

Unit 11.

☀ Status	완료
👤 담당자	
📅 마감일	
📅 완료일	@2022년 11월 22일

Unit 11. 시퀀스 자료형 활용하기

1. 지금까지 배운 리스트, 튜플, 레인지, 문자열을 보면 값이 연속적이다

리스트: [0, 10, 20, 30, 40, 50]

0	10	20	30	40	50
---	----	----	----	----	----

튜플: (38, 76, 53, 62, 19)

38	76	53	62	19
----	----	----	----	----

range: range(5)

0	1	2	3	4
---	---	---	---	---

문자열: 'Hello'

H	e	l	l	o
---	---	---	---	---

2. 값이 연속적으로 이어진 자료형을 시퀀스 자료형

11.1 시퀀스 자료형의 공통 기능 사용하기

- 시퀀스 자료형의 특징은 공통된 동작과 기능을 제공한다
- 시퀀스 자료형으로 만든 객체를 시퀀스 객체라고 하며, 시퀀스 객체에 들어있는 각 값을 요소(element)라고 부른다

11.1.1 특정 값이 있는지 확인하기

- 특정 값이 있는지 확인할때 in 연산자나 not in 연산자 사용
- ex) 값 in 시퀀스 객체, ex) 값 not in 시퀀스 객체

11.1.2 시퀀스 객체 연결하기

- 시퀀스 객체는 +연산자를 사용해서 객체를 서로 연결 가능

```
>>> a = [0, 10, 20, 30]
>>> b = [9, 8, 7, 6]
>>> a + b
[0, 10, 20, 30, 9, 8, 7, 6]
```

!!! range는 +연산자로 객체를 연결 할 수 없어서 리스트나 튜플로 만들어서 연결

```
>>> list(range(0, 10)) + list(range(10, 20))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> tuple(range(0, 10)) + tuple(range(10, 20))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19)
```

11.1.3 시퀀스 객체 반복하기

- *연산자를 사용하여 특정 횟수만큼 반복하여 새 시퀀스 객체를 만든다

```
>>> [0, 10, 20, 30] * 3
[0, 10, 20, 30, 0, 10, 20, 30, 0, 10, 20, 30]
```

!!! range는 *연산자로 반복할 수 없다

- 이때도 리스트나 튜플로 만들어서 반복

```
>>> list(range(0, 5, 2)) * 3
[0, 2, 4, 0, 2, 4, 0, 2, 4]
>>> tuple(range(0, 5, 2)) * 3
(0, 2, 4, 0, 2, 4, 0, 2, 4)
```

11.2.1 리스트와 튜플의 요소 개수 구하기

- 요소의 개수를 구할때 len 함수를 사용한다

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> len(a)
10
```

11.2.2 range의 숫자 생성 개수 구하기

- range에 len 함수를 사용하면 숫자가 생성되는 개수를 구한다.
- 지금까지는 구하는 요소들이 한눈에 보였지만 뒤로 갈수록 요소가 한눈에 보이지 않아서 len함수가 필요하다.

11.2.3 문자열 길이 구하기

```
>>> hello = 'Hello, world!'
>>> len(hello)
13
```

11.3 인덱스 사용하기

1. 시퀀스 객체의 각 요소에는 순서가 정해져 있고 이 순서를 인덱스라고 한다.
2. 시퀀스 객체에 []를 붙이고 [] 안에 각 요소의 인덱스를 지정

- 시퀀스 객체[인덱스]

1. 인덱스는 항상 0부터 시작

```
>>> a = [38, 21, 53, 62, 19]
>>> a[0]      # 리스트의 첫 번째(인덱스 0) 요소 출력
38
>>> a[2]      # 리스트의 세 번째(인덱스 2) 요소 출력
53
>>> a[4]      # 리스트의 다섯 번째(인덱스 4) 요소 출력
19
```

- 튜플, range, 문자열도 []에 인덱스를 지정하면 해당 요소를 가져올 수 있습니다

```
>>> b = (38, 21, 53, 62, 19)
>>> b[0]      # 튜플의 첫 번째(인덱스 0) 요소 출력
38
```

11.3.1 음수 인덱스 지정하기

- 인덱스를 음수로 지정하면 뒤에서부터 요소에 접근 -1은 뒤에서 첫번째

```
>>> a = [38, 21, 53, 62, 19]
>>> a[-1]      # 리스트의 뒤에서 첫 번째(인덱스 -1) 요소 출력
19
>>> a[-5]      # 리스트의 뒤에서 다섯 번째(인덱스 -5) 요소 출력
38
```

11.4 슬라이스 사용하기

- 시퀀스객체[시작인덱스:끝인덱스]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[0:4]      # 인덱스 0부터 3까지 잘라서 새 리스트를 만들
[0, 10, 20, 30]
```

