컴퓨터정보과

파이선 프로그래밍

04주차 – 자료형

자료형과 기본 연산 (2장)

그룹	자료형	생성자	리터럴 표현	내용
수치형 (Number)	Integer	int()	1	정수
	Floating point number	float()	1.1	실수
	Complex Number	complex()	2 + 3j	복소수
논리형 (Boolean)	Boolean	bool()	True	참(True)과 거짓(False)만 값을 갖음
군집형 (Collection)	String	str()	“1” 혹은 ‘1’	문자열 (순서 O, 수정 X, 중복 O, 구성요소한정 O)
	List	**list()**	**[1,2]**	**리스트 (순서 O, 수정 O, 중복 O, 구성요소한정 X)**
	Tuple	tuple()	(1,2)	튜플 (순서 O, 수정 X, 중복 O, 구성요소한정 X)
	Set	set()	{1,2}	집합, 세트 (순서 X, 수정 O, 중복 X, 구성요소한정 X)
	Dictionary	dict()	{1:”1”, 2:”2}	사전, 딕셔너리 (순서 X, 수정 O, 중복 X, 구성요소한정 X) {key:value, … , key_n:$value_n$}

리스트

- List p.77
  - 순서가 있는 항목의 모음
  - 모든 종류의 값을 요소로 넣을 수 있음
    - 숫자, 문자열, 불, 기타 등등~ 모든…
  - 리스트에 들어있는 요소는 서로 연관될 필요가 없음
  - [] 대괄호로 표현, 요소는 , 콤마로 구분
  - 예제

```
list_1 = list()
list_2 = []
list_3 = list(["1", 2, 3.3])
list_4 = ["1", 2, 3.3]
list_5 = ["1", [2, 3.3]]
```

- *c언어의 배열과 다른 점은? -> 가변길이, 메모리에 비연속 할당*

리스트

- Indexing p.78
  - 문자열의 인덱싱과 동일하게 동작
    - 인덱싱은 순서번호가 있는 군집형의 타입에 쓰는 공통의 문법
  - 예제

```
data1 = [1, 2, 3, 4, 5]
print(data1[0])
print(data1[-1])
print(data1[0] + data1[-1])

data2 = [1, 2, 3, ["a", "b", "c"]]
print(data2[0], type(data2[0]))
print(data2[-1], type(data2[-1]))
print(data2[3][1], type(data2[3][1]))
```

```
>>> data1 = [1, 2, 3, 4, 5]
... print(data1[0])
... print(data1[-1])
... print(data1[0] + data1[-1])
...
1
5
6
```

  - *리스트 인덱싱의 결과는 요소 타입*

리스트

- Slicing p.80
  - 문자열의 슬라이싱과 동일하게 동작
    - 인덱싱은 순서번호가 있는 군집형의 타입에 쓰는 공통의 문법
  - 예제

```
data1 = [1, 2, 3, 4, 5]
print(data1[:2])
print(data1[2:])
print(data1[2:-1])

data2 = [1, 2, 3, ["a", "b", "c"]]
print(data2[2:], type(data2[2:0]))
print(data2[3][:2], type(data2[3][:2])) # 인덱싱 & 슬라이싱
```

```
>>> data1 = [1, 2, 3, 4, 5]
... print(data1[:2])
... print(data1[2:])
... print(data1[2:-1])
...
[1, 2]
[3, 4, 5]
[3, 4]
```

```
>>> data2 = [1, 2, 3, ["a", "b", "c"]]
... print(data2[2:], type(data2[2:0]))
... print(data2[3][:2], type(data2[3][:2]))
...
[3, ['a', 'b', 'c']] <class 'list'>
['a', 'b'] <class 'list'>
```

  - *리스트 슬라이싱의 결과는 리스트 타입*
  - *[:]의 결과는? 그리고 어떤 효과가? → 복사 (문자열 제외)*

```
>>> a = [1,2,3]
>>> b = a
>>> c = a[:]
>>> id(a)
2578081124928
>>> id(b)
2578081124928
>>> id(c)
2578072821952
```

리스트

- 관련 연산 및 함수 p.82
  - 연결하기 : +
  - 반복하기 : *
  - 길이 구하기 : len() 함수
  - 예제

```
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b
d = [a, b]
e = a * 3
print(len(c))
print(len(d))
print(len(e))
```

```
>>> a = [1, 2, 3]
... b = [4, 5, 6]
... c = a + b
... d = [a, b]
... e = a * 3
... print(len(c))
... print(len(d))
... print(len(e))
...
6
2
9
```

리스트

- 요소의 수정과 삭제 p.83
  - 수정
    - 인덱싱을 이용한 수정

```
motorcycles = ['honda', 'yamaha', 'suzuki ’ ]
motorcycles[0] = 'bmw'
```

  - 삭제
    - del 명령과 인덱싱/슬라이싱 이용

```
motorcycles = ['honda', 'yamaha', 'suzuki ’ , ‘ daelim ’ ]
del motorcycles[0]
del motorcycles[0:2]
```

리스트

- 관련 메소드 p.84

  메소드 : 리스트만 사용할 수 있는 전용 함수

  - append() : 리스트 끝에 요소를 추가 p.84
  - insert() : 리스트 특정 위치에 요소를 삽입 p.86
  - remove() : 리스트의 앞에서 검색해서 일치하는 요소 하나를 삭제 P.87
  - pop() : 리스트의 제일 뒤 요소를 제거하고 반환 p.87
    - 특정 위치를 사용해도 됨. (굳이 비교하면 뒤부터 move)
  - index() : 리스트에서 요소의 위치(인덱스) 찾기 p.86
    - 못 찾으면 오류 발생
  - count() : 리스트에 포함된 요소의 개수 p.88
  - extend() : 리스트에 다른 리스트를 추가 p.88
    - += 연산자와 동일
  - sort() : 리스트의 정렬 p.85
    - 원본 손상 (vs. sorted() 함수)
  - reverse() : 리스트의 요소 순서를 반대로 변경 p.85

