기초 PYTHON 프로그래밍

10. 리스트(list), 튜플(tuple)

1. 리스트와 연산
2. 리스트 사용하기
3. 튜플과 연산
4. 튜플 사용하기

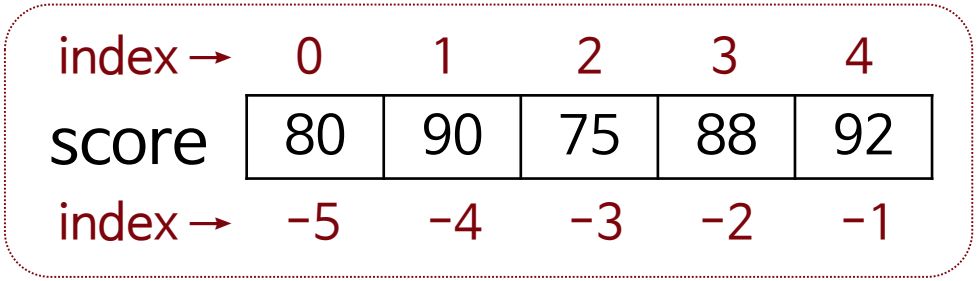
서강대학교
SOGANG UNIVERSITY

1. 리스트와 연산

서강대학교 SOGANG UNIVERSITY

- 대괄호 []를 사용하여 여러 데이터를 저장할 수 있는 자료형
- 순서가 있고, 인덱스를 이용하여 데이터에 접근할 수 있다.
- 리스트는 **mutable** 하다

```
>>> score = [80, 90, 75, 88, 92]
>>> print(score)
[80, 90, 75, 88, 92]
>>> print(score[2])
75
>>> print(score[-2])
88
```




```
>>> score[2] = 78      # mutable
>>> print(score)
[80, 90, 78, 88, 92]
```

1. 리스트와 연산

서강대학교 SOGANG UNIVERSITY

◆ 슬라이싱 (slicing)

A = [15, 27, 32, 20, 17, 13, 10, 22]

0	1	2	3	4	5	6	7
15	27	32	20	17	13	10	22
-8	-7	-6	-5	-4	-3	-2	-1

```
A[:]       # [15, 27, 32, 20, 17, 13, 10, 22]
A[:5]      # [15, 27, 32, 20, 17]
A[0:5]     # [15, 27, 32, 20, 17]
A[2:5]     # [32, 20, 17]
A[2:-1]    # [32, 20, 17, 13, 10]
A[:-1]     # [15, 27, 32, 20, 17, 13, 10]
A[-3:-1]   # [13, 10]
```

1. 리스트와 연산

서강대학교 SOGANG UNIVERSITY

◆ 슬라이싱 (slicing)

B = ['a','b','c','d','e','f','g','h','i','j','k']

0	1	2	3	4	5	6	7	8	9	10
a	b	c	d	e	f	g	h	i	j	k
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
B[::]          # ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k']
B[2:9:3]       # ['c', 'f', 'i']
B[0:-1:2]      # ['a', 'c', 'e', 'g', 'i']
B[::2]         # ['a', 'c', 'e', 'g', 'i', 'k']
B[-1:-7:-2]    # ['k', 'i', 'g']
```

1. 리스트와 연산

서강대학교 SOGANG UNIVERSITY

- 리스트에 대한 +(연결) , *(반복) 연산

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

>>> L + M              # 리스트 L과 리스트 M을 연결한 리스트를 반환한다.
[1, 3, 5, 7, 9, 2, 4, 6, 8]

>>> print(L, M)    # L과 M은 변하지 않는다.
[1, 3, 5, 7, 9]  [2, 4, 6, 8]

>>> K = L + M
>>> print(K)
[1, 3, 5, 7, 9, 2, 4, 6, 8]

>>> L * 3              # 리스트 L을 3회 반복한 리스트를 반환한다.
[1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
```

1. 리스트와 연산

서강대학교 SOGANG UNIVERSITY

- in, not in, len()

```
>>> L = [1,3,5,7,9]
>>> 7 in L          # 리스트 L에 데이터 7이 있다면 True
True
>>> 10 in L
False
>>> 10 not in L    # 리스트 L에 데이터 10이 없다면 True
True
>>> len(L)
5
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 메소드 (method) - 데이터 객체에 대해서 어떤 일을 처리할 수 있도록 하는 코드
- 리스트에서 사용할 수 있는 메소드

```
〉〉〉 dir(list)
['__add__', '__class__', …… '__subclasshook__', 'append',
'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
```

- 리스트.메소드 의 형태로 사용해야 한다.

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

메소드	설명
append (x)	데이터 x를 리스트 끝에 추가한다.
clear()	리스트를 비운다.
copy()	리스트를 복사한다.
count (x)	데이터 x의 개수를 알아낸다.
extend (M)	리스트 M을 연결한다.
index (x)	데이터 x의 위치를 알아낸다 (인덱스를 알려준다).
insert (i,x)	데이터 x를 지정한 위치 (인덱스 i)에 삽입한다.
pop()	리스트의 지정한 값 하나를 읽어 내고 삭제한다.
remove (x)	리스트에서 데이터 x를 삭제한다.
reverse()	리스트의 순서를 역순으로 바꾼다.
sort()	리스트를 정렬한다.

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 리스트에 데이터 추가하기
  - append(x) - 데이터 x를 리스트 끝에 추가한다.
  - insert(i,x) - 인덱스 i에 데이터 x를 삽입한다.

