 – zfill(), replace(), maketrans(), translate()

```
# str - zfill

word = "hello"
word_zfill = word.zfill(10)

print("word = ", word)
print("word.zfill(10) = ", word_zfill)
```

```
word =  hello
word.zfill(10) =  00000hello
```

```
# str - replace, transplate

shapes = "circle, triangle, rectangle, square, pentagon, hexagon"
shapes_replace = shapes.replace('g', 'G', 3)
print("shapes = ", shapes)
print("shapes.replace('g', 'G', 3) = ", shapes_replace)

transTbl = str.maketrans('t', 'T', 'x')
shapes_trans = shapes.translate(transTbl)
print("shapes.translate(transTbl) = ", shapes_trans)
```

```
shapes =  circle, triangle, rectangle, square, pentagon, hexagon
shapes.replace('g', 'G', 3) =  circle, trianGle, rectanGle, square, pentaGon, hexagon
shapes.translate(transTbl) =  circle, Triangle, recTangle, square, penTagon, heagon
```

이스케이프 시퀀스 (escape sequence)

Escape Sequence	의미
\\	역슬래시 (backslash)
\'	작은 따옴표 (Single quote)
\"	큰 따옴표 (Double quote)
\a	경고음 (ASCII bell (BEL))
\b	한칸 뒤로가기 (ASCII Backspace (BS))
\f	한 페이지 단위 전진 (ASCII Formfeed (FF))
\n	줄 바꿈 (ASCII Linefeed (LF))
\r	줄의 맨 처음으로 이동 (ASCII Carriage Return (CR))
\t	수평 탭 (ASCII Horizontal Tab (TAB))
\v	수직 탭 (ASCII Vertical Tab (VT))
\ooo	8진법 표시 (Octal digit ooo)
\xhh	16진법 표시 (Hexa-decimal digit hh)
\N{name}	유니코드에서 정의되어 있는 유니코드 문자열 (예: BEL, NULL, NUL, TAB)
\uxxxx	16-bit 16진수 데이터 xxxx
\Uxxxxxxxx	32-bit 16진수 데이터 xxxxxxxx

bytes, bytearray, memoryview

bytes 자료형

◆ 바이츠 객체 생성

```
# bytes

B0 = bytes()
B1 = bytes(10)
B2 = bytes(range(10))
B3 = bytes(b"ABCDEFG")
print("bytes()          : {}".format(B0))
print("bytes(10)        : {}".format(B1))
print("bytes(range(10)) : {}".format(B2))
print("bytes(b'ABCDEFG') : {}".format(B3))
```

```
bytes()          : b''
bytes(10)        : b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
bytes(range(10)) : b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t'
bytes(b'ABCDEFG') : b'ABCDEFG'
```

리스트를 사용한 바이츠 객체 생성

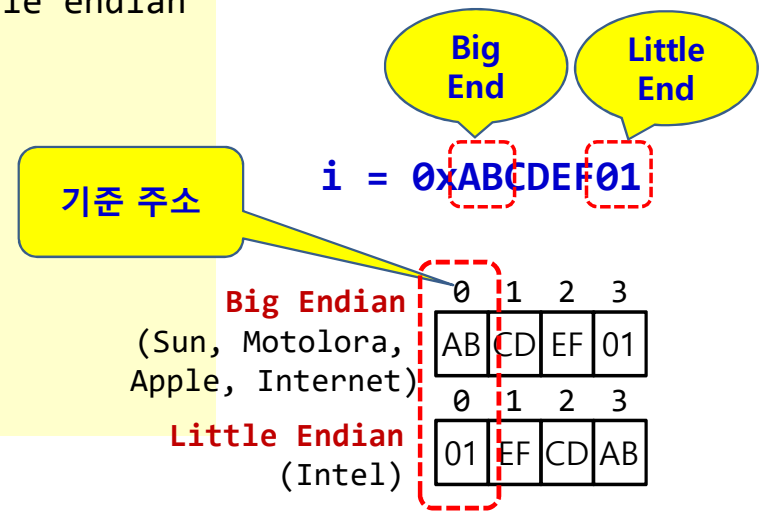
```
# bytes - list argument

L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

B = bytes(L)
print("L = ", L)
print("B = bytes(L) =  ", B)

print("Access individual element in B : ")
for i in range(len(B)):
    print("0x{:02X}".format(B[i]), end=' ')
print()

# next statement generates error
B[0] = 0xAB # bytes is immutable
```

```
L =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
B = bytes(L) =   b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f'
Access individual element in B :
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
Traceback (most recent call last):
  File "C:/MyPyPackage/TextBook - 2019/ch 5 Python Adv Data types, Data Structures,
, bytearray, memoryview/(2) bytes from list.py", line 15, in <module>
    B[0] = 0xAB
TypeError: 'bytes' object does not support item assignment
```

bytes 자료형 – to_bytes(), from_bytes(), byte ordering

```
# bytes - to_bytes, from_bytes

i = 0xABCDEF01
i_bytes_BE = int.to_bytes(i, length=4, byteorder='big') # big endian
i_bytes_LE = int.to_bytes(i, length=4, byteorder='little') # little endian

print("i = 0x{:X}".format(i))
print("i_bytes_BE = ", i_bytes_BE)
print("i_bytes_LE = ", i_bytes_LE)

i_BE = int.from_bytes(i_bytes_BE, byteorder='big')
i_LE = int.from_bytes(i_bytes_LE, byteorder='little')
print("i_BE = 0x{:X}".format(i_BE))
print("i_LE = 0x{:X}".format(i_LE))
```

```
i = 0xABCDEF01
i_bytes_BE =  b'\xab\xcd\xef\x01'
i_bytes_LE =  b'\x01\xef\xcd\xab'
i_BE = 0xABCDEF01
i_LE = 0xABCDEF01
```



바이트 배열 (bytearray)

```
# creation of bytearray

BA0 = bytearray()
BA1 = bytearray(10)
BA2 = bytearray(range(10))
BA3 = bytearray("ABCDEFG", "ascii")
print("bytearray()           : {}".format(BA0))
print("bytearray(10)         : {}".format(BA1))
print("bytearray(range(10)) : {}".format(BA2))
print("bytearray('ABCDEFG', 'ascii'): {}".format(BA3))
```

```
bytearray()           : bytearray(b'')
bytearray(10)         : bytearray(b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00')
bytearray(range(10))  : bytearray(b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t')
bytearray('ABCDEFG', 'ascii'): bytearray(b'ABCDEFG')
```

바이트 배열 (bytearray)

```
# bytearray - list

L1 = [0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27]
print("L1 : ", end='')
for i in range(len(L1)):
    print("0x{:02X}".format(L1[i]), end=' ')
print()

BA1 = bytearray(L1)
print("BA = bytearray(L1) = ", end = '')
for i in range(len(L1)):
    print("0x{:02X}".format(BA1[i]), end=' ')
print()

for i in range(len(BA1)):
    BA1[i] += 0x20
print("After adding 0x20, BA1 = {}".format(BA1))

BA2 = bytearray("ABCDEFG", "ascii")
print("BA2 = {}".format(BA2))

for i in range(len(BA2)):
    BA2[i] += 0x20
print("After adding 0x20, BA2 = {}".format(BA2))
```

```
L1 : 0x21 0x22 0x23 0x24 0x25 0x26 0x27
BA = bytearray(L1) = 0x21 0x22 0x23 0x24 0x25 0x26 0x27
After adding 0x20, BA = bytearray(b'ABCDEFG')
BA2 = bytearray(b'ABCDEFG')
After adding 0x20, BA = bytearray(b'abcdefg')
```

메모리뷰 (memoryview) 자료형

◆ 메모리뷰 자료형

- 파이썬 프로그램에서 사용되는 객체나 변수들이 저장된 메모리상의 바이트들을 직접 살펴볼 수 있게 함
- bytes나 list의 다른 자료형으로 변환시키거나 다른 메모리 뷰 자료형으로 변경시킬 수 있게 함

