02-3 리스트 자료형

지금까지 우리는 숫자와 문자열에 대해서 알아보았다. 하지만 숫자와 문자열만으로 프로그래밍을 하기엔 부족한 점이 많다. 예를 들어 1부터 10까지의 숫자 중 홀수 모음인 1, 3, 5, 7, 9의 집합을 생각해 보자. 이런 숫자 모음을 숫자나 문자열로 표현하기는 쉽지 않다. 파이썬에는 이러한 불편함을 해소할 수 있는 자료형이 존재한다. 그것이 바로 여기에서 공부할 리스트(List)이다.

리스트는 어떻게 만들고 사용할까?

리스트를 사용하면 1, 3, 5, 7, 9 숫자 모음을 다음과 같이 간단하게 표현할 수 있다.

```
>>> odd = [1, 3, 5, 7, 9]
```

리스트를 만들 때는 위에서 보는 것과 같이 대괄호([ ])로 감싸 주고 각 요솟값은 쉼표(,)로 구분해 준다.

```
리스트명 = [요소1, 요소2, 요소3, ...]
```

여러 가지 리스트의 생김새를 살펴보면 다음과 같다.

```
>>> a = []
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]
```

리스트는 a처럼 아무것도 포함하지 않아 비어 있는 리스트([ ])일 수도 있고 b처럼 숫자를 요솟값으로 가질 수도 있고 c처럼 문자열을 요솟값으로 가질 수도 있다. 또한 d처럼 숫자와 문자열을 함께 요솟값으로 가질 수도 있으며 e처럼 리스트 자체를 요솟값으로 가질 수도 있다. 즉 리스트 안에는 어떠한 자료형도 포함시킬 수 있다.

> ※ 비어 있는 리스트는 a = list()로 생성할 수도 있다.

리스트의 인덱싱과 슬라이싱

리스트도 문자열처럼 인덱싱과 슬라이싱이 가능하다. 백문이 불여일견. 말로 설명하는 것보다 직접 예를 실행해 보면서 리스트의 기본 구조를 이해하는 것이 쉽다. 대화형 인터프리터로 따라 하며 확실하게 이해하자.

리스트의 인덱싱

리스트 역시 문자열처럼 인덱싱을 적용할 수 있다. 먼저 a 변수에 [1, 2, 3] 값을 설정한다.

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

a[0]은 리스트 a의 첫 번째 요솟값을 말한다.

```
>>> a[0]
1
```

다음 예는 리스트의 첫 번째 요소인 a[0]과 세 번째 요소인 a[2]의 값을 더한 것이다.

```
>>> a[0] + a[2]
4
```

이것은 1 + 3으로 해석되어 값 4를 출력한다.

문자열을 공부할 때 이미 살펴보았지만 파이썬은 숫자를 0부터 세기 때문에 a[1]이 리스트 a의 첫 번째 요소가 아니라 a[0]이 리스트 a의 첫 번째 요소임을 명심하자. a[-1]은 문자열에서와 마찬가지로 리스트 a의 마지막 요솟값을 말한다.

```
>>> a[-1]
3
```

이번에는 다음 예처럼 리스트 a를 숫자 1, 2, 3과 또 다른 리스트인 ['a', 'b', 'c']를 포함하도록 만들어 보자.

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

다음 예를 따라 해 보자.

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
>>> a[3]
['a', 'b', 'c']
```

예상한 대로 a[-1]은 마지막 요솟값 ['a', 'b', 'c']를 나타낸다. a[3]은 리스트 a의 네 번째 요소를 나타내기 때문에 마지막 요소를 나타내는 a[-1]과 동일한 결괏값을 보여 준다.

그렇다면 여기에서 리스트 a에 포함된 ['a', 'b', 'c'] 리스트에서 'a' 값을 인덱싱을 사용해 끄집어낼 수 있는 방법은 없을까? 다음 예를 보자.

```
>>> a[-1][0]
'a'
```

위와 같이 하면 'a'를 끄집어낼 수 있다. a[-1]이 ['a', 'b', 'c'] 리스트라는 것은 이미 말했다. 바로 이 리스트에서 첫 번째 요소를 불러오기 위해 [0]을 붙여 준 것이다.

다음 예도 마찬가지 경우이므로 어렵지 않게 이해될 것이다.

```
>>> a[-1][1]
'b'
>>> a[-1][2]
'c'
```

점프 투 파이썬

[삼중 리스트에서 인덱싱하기]

조금 복잡하지만 다음 예를 따라 해 보자.

```
>>> a = [1, 2, ['a', 'b', ['Life', 'is']]]
```

리스트 a 안에 ['a', 'b', ['Life', 'is']] 리스트가 포함되어 있고, 그 리스트 안에 다시 ['Life', 'is'] 리스트가 포함되어 있다. 삼중 구조의 리스트이다.

이 경우 'Life' 문자열만 끄집어내려면 다음과 같이 해야 한다.