```
>>> L = [1,4,7,8]
>>> L.append(5)
>>> print(L)
[1, 4, 7, 8, 5]
>>> L.insert(2,10)
>>> print(L)
[1, 4, 10, 7, 8, 5]
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 리스트에서 데이터 삭제하기
  - pop() - 리스트의 맨 마지막 데이터를 반환하고 삭제한다.
  - pop(i) - 리스트에서 인덱스 i 에 있는 데이터를 반환하고 삭제한다.
  - remove(x) - 리스트에서 데이터 x를 삭제한다. x가 여러 개 있으면 맨 처음 x를 삭제한다.

```
>>> L = [1,2,3,4,5]
>>> L.pop()
5
>>> print(L)
[1, 2, 3, 4]
>>> L.pop(2)
3
>>> print(L)
[1, 2, 4]
```

```
>>> L = [4,2,5,3,2,7,1,2,5]
>>> L.remove(7)
>>> print(L)
[4, 2, 5, 3, 2, 1, 2, 5]
>>> L.remove(2)
>>> print(L)
[4, 5, 3, 2, 1, 2, 5]
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 리스트에서 데이터 위치 찾기 / 데이터 개수 세기
  - index(x) - 리스트에서 데이터 x의 인덱스를 반환한다.
  - count(x) - 리스트에서 데이터 x의 개수를 반환한다.

```
>>> L = ['g','a','c','f','g','b']
>>> L.index('f')
3
>>> L.index('g')   # 데이터가 여러 개이면 첫 번째 인덱스 반환
0
>>> L.count('g')
2
>>> L.count('b')
1
>>> L.count('m')
0
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 빈 리스트와 리스트 비우기 – clear()

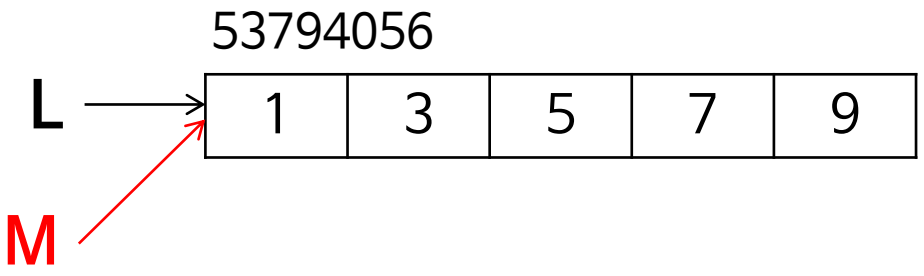
```
>>> A = []    # 빈 리스트
>>> A.append('red')
>>> A.append('blue')
>>> A.append('yellow')
>>> print(A)
['red', 'blue', 'yellow']
>>> A.clear()
>>> print(A)
[]
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 리스트 복사하기 ('=' 이용하기)

```
>>> L = [1,3,5,7,9]
>>> id(L)
53794056
>>> M = L
>>> M[2] = 100
>>> print(L)
[1, 3, 100, 7, 9]
>>> print(M)
[1, 3, 100, 7, 9]
>>> id(M)
53794056
```

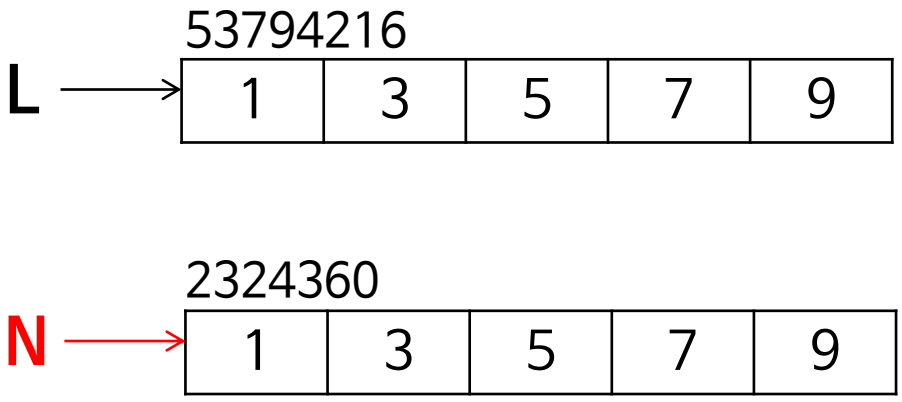


2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

◆ 리스트 복사하기 - copy()