◆ 메모리뷰 관련 메소드

메모리뷰 메소드	설 명
mv.readonly	읽기 전용 (readonly)인가를 확인
mv.format()	메모리뷰 형식 지정
mv.itemsize()	메모리뷰의 항목 (entry)의 크기를 바이트 단위로 반환
mv.obj()	메모리뷰가 어떤 객체를 대상으로 하는지 확인
mv.nbytes()	전체 바이트 수; len(memoryview.tobytes())와 동일
mv.tobytes()	메모리뷰의 내용을 메모리 버퍼에 문자열로 저장시켜줌; bytes()와 동일한 결과
mv.tolist()	메모리 버퍼에 데이터의 리스트를 저장; list()와 동일한 결과
mv.release()	메모리 버퍼에 있는 데이터를 해제시킴
mv.cast()	메모리뷰를 새로운 형식이나 구성으로 변환시킨 새로운 메모리 뷰를 반환

memoryview와 to_bytes(), from_bytes()

```
# memoryview - to_bytes, bytearray, big-endian, little-endian

n = 0xABCDEF01 # hexadecimal integer (immutable)
print("n = {:X}".format(n))
n_bytes_BE = int.to_bytes(n, length=4, byteorder='big')
mv_n_BE = memoryview(n_bytes_BE)
print("mv_n_BE = memoryview(n_bytes_BE) = ", mv_n_BE)
print("type(mv_n_BE) = ", type(mv_n_BE))
for i in range(len(mv_n_BE)):
    print("mv_n_BE[{:2}] = {:02X}".format(i, mv_n_BE[i]))
print("mv_n_BE.readonly = ", mv_n_BE.readonly)

n_bytes_LE = int.to_bytes(n, length=4, byteorder='little')
mv_n_LE = memoryview(n_bytes_LE)
print("mv_n_LE = memoryview(n_bytes_LE) = ", mv_n_LE)
print("type(mv_n_LE) = ", type(mv_n_LE))
for i in range(len(mv_n_LE)):
    print("mv_n_LE[{:2}] = {:02X}".format(i, mv_n_LE[i]))

L = [1, 2, 3, 4, 5] # list is mutable
bytearray_L = bytearray(L)
mv_list = memoryview(bytearray_L)
print("mv_list = ", mv_list)
for i in range(len(mv_list)):
    print("mv_list[{:2}] = {:02X}".format(i, mv_list[i]))
print("mv_list.readonly = ", mv_list.readonly)
mv_list[0] = 0xFF
print("After mv_list[0] = 0xFF:")
for i in range(len(mv_list)):
    print("mv_list[{:2}] = {:02X}".format(i, mv_list[i]))
```

```
n = ABCDEF01
mv_n_BE = memoryview(n_bytes_BE) =  <memory at 0x039CAE68>
type(mv_n_BE) =  <class 'memoryview'>
mv_n_BE[ 0] = AB
mv_n_BE[ 1] = CD
mv_n_BE[ 2] = EF
mv_n_BE[ 3] = 01
mv_n_BE.readonly =  True
mv_n_LE = memoryview(n_bytes_LE) =  <memory at 0x039CAF28>
type(mv_n_LE) =  <class 'memoryview'>
mv_n_LE[ 0] = 01
mv_n_LE[ 1] = EF
mv_n_LE[ 2] = CD
mv_n_LE[ 3] = AB
mv_list =  <memory at 0x03C0E028>
mv_list[ 0] = 01
mv_list[ 1] = 02
mv_list[ 2] = 03
mv_list[ 3] = 04
mv_list[ 4] = 05
mv_list.readonly =  False
After mv_list[0] = 0xFF:
mv_list[ 0] = FF
mv_list[ 1] = 02
mv_list[ 2] = 03
mv_list[ 3] = 04
mv_list[ 4] = 05
```

List

리스트 (list) 자료형

◆ 리스트 자료형 개요

- 다양한 자료형의 원소들을 배열처럼 사용할 수 있게 하는 시퀀스 자료형
- 대괄호 ([,])로 표현하며, 각 원소들을 콤마(,)로 구분하며 열거
  - L_data = [1, 2, 3, 4, 5, 6, 7, 8, 9]
  - L_str = [“one”, “two”, “three”]
  - L_data = [1, 2, “one”, “two”, (1, 2, 3, 4), [1, 2, 3, 4], {1:”one”, 2:”two”}]
  - L_students = [(“Kim”, 1234), (“Yoon”, 2345), (“Park”, 3456), (“Lee”, 5678)]
- 리스트에 포함된 각 요소들을 인덱싱 (indexing)을 사용하여 개별적으로 열거할 수 있음
- 리스트에 포함된 요소들을 슬라이싱(slicing)을 사용하여 추출할 수 있음
- 리스트의 생성:
  - str
  - list
  - tuple

리스트 관련 함수 (1)

구분	list (L) operation	설명
리스트 생성	L = []	비어있는 (empty) 리스트 생성
	L = [123, 'abc', 1.23, {}]	four items: indexes 0..3
	L = ['Bob', 40.0, ['dev', 'mgr']]	리스트 내부에 리스트가 포함된 중첩 nested sublists
	L = list("ABCDEFG")	list of an iterable's items
	L = list(range(-4, 4))	list of successive integers
	L = [x**2 for x in range(5)]	range()와 계산식을 사용하여 원소 초기값 설정
	L_num = list(map(int, ['1', '2', '3', '4']) L_ascii = list(map(ord, "ABCDE"))	L_num = [1, 2, 3, 4] L_ascii = [65, 66, 67, 68, 69]
기본 연산	len(L)	length
	L1 + L2	접합 (concatenation)
	L * 3	반복 (repeat)
	L.append(4)	첨가
	L.extend([5, 6, 7])	확장
	L.clear()	리스트 원소들을 모두 삭제
	L.copy()	리스트를 복사
	L.count(x)	리스트에 x가 포함되어 있는 개수 반환
	L.remove(x)	리스트로부터 원소 x를 삭제
	3 in L	membership

리스트 관련 함수 (2)

구분	list (L) operation	설명
정렬	L.sort()	정렬
	L.reverse()	역순 정렬
인덱싱	L[i] = 3	index
	L.insert(i, x)	i번째 위치에 x를 삽입
	L[i][j]	index of index
	L.pop(i)	L.pop() : 맨 끝 항목을 삭제 L.pop(0): 0번째 (맨 처음) 항목을 삭제 L.pop(-1): 맨 끝 항목을 삭제
	del L[i]	i번째 원소를 삭제
슬라이싱	L[i:j] = [4, 5, 6]	i번째~j-1번째 구간의 원소를 주어진 값으로 변경
	L[i:j] = []	i번째~j-1번째 구간의 원소를 삭제
	del L[i:j]	i번째~j-1번째 구간의 원소를 삭제
반복문	for x in L: print(x)	iteration

리스트 (list) – [] 연산자와 list() 함수

```
# list creations

L1 = [1, 2, 3, 4, 5]
print("L1 = {}, type(L1) = {}".format(L1, type(L1)))

L2 = ["abc", "def", "gold", "silver", "bronze"]
print("L2 = {}, type(L2) = {}".format(L2, type(L2)))

L3 = list("ABCDEFG")
print("L3 = {}, type(L3) = {}".format(L3, type(L3)))
```

```
L1 = [1, 2, 3, 4, 5], type(L1) = <class 'list'>
L2 = ['abc', 'def', 'gold', 'silver', 'bronze'], type(L2) = <class 'list'>
L3 = ['A', 'B', 'C', 'D', 'E', 'F', 'G'], type(L3) = <class 'list'>
```

리스트 (list) – range()를 사용한 list 생성

```
# Creations of list - range(), len()

L1 = list(range(-4, 4))
print("L1 = ", L1)

L2 = [x**2 for x in range(5)]
print("L2 = ", L2)

L3 = [1234, "ABC", 1.234, ('a', 'b', 'c'), {'x':1, 'y':2, 'z':3}]
for i in range(len(L3)):
    print("L3[{}]: {}".format(i, L3[i]))
```

```
L1 =  [-4, -3, -2, -1, 0, 1, 2, 3]
L2 =  [0, 1, 4, 9, 16]
L3[0]: 1234
L3[1]: ABC
L3[2]: 1.234
L3[3]: ('a', 'b', 'c')
L3[4]: {'x': 1, 'y': 2, 'z': 3}
```

리스트 (list) 기본 연산 – append(), extend()

```
# List - append(), extend()

A = [1, 2, 3, 4, 5]
B = [3, 4, 5, 6, 7]
C = [10, 11, 12, 13]

print("A : ", A)
print("B : ", B)
print("C : ", C)

A.append(C)
print("A.append(C) ==> A: ", A)
B.extend(C)
print("B.extend(C) ==> B: ", B)
```

```
A :  [1, 2, 3, 4, 5]
B :  [3, 4, 5, 6, 7]
C :  [10, 11, 12, 13]
A.append(C) ==> A:  [1, 2, 3, 4, 5, [10, 11, 12, 13]]
B.extend(C) ==> B:  [3, 4, 5, 6, 7, 10, 11, 12, 13]
```