자료형과 기본 연산 (2장)

그룹	자료형	생성자	리터럴 표현	내용
수치형 (Number)	Integer	int()	1	정수
	Floating point number	float()	1.1	실수
	Complex Number	complex()	2 + 3j	복소수
논리형 (Boolean)	Boolean	bool()	True	참(True)과 거짓(False)만 값을 갖음
군집형 (Collection)	String	str()	“1” 혹은 ‘1’	문자열 (순서 O, 수정 X, 중복 O, 구성요소한정 O)
	List	list()	[1,2]	리스트 (순서 O, 수정 O, 중복 O, 구성요소한정 X)
	Tuple	**tuple()**	**(1,2)**	**튜플 (순서 O, 수정 X, 중복 O, 구성요소한정 X)**
	Set	set()	{1,2}	집합, 세트 (순서 X, 수정 O, 중복 X, 구성요소한정 X)
	Dictionary	dict()	{1:”1”, 2:”2}	사전, 딕셔너리 (순서 X, 수정 O, 중복 X, 구성요소한정 X) {key:value, … , key_n:$value_n$}

튜플

- Tuple p.89
  - 모든 종류의 값이나 넣을 수 있음
    - 숫자, 문자열, 불, 기타 등등~ 모든…
  - 튜플에 들어있는 요소는 서로 연관될 필요가 없음
  - () 소괄호로 표현, 요소는 , 콤마로 구분
  - 튜플 생성 후 요소의 변경(생성,삭제,수정)은 불가능
    → 리스트와 다른 점
  - 예제

```
tuple_1 = tuple()
tuple_2 = ()
tuple_3 = (1,)
tuple_no = (1)
tuple_4 = "1", 2, 3.3
tuple_5 = ("1", (2, 3.3))
```

튜플

- Tuple 요소 변경 p.90
  - 삭제/변경 불가능
- 가능한 연산 및 함수 p.91
  - + / * / len()

```
a = (1, 2, 3)
b = (4, 5, 6)
c = a + b
d = a * 3
print(len(c))
print(len(d))
```

  - 인덱싱 / 슬라이싱

```
a = (1, 2, 3, 4, 5)
b = a[1] #결과는 요소 타입
c = a[1:] #결과는 tuple
```

```
>>> a = (1, 2, 3)
... b = (4, 5, 6)
... c = a + b
... d = a * 3
... print(len(c))
... print(len(d))
...
6
9
```

자료형과 기본 연산 (2장)

그룹	자료형	생성자	리터럴 표현	내용
수치형 (Number)	Integer	int()	1	정수
	Floating point number	float()	1.1	실수
	Complex Number	complex()	2 + 3j	복소수
논리형 (Boolean)	Boolean	bool()	True	참(True)과 거짓(False)만 값을 갖음
군집형 (Collection)	String	str()	“1” 혹은 ‘1’	문자열 (순서 O, 수정 X, 중복 O, 구성요소한정 O)
	List	list()	[1,2]	리스트 (순서 O, 수정 O, 중복 O, 구성요소한정 X)
	Tuple	tuple()	(1,2)	튜플 (순서 O, 수정 X, 중복 O, 구성요소한정 X)
	Set	**set()**	**{1,2}**	**집합, 세트 (순서 X, 수정 O, 중복 X, 구성요소한정 X)**
	Dictionary	dict()	{1:”1”, 2:”2}	사전, 딕셔너리 (순서 X, 수정 O, 중복 X, 구성요소한정 X) {key:value, … , keyn:valuen}

집합

- Set p.102
  - 집합에 관련한 것을 쉽게 처리하기 위해 만든 자료형
    - 형태 : { 값1, 값2, 값3} (중괄호를 사용해 값을 감싼다)
    - 특징 : 중복 불허, 순서 없음 (인덱싱으로 값을 얻을 수 없음)
    - 중복을 제거하기 위한 필터용으로도 사용된다.

```python
s0 = {1, 2, 3, 3, 4, 5}
s1 = {1, 4, 4, 4, 5, 6, 7, 7, 10}
s2 = set("Hello")
print(s0)
print(s1)
print(s2)

s3 = list(s1)  # 리스트로 변환
s4 = tuple(s1)  # 튜플로 변환
print(s3)
print(s4)
```

```
{1, 2, 3, 4, 5}
{1, 4, 5, 6, 7, 10}
{'e', 'o', 'H', 'l'}
[1, 4, 5, 6, 7, 10]
(1, 4, 5, 6, 7, 10)
```

```python
# 교집합
isc1 = s0 & s1
isc2 = s0.intersection(s1)
print(isc1)
print(isc2)

# 합집합
uni1 = s0 | s1
uni2 = s0.union(s1)
print(uni1)
print(uni2)

# 차집합
dif1 = s0 - s1
dif2 = s0.difference(s1)
print(dif1)
print(dif2)
```

```
{1, 4, 5}
{1, 4, 5}
{1, 2, 3, 4, 5, 6, 7, 10}
{1, 2, 3, 4, 5, 6, 7, 10}
{2, 3}
{2, 3}
```

```python
# 특정값 추가
s5 = {1, 2, 3}
s5.add(44)
print(s5)

# 여러값 추가
s5.update([4, 5, 6])
print(s5)

# 특정값 제거
s5.remove(1)
print(s5)
```

```
{1, 2, 3, 44}
{1, 2, 3, 4, 5, 6, 44}
{2, 3, 4, 5, 6, 44}
```