2. 파이썬 자료형
(2) 리스트, 튜플, 딕셔너리

2023-1 세종과학고등학교 1학년 정보

이예린T

3) 리스트

(1) 리스트

- 리스트(list)
  - 여러 요소들을 연속적으로 저장하는 자료형
  - 새로운 원소의 추가, 갱신, 삭제 가능
  - 리스트 생성: 리스트명=[요소, 요소, 요소, ...]

```
>>> a = [ ]        # 빈 리스트 생성, a = list()도 가능
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]   # 리스트 안에 리스트 저장 가능
```

(2) 리스트 요소 접근

- 리스트 인덱싱
  - 인덱스를 통해 리스트 내부의 요소 각각에 접근할 수 있다.
  - 리스트명[인덱스]로 사용
  - 0부터 시작하고, 마지막 요소는 -1로 가리킬 수 있다.
  - 인덱싱을 통해 요소의 값을 바꿀 수 있다.

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
>>> a[0]
???
>>> a[-1]
???
>>> a[3]
???
>>> a[-1][0]
???
>>> a[-1][-1]
???
```

- 리스트 슬라이싱
  - 리스트의 일부 범위를 선택해 접근한다.
  - 리스트명[시작인덱스:끝인덱스] → 리스트의 시작인덱스부터 끝인덱스 **-1** 까지 리스트로 추출
  - 시작인덱스가 비어있으면 처음부터, 끝인덱스가 비어있으면 맨 끝까지
  - 슬라이싱을 사용해도 요소 값을 바꿀 수 있다.

(3) 리스트와 연산자

- 리스트+리스트: 리스트 두 개를 연결하여 새로운 리스트 생성

- 리스트*수: 리스트의 값들을 반복하여 새로운 리스트 생성

- 리스트 길이 구하기
- len() 함수 사용
- len(리스트명)을 사용하면 요소의 개수가 구해진다.

- 값 in 리스트 / 값 not in 리스트
- 특정 값이 리스트에 있으면/없으면 True, 아니면 False를 반환한다.

(4) 리스트와 함수

- append
- 리스트명.append(추가하려는 요소)
- 리스트의 맨 마지막 요소로 값이 추가된다.

- insert
- 리스트명.insert(삽입하려는 위치의 인덱스, 삽입하려는 값)

- reverse
- 리스트명.reverse()
- 리스트 내부 값들의 순서를 뒤집는다.

(4) 리스트와 함수

- sort
- 리스트명.sort()
- 리스트 안의 값들을 오름차순(작은 값부터 큰 값 순서)으로 정렬한다.
- 내림차순으로 정렬하려면 리스트명.sort(reverse=True)

- index
- 리스트명.index(찾으려는 값)
- 값이 위치한 인덱스를 반환한다.

- count
- 리스트명.count(찾으려는 값)
- 리스트 안에 해당 값이 몇 개 있는지 조사하여 그 개수를 돌려줌

(4) 리스트와 함수

- remove
- 리스트명.remove(삭제할 값)
- 리스트에서 첫 번째로 나오는 '값'을 삭제

- del
- del 리스트명[인덱스]
- 리스트에서 해당 인덱스의 요소를 삭제한다.
- 슬라이싱 사용 가능

- pop
- 리스트명.pop()
- 맨 마지막 요소를 돌려주고 리스트에서 그 요소는 삭제
- 리스트명.pop(삭제하고자 하는 인덱스)로 사용하면 특정 인덱스의 값을 돌려주고 삭제

4) 튜플

(1) 튜플(tuple)

- 튜플은 리스트와 유사하지만, ( ) 기호를 사용하고 요소를 수정할 수 없다는 점이 다르다.
- 프로그램 실행 중 값이 변해야 할 때는 리스트, 변하지 않을 때는 튜플을 사용하는 것이 적절
- 튜플명=(요소, 요소, 요소, ...) 로 선언
- 리스트와 같이 인덱스와 슬라이싱을 사용해 내부의 요소에 접근
- 덧셈, 곱셈 연산자 사용 가능

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0]
1
>>> t1[3]
'b'
```

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[1:]
(2, 'a', 'b')
```

```
>>> t1 = (1, 2, 'a', 'b')
>>> t2 = (3, 4)
>>> t1 + t2
(1, 2, 'a', 'b', 3, 4)
```

```
>>> t2 * 3
(3, 4, 3, 4, 3, 4)
```

(1) 튜플(tuple)

- 요소 한 개짜리 튜플을 만들 때는 요소 뒤에 콤마를 꼭 넣어야 한다.
- 괄호()를 생략해도 된다.

```
>>> t1 = (1,)
```

```
>>> t2 = 1,2,3
```

ex)

a,b=b,a

[a,b]=b,a

(a,b)=b,a

5) 딕셔너리

(1) 딕셔너리(Dictionary)

- Key와 value의 쌍 여러 개를 중괄호 { }로 둘러싼다.

```
dic = {Key1:Value1, Key2:Value2, …}
```

```
dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

- Key를 통해 value에 접근하는 형태로 사용
- Value에 리스트 사용 가능

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey'] #Key가 'pey'인 Value
10
>>> grade['julliet'] #Key가 'julliet'인 Value
99
```

(2) 딕셔너리(Dictionary) 요소 추가, 삭제

- 딕셔너리명[key]=value

```
>>> a = {1: 'a'}
>>> a[2] = 'b'    # {2:'b'} 쌍 추가
>>> a             # key가 2이고 value가 'b'인 쌍
{1: 'a', 2: 'b'}
```

- del 딕셔너리명[key]

```
>>> del a[1]    # Key가 1인 Key:value 쌍 삭제
>>> a
{2: 'b'}
```

(3) 딕셔너리(Dictionary) 사용 시 주의할 점

- 딕셔너리 내의 key값 중복 금지, 중복될 경우 하나만 유지하고 나머지는 무시

```
>>> a = {1:'a', 1:'b'}   #1 이라는 Key값이 중복
>>> a
{1: 'b'}    #1:'a' 쌍이 무시됨
```

- Key에 리스트 사용 불가, 튜플 사용 가능 - 변하는 값 사용 불가

```
>>> a = {[1,2] : 'hi'}
Traceback (most recent call last): File "<stdin>",
line 1, in <module>
TypeError: unhashable type: 'list'
```

(4) 딕셔너리 요소 사용하기

- 딕셔너리명.keys() → key만 모아서 dict_keys 객체를 반환

```
>>> a = {'name': 'pey', 'phone': '0119993323',
'birth': '1118'}
>>> a.keys()
dict_keys(['name', 'phone', 'birth'])
```

- 딕셔너리명.values() → value만 모아서 dict_values 객체를 반환

```
>>> a.values()
dict_values(['pey', '0119993323', '1118'])
```

- 딕셔너리명.items() → key,value 쌍을 튜플로 묶어 모아서 dict_items 객체로 반환

```
>>> a.items()
dict_items([('name', 'pey'), ('phone', '0119993323'),
('birth', '1118')])
```

(4) 딕셔너리 요소 사용하기

- 딕셔너리명.get(key값)
  - key값에 대응되는 value 반환
  - 존재하지 않는 key값일 경우 None 반환 (비교: 딕셔너리명[key값]은 없으면 에러 발생)

- Key값 in 딕셔너리명
  - Key가 있는지 True/False로 반환

- 딕셔너리명.clear()
  - 쌍 모두 삭제