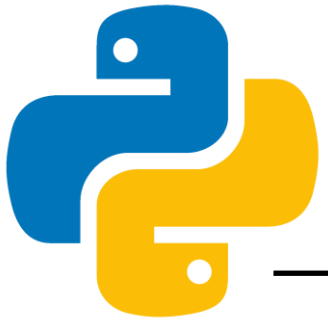


창직 IoT 종합설계입문

파이썬 (4)

컴파운드 자료형

Compound 자료형

- 다른 값들을 덩어리로 묶는데 사용되는 여러가지 컴파운드 자료형이 존재합니다.

- 리스트 (List)
- 튜플 (Tuple)
- 집합 (Set)
- 사전 (Dictionary)

리스트 (1)

리스트 생성

- 리스트는 대괄호 사이에 쉼표로 구분된 값 (항목)들의 목록으로 표현될 수 있습니다.
- 리스트는 서로 다른 형의 항목 등을 포함할 수 있습니다.

```
# 빈 리스트의 생성
empty_list = []
print(empty_list)

# 항목을 갖고 있는 리스트 생성
squares = [1, 4, 9, 16, 25]
print(squares)
```

```
[]
[1, 4, 9, 16, 25]
```

리스트 (2)

인덱싱과 슬라이싱

- 문자열 (그리고, 다른 모든 내장 시퀀스 형들)처럼 리스트는 인덱싱하고 슬라이싱할 수 있습니다.

```
# 항목을 갖고 있는 리스트 생성
squares = [1, 4, 9, 16, 25]
print(squares, '\n')

print(squares[0])
print(squares[-1])
print(squares[-3:])
```

```
[1, 4, 9, 16, 25]

1
25
[9, 16, 25]
```

리스트 (3)

Mutable

- 리스트는 문자열과 다르게 내용을 변경할 수 있습니다.

```
# 항목을 갖고 있는 리스트 생성
squares = [1, 4, 9, 16, 25]
print(squares, '\n')

squares[4] = 49
print(squares)
```

```
[1, 4, 9, 16, 25]

[1, 4, 9, 16, 49]
```

리스트 연산

리스트 연산

- 리스트는 문자열처럼 덧셈과 곱셈을 지원합니다.

```
# 항목을 갖고 있는 리스트 생성
squares = [1, 4, 9, 16, 25]
another_squares = [36, 49]
print(squares)
print(another_squares, '\n')

new_squares = squares + another_squares
print(new_squares)
print(another_squares * 2)
```

```
[1, 4, 9, 16, 25]
[36, 49]

[1, 4, 9, 16, 25, 36, 49]
[36, 49, 36, 49]
```

리스트 메서드

리스트 메서드

- 리스트 자료형은 몇 가지 메서드들을 가지고 있습니다.

- List.append()
- List.extend()
- List.insert()
- List.remove()
- List.pop()
- List.clear()
- List.sort()
- List.reverse()
- List.copy()

List.append()

List.append()

- **리스트의 끝에 항목을 더합니다.**

```
# 빈 집합 생성
squares = []

# 항목 추가
squares.append(1)
squares.append(4)
squares.append(9)
print(squares)
```

```
[1, 4, 9]
```

List.extend()

List.extend(*iterable*)

리스트의 끝에 이터러블의 모든 항목을 붙여 확장합니다.

```
# 빈 집합 생성
squares = []

# 항목 추가
squares.extend([1, 4, 9])
print(squares)
```

```
[1, 4, 9]
```

List.insert()

List.insert(*i*, *x*)

- 주어진 위치에 항목을 삽입합니다. 첫 번째 인자는 삽입되는 요소가 갖게 될 인덱스입니다.

```
# 빈 집합 생성
squares = []

# 항목 추가
squares.insert(0, 4)
squares.insert(0, 1)
print(squares)
```

```
[1, 4]
```

List.remove()

List.remove(*x*)

- 리스트에서 값이 x와 같은 첫 번째 항목을 삭제합니다. 그런 항목이 없으면 ValueError를 일으킵니다.

```
squares = [1, 4, 9, 16]

squares.remove(1)
print(squares)
```

```
[4, 9, 16]
```

List.pop()

List.pop([*i*])

- 주어진 위치에 있는 항목을 삭제하고, 그 항목을 돌려줍니다. 인덱스를 지정하지 않으면, 리스트의 마지막 항목을 삭제하고 돌려줍니다.

```
squares = [1, 4, 9, 16]

squares.pop(2)
print(squares)
```

```
[1, 4, 16]
```

List.clear()

List.clear()

- 리스트의 모든 항목을 삭제합니다.

```
squares = [1, 4, 9, 16]

squares.clear()
print(squares)
```

```
[]
```

List.sort()

List.sort*(key=None, reverse=False)*

- 리스트의 항목들을 제자리에서 정렬합니다.

```
numbers = [1, 3, 5, 6, 4, 2]

numbers.sort()
print(numbers)
```

```
[1, 2, 3, 4, 5, 6]
```

List.reverse()

List.reverse()

- 리스트의 요소들을 제자리에서 뒤집습니다.

```
numbers = [1, 3, 5, 6, 4, 2]

numbers.reverse()
print(numbers)
```

```
[2, 4, 6, 5, 3, 1]
```

List.copy()

List.copy()

- 리스트의 얕은 사본을 돌려줍니다.

```
numbers = [1, 3, 5, 6, 4, 2]

copied_numbers = numbers.copy()
copied_numbers[0] = 8
print(numbers)

copied_numbers = numbers
copied_numbers[0] = 8
print(numbers)
```

```
[1, 3, 5, 6, 4, 2]
[8, 3, 5, 6, 4, 2]
```

튜플 (1)

