Python 기초 – 자료 구조(나머지)

Made by Jay Hong

Check

1. 자료 구조?

1. 자료 구조?

1. 리스트
2. 튜플
3. 딕셔너리
4. 집합

1) 리스트 뜻

- ✓ 리스트 : 값을 “순서대로“ 저장하는 자료 구조
- ✓ 리스트를 사용해 행렬이나 벡터를 표현할 수 있다.

2) 리스트 생성하기

- ✓ 리스트는 대괄호([])로 감싸서 만든다.
- ✓ 대괄호 속에 들어가는 “원소”는 순서의 의미를 갖는다.
  - ✓ List 속 값을 “원소”라고 부른다.
  - ✓ 순서를 이용한 다양한 기법들이 있다.
- ✓ 리스트 속에는 다양한 형태의 원소가 들어갈 수 있다.
  - ✓ 숫자형과 문자열이 동시에 존재할 수 있다.
- ✓ list() 를 사용해 형태를 변환할 수 있다.

3) 인덱싱

※ len() 연산자 : 원소의 개수를 확인할 수 있다.

- ✓ 인덱싱 : 한 개의 원소를 선택할 수 있다.
  - ✓ 형태 : x[n]

  - ✓ n에 0부터 len(x)-1 사이의 값을 입력하면, 해당 위치를 반환한다.
    - ✓ Python은 0부터 시작하기 때문에, 길이를 그대로 입력하면 에러 발생
    - ✓ x[1] : 2번째 위치에 있는 원소 (Python은 0부터 시작)

  - ✓ n에 -1부터 –len(x)까지 입력하면, 뒤에서부터 해당 위치의 값을 반환한다.
    - ✓ x[-1] : 가장 마지막 원소
    - ✓ x[-2] : 뒤에서 두번째 원소
    - ✓ x[-len(x)] : 가장 첫 원소

4) 슬라이싱

- ✓ 슬라이싱 : 여러 개의 원소를 선택할 수 있다.
  - ✓ 형태 : x[a : b:c]
    - ✓ a : 시작 위치
    - ✓ b : 끝 위치 (미포함) : b위치 앞까지의 원소를 반환한다.
    - ✓ c : 간격
  - ✓ 가장 처음 혹은 가장 마지막을 표현하지 않아도 된다
    - ✓ x[:b] = 처음부터 b-1번째 까지
    - ✓ x[a:] = a부터 끝까지
  - ✓ 마찬가지로 음수를 사용할 수 있다
    - ✓ x[:-1] = 처음부터 가장 마지막에서 두번째 까지
    - ✓ x[::-1] = 생각해보기

5) 리스트 관련 연산자

- ✓ .append() : () 안에 들어가는 값을 리스트에 추가한다.
- ✓ .pop() 해당 위치의 값을 하나를 제거하고, 그 값을 반환한다
- ✓ del x[n] : x의 n번째에 위치한 값을 제거한다.
- ✓ x.index() : () 안에 들어가는 값의 위치를 찾는다. (중복되면 가장 처음 것을 반환)
- ✓ x[n] = k : x의 n번째에 위치한 원소를 k로 바꾼다.
- ✓ a + b : 두 리스트를 합치면 두 리스트 속 원소들이 하나의 리스트에 담긴다.
- ✓ a * 2 : a 리스트 속 원소들이 2번 생긴다.
- ✓ a.sort() : a 리스트 속 원소들을 정렬한다.
- ✓ a.reverse() : a 리스트 속 원소들을 역순으로 배열한다.

1) 튜플의 뜻

- ✓ 튜플 (tuple)은 리스트와 동일하게 값을 순서대로 저장하는 자료 구조이다.

- ✓ 단, 튜플은 수정하거나 변경할 수 있다.
  - ✓ 추가, 삭제, 수정 등에 대비하지 않기에 효율적인 내부 구조를 가진다.

2) 튜플 생성하기

- ✓ 튜플 (tuple)은 소괄호()로 감싸서 만든다.
- ✓ tuple()을 사용해 바꿀 수 있다.

3) 인덱싱

- ✓ 리스트와 동일하다.

4) 슬라이싱

- ✓ 리스트와 동일하다.