```
>>> a[2][2][0]
'Life'
```

위 예는 리스트 a의 세 번째 요소인 리스트 ['a', 'b', ['Life', 'is']]에서 세 번째 요소인 리스트 ['Life', 'is']의 첫 번째 요소를 나타낸다.

이렇듯 리스트를 삼중으로 중첩해서 쓰면 혼란스럽기 때문에 자주 사용하지는 않지만 알아두는 것이 좋다.

리스트의 슬라이싱

문자열과 마찬가지로 리스트에서도 슬라이싱 기법을 적용할 수 있다. 슬라이싱은 '나눈다'는 뜻이라고 했다.

자, 그럼 리스트의 슬라이싱에 대해서 살펴보자.

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

앞의 예를 문자열에서 슬라이싱했던 것과 비교해 보자.

```
>>> a = "12345"
>>> a[0:2]
'12'
```

2가지가 완전히 동일하게 사용되었음을 눈치챘을 것이다. 문자열에서 했던 것과 사용법이 완전히 동일하다.

몇 가지 예를 더 들어 보자.

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a[:2]
>>> c = a[2:]
>>> b
[1, 2]
>>> c
[3, 4, 5]
```

b 변수는 리스트 a의 첫 번째 요소부터 두 번째 요소인 a[1]까지 나타내는 리스트이다. 물론 a[2] 값인 3은 포함되지 않는다. c라는 변수는 리스트 a의 세 번째 요소부터 끝까지 나타내는 리스트이다.

점프 투 파이썬

[중첩된 리스트에서 슬라이싱하기]

리스트가 포함된 중첩 리스트 역시 슬라이싱 방법은 똑같이 적용된다.

```
>>> a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]
>>> a[2:5]
[3, ['a', 'b', 'c'], 4]
>>> a[3][:2]
['a', 'b']
```

위 예에서 a[3]은 ['a', 'b', 'c']를 나타낸다. 따라서 a[3][:2]는 ['a', 'b', 'c']의 첫 번째 요소부터 세 번째 요소 직전까지의 값, 즉 ['a', 'b']를 나타내는 리스트가 된다.

리스트 연산하기

리스트 역시 + 기호를 사용해서 더할 수 있고 * 기호를 사용해서 반복할 수 있다. 문자열과 마찬가지로 리스트에서도 되는지 직접 확인해 보자.

리스트 더하기(+)

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

리스트 사이에서 + 기호는 2개의 리스트를 합치는 기능을 한다. 문자열에서 "abc" + "def" = "abcdef" 가 되는 것과 같은 이치이다.

리스트 반복하기(*)

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

위에서 볼 수 있듯이 [1, 2, 3] 리스트가 세 번 반복되어 새로운 리스트를 만들어낸다. 문자열에서 `"abc" * 3 = "abcabcabc"` 가 되는 것과 같은 이치이다.

리스트 길이구하기

리스트 길이를 구하기 위해서는 다음처럼 len 함수를 사용해야 한다.

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

len 함수는 문자열, 리스트 외에 앞으로 배울 튜플과 딕셔너리에도 사용할 수 있는 함수이다. 실습에서 자주 사용하니 잘 기억해 두자.

점프 투 파이썬

[초보자가 범하기 쉬운 리스트 연산 오류]

다음 소스 코드를 입력했을 때 결괏값은 어떻게 나올까?

```
>>> a = [1, 2, 3]
>>> a[2] + "hi"
```

a[2]의 값인 3과 문자열 hi가 더해져서 3hi가 출력될 것이라고 생각할 수 있다. 하지만 다음 결과를 보자. 형 오류(TypeError)가 발생했다. 오류의 원인은 무엇일까?

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

a[2]에 저장된 값은 3이라는 정수인데 "hi"는 문자열이다. 정수와 문자열은 당연히 서로 더할 수 없기 때문에 형 오류가 발생한 것이다.

만약 숫자와 문자열을 더해서 '3hi'처럼 만들고 싶다면 숫자 3을 문자 '3'으로 바꾸어 주어야 한다. 다음과 같이 할 수 있다.

```
>>>str(a[2]) + "hi"
```

str 함수는 정수나 실수를 문자열의 형태로 바꾸어 주는 파이썬의 내장 함수이다.

리스트의 수정과 삭제

리스트는 값을 수정하거나 삭제할 수 있다.

리스트에서 값 수정하기

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

a[2]의 요솟값 3이 4로 바뀌었다.

del 함수 사용해 리스트 요소 삭제하기

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

del a[x]는 x번째 요솟값을 삭제한다. 여기에서는 a 리스트에서 a[1]을 삭제하는 방법을 보여준다. del 함수는 파이썬이 자체적으로 가지고 있는 삭제 함수이며 다음과 같이 사용한다.

```
del 객체
```

※ 객체란 파이썬에서 사용되는 모든 자료형을 말한다.