튜플

- 튜플은 (소괄호 사이에) 쉼표로 구분된 값 (항목)들의 목록으로 표현될 수 있습니다.
- 튜플은 서로 다른 형의 항목 등을 포함할 수 있습니다.

```
# 빈 튜플 생성
empty_tuple = ()
print(empty_tuple)

# 하나의 항목을 갖는 튜플 생성
single_tuple = 'Hongik',
print(single_tuple)

# 여러 항목을 갖는 튜플 생성
multi_tuple = 'Hongik', 'University', 20
print(multi_tuple)
```

```
()
('Hongik',)
('Hongik', 'University', 20)
```

튜플 (2)

인덱싱과 슬라이싱

- 문자열 (그리고, 다른 모든 내장 시퀀스 형들)처럼 튜플은 인덱싱하고 슬라이싱 할 수 있습니다.

```
squares = (1, 4, 9, 16, 25)

print(squares[0])
print(squares[-3])
print(squares[:3])
```

```
1
9
(1, 4, 9)
```

튜플 (3)

Immutable

- 튜플은 리스트와 다르게 생성한 이후 내용을 변경할 수 없습니다.

```
squares = (1, 4, 9, 16, 25)

squares[0] = 36
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-63-33411ad87b63> in <module>()
      1 squares = (1, 4, 9, 16, 25)
      2
----> 3 squares[0] = 36

TypeError: 'tuple' object does not support item assignment

SEARCH STACK OVERFLOW
```

튜플 메서드

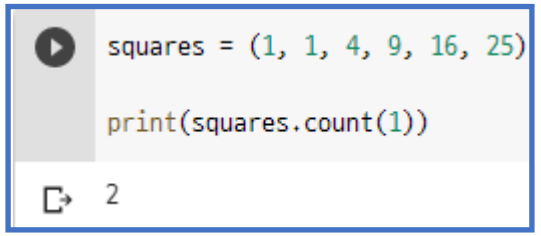
튜플 메서드

- tuple.count()
- tuple.index()

tuple.count()

Tuple.count(*x*)

- Tuple에 등장하는 x의 총 수를 반환합니다.



집합 (2)

Tuple.index(*x*, [, *i* [, *j*]])

- (인덱스 i 또는 그 이후에, 인덱스 j 전에 등장하는) Tuple의 첫 번째 x의 인덱스를 반환합니다.

```
squares = (1, 1, 4, 9, 16, 25)

print(squares.index(9))
```

```
3
```

집합 (1)

집합

- 집합은 중괄호 사이에 쉼표로 구분된 값 (항목)들의 목록으로 표현될 수 있습니다.
- 집합은 중복되는 요소가 없는 순서 없는 컬렉션입니다.

```
# 빈 집합 생성
empty_set = set()
print(empty_set)

# 항목을 갖고 있는 집합 생성
fruits = {'orange', 'banana', 'pear'}
print(fruits)
numbers = {1, 4, 2, 3, 2}
print(numbers)
```

```
set()
{'pear', 'banana', 'orange'}
{1, 2, 3, 4}
```

집합 연산

집합의 연산

- 교집합, 합집합, 차집합 등 집합의 연산을 지원합니다.

```
squares ={1, 4, 9, 16}
integers = {1, 2, 3, 4}

# 교집합
print(squares & integers)
# 합집합
print(squares | integers)
# 차집합
print(squares - integers)
# XOR 집합
print(squares ^ integers)
```

```
{1, 4}
{1, 2, 3, 4, 9, 16}
{16, 9}
{2, 3, 9, 16}
```

집합 메서드

집합 메서드

- Set.add()
- Set.remove()
- Set.discard()
- Set.pop()
- Set.clear()

Set.add()

Set.add(*elem*)

- 원소 elem을 집합에 추가합니다.

```
squares ={1, 4, 9, 16}

squares.add(25)
print(squares)
```

```
{1, 4, 9, 16, 25}
```

Set.remove()

Set.remove(*elem*)

- 원소 elem을 집합에서 제거합니다.
  Elem가 집합에 포함되어 있지 않으면
  KeyError를 일으킵니다.

```
squares ={1, 4, 9, 16}

squares.remove(9)
print(squares)
```

```
{16, 1, 4}
```

Set.discard()

Set.discard(*elem*)

- 원소 elem이 집합에 포함되어 있으면 제거합니다.

```
squares ={1, 4, 9, 16}

squares.discard(4)
print(squares)
```

```
{16, 1, 9}
```

Set.clear()

Set.clear()

- 집합의 모든 원소를 제거합니다.

```
squares ={1, 4, 9, 16}

squares.clear()
print(squares)
```

```
set()
```

사전 (1)

사전의 생성

- 집합은 중괄호 사이에 쉼표로 구분된 Key:Value 쌍들의 목록으로 표현될 수 있습니다.

```
# 빈 사전 생성
nation_capital = {}
print(nation_capital)

# 항목을 갖는 사전 생성
nation_capital = {"한국" : "서울", "일본" : "도쿄"}
print(nation_capital)
```

```
{}
{'한국': '서울', '일본': '도쿄'}
```

사전 (2)

사전 요소 추가 및 제거

- 다음과 같이 사전에 요소를 추가하거나 제거할 수 있습니다.

```
nation_capital = {"한국" : "서울", "일본" : "도쿄"}
print(nation_capital)

# 요소 추가
nation_capital["미국"] ="워싱턴 DC"
print(nation_capital)

# 요소 제거
del nation_capital["일본"]
print(nation_capital)
```

```
{'한국': '서울', '일본': '도쿄'}
{'한국': '서울', '일본': '도쿄', '미국': '워싱턴 DC'}
{'한국': '서울', '미국': '워싱턴 DC'}
```