리스트 (list) – split()과 map()

```
# Creation of list of integers using split(), map()

input_int_strings = input("input integers : ")
int_strs = input_int_strings.split(sep=' ')
print("int_strs = ", int_strs)
L = list(map(int, int_strs))
print("L = ", L)
```

```
input integers : 123 456 789 1234 5678
int_strs =  ['123', '456', '789', '1234', '5678']
L =  [123, 456, 789, 1234, 5678]
```

한 줄에 여러 정수 (integer) 데이터 입력 - split(), map()

```
# input multiple numbers from one line using split() and map()

input_data_str = input("input data : ")
decimal_strings = input_data_str.split(sep=' ')
print("Input decimal_strings : ", decimal_strings)
L = list(map(int, decimal_strings))
print("Input integers : ", L)
```

```
input data : 1 2 3 4 5 6 7 8 9 10
Input decimal_strings :  ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Input integers :  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

input data : 1 1234 6543 9876
Input decimal_strings :  ['1', '1234', '6543', '9876']
Input integers :  [1, 1234, 6543, 9876]
```

한 줄에 여러 실수 (float) 데이터 입력 - split(), map()

```
# Getting multiple float numbers in one line using split() and map()

input_data_str = input("input float data : ")
float_strings = input_data_str.split(sep=' ')
print("Input decimal_strings : ", float_strings)
L = list(map(float, float_strings))
print("Input float data : ", L)
```

```
input float data : 1.23 4.56 7.78 10.12345
Input decimal_strings :  ['1.23', '4.56', '7.78', '10.12345']
Input float data :  [1.23, 4.56, 7.78, 10.12345]
```

한 줄에 연, 월, 일 입력 하기

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

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))
print("Input date : year({}), month({}), day({})".format(year, month, day))
```

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

리스트 (list) 기본 연산 – sort(), reverse()

```
# List functions

A = [4, 2, 9, 7, 1, 3, 5, 6, 8, 0]
B = ['b', 'f', 'a', 'd', 'e', 'c', 'g']

print("A = ", A)
A.sort()
print("After A.sort(), A = ", A)
A.reverse()
print("After A.reverse(), A = ", A)

print("B = ", B)
B.sort()
print("After B.sort(), B = ", B)
B.reverse()
print("After B.reverse(), B = ", B)
```

```
A =  [4, 2, 9, 7, 1, 3, 5, 6, 8, 0]
After A.sort(), A =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
After A.reverse(), A =  [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
B =  ['b', 'f', 'a', 'd', 'e', 'c', 'g']
After B.sort(), B =  ['a', 'b', 'c', 'd', 'e', 'f', 'g']
After B.reverse(), B =  ['g', 'f', 'e', 'd', 'c', 'b', 'a']
```

리스트 (list) 기본 연산 – indexing, pop(), remove()

```
# List indexing, pop(), remove()

A = [0, 1, 2, 3, 4, 5]
print("A = ", A)

A[0] = 100
print("After A[0] = 100, A = ", A)
A.insert(0, 200)
print("After A.insert(0, 200), A = ", A)
del A[0]
print("After del A[0], A = ", A)
A.pop(0)
print("After A.pop(0), A = ", A)
A.pop()
print("After A.pop(),  A = ", A)
A.pop(-1)
print("After A.pop(-1),A = ", A)
A.remove(3)
print("After A.remove(3),A=", A)
print("3 in A = ", 3 in A)
```

```
A =  [0, 1, 2, 3, 4, 5]
After A[0] = 100, A =  [100, 1, 2, 3, 4, 5]
After A.insert(0, 200), A =  [200, 100, 1, 2, 3, 4, 5]
After del A[0], A =  [100, 1, 2, 3, 4, 5]
After A.pop(0), A =  [1, 2, 3, 4, 5]
After A.pop(),  A =  [1, 2, 3, 4]
After A.pop(-1),A =  [1, 2, 3]
After A.remove(3),A= [1, 2]
3 in A =  False
```

리스트 (list) 기본 연산 – 인덱싱 (indexing)과 슬라이싱 (slicing)

◆ 리스트 인덱싱과 슬라이싱

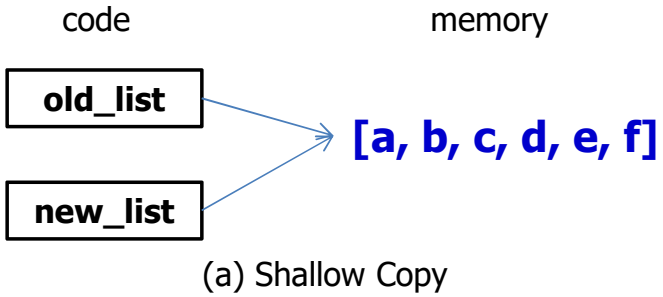
```
# List – indexing and slicing

A = [0, 1, 2, 3, 4, 5]

print("A = ", A)
print("A[1] = ", A[1])  # indexing
print("A[-1] = ", A[-1])

print("A[0:3] = ", A[0:3])
A[0:3] = [10, 11, 12] # slicing
print("After A[0:3] = [10, 11, 12], A = ", A)
A[0:3] = []
print("After A[0:3] = [], A = ", A)
A = [0, 1, 2] + A
print("After A = [0, 1, 2] + A, A = ", A)
```

```
A =  [0, 1, 2, 3, 4, 5]
A[1] =  1
A[-1] =  5
A[0:3] =  [0, 1, 2]
After A[0:3] = [10, 11, 12], A =  [10, 11, 12, 3, 4, 5]
After A[0:3] = [], A =  [3, 4, 5]
After A = [0, 1, 2] + A, A =  [0, 1, 2, 3, 4, 5]
```

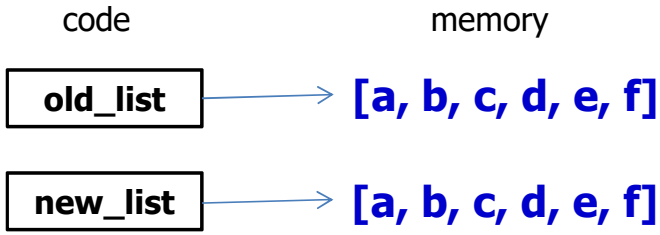
리스트 복사 (copy)

◆ Shallow copy vs. Deep Copy

- shallow copy (얕은 복사) :
  - 동일한 객체 (object)를 이름을 다르게 하여 사용
  - 하나의 이름으로 객체의 내용을 변경할 때, 다른 이름에서도 변경된 내용이 반영됨
- deep copy (깊은 복사)
  - 서로 다른 객체로 사용될 수 있게 관리
  - 한 객체의 내용을 변경할 때, 깊은 복사된 다른 객체에는 영향이 없음



(a) Shallow Copy



(a) Deep Copy

리스트 복사 (copy) – copy()

```
A = [1, 2, [31, 32, 33]]
B = A # shallow copy
print("A : ", A)
print("B = A # shallow copy")
print("B : ", B)
print("id(A) : {0}, id(B) : {1}".format(id(A), id(B)))
print("A is B : ", A is B)
B[1] = 20
print("A : ", A)
print("B : ", B)

C = A.copy()
print("\nC = A.copy()")
print("A : ", A)
print("C : ", C)
print("id(A) : {0}, id(C) : {1}".format(id(A), id(C)))
print("A is C : ", A is C)
C[1] = 200
C[2][0] = 201
print("A : ", A)
print("C : ", C)
```

```
A :  [1, 2, [31, 32, 33]]
B = A # shallow copy
B :  [1, 2, [31, 32, 33]]
id(A) : 1757649798528, id(B) : 1757649798528
A is B :  True
A :  [1, 20, [31, 32, 33]]
B :  [1, 20, [31, 32, 33]]

C = A.copy()
A :  [1, 20, [31, 32, 33]]
C :  [1, 20, [31, 32, 33]]
id(A) : 1757649798528, id(C) : 1757689241216
A is C :  False
A :  [1, 20, [201, 32, 33]]
C :  [1, 200, [201, 32, 33]]
```

리스트 복사 (copy) – copy module

◆ list copy using copy module