다음처럼 슬라이싱 기법을 사용하여 리스트의 요소 여러 개를 한꺼번에 삭제할 수도 있다.

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

a[2:]에 해당하는 리스트의 요소들이 삭제되었다.

리스트의 요소를 삭제하는 방법에는 2가지가 더 있다. 그것은 리스트의 remove와 pop 함수를 사용하는 방법인데 이것에 대해서는 바로 이어지는 리스트 관련 함수에서 설명한다.

리스트 관련 함수들

문자열과 마찬가지로 리스트 변수 이름 뒤에 '.'를 붙여서 여러 가지 리스트 관련 함수를 사용할 수 있다. 유용하게 사용되는 리스트 관련 함수 몇 가지에 대해서만 알아보기로 하자.

리스트에 요소 추가(append)

append를 사전에서 검색해 보면 "덧붙이다, 첨부하다"라는 뜻이 있다. 이 뜻을 안다면 다음 예가 바로 이해될 것이다. append(x)는 리스트의 맨 마지막에 x를 추가하는 함수이다.

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

리스트 안에는 어떤 자료형도 추가할 수 있다.

다음 예는 리스트에 다시 리스트를 추가한 결과이다.

```
>>> a.append([5,6])
>>> a
[1, 2, 3, 4, [5, 6]]
```

리스트 정렬(sort)

sort 함수는 리스트의 요소를 순서대로 정렬해 준다.

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

문자 역시 알파벳 순서로 정렬할 수 있다.

```
>>> a = ['a', 'c', 'b']
>>> a.sort()
>>> a
['a', 'b', 'c']
```

리스트 뒤집기(reverse)

reverse 함수는 리스트를 역순으로 뒤집어 준다. 이때 리스트 요소들을 순서대로 정렬한 다음 다시 역순으로 정렬하는 것이 아니라 그저 현재의 리스트를 그대로 거꾸로 뒤집는다.

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

위치 반환(index)

index(x) 함수는 리스트에 x 값이 있으면 x의 위치 값을 돌려준다.

```
>>> a = [1,2,3]
>>> a.index(3)
2
>>> a.index(1)
0
```

위 예에서 리스트 a에 있는 숫자 3의 위치는 a[2]이므로 2를 돌려주고, 숫자 1의 위치는 a[0]이므로 0을 돌려준다.

다음 예에서 값 0은 a 리스트에 존재하지 않기 때문에 값 오류(ValueError)가 발생한다.

```
>>> a.index(0)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 0 is not in list
```

리스트에 요소 삽입(insert)

insert(a, b)는 리스트의 a번째 위치에 b를 삽입하는 함수이다. 파이썬에서는 숫자를 0부터 센다는 것을 반드시 기억하자.

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a
[4, 1, 2, 3]
```

위 예는 0번째 자리, 즉 첫 번째 요소(a[0]) 위치에 값 4를 삽입하라는 뜻이다.

```
>>> a.insert(3, 5)
>>> a
[4, 1, 2, 5, 3]
```

위 예는 리스트 a의 a[3], 즉 네 번째 요소 위치에 값 5를 삽입하라는 뜻이다.

리스트 요소 제거(remove)

remove(x)는 리스트에서 첫 번째로 나오는 x를 삭제하는 함수이다.

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```

a가 3이라는 값을 2개 가지고 있을 경우 첫 번째 3만 제거되는 것을 알 수 있다.

```
>>> a.remove(3)
>>> a
[1, 2, 1, 2]
```

remove(3)을 한 번 더 실행하면 다시 3이 삭제된다.

리스트 요소 끄집어내기(pop)

pop()은 리스트의 맨 마지막 요소를 돌려주고 그 요소는 삭제한다.

```
>>> a = [1,2,3]
>>> a.pop()
3
>>> a
[1, 2]
```

a 리스트 [1, 2, 3]에서 3을 끄집어내고 최종적으로 [1, 2]만 남는 것을 볼 수 있다.

pop(x)는 리스트의 x번째 요소를 돌려주고 그 요소는 삭제한다.

```
>>> a = [1,2,3]
>>> a.pop(1)
2
>>> a
[1, 3]
```

a.pop(1)은 a[1]의 값을 끄집어낸다. 다시 a를 출력해 보면 끄집어낸 값이 삭제된 것을 확인할 수 있다.

리스트에 포함된 요소 x의 개수 세기(count)

count(x)는 리스트 안에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수이다.

```
>>> a = [1,2,3,1]
>>> a.count(1)
2
```

1이라는 값이 리스트 a에 2개 들어 있으므로 2를 돌려준다.

리스트 확장(extend)

extend(x)에서 x에는 리스트만 올 수 있으며 원래의 a 리스트에 x 리스트를 더하게 된다.

```
>>> a = [1,2,3]
>>> a.extend([4,5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

a.extend([4, 5])는 a += [4, 5]와 동일하다.

> `a += [4, 5]`는 `a = a + [4, 5]`와 동일한 표현식이다.