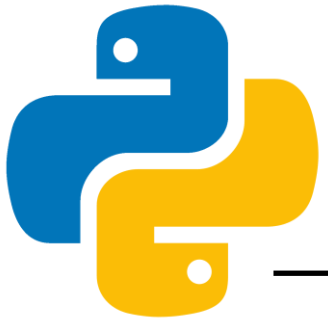

창직 IoT 종합설계입문

파이썬 (3)

문자열

문자열 (String)

- 문자열이란 문자, 단어 등으로 구성된 문자들의 집합입니다.

```
strName1 = "Python"
strName2 = 'python'
strName3 = " I'm Python."
strName4 = '"Python"'

print(strName1)
print(strName2)
print(strName3)
print(strName4)
```

```
Python
python
 I'm Python.
"Python"
```

문자열의 연산

문자열 연산

- 파이썬은 문자열에 대해 덧셈과 곱셈을 할 수 있습니다.

```
strName1 = "I'm "
strName2 = 'python.'

print(strName1)
print(strName2)
print(strName1 + strName2) # 문자열 덧셈
print(strName2 * 2)        # 문자열 곱셈
```

```
I'm
python.
I'm python.
python.python.
```

문자열 인덱싱

인덱싱(Indexing)

- **문자열의 특정 인덱스에 있는 문자를 출력합니다.**

```
strName1 = "Python"

# 문자열의 길이를 반환하는 내장 함수
print(len(strName1))

print(strName1[0]) # 문자열의 0번째 인덱스
print(strName1[2]) # 문자열의 4번째 인덱스
print(strName1[-3]) # 문자열의 4(-3)번째 인덱스
```

```
6
P
t
h
```

0	1	2	3	4	5
P	y	t	h	o	n

문자열 슬라이싱

슬라이싱(Slicing)

- **문자열 내 특정 문자열을 출력합니다.**

```
strName1 = "Whatever you do, make it pay."

# 문자열의 길이를 반환하는 내장 함수
print(len(strName1))

# make를 출력하는 방법.
print(strName1[17] + strName1[18] \
      + strName1[19] + strName1[20])

print(strName1[17:21]) # 17 이상 21 미만.

# 문자열 전체를 출력하는 방법
print(strName1)
print(strName1[:])

# 문자열 부분을 출력하는 방법
print(strName1[:8]) # 시작부터 8 미만까지
print(strName1[9:]) # 9 이상부터 끝까지
```

```
29
make
make
Whatever you do, make it pay.
Whatever you do, make it pay.
Whatever
you do, make it pay.
```

문자열 내장함수

문자열 변경

- 인덱싱을 이용하여 변경 시 오류가 발생합니다.

```
strName1 = "Whatever you do, make it pay."

# 문자열의 길이를 반환하는 내장 함수
print(len(strName1))

strName1[0] = "w"
```

```
29
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-57-7751b09b1f12> in <module>()
      4 print(len(strName1))
      5
----> 6 strName1[0] = "w"

TypeError: 'str' object does not support item assignment

SEARCH STACK OVERFLOW
```

문자열 변경 (2)

슬라이싱과 덧셈 이용

- 다음과 같이 슬라이싱과 문자열 덧셈을 통해 새 문자열을 구성할 수 있습니다.

```
strName1 = "python"
print(strName1)

# 소문자 'p'를 대문자 "P"로 변경하려면?
print("P" + strName1[1:])
```

```
python
Python
```

문자열 메서드 (1)

문자열 메서드

- 파이썬의 덱스트 데이터는 str 또는 문자열 (strings), 객체를 사용하여 처리됩니다.