```
# list copy using copy module
import copy
print("\nTesting copy function in copy module")
A = [1, 2, [31, 32, 33]]
D = copy.copy(A) # shallow copy function in copy module
print("D = copy.copy(A) # shallow copy")
print("A : ", A)
print("D : ", D)
print("id(A) : {0}, id(D) : {1}".format(id(A), id(D)))
print("D is A : ", D is A)
D[1] = 300
D[2][0] = 301
print("A : ", A)
print("D : ", D)
print("D[1] is A[1] : ", D[1] is A[1])

E = copy.deepcopy(A) # deep copy function in copy module
print("\nE = copy.deepcopy(A) #deep copy")
print("A : ", A)
print("E : ", E)
print("id(A) : {0}, id(E) : {1}".format(id(A), id(E)))
print("E is A : ", E is A)
E[1] = 500
E[2][0] = 501
print("A : ", A)
print("E : ", E)
```

```
Testing copy function in copy module
D = copy.copy(A) # shallow copy
A :  [1, 2, [31, 32, 33]]
D :  [1, 2, [31, 32, 33]]
id(A) : 45608672, id(D) : 45608872
D is A :  False
A :  [1, 2, [301, 32, 33]]
D :  [1, 300, [301, 32, 33]]
D[1] is A[1] :  False

E = copy.deepcopy(A) #deep copy
A :  [1, 2, [301, 32, 33]]
E :  [1, 2, [301, 32, 33]]
id(A) : 45608672, id(E) : 45608752
E is A :  False
A :  [1, 2, [301, 32, 33]]
E :  [1, 500, [501, 32, 33]]
```

2차원 리스트를 사용한 행렬 표현

```
# two-dimensional list for matrix

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

print("A : ", A)
print("B : ", B)
for i in range (3):
    for j in range (3):
        C[i][j] = A[i][j] + B[i][j]
print("A+B => ", C)

for i in range (3):
    for j in range (3):
        D[i][j] = A[i][j] - B[i][j]
print("A-B => ", D)

for i in range (3):
    for j in range (3):
        for k in range (3):
            E[i][j] += A[i][k] * B[k][j]
print("A*B => ", E)
```

```
A :  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
B :  [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
A+B =>  [[2, 2, 3], [4, 6, 6], [7, 8, 10]]
A-B =>  [[0, 2, 3], [4, 4, 6], [7, 8, 8]]
A*B =>  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

튜플 (tuple)

튜플 (tuple) 자료형

◆ 튜플 자료형 개요

- 여러 개의 요소들을 묶음으로 사용할 수 있는 시퀀스 자료형
- 소괄호로 표현 : ('name', student_id, age, score)
- 튜플에 포함된 각 요소들을 개별적으로 열거할 수 있음
- 튜플의 생성:
  - str
  - list
  - range()
  - zip()

튜플에 사용할 수 있는 연산

연산, 함수	기능	사용 예
len()	튜플에 포함된 객체의 개수	len((1, 2, 3, 4, 5))
+	2개의 튜플을 접합 (concatenation)	(1, 2, 3) + (4, 5, 6)
*	지정된 횟수만큼 시퀀스를 반복 (repetition)	('a', 'b', 'c') * 3
in	튜플에 포함되어 있는지 확인	tuple_A = ('Mon', 'Tue', Wed') 'Mon' in tuple_A
not in	튜플에 포함되어 있지 않는지 확인	'x' not in ('a', 'b', 'c', 'd', 'e')
[i]	인덱싱	T = (3, 1, 5, 7, 0, 4, 2, 6) T[0]
[start : end : step]	슬라이싱	T = (3, 1, 5, 7, 0, 4, 2, 6) T_1 = T[0:5:2] # get subtuple
min()	튜플에서 제일 작은 요소	min((3, 1, 5, 7))
max()	튜플에서 제일 큰 요소	max((3, 1, 5, 7))
sum()	튜플에 포함된 원소들의 합 (tuple에 포함된 원소에 + 연산이 가능한 경우에만 사용)	sum((3, 1, 5, 7))
sorted()	튜플에 포함된 원소들을 정렬한 결과를 반환	T = (3, 1, 5, 7, 0, 4, 2, 6) T_sorted = sorted(T)
for 반복문	반복문 구성	for n in (3, 1, 5, 7, 0, 4, 2, 6) : print(n)

튜플 – 문자열, 리스트, range를 사용한 튜플 생성

```
# tuple - creations

t1 = tuple("one") # tuple creation with string
print("t1 = {}, type(t1) = {}".format(t1, type(t1)))

t2 = tuple(["one", "two", "three"]) # tuple creation with list
print("t2 = {}, type(t2) = {}".format(t2, type(t2)))

t3 = tuple(range(5)) # tuple creation with range()
print("t3 = {}, type(t3) = {}".format(t3, type(t3)))

t4 = tuple([0, 1.0, 'a', b"xyz"]) # tuple creation with list of different types
print("t4 = {}, type(t4) = {}".format(t4, type(t4)))
```

```
t1 = ('o', 'n', 'e'), type(t1) = <class 'tuple'>
t2 = ('one', 'two', 'three'), type(t2) = <class 'tuple'>
t3 = (0, 1, 2, 3, 4), type(t3) = <class 'tuple'>
t4 = (0, 1.0, 'a', b'xyz'), type(t4) = <class 'tuple'>
```

튜플 (tuple)의 생성 – ()

```
# tuple definition without parenthesis

fruits = 'apple', 'orange', 'banana', 'mango', 'cherry', 'plum', 'pear', 'peach'
animals = 'lion', 'tiger', 'monkey', 'rabbit', 'cat', 'dog'

print("type(fruits) = ", type(fruits))
print("fruits = ", fruits)

print("\ntype(animals) = ", type(animals))
print("fruits = ", animals)
```

```
type(fruits) =  <class 'tuple'>
fruits =  ('apple', 'orange', 'banana', 'mango', 'cherry', 'plum', 'pear', 'peach')

type(animals) =  <class 'tuple'>
fruits =  ('lion', 'tiger', 'monkey', 'rabbit', 'cat', 'dog')
```

튜플 (tuple)의 생성 – zip()

```
# creation of list of tuples with zip

A = ["Kim", "Lee", "Park", "Yoon", "Choi"]
B = [20, 30, 40, 50, 60]
C = ["Male", "Female", "Female", "Male", "Female",]

Persons = list(zip(A, B, C))
print("A = ", A)
print("B = ", B)
print("C = ", C)
print("Persons = list(zip(A, B, C)) = \n", Persons)
for i in range(len(Persons)):
    print("Persons[{:2}] = {}".format(i, Persons[i]))
```

```
A =  ['Kim', 'Lee', 'Park', 'Yoon', 'Choi']
B =  [20, 30, 40, 50, 60]
C =  ['Male', 'Female', 'Female', 'Male', 'Female']
Persons = list(zip(A, B, C)) =
 [('Kim', 20, 'Male'), ('Lee', 30, 'Female'), ('Park', 40, 'Female
'), ('Yoon', 50, 'Male'), ('Choi', 60, 'Female')]
Persons[ 0] = ('Kim', 20, 'Male')
Persons[ 1] = ('Lee', 30, 'Female')
Persons[ 2] = ('Park', 40, 'Female')
Persons[ 3] = ('Yoon', 50, 'Male')
Persons[ 4] = ('Choi', 60, 'Female')
```

튜플 (tuple)의 복사

```
# Testing copy operations of tuple
import copy

t1 = (1, 2, 3)
print('t1 : ', t1)

print('t2 = t1')
t2 = t1
print("t2 == t1 : ", t2 == t1)
print('id(t1) : ', id(t1))
print('id(t2) : ', id(t2))
print("t2 is t1 : ", t2 is t1)

print('\nt3 = copy.copy(t1)')
#t3 = t1.copy()
# tuple does not have .copy() member function
t3 = copy.copy(t1)
print('t3 : ', t3)
print("t3 == t1 : ", t3 == t1)
print('id(t1) : ', id(t1))
print('id(t3) : ', id(t3))
print("t3 is t1 : ", t3 is t1)

print('\nt4 = copy.deepcopy(t1)')
t4 = copy.deepcopy(t1)
print('t4 : ', t4)
print("t4 == t1 : ", t4 == t1)
print('id(t1) : ', id(t1))
print('id(t4) : ', id(t4))
print("t4 is t1 : ", t4 is t1)
```

```
t1 :  (1, 2, 3)
t2 = t1
t2 == t1 :  True
id(t1) :  3232712
id(t2) :  3232712
t2 is t1 :  True

t3 = copy.copy(t1)
t3 :  (1, 2, 3)
t3 == t1 :  True
id(t1) :  3232712
id(t3) :  3232712
t3 is t1 :  True

t4 = copy.deepcopy(t1)
t4 :  (1, 2, 3)
t4 == t1 :  True
id(t1) :  3232712
id(t4) :  3232712
t4 is t1 :  True
```