```
>>> L = [1,3,5,7,9]
>>> N = L.copy()
>>> N[2] = 100
>>> print(L)
[1, 3, 5, 7, 9]
>>> print(N)
[1, 3, 100, 7, 9]
>>> id(L)
53794216
>>> id(N)
2324360
```



2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

◆ 리스트 연결하기 - extend(M)

```
>>> L = [1,3,5]
>>> M = [10,20]
>>> L.extend(M)      # L에 M를 연결한다.
>>> print(L)
[1, 3, 5, 10, 20]
>>> print(M)
[10, 20]
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

◆ 리스트 정렬하기 - sort()

```
>>> L = [3,5,1,4,2]
>>> L.sort()                    # 오름차순으로 정렬
>>> print(L)
[1, 2, 3, 4, 5]
>>> L = [3,5,1,4,2]
>>> L.sort(reverse=True)    # 내림차순으로 정렬
>>> print(L)
[5, 4, 3, 2, 1]
```

2. 리스트 사용하기

서강대학교 SOGANG UNIVERSITY

- 리스트 역순으로 바꾸기 - reverse()

```
>>> L = [3,5,1,4,2]
>>> L.reverse()
>>> print(L)
[2, 4, 1, 5, 3]
```

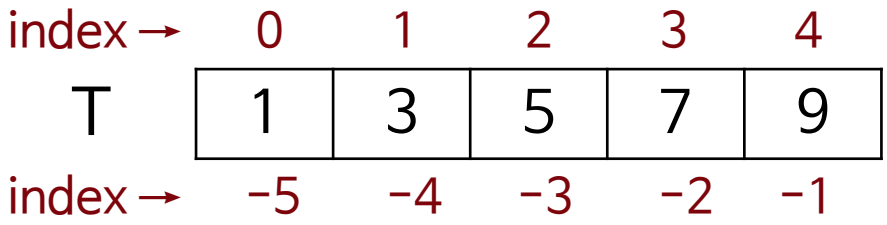
3. 튜플과 연산

서강대학교 SOGANG UNIVERSITY

- 리스트처럼 여러 데이터를 저장할 수 있는 자료형
- 인덱싱, 슬라이싱 이용할 수 있다.
- +, *, in, not in, len() 사용할 수 있다.
- [] 대신에 ( )를 사용
- 리스트와의 차이점 - **튜플은 immutable 하다.**

3. 튜플과 연산

```
>>> T = (1,3,5,7,9)
>>> M = (2,4,6,8)
>>> print(T[2])
5
>>> T + M
(1, 3, 5, 7, 9, 2, 4, 6, 8)
>>> M * 3
(2, 4, 6, 8, 2, 4, 6, 8)
>>> 7 in T
True
>>> len(T)
5
```



튜플에서는 다음과 같이 데이터를 바꾸는 일이 불가능하다.

```
>>> T[2] = 10
……
TypeError: 'tuple' object does not support item assignment
```

4. 튜플 사용하기

서강대학교 SOGANG UNIVERSITY

메소드	설명
index(x)	튜플에서 데이터 x의 인덱스를 반환한다.
count(x)	튜플에서 데이터 x의 개수를 반환한다.

```
>>> T = (4,3,5,7,6,5,7,4,7,2)
>>> T.count(7)    # 7의 개수
3
>>> T.index(2)    # 데이터 2의 첫 번째 인덱스
9
>>> T.index(7)    # 데이터 7의 첫 번째 인덱스. 없으면 에러 발생.
3
```

```
>>> dir(tuple)
['__add__', '__class__', …… '__subclasshook__', 'count', 'index']
```

4. 튜플 사용하기

서강대학교 SOGANG UNIVERSITY

- 빈 튜플

```
T = ()
T = tuple()
```

- 원소가 하나인 튜플 만들기

```
>>> T = (1,)        # 콤마 필요
>>> type(T)
<class 'tuple'>

>>> S = (1)     # S = 1 과 같음
>>> type(S)
<class 'int'>
```

```
>>> A = 1,2,3
>>> type(A)
<class 'tuple'>
>>> B = (4,5,6)
>>> type(B)
<class 'tuple'>
```

```
>>> C = 10
>>> D = 20,
>>> type(C)
<class 'int'>
>>> type(D)
<class 'tuple'>
```

콤마를 사용하면 괄호가 없어도 튜플로 인식한다.

4. 튜플 사용하기

서강대학교 SOGANG UNIVERSITY

- 튜플은 여러 변수에 값을 동시에 할당할 수 있도록 한다.

```
>>> a,b,c = 1,2,3
>>> (a,b,c) = (1,2,3)
```

```
>>> a = 1
>>> b = 2
>>> c = 3
```

- 튜플 이용한 swap

```
>>> x=10; y=20
>>> print(x,y)
10 20
>>> x,y = y,x
>>> print(x,y)
20 10
```

- 튜플은 함수에서 유용하게 사용된다.
- 튜플은 immutable하므로 프로그램 수행 동안 변경하지 않아야 하는 데이터는 튜플로 저장하는 것이 좋다.