문자열 메서드 (2)

str.capitalize()

- 첫 문자가 대문자이고 나머지가 소문자인 문자열의 복사본을 반환합니다.

```
strName1 = "python"
print(strName1)

# 소문자 'p'를 대문자 "P"로 변경하려면?
print(strName1.capitalize()) # strName1의 복사본 반환.
print(strName1) # 따라서 strName1의 값은 변하지 않습니다.
```

```
python
Python
python
```

문자열 메서드 (3)

str.count(sub, start[, end])

- 범위 [start, end]에서 부분 문자열 sub가 중첩되지 않고 등장하는 횟수를 반환합니다.
- 선택적 인자 start와 end는 슬라이스 표기법으로 해석됩니다.

```
strName1 = "python"
print(strName1)

# 문자열 안에 소문자 'y'가 몇 개
# 존재하는지 확인하려면?
print(strName1.count("y")) # 문자열 전체를 검색
print(strName1.count("y", 2, 4)) # 2 이상 4 미만의 문자열 내에서 검색
```

```
python
1
0
```

문자열 메서드 (4)

str.find(sub[, start[, end]])

- 부분 문자열 sub가 슬라이스 내에 등장하는 가장 작은 문자열의 인덱스를 돌려줍니다.
- 선택적 인자 start와 end는 슬라이스 표기법으로 해석됩니다.

```
strName1 = "python"
print(strName1)

# 문자열 안에 문자열 ' py'가
# 존재하는지 확인하려면?
print(strName1.find("on"))  # 문자열 'on'의 앞 글자
                            # 'o'의 인덱스를 반환.
print(strName1.find("pys")) # 존재하지 않는다면 -1을 반환.
```

```
python
4
-1
```

문자열 메서드 (5)

str.isalnum(), str.isalpha()

- str.isalnum()는 문자열 내의 모든 문자가 알파벳과 숫자이고, 적어도 하나의 문자가 존재하는 경우 True를 반환하며 그렇지 않다면 False를 반환한다.
- str.isalpha()는 문자열 내의 모든 문자가 알파벳이고, 적어도 하나의 문자가 존재하는 경우, True를 반환하며 그렇지 않다면 False를 반환한다.

```
strName1 = "python1"
print(strName1)

print(strName1.isalnum())
print(strName1.isalpha())
```

```
python1
True
False
```

문자열 메서드 (6)

str.startswith(), str.endswith()

- 문자열이 지정된 문자로 시작하거나 끝나면 True를 반환하면 그렇지 않으면 False를 반환합니다.

```
strName1 = "python1"
print(strName1)

# 문자열이 지정된 문자로 시작하거나
# 끝나는지 확인하려면?
print(strName1.startswith("p"))
print(strName1.endswith("1"))
```

```
python1
True
True
```

문자열 메서드 (7)

str.upper(), str.lower()

- 모든 케이스 문자가 대문자 또는 소문자로 변환된 문자열의 복사본을 반환합니다.

```
strName1 = "PYTHON"
strName2 = "python"
print(strName1)
print(strName2)

print(strName2.upper())
print(strName1.lower())
```

```
PYTHON
python
PYTHON
python
```

문자열 메서드 (8)

str.replace(old, new[, count])

- 모든 부분 문자열 old가 new로 치환된 문자열의 복사본을 돌려줍니다.

```
strName1 = "PYTHON"
print(strName1)

print(strName1.replace("YTHON", "ython"))
print(strName1)
```

```
PYTHON
Python
PYTHON
```

문자열 메서드 (9)

str.split(sep=None, maxsplit=-1)

- Sep를 구분자 문자열로 사용하여 문자열에 있는 단어들의 리스트를 돌려줍니다.
- Maxsplit이 주어지면 최대 maxsplit 번의 분할이 수행됩니다.

```
strName1 = "I'M PYTHON"
print(strName1)

print(strName1.split(" "))
```

```
I'M PYTHON
["I'M", 'PYTHON']
```

문자열 메서드 (10)

str.strip([chars])

- 선행과 후행 문자가 제거된 문자열의 복사본을 돌려줍니다.
- 모든 조합 값이 제거됩니다.

```
strName1 = "       PYTHON"
strName2 = "www.naver.com"

print(strName1.strip())
print(strName2.strip("cowm."))
```

```
PYTHON
naver
```