튜플의 개별 원소의 자료형에 따른 변경 가능 여부

```
# tuple - immutability

a = (1, 2, 3, [10, 20, 30])
print("a = ", a)
a[3][0] = 100 # list is mutable
print("after a[3][0] = 100, a = ", a)
del a[3][2]
print("after del a[3][2], a = ", a)

a[0] = 10 # integer is immutable
```

```
a =  (1, 2, 3, [10, 20, 30])
after a[3][0] = 100, a =  (1, 2, 3, [100, 20, 30])
after del a[3][2], a =  (1, 2, 3, [100, 20])
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 5 Python Adv Data
\(6) tuple - immutable.py", line 10, in <module>
    a[0] = 10 # integer is immutable
TypeError: 'tuple' object does not support item assignment
```

튜플의 접합 (+) 연산, 반복(*) 연산

```
# tuple - concatenation, repeation

t1 = ('a', 'b', 'c')
t2 = ('x', 'y', 'z')
print('t1 = {}'.format(t1))
print('t2 = {}'.format(t2))

t3 = t1 + t2
print('t3 = t1 + t2 = {}'.format(t3))

t4 = t1 * 3
print('t4 = t1 * 3 = {}'.format(t4))
```

```
t1 = ('a', 'b', 'c')
t2 = ('x', 'y', 'z')
t3 = t1 + t2 = ('a', 'b', 'c', 'x', 'y', 'z')
t4 = t1 * 3 = ('a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c')
```

튜플의 in, not in 연산

```
# tuple - concatenation, repeation, in, not in

t1 = ('a', 'b', 'c')
t2 = ('x', 'y', 'z')
print('t1 = {}'.format(t1))
print('t2 = {}'.format(t2))

result = 'a' in t1
print("'a' in t1 : ", result)

result = 'x' in t1
print("'x' in t1 : ", result)

result = 'x' not in t1
print("'x' not in t1 : ", result)
```

```
t1 = ('a', 'b', 'c')
t2 = ('x', 'y', 'z')
'a' in t1 :  True
'x' in t1 :  False
'x' not in t1 :  True
```

튜플의 min(), max(), sum()

```
# tuple - =, +, min(), max(), sum()

t1 = ('a', 'b', 'c')
t2 = ('d', 'e', 'f')
print('t1 : {}, id(t1) = {}'.format(t1, id(t1)))
print('t2 : {}, id(t2) = {}'.format(t2, id(t2)))
print("t1 == t2 : ", t1 == t2)

print('\nt3 = t1')
t3 = t1
print('t3 : {}, id(t3) = {}'.format(t3, id(t3)))
print("t3 == t1 : ", t3 == t1)
print("t3 is t1 : ", t3 is t1)

t4 = t1 + t2
print('t4= t1 + t2 = {}, id(t4) = {}'.format(t4, id(t4)))
print("min(t4) = {}, max(t4) = {}".format(min(t4), max(t4)))

t5 = (1, 3, 5, 7, 9)
print('t5 : {}, id(t5) = {}'.format(t5, id(t5)))
print("sum(t5) = {}".format(sum(t5)))

# following sum() generates error
print("sum(t4) = {}".format(sum(t4)))
```

```
t1 : ('a', 'b', 'c'), id(t1) = 49241704
t2 : ('d', 'e', 'f'), id(t2) = 49200504
t1 == t2 :  False

t3 = t1
t3 : ('a', 'b', 'c'), id(t3) = 49241704
t3 == t1 :  True
t3 is t1 :  True
t4= t1 + t2 = ('a', 'b', 'c', 'd', 'e', 'f'), id(t4) = 46808736
min(t4) = a, max(t4) = f
t5 : (1, 3, 5, 7, 9), id(t5) = 49178640
sum(t5) = 25
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 5 Python Adv Data type
\(9) tuple - min, max, sum.py", line 24, in <module>
    print("sum(t4) = {}".format(sum(t4)))
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

튜플의 비교 - <, <=, >, >=, ==

```
# tuple - comparisons by >, >=, <, <=, ==

s1 = ("Kim", "CE", 4.17, 19001234)
s2 = ("Lee", "EE", 3.78, 21003234)
s3 = ("Lee", "ICE", 4.13, 18001547)
s4 = ("Park", "ICE", 4.10, 20001545)
s5 = ("Park", "ICE", 4.10, 20001546)

print("{}  >  {} = {}".format(s1, s2, s1 > s2))
print("{}  <  {} = {}".format(s1, s2, s1 < s2))
print("{}  >  {} = {}".format(s3, s4, s3 > s4))
print("{} >=  {} = {}".format(s4, s5, s4 >= s5))
print("{} <  {} = {}".format(s4, s5, s4 < s5))
```

```
('Kim', 'CE', 4.17, 19001234)  >  ('Lee', 'EE', 3.78, 21003234) = False
('Kim', 'CE', 4.17, 19001234)  <  ('Lee', 'EE', 3.78, 21003234) = True
('Lee', 'ICE', 4.13, 18001547)  >  ('Park', 'ICE', 4.1, 20001545) = False
('Park', 'ICE', 4.1, 20001545) >=  ('Park', 'ICE', 4.1, 20001546) = False
('Park', 'ICE', 4.1, 20001545) <  ('Park', 'ICE', 4.1, 20001546) = True
```

튜플의 정렬 (sorting) – sorted()

```
# tuple - sorted()

t1 = ('x', 'y', 'z', 'w', 'a', 'b', 'c')
print('t1 : {}, id(t1) = {}'.format(t1, id(t1)))
t2 = sorted(t1)
print('t2 : {}, id(t2) = {}'.format(t2, id(t2)))

t3 = (3, 5, 1, 0, 6, 9, 2, 8, 7, 4)
print('t3 : {}, id(t3) = {}'.format(t3, id(t3)))
t4 = sorted(t3)
print('t4 : {}, id(t4) = {}'.format(t4, id(t4)))
```

```
t1 : ('x', 'y', 'z', 'w', 'a', 'b', 'c'), id(t1) = 49221168
t2 : ['a', 'b', 'c', 'w', 'x', 'y', 'z'], id(t2) = 37503264
t3 : (3, 5, 1, 0, 6, 9, 2, 8, 7, 4), id(t3) = 49361952
t4 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], id(t4) = 37503064
```

튜플의 인덱싱, 슬라이싱

```
# tuple - indexing, slicing

t1 = (3, 5, 1, 0, 6, 9, 2, 8, 7, 4)
print('t1 : {}, id(t1) = {}'.format(t1, id(t1)))
t2 = sorted(t1)
print('t2 : {}, id(t2) = {}'.format(t2, id(t2)))

# testing slicing
t3 = t2[0:len(t2):2]
print('t3 = t2[0:len(t2):2] = {}'.format(t3))
t4 = t2[1:len(t2):2]
print('t4 = t2[1:len(t2):2] = {}'.format(t4))

# testing indexing => following statement generates error
t1[0] = 11
```

```
t1 : (3, 5, 1, 0, 6, 9, 2, 8, 7, 4), id(t1) = 49349448
t2 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], id(t2) = 45498776
t3 = t2[0:len(t2):2] = [0, 2, 4, 6, 8]
t4 = t2[1:len(t2):2] = [1, 3, 5, 7, 9]
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 5 Python Adv Data
\(12) tuple - indexing, slicing.py", line 15, in <module>
    t1[0] = 11
TypeError: 'tuple' object does not support item assignment
```

튜플 리스트

```
# tuple - dates, sorted

dates = [
    (2000, 12, 25),
    (2019, 9, 1),
    (2019, 3, 1),
    (1, 1, 1)
    ]

print("dates = ", dates)
sorted_dates = sorted(dates)

print("sorted_dates = ", sorted_dates)
```

```
dates =  [(2000, 12, 25), (2019, 9, 1), (2019, 3, 1), (1, 1, 1)]
sorted_dates =  [(1, 1, 1), (2000, 12, 25), (2019, 3, 1), (2019, 9, 1)]
```

학생 정보 (student record) 튜플의 리스트 구성

