03. 데이터 구조 #src: 8_Python/ch03_데이터구조 참고

1) 리스트 []

(1) 리스트 생성; []나 list()를 이용해서 생성, 여러 개 값을 하나의 변수에 저장 및 관리 가능

(2) 기본 정보 조회: len(), max(), min() #2차원 리스트에서 max(), min()은 리스트 첫 요소만 비교

(3) **요소(데이터) 추가**

① +: 두 리스트를 연결

② *: 리스트를 곱한 수 만큼 반복

③ **append()**; 요소를 맨 뒤에 추가

④ **extend()**; 리스트를 요소별로 맨 뒤에 추가

⑤ insert(index, data); 지정된 index 위치에 data 삽입

(4) **인덱싱** #처음:0, 맨 마지막:-1

① count(); 리스트에서 데이터의 개수 반환

② index(): 해당 요소의 첫번째 위치 반환 index(a, b); a 인덱스 이후부터 첫 b의 인덱스 반환

③ [index]; 인덱스를 이용한 접근

(5) **원하는 리스트 내용만 추출; 리스트 슬라이싱** #인덱스보다 많이 사용

① [from:to:by]; from부터 to 바로 앞까지 by씩 증가하는 index 아이템 추출

② from, to, by 생략 가능. 음수 인덱스 가능 #처음:0, 맨 마지막:-1

(6) 요소 수정하기; 인덱싱으로 데이터 수정

① 지정한 인덱스 위치의 데이터를 다른 데이터로 바꿈

② 지정한 범위의 리스트 항목을 다른 항목들로 통째로 바꿈(슬라이싱으로 데이터 수정)

③ 바꾸려는 항목이 더 많거나 적어도 일괄적으로 변경 가능 #by사용시 항목 개수 일치必

(7) **삭제하기**

① pop(); 가장 마지막 요소 반환 및 삭제, pop(n); n번째 요소 반환 및 삭제

② remove(data); 해당 데이터가 삭제

③ del xx[n]; 지정한 위치 항목 삭제, 변수 삭제

④ clear(); 모든 항목 삭제 (결과: []) #del 변수[:]와 동일한 결과

(8) **정렬**

① sort(); 기본 오름차순 정렬(reverse=True: 내림차순 정렬) #원본 데이터 변경

② reverse(); 역순으로 나열 #원본 데이터 변경

③ [::-1]; 역순으로 출력 #원본 데이터 변경X

(9) **리스트 복제**

① =; 주소를 복사해 같은 객체를 참조(둘 중에 하나를 변경하면 다른 것도 변경)

② copy(); 복제된 새로운 객체를 생성(주소값이 다름)

2) 튜플 ()

(1) 리스트와 유사하지만 읽기 전용으로 수정이 필요 없는 데이터 타입에서 사용

(2) 튜플에 데이터 추가, 수정, 삭제 불가(제공되는 함수도 많지 않음)

(3) ()이용해서 생성

3) 딕셔너리 {}

(1) 중괄호{}를 이용해서 딕셔너리 생성

(2) 키(key)와 값(value)의 쌍으로 구성된 자료구조(cf. 자바 hashmap과 유사)

(3) 키는 유일한 값(중복허용X), 키에 리스트는 사용 불가(튜플은 사용 가능)

(4) 값은 중복 가능하며 모든 데이터 타입 가능

(5) 없는 키 값을 참조시 에러 발생

(6) 인덱스를 이용한 데이터 참조는 불가능

(7) 할당과 복제

① =; 주소 복사

② copy(); 복제된 새로운 객체 생성(주소값 다름)

4) 셋

(1) 중복을 허용하지 않는 집합(순서X, 인덱스를 통한 접근 불가)

(2) {}나 set()함수를 이용하여 생성

- set(딕셔너리 타입의 변수); 딕셔너리 변수의 키만 셋에 추가

(3) 집합 연산자(&:교집합, |:합집합, -:차집합)

5) enumerate

(1) 반복자(iterator) 또는 순서(sequence) 객체로 반복문 사용시 인덱스 처리 지원

(2) enumerate(객체) 사용: 객체를 (0, 객체[0]), (1, 객체[1]), ... 형식으로 반환