# 문자열을 다루는 다양한 방법들

## 들어가기

- 계속해서 문자열 타입의 내장 메서드들에 대해서 알아보자.

## 학습 목표

- 파이썬 문자열 타입의 내장 메서드를 이해하고 사용할 수 있다.

## 핵심 키워드

- 문자열
- 내장 메서드

## 학습 내용

### 문자열 슬라이싱

- 파이썬에서는 특정 범위에 있는 문자를 가져올 수 있다.

```
myString = 'Monty Python'
print(myString[0:4])
# Mont가 출력됩니다. 여기서 0 to 4에서 4에 대한 인덱스는 출력되는 값에 포함되지 않는 것을 확인하여야 합니다.
print(myString[6:7])
# P가 출력됩니다.
print(myString[6:20])
# Python이 출력됩니다.
print(myString[:2])
# index값이 2에 해당하는 문자 앞부터 출력됩니다.
print(myString[8:])
# index값이 8에 해당하는 문자부터 출력됩니다.
print(myString[:])
# 전체가 출력됩니다.
```

### 문자열 합치기

- 문자열 연결은 수리 연산자인 `+` 를 사용해서 가능하게 만들 수 있다.

```
firstString = 'Hello'
secondString = 'There'
print(firstString + secondString)
# HelloThere로 출력됩니다.
thirdString = firstString + ' ' + secondString
print(thirdString)
# Hello There로 출력됩니다.
```

## `In` 을 논리 연산자로 사용하기

- 특정 문자열에 우리가 확인하고자 하는 문자가 있는지 확인하기 위해서 `in` 을 사용한다.

```
fruit = 'banana'
print('n' in fruit)
# True로 출력됨
print('m' in fruit)
# False로 출력됨
print('nan' in fruit)
# True로 출력됨
if 'a' in fruit :
    print('Found it!')
# Found it으로 출력됨
```

## 문자열 라이브러리

- 문자열 타입의 객체에서 우리는 다양한 메서드를 사용할 수 있다.

```
greet = 'Hello Bob'
zap = greet.lower()
print(zap)
# hello bob으로 출력됨
print(greet)
# Hello Bob으로 출력됨
print('Hi There'.lower())
# hi there로 출력됨
print(greet.upper())
# HELLO BOB으로 출력됩니다.
```

## Strip 메서드

- 문자열에서 공백을 제거하는 메서드
- `lstrip()` : 왼쪽 공백 제거

- `rstrip()` : 오른쪽 공백 제거
- `strip()` : 양쪽 공백 제거

```
greet = '                    Hello Bob       '
greet.lstrip()
# 왼쪽의 공백이 삭제됨
greet.rstrip()
# 오른쪽의 공백이 삭제됨
greet.strip()
# 양쪽의 공백이 삭제됨
```

## 시작 문자열 찾기

- startswith() 메서드를 통해서 특정 문자로 문자열이 시작되는지 확인할 수 있다.
- 결과는 불리언 타입으로 반환된다.
  - Yes → True
  - No → False

```
line = 'Please have a nice day'
print(line.startswith('Please'))
# True가 출력됨
print(line.startswith('p'))
# False가 출력됨 : 대소문자 구분
```