```
# tuple - student, sorting by key element, reverse order

students = [
    ("Kim", 19001234, "CE", 4.17),\
    ("Lee", 18003234, "EE", 3.78),\
    ("Park", 21001547, "ICE", 4.13),\
    ("Yoon", 17002571, "ME", 3.55),\
    ("Hong", 20003257, "ICE", 4.45)]

for i in range(len(students)):
    (name, st_id, major, GPA) = students[i]
    print("students[{:2}] : name({:^8s}), st_id({:8d}), major({:3s}, GPA({:5.2f}))"\
          .format(i, name, st_id, major, GPA))

sorted_students = sorted(students, key=lambda student: student[3], reverse=True)

print("\nAfter sorting according to GPA, decreasing order :")
for i in range(len(sorted_students)):
    (name, st_id, major, GPA) = sorted_students[i]
    print("students[{:2}] : name({:^8s}), st_id({:8d}), major({:3s}, GPA({:5.2f}))"\
          .format(i, name, st_id, major, GPA))
```

```
students[ 0] : name(  Kim   ), st_id(19001234), major(CE , GPA( 4.17))
students[ 1] : name(  Lee   ), st_id(18003234), major(EE , GPA( 3.78))
students[ 2] : name(  Park  ), st_id(21001547), major(ICE, GPA( 4.13))
students[ 3] : name(  Yoon  ), st_id(17002571), major(ME , GPA( 3.55))
students[ 4] : name(  Hong  ), st_id(20003257), major(ICE, GPA( 4.45))

After sorting according to GPA, decreasing order :
students[ 0] : name(  Hong  ), st_id(20003257), major(ICE, GPA( 4.45))
students[ 1] : name(  Kim   ), st_id(19001234), major(CE , GPA( 4.17))
students[ 2] : name(  Park  ), st_id(21001547), major(ICE, GPA( 4.13))
students[ 3] : name(  Lee   ), st_id(18003234), major(EE , GPA( 3.78))
students[ 4] : name(  Yoon  ), st_id(17002571), major(ME , GPA( 3.55))
```

딕셔너리 (dict)

dict 자료형

◆ dict 자료형

- 사전 (dictionary)에서 단어 (keyword)에 대한 뜻 (value)를 찾는것과 동일한 개념으로 사용
- dict의 항목 (item)은 key와 value의 쌍으로 표현
  {key_1:value_1, key_2:value_2, ... }
- key는 숫자, 문자, 문자열, 튜플 등 다양한 자료형이 될 수 있음
- value도 숫자, 문자, 문자열, 튜플 등 다양한 자료형이 될 수 있음

◆ dict 자료형의 사용 예

- `d1 = {"one":1, "two":2, "three":3}`
- month_name = {1:”January”, 2:”February”, 3:”March”}
- inter_city_distance = {(“Seoul”, “DaeGu”):300, (“Seoul”, “Daejon”):150}
- students = {1234:(“Kim”, “EE”, 4.12), 3456:(“Lee”, “KK”, 3.98)}

dict에서의 연산 (1)

기능 분야	사전 (dict d)에서의 연산	의미
공통	len(d)	사전 d의 항목 개수
복사, 삭제	d.copy()	d의 얕은 복사 (shallow copy)로 새로운 사전 반환
	d.clear()	사전 d의 모든 항목을 삭제
dict 항목, 키, 값 추출	d.items()	d의 항목 item(key, value)에 대한 사전 뷰 (dictionary views) 객체 반환
	d.keys()	d의 키(keys)의 사전 뷰 (dictionary views) 객체 반환
	d.values()	d의 값 (values)의 사전 뷰 (dictionary views) 객체를 반환
키 포함 여부 확인	key in d	d에 key가 있으면 True
	key not in d	d에 key가 없으면 True
키를 사용한 항목 추출	d[key]	d에서 key인 항목의 value를 반환, key가 없으면 KeyError
	d[key] = value	d[key]의 값을 value로 변경
	d.get(key[, default])	key가 있으면 value를 반환 (d[key]와 동일), key가 없으면 default 값 반환 (암묵적으로 default = None)

dict에서의 연산 (1)

기능 분야	사전 (dict d)에서의 연산	의미
키를 사용한 항목 추출	del d[key]	d[key]의 항목을 삭제, key가 없으면 KeyError
	d.pop(key[, default])	key가 있으면 해당 항목을 삭제하고 value를 반환, 만약 key가 없으면 default를 반환하며, key가 없고 default가 없으면 KeyError 발생
	d.fromkeys(seq[, value])	시퀀스 seq를 키로 설정하고, value를 값으로 설정한 새로운 사전을 반환, 디폴트는 value = None.
항목 삭제	d.popitem()	d에서 임의의 항목을 삭제하고, 튜플 (key, value)로 반환; d가 공백이면 KeyError
설정 및 갱신	d.setdefault(key, [, default])	d에 key가 있으면 value를 반환, 없으면 key: default항목을 추가하고 default 반환 (기본적으로 default = None)
	d.update(key = new_value])	사전 객체 또는 key와 new_value 쌍의 반복 가능한 other를 가지고 사전 d를 갱신하며 None을 반환 key가 있으면 갱신하고, 없으면 삽입
열거자	iter(d)	d의 key에 의한 열거자 (iterator)를 반환, iter(d.keys())

딕셔너리 (dict)의 생성

```
# dict - creations by {key:value}, dict()

d1 = {"one":1, "two":2, "three":3}
print("d1 = ", d1)
print("type(d1) = ", type(d1))

d2 = dict(one=1, two=2, three=3)
print("d2 = ", d2)
print("type(d2) = ", type(d2))
print("d1 == d2 = ", d1==d2)
print("d1 is d2 = ", d1 is d2)
```

```
d1 =  {'one': 1, 'two': 2, 'three': 3}
type(d1) =  <class 'dict'>
d2 =  {'one': 1, 'two': 2, 'three': 3}
type(d2) =  <class 'dict'>
d1 == d2 =  True
d1 is d2 =  False
```

딕셔너리 (dict)의 생성 – (key, value), 튜플(튜플)

```
# dict - creations by list of tuple, tuple of tuple

d3 = dict([("one", 1), ("two", 2), ("three", 3)])
print("d3 = ", d3)
print("type(d3) = ", type(d3))
d4 = dict((("one", 1), ("two", 2), ("three", 3)))
print("d4 = ", d4)
print("type(d4) = ", type(d4))
```

```
d3 =  {'one': 1, 'two': 2, 'three': 3}
type(d3) =  <class 'dict'>
d4 =  {'one': 1, 'two': 2, 'three': 3}
type(d4) =  <class 'dict'>
```

딕셔너리 (dict)의 생성 – zip()

```
# dict - creations by keys, values, zip()

d_keys = ["one", "two", "three"]
d_values = [1, 2, 3]

d5 = dict(zip(d_keys, d_values))
print("d5 = ", d5)
print("type(d5) = ", type(d5))
```

```
d5 =  {'one': 1, 'two': 2, 'three': 3}
type(d5) =  <class 'dict'>
```

딕셔너리 (dict) 기본 함수 – items(), keys(), values()

```
# dict - creations

a = {"one":1, "two":2, "three":3}
print("a = ", a)

a_items = a.items()
a_keys = a.keys()
a_values = a.values()


print("a.items() = ", a_items)
print("a.keys() = ", a_keys)
print("a.values() = ", a_values)

b = dict(zip(a_keys, a_values))
print("b = ", b)

print("a == b : ", a == b)
```

```
a =  {'one': 1, 'two': 2, 'three': 3}
a.items() =  dict_items([('one', 1), ('two', 2), ('three', 3)])
a.keys() =  dict_keys(['one', 'two', 'three'])
a.values() =  dict_values([1, 2, 3])
b =  {'one': 1, 'two': 2, 'three': 3}
a == b :  True
```

딕셔너리 (dict) 기본 함수 – in, not in, iter()

```
# dict - in, not in, iter()

a = {"one":1, "two":2, "three":3}
print("a = ", a)

print("'one' in a : ", 'one' in a)
print("'five' in a : ", 'five' in a)
print("'five' not in a : ", 'five' not in a)

print("\nListing items in a using iter():")
keys = list(iter(a))
for k in keys:
    print("a[{:5}] = {}".format(k, a[k]))
```

```
a =  {'one': 1, 'two': 2, 'three': 3}
'one' in a :  True
'five' in a :  False
'five' not in a :  True