!!! 끝 인덱스는 실제로 가져오려는 인덱스보다 1을 크게 지정해야 한다

11.4.2 인덱스 증가폭 사용하기

- 시퀀스객체[시작인덱스:끝인덱스:인덱스증가폭]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:8:3]    # 인덱스 2부터 3씩 증가시키면서 인덱스 7까지 가져옴
[20, 50]
```

11.4.3 인덱스 생략하기

- 시퀀스 객체의 길이를 몰라도 될때 인덱스를 생략
- 시퀀스객체[:끝인덱스]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[:7]       # 리스트 처음부터 인덱스 6까지 가져옴
[0, 10, 20, 30, 40, 50, 60]
```

- 시작 인덱스 생략하기
- 시퀀스객체[시작인덱스:]

```
>>> a[7:]       # 인덱스 7부터 마지막 요소까지 가져옴
[70, 80, 90]
```

- 시작 인덱스와 끝 인덱스 둘다 생략
- 시퀀스객체[:]

```
>>> a[:]        # 리스트 전체를 가져옴
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

11.4.4 인덱스를 생략하면서 증가폭 사용하기

- 시퀀스객체[:끝인덱스:증가폭]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[:7:2]     # 리스트의 처음부터 인덱스를 2씩 증가시키면서 인덱스 6까지 가져옴
[0, 20, 40, 60]
```

• 시퀀스객체[시작인덱스:증가폭]

```
>>> a[7::2]      # 인덱스 7부터 2씩 증가시키면서 리스트의 마지막 요소까지 가져옴
[70, 90]
```

11.4.5 len 응용하기

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[0:len(a)]   # 시작 인덱스에 0, 끝 인덱스에 len(a) 지정하여 리스트 전체를 가져옴
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[:len(a)]    # 시작 인덱스 생략, 끝 인덱스에 len(a) 지정하여 리스트 전체를 가져옴
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

11.4.6 range, 튜플, 문자열에 슬라이스 사용하기

- 튜플은 리스트와 동일
- range는 리스트와 튜플과 다르게 생성 범위만 표시된다

```
>>> r = range(10)
>>> r
range(0, 10)
>>> r[4:7]        # 인덱스 4부터 6까지 숫자 3개를 생성하는 range 객체를 만들
range(4, 7)
>>> r[4:]         # 인덱스 4부터 9까지 숫자 6개를 생성하는 range 객체를 만들
range(4, 10)
>>> r[:7:2]       # 인덱스 0부터 2씩 증가시키면서 인덱스 6까지 숫자 4개를 생성하는 range 객체를 만들
range(0, 7, 2)
```

- 문자열[시작인덱스:끝인덱스]
- 문자열[시작인덱스:끝인덱스:인덱스증가폭]

11.4.7 슬라이스에 요소 할당하기

• 시퀀스객체[시작인덱스:끝인덱스] = 시퀀스객체

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:5] = ['a', 'b', 'c']    # 인덱스 2부터 4까지 값 할당
>>> a
[0, 10, 'a', 'b', 'c', 50, 60, 70, 80, 90]
```

- 요소 개수를 맞추지 않아도 가능
- 요소의 개수가 적을때

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:5] = ['a']              # 인덱스 2부터 4까지에 값 1개를 할당하여 요소의 개수가 줄어듦
>>> a
[0, 10, 'a', 50, 60, 70, 80, 90]
```

- 요소의 개수가 많을때

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:5] = ['a', 'b', 'c', 'd', 'e'] # 인덱스 2부터 4까지 값 5개를 할당하여 요소의 개수가 늘어남
>>> a
[0, 10, 'a', 'b', 'c', 'd', 'e', 50, 60, 70, 80, 90]
```

- 인덱스 증가폭을 지정해서 인덱스를 건너뛰면서 할당(이때는 슬라이스 범위와 요소 개수가 일치해야 한다.

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:8:2] = ['a', 'b', 'c']      # 인덱스 2부터 2씩 증가시키면서 인덱스 7까지 값 할당
>>> a
[0, 10, 'a', 30, 'b', 50, 'c', 70, 80, 90]
```

- 튜플, range, 문자열은 슬라이스 범위를 지정하더라도 요소를 할당할 수 없다

11.4.8 del로 슬라이스 삭제하기

- del 시퀀스객체[시작인덱스:끝인덱스]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> del a[2:5]      # 인덱스 2부터 4까지 요소를 삭제
>>> a
[0, 10, 50, 60, 70, 80, 90]
```

- 튜플, range, 문자열은 del로 슬라이스 삭제 불가능