5) 합치기 & 반복하기

- ✓ a + b

6) 수정하기

- ✓ 기본적으로 tuple은 수정이 불가능하다.
- ✓ 수정하기 위해서는 list()로 바꾸어 수정 후 다시 tuple() 형태로 바꾸면 된다.

1) 딕셔너리의 뜻

- ✓ 사전입니다.
- ✓ 사전은 “단어의 이름“ : “단어의 의미“ 의 형태로 구성되어 있습니다.
- ✓ 딕셔너리도 마찬가지로 key : value의 형태로 구성됩니다.
- ✓ 즉, 딕셔너리는 key에 해당하는 정보인 value를 매칭시켜 저장한 책입니다.
- ✓ 따라서, 딕셔너리는 순서의 의미가 존재하지 않습니다.

2) 딕셔너리 만들기

- ✓ 딕셔너리는 중괄호로 key : value를 감싸 만든다.
- ✓ {key1 : value1, key2 : value2, …}

3) 인덱싱

- ✓ 딕셔너리는 순서의 의미가 없음을 말씀 드렸습니다.
- ✓ 하지만 딕셔너리는 key라는 고유 값들이 딕셔너리에 존재합니다.
- ✓ 따라서 key를 이용해 인덱싱합니다.
- ✓ 이를 이용해서 값을 덮어쓸 수도 있습니다.
- ✓ example) 심부름에는 지금까지 구매했던 상품과 그 상품의 가격이 담겨있습니다.
  - ✓ 심부름['두부']
    심부름이라는 딕셔너리에서 두부에 해당하는 value를 가져옵니다.
  - ✓ 심부름['두부'] = 2000
    두부의 가격을 2000원으로 변경했습니다.

4) 원소 추가하기

- ✓ 인덱싱과 동일한 형태를 지닙니다.
- ✓ 해당 딕셔너리에 없는 key와 값을 저장해주면 됩니다.
- ✓ x[a] = b
  - ✓ x라는 딕셔너리에 a라는 key와 그에 해당하는 value b가 저장됩니다.
- ✓ example)
  - ✓ 심부름['파워에이드'] = 2000
    심부름이라는 딕셔너리에 "파워에이드 " 라는 key와 2000원 이라는 값이 저장 됐습니다.

5) 유의 사항

- ✓ 딕셔너리 생성시, 중복된 키가 있다면, 가장 먼저 나온 key를 사용합니다.

- ✓ 사전의 키에는 숫자, 문자, 튜플 등 “불변”하는 값들만 들어갈 수 있습니다.
  - ✓ 리스트와 같이 가변적인 값은 사용할 수 없습니다.

    단순히 생각해도, 이름이 매번 바뀌는 값이 사전의 단어로 선정되기 어렵습니다.

6) 관련 Method

- ✓ .keys() : 사전의 키를 모두 불러옵니다.
- ✓ .values() : 사전의 값들을 모두 불러옵니다.
- ✓ .items() : 사전의 키와 값들을 짝지어서 모두 불러옵니다.
- ✓ .get(key) : key에 해당하는 값을 불러옵니다.
  - ✓ 없는 값이더라도 오류를 내뱉지 않습니다.
- ✓ in : 사전에 해당 key가 있는지 봅니다.
  - ✓ 사과 in 심부름
- ✓ del 사전[key] : 해당 key를 삭제합니다.
- ✓ .clear() : 사전을 비웁니다.

1) 집합의 뜻

- ✓ 수학의 집합과 같은 용도로 사용
- ✓ 반복되는 값을 허용하지 않는다.
- ✓ 순서의 의미가 없다.

2) 집합 생성하기

- ✓ {}로 감싸 만든다.
- ✓ 딕셔너리와는 다르게 key와 value의 매칭이 없다.
- ✓ set()를 사용해 만들 수 있다.

3) 원소 추가와 삭제

※ len() 연산자 : 원소의 개수를 확인할 수 있다.

- ✓ .add() : 안에 들어가는 원소를 추가한다.
- ✓ .update() : 여러 개의 원소를 추가한다.
- ✓ .remove() : 원소를 제거한다.
- ✓ in : 안에 원소가 있는지 확인한다.

4) 집합 연산자

- ✓ | : 합집합
- ✓ & : 교집합
- ✓ - : 차집합