Listing items in a using iter():
a[one  ] = 1
a[two  ] = 2
a[three] = 3
```

집합 (set), 동결집합(frozenset)

집합(set), 동결집합(frozenset)

◆ class set([iterable]), class frozenset([iterable])

- set 자료형은 정렬되어 있지 않으며, 중복되지 않은 항목들로 만 구성된 데이터의 집합
- set 자료형의 항목은 add()를 사용하여 추가할 수 있고, remove()를 사용하여 삭제할 수 있음
- 동결집합 (frozenset) 자료형은 변경할 수 없음 (immutable)
- set/frozenset은 집합 관련 연산에 사용됨: membership, removal of duplications, intersection, union, difference
- set/frozen set 자료형은 정렬 기능이 제공되지 않으며, 시퀀스 기능 (indexing, slicing)도 제공되지 않음

모든 집합(set, frozenset)에서 가능한 연산

집합 S와 T의 연산	의미
len(S)	집합 S의 원소 개수
x in S	x ∈ S, 집합 S에 x가 있으면 True
x not in S	x ∉ S, 집합 S에 x가 없으면 True
S.isdisjoint(T)	S ∩ T = ∅ 집합 S와 집합 T의 교집합이 공집합이면 True
S.issubset(T) S <= T	S ⊂ T, 집합 S가 집합 T의 부분 집합이면 True
S < T	S ⊂ T and S ≠ T 집합 S가 집합 T의 진부분집합이면 True
S.issuperset(T) S >= T	S ⊃ T, 집합 T가 집합 S의 부분 집합이면 True
S > T	S ⊃ T and S ≠ T 집합 T가 집합 S의 진부분집합이면 True
S.union(T, ...) S | T | ...	S ∪ T 집합 S와 집합 T의 합집합을 계산하여 새로운 집합을 반환
S.intersection(T, ...) S & T & ...	S ∩ T 집합 S와 집합 T의 교집합을 계산하여 새로운 집합을 반환
S.difference(T, ...) S – T - ...	S – T 집합 S와 집합 T의 차집합을 계산하여 새로운 집합을 반환
S.symmetircdifference(T) S ^ T	(S – T) ∪ (T – S) 두 집합에 모두 있지 않는 요소들의 새로운 집합을 반환
S.copy()	집합 S의 얕은 복사 (shallow copy)로 새로운 집합을 반환

set 자료형 객체 생성 – {}, set()

```
# Creations of set

S1 = {'a', 'b', 'c'}
print("S1 = ", S1)
S2 = {1, 2, 3, 4, 1, 2, 3}
print("S2 = ", S2) # set maintains non-overlapped items
S3 = set(range(5))
print("S3 = ", S3)
```

```
S1 =  {'c', 'a', 'b'}
S2 =  {1, 2, 3, 4}
S3 =  {0, 1, 2, 3, 4}
```

집합 객체에 대한 in, not in, isdisjoint(), issubset(), issuperset()

```
# set - in, not in, isdisjoint(), issubset(), issuperset()

S1 = {'a', 'c', 'b', 'd', 'e'}
S2 = set("abc")
S3 = {'d', 'g', 'h'}
print("S1 = ", S1)
print("S2 = ", S2)
print("S3 = ", S3)
print("'c' in S1 = ", 'c' in S1)
print("'g' in S1 = ", 'g' in S1)
print("'a' not in S3 = ", 'a' not in S3)
print("S1.isdisjoint(S2) = ", S1.isdisjoint(S2))
print("S2.issubset(S1) = ", S2.issubset(S1))
print("S1.issuperset(S2) = ", S1.issuperset(S2))
```

```
S1 =  {'b', 'c', 'a', 'e', 'd'}
S2 =  {'c', 'b', 'a'}
S3 =  {'d', 'g', 'h'}
'c' in S1 =  True
'g' in S1 =  False
'a' not in S3 =  True
S1.isdisjoint(S2) =  False
S2.issubset(S1) =  True
S1.issuperset(S2) =  True
```

집합 객체에 대한 union(), intersecton(), difference(), symmetric_difference()

```
# set - union(), intersection(), difference(), symmetric_difference()

S1 = {'a', 'c', 'b', 'd', 'e'}
S2 = set("abc")
S3 = {'d', 'g', 'h'}

print("S1.union(S3) = ", S1.union(S3))
print("S1.intersection(S2) = ", S1.intersection(S2))
print("S1.intersection(S3) = ", S1.intersection(S3))
print("S1.difference(S2) = ", S1.difference(S2))
print("S1.symmetric_difference(S2) = ", S1.symmetric_difference(S2))
print("S1.symmetric_difference(S3) = ", S1.symmetric_difference(S3))
```

```
S1.union(S3) =  {'g', 'b', 'd', 'e', 'h', 'c', 'a'}
S1.intersection(S2) =  {'b', 'c', 'a'}
S1.intersection(S3) =  {'d'}
S1.difference(S2) =  {'d', 'e'}
S1.symmetric_difference(S2) =  {'d', 'e'}
S1.symmetric_difference(S3) =  {'g', 'e', 'h', 'b', 'c', 'a'}
```

set에서만 가능한 연산

◆ Operations only for set

집합 S와 T의 연산	의미
S.update(T, ...) S |= T |	S = S ∪ T 집합 S와 집합 T의 합집합을 계산하여 S를 갱신
S.intersection_update(T, . . .) S &= T & . . .	S = S ∩ T 집합 S와 집합 T의 교집합을 계산하여 S를 갱신
S.difference_update(T, . . .) S -= T | . . .	S = S – T 집합 S와 집합 T의 차집합을 계산하여 S를 갱신
S.symmetric_difference_update(T) S ^= T	S = (S – T) ∪ (T – S) 두 집합에 모두 있지 않는 요소들의 새로운 집합을 계산하여 S를 갱신
S.add(elem)	집합 S에 원소 elem을 추가
S.remove(elem)	집합 S에 원소 elem이 있으면 삭제 원소가 없으면 KeyError 발생
S.discard(elem)	집합 S에 원소 elem이 있으면 삭제
S.pop()	집합 S에서 임의의 원소 하나를 삭제하고, 반환 공 집합이면 KeyError 발생
S.clear()	집합 S의 모든 원소를 삭제

집합의 update - |=, &=, add(), pop(), clear()

```
# set - |=, &=, add(), pop(), clear()
s1 = {1, 2, 3, 4}
s2 = {5, 6, 7, 8}
print('s1 : ', s1)
print('s2 : ', s2)

s1 |= s2
print("after s1 |= s2, s1 = ", s1)
s1 &= s2
print("after s1 &= s2, s1 = ", s1)
s1.add(1)
print("after s1.add(1), s1 = ", s1)
s1.pop()
print("after s1.pop(), s1 = ", s1)
s1.clear()
print("after s1.clear(), s1 = ", s1)
```

```
s1 :  {1, 2, 3, 4}
s2 :  {8, 5, 6, 7}
after s1 |= s2, s1 =  {1, 2, 3, 4, 5, 6, 7, 8}
after s1 &= s2, s1 =  {8, 5, 6, 7}
after s1.add(1), s1 =  {1, 5, 6, 7, 8}
after s1.pop(), s1 =  {5, 6, 7, 8}
after s1.clear(), s1 =  set()
```

집합에 대한 연산 – add(), remove(), discard(), pop(), clear()

```
# set - add(), pop(), remove(), discard(), clear()
s1 = {1, 2, 3, 4, 5, 6, 7, 8}
print('s1 : ', s1)

s1.add(9)
print("after s1.add(1), s1 = ", s1)
s1.pop()
print("after s1.pop(), s1 = ", s1)
s1.remove(9)
print("after s1.remove(9), s1 = ", s1)
s1.discard(2)
print("after s1.discard(2), s1 = ", s1)
s1.clear()
print("after s1.clear(), s1 = ", s1)
```

```
s1 :  {1, 2, 3, 4, 5, 6, 7, 8}
after s1.add(1), s1 =  {1, 2, 3, 4, 5, 6, 7, 8, 9}
after s1.pop(), s1 =  {2, 3, 4, 5, 6, 7, 8, 9}
after s1.remove(9), s1 =  {2, 3, 4, 5, 6, 7, 8}
after s1.discard(2), s1 =  {3, 4, 5, 6, 7, 8}
after s1.clear(), s1 =  set()
```

Homework 4

Homework 4.1

4.1 파이썬 프로그램 실행 컴퓨터의 바이트 저장 순서 (byte ordering) 확인

- 16진수 0xABCDEF01을 big endian과 little endian의 바이트 저장 순서의 바이츠로 저장한 후, 이를 출력하여 바이트 저장 순서의 차이를 확인하라.
- Big endian과 little endian의 차이점에 대하여 설명하라.
- (실행 예)

