전처리 과정 : 불필요한 데이터 삭제 및 추출

-> 문자열 찾기, 문자열 조작, 문자열 치환 등 활용

**1. 문자열 연산**

**문자열의 연결**

message = ‘안녕하세요’

guest = ‘홍길동’

greeting = guest + ‘님, ‘ + message

print(greeting) => 홍길동님, 안녕하세요

**문자열의 반복**

Print(‘=’ \* 40)

Print(‘-‘ \* 13, end=’’)

**문자열의 범위 선택**

Data\_str[시작연산자:종료연산자] -> 종료인덱스의 -1까지만 결과값이 나옴

**2. 문자열 함수**

**문자열 출현 횟수 확인**

data\_str.count(input\_str) -> data\_str에 input\_str 문자가 얼마나 나왔는지 반환

(input\_str은 문자열도 가능하다)

**문자열의 길이**

Len(data\_str) -> 문자열의 길이 반환

**문자열 찾기**

data\_str.find(input\_str, 시작위치) -> 문자열을 찾을 경우 시작 인덱스 반환, 못 찾을 경우 -1 반환

data\_str.rfind(input\_str) -> 문자열을 찾을 경우 시작 인덱스 반환, 못 찾을 경우 -1 반환

cf) find는 왼쪽에서 오른쪽으로 찾고 rfind는 오른쪽에서 왼쪽으로 찾음

data\_str.index(input\_str) -> 문자열을 못 찾을 경우 Value Error 발생

**문자열의 삽입**

Join 함수 : 문자열 삽입 함수

data\_str = ‘가나다라마바사아자차카타파하’

comma\_space = ‘, ‘

output = comma\_space.join(data\_str) -> comma\_space에 저장된 값을 구분 문자열로 data\_str의 각 항목 사이에 삽입

=> 가, 나, 다, 라, 마, 바, 사, 아, 자, 차, 카, 타, 파, 하

**대소문자 바꾸기**

capitalize, lower, upper : 대소문자 변경 관련 함수

data\_str = ‘better tomorrow’

data\_str = data\_str.capitalize() -> 첫 번째 문자를 대문자로 한 새로운 문자열 반환

data\_str = data\_str.lower() -> 모든 문자를 소문자로 한 새로운 문자열 반환

data\_str = data\_str.upper() -> 모든 문자를 대문자로 한 새로운 문자열 반환

**공백 제거**

Lstrip, rstrip, strip : 왼쪽, 오른쪽, 양쪽 공백 제거 함수

data\_str = data\_str.lstrip(‘ ‘) -> 인자로 전달된 문자열을 왼쪽에서 제거

data\_str = data\_str.rstrip(‘\_ ‘) -> 인자로 전달된 문자열을 오른쪽에서 제거

data\_str = data\_str.strip(‘ 0?\_#’) -> 인자로 전달된 문자열을 문자열의 왼쪽과 오른쪽에서 제거

**문자열 교체**

replace 함수 : 찾을 문자열과 교체 문자열을 인자로 사용해 교체

data\_str = data\_str.replace(‘.’ \* 4, ‘\t’) -> ….를 tab만큼 떨어지게 만듦

**문자열 자르기**

data\_str = data\_str.replace(‘ ‘, ‘’) -> 문자열 사이에 있는 공백 문자열을 없애기 위해 공백문자열 ‘ ‘을 널문자열 ‘’로 치환

data\_list = data\_str.split(‘,’) -> split 함수 : 전달된 인자로 문자열을 잘라 이를 항목으로 갖는 리스트 생성 / data\_str 문자열 객체의 split 함수에 문자열 콤마(,)를 전달해 리스트 객체 data\_list 생성

**문자열 구성 확인**

Isdigit 함수 : 숫자 문자열인 경우 True를 반환하는 함수

문자열의 구성을 확인하는 기능을 이용해 다양한 유효성 검사를 하는 기능을 만들 수 있음