```
d = 0xABCDEF01
d_bytes_BE =  b'\xab\xcd\xef\x01'
d_bytes_LE =  b'\x01\xef\xcd\xab'
```

Homework 4.2

4.2 2차원 리스트로 구성된 행렬의 곱셈 계산

- 3 x 4 크기의 2차원 리스트로 구성된 행렬 A와 4 x 3 크기의 2차원 리스트로 구성된 행렬 B를 준비하라.
- 행렬 A와 B의 곱셈을 계산하여 행렬 C에 저장하라.
- 행렬 A, B, C를 출력하여 정확하게 행렬 곱셈 계산이 된 것을 확인하라.
- (실행 예)

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

Homework 4.3

4.3 튜플 (year, month, day)로 표현되는 날짜 (date)을 표준 입력장치로 부터 10개 입력하여 날짜 튜플 리스트 (list of date-tuples)에 포함시킨 후, 이 날짜들을 오름차순으로 정렬하는 파이썬 프로그램을 작성하라. 입력 날짜는 무작위로 설정할 것.
(실행 예)

```
Input 10 dates in (year month day) format :
input year, month, day : 2021 3 20
L_dates = [(2021, 3, 20)]
input year, month, day : 2020 12 25
L_dates = [(2021, 3, 20), (2020, 12, 25)]
input year, month, day : 2021 1 1
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1)]
input year, month, day : 2021 12 31
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31)]
input year, month, day : 1 1 1
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1)]
input year, month, day : 2020 1 1
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (20
20, 1, 1)]
input year, month, day : 2019 12 25
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (20
20, 1, 1), (2019, 12, 25)]
input year, month, day : 2010 3 1
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (20
20, 1, 1), (2019, 12, 25), (2010, 3, 1)]
input year, month, day : 2021 4 1
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (20
20, 1, 1), (2019, 12, 25), (2010, 3, 1), (2021, 4, 1)]
input year, month, day : 10 10 10
L_dates = [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (20
20, 1, 1), (2019, 12, 25), (2010, 3, 1), (2021, 4, 1), (10, 10, 10)]
After input of 10 dates :
L_dates =  [(2021, 3, 20), (2020, 12, 25), (2021, 1, 1), (2021, 12, 31), (1, 1, 1), (2
020, 1, 1), (2019, 12, 25), (2010, 3, 1), (2021, 4, 1), (10, 10, 10)]
After sorting, L_dates =  [(1, 1, 1), (10, 10, 10), (2010, 3, 1), (2019, 12, 25), (202
0, 1, 1), (2020, 12, 25), (2021, 1, 1), (2021, 3, 20), (2021, 4, 1), (2021, 12, 31)]
```

Homework 4.4

4.4 튜플을 사용한 학생 정보 처리

튜플 (학생이름, 학번, 학과명, 평균성적)로 표현되는 학생 (Student)의 정보를 10개 준비하여 학생 튜플 리스트 (list of student-tuples)를 생성하라. 이 학생 튜플을 오름차순으로 정렬 (별도의 정렬 기준 설정 없이)하고, 결과를 출력하라. 다음으로 학생튜플을 학점을 기준으로 역순으로 정렬하여 출력하라.

(실행 예)

```
students[ 0] : name(Kim, S.C.), st_id(12001234), major(CE , GPA( 4.10))
students[ 1] : name(Choi, Y.B.), st_id(119003234), major(EE , GPA( 3.78))
students[ 2] : name(Hong, C.H.), st_id(21001547), major(ICE, GPA( 4.13))
students[ 3] : name(Yoon, J.H.), st_id(17002571), major(ME , GPA( 3.55))
students[ 4] : name(Lee, S.H.), st_id(20003257), major(ICE, GPA( 4.45))
students[ 5] : name(Kim, H.Y.), st_id(19001234), major(CE , GPA( 4.17))
students[ 6] : name(Lee, J.K), st_id(18003234), major(EE , GPA( 3.78))
students[ 7] : name(Park, S.Y.), st_id(21001643), major(ICE, GPA( 4.13))
students[ 8] : name(Jang, S.H.), st_id(19002567), major(ME , GPA( 3.35))
students[ 9] : name(Yeo, C.S), st_id(20005243), major(CE , GPA( 4.45))

After sorting in increasing order :
students[ 0] : name(Choi, Y.B.), st_id(119003234), major(EE , GPA( 3.78))
students[ 1] : name(Hong, C.H.), st_id(21001547), major(ICE, GPA( 4.13))
students[ 2] : name(Jang, S.H.), st_id(19002567), major(ME , GPA( 3.35))
students[ 3] : name(Kim, H.Y.), st_id(19001234), major(CE , GPA( 4.17))
students[ 4] : name(Kim, S.C.), st_id(12001234), major(CE , GPA( 4.10))
students[ 5] : name(Lee, J.K), st_id(18003234), major(EE , GPA( 3.78))
students[ 6] : name(Lee, S.H.), st_id(20003257), major(ICE, GPA( 4.45))
students[ 7] : name(Park, S.Y.), st_id(21001643), major(ICE, GPA( 4.13))
students[ 8] : name(Yeo, C.S), st_id(20005243), major(CE , GPA( 4.45))
students[ 9] : name(Yoon, J.H.), st_id(17002571), major(ME , GPA( 3.55))

After sorting according to GPA in decreasing order :
students[ 0] : name(Lee, S.H.), st_id(20003257), major(ICE, GPA( 4.45))
students[ 1] : name(Yeo, C.S), st_id(20005243), major(CE , GPA( 4.45))
students[ 2] : name(Kim, H.Y.), st_id(19001234), major(CE , GPA( 4.17))
students[ 3] : name(Hong, C.H.), st_id(21001547), major(ICE, GPA( 4.13))
students[ 4] : name(Park, S.Y.), st_id(21001643), major(ICE, GPA( 4.13))
students[ 5] : name(Kim, S.C.), st_id(12001234), major(CE , GPA( 4.10))
students[ 6] : name(Choi, Y.B.), st_id(119003234), major(EE , GPA( 3.78))
students[ 7] : name(Lee, J.K), st_id(18003234), major(EE , GPA( 3.78))
students[ 8] : name(Yoon, J.H.), st_id(17002571), major(ME , GPA( 3.55))
students[ 9] : name(Jang, S.H.), st_id(19002567), major(ME , GPA( 3.35))
```

Homework 4.5

4.5 집합과 딕셔너리를 사용한 정보 검색

- 총 10개의 국가에 대하여 문자열 (str) 자료형의 국가 이름과 수도 이름을 입력받고, 국가의 이름을 딕셔너리의 key로 사용하고, 그 국가의 수도의 이름 (문자열 자료형)을 value로 사용하는 딕셔너리 (dict_country_capital)를 구성하라.
- 입력장치로 부터 국가 이름을 입력받아 해당 국가의 수도 이름을 찾아내는 파이썬 프로그램을 작성하라.
- (실행 예)

```
Input nation and its capital (. to quit) : Korea Seoul
Input nation and its capital (. to quit) : Japan Tokyo
Input nation and its capital (. to quit) : China Beijing
Input nation and its capital (. to quit) : USA WashingtonDC
Input nation and its capital (. to quit) : UK London
Input nation and its capital (. to quit) : France Paris
Input nation and its capital (. to quit) : Italy Roma
Input nation and its capital (. to quit) : Germany Berlin
Input nation and its capital (. to quit) : Canada Ottawa
Input nation and its capital (. to quit) : Spain Madrid
dict_nation_capital :  {'Korea': 'Seoul', 'Japan': 'Tokyo', 'China': 'Beijing', 'USA': 'W
ashingtonDC', 'UK': 'London', 'France': 'Paris', 'Italy': 'Roma', 'Germany': 'Berlin', 'C
anada': 'Ottawa', 'Spain': 'Madrid'}
Input nation to find its capital (. to quit) : Canada
The capital of Canada is Ottawa
Input nation to find its capital (. to quit) : Korea
The capital of Korea is Seoul
Input nation to find its capital (. to quit) : USA
The capital of USA is WashingtonDC
Input nation to find its capital (. to quit) : .
```