

코드로 한 번에 이해되는 파이썬 (4)

실습 코드 : https://github.com/hansikyung/Python_OneTime

Today's point!



- ✓ 리스트란? 리스트의 정의, 추가, 삭제
- ✓ 리스트의 인덱싱과 슬라이싱
- ✓ 리스트가 가지고 있는 함수
- ✓ 딕셔너리란? 딕셔너리의 정의, 추가, 삭제
- ✓ 딕셔너리가 가지고 있는 함수

리스트란?

파이썬 데이터 타입



int

float

string

bool

tuple

set

list

dictionary

리스트

컬렉션 타입의 데이터로, 같거나 다른 자료형의 데이터 여러 개를 한 개로 묶어서 다루는 데이터 형식이다.
데이터에는 순서가 있으며, 여러 개의 데이터를 ,(쉼표)로 구분하고, [](대괄호)로 감싸 표현한다.

정의

```
listA = []  
listB = [1, 2, 3, 4, 5]  
listC = [3, '강아지', 2, 4, '고양이']
```

추가

```
listA.append( 추가하려는 데이터 ) #바로 뒤에 데이터를 추가한다.  
listB.insert( 자리 수, 추가하려는 데이터 ) # '자리' 에 데이터를 추가한다.
```

삭제

```
listA.remove( 삭제하려는 데이터 ) #리스트에서 첫번째로 나타난 '삭제하려는 데이터'를 삭제한다.  
del listA[n] #리스트의 n번째 요소를 삭제한다.
```

리스트

컬렉션 타입의 데이터로, 같거나 다른 자료형의 데이터 여러 개를 한 개로 묶어서 다루는 데이터 형식이다. 데이터에는 순서가 있으며, 여러 개의 데이터를 ,(쉼표)로 구분하고, [](대괄호)로 감싸 표현한다.

리스트 : 인덱싱과 슬라이싱

리스트에 포함된 여러 개의 데이터 중 특정한 데이터에 접근하려면 어떻게 해야 할까?

리스트는 주로 데이터의 위치를 가지고 데이터의 자료를 가져올 수 있다. 이를 '인덱싱' 이라고 한다.

그리고 리스트의 값 중 일부를 잘라 가져오는 것을 '슬라이싱' 이라고 한다.

인덱싱

```
listC = [3, '강아지', 2, 4, '고양이']
```

`listC[n]` #단, n은 0부터 시작한다.

슬라이싱

```
listC = [3, '강아지', 2, 4, '고양이']
```

`listC[1:3]` # list의 이름[처음:끝]

`listC[2:]` # list의 이름[처음:]

`listC[:3]` # list의 이름[:끝]

`listC[::2]` # list의 이름[::몇 개씩 건너뛰겠다!]

`listC[-1]` # -가 붙은 숫자는 리스트의 요소를 뒤쪽부터 센다는 뜻이다.

리스트 : 리스트의 함수

리스트가 가지고 있는 리스트의 함수에는 `len()`, `index()`, `pop()`, `sort()`, `reverse()` 등이 있다.

```
listC = [3, '강아지', 2, 4, '고양이']
```

len

`len(listC)` #리스트의 길이(리스트 안에 있는 데이터의 수)를 알려준다.

index

`listC[3]` #listC안에 3이 있는 데이터 위치를 반환한다.

pop

`listC.pop` #listC의 가장 마지막 데이터를 꺼내어 삭제한다.

sort

`listC.sort()` #listC의 요소를 순서대로 정렬한다.

reverse

`listC.reverse()` #listC의 요소를 거꾸로 정렬한다.

덕셔너리란?

파이썬 데이터 타입



int

float

string

bool

tuple

set

list

dictionary

딕셔너리

컬렉션 타입의 데이터로, **키-값의 한 쌍으로 구성된** 데이터 형식이다.

리스트와 달리, 순서가 큰 의미를 갖지는 않는다. 키와 값은 : (콜론)으로 짝짓고, {}(중괄호)로 감싸 표현한다.

정의

```
dictA = { }  
dictB = {'A' : 1, 'B': 'Baby', 2:3, 4: 'four'}  
dictC = {'flower' : ['진달래', '벚꽃', '개나리'], 3:31}
```

추가

`dictA[추가할 키] = 추가할 값` # []로 키를 지정하고, = 뒤에 값을 적어 키-값 쌍을 지정한다.

삭제

`del dictA[삭제할 요소의 키]` # 정해진 딕셔너리의 키를 기준으로, 키-값 쌍이 삭제된다.

접근

`dictA[키]` # '키'에 맞는 '값'을 출력할 수 있다.

딕셔너리 : 딕셔너리의 함수

딕셔너리의 함수에는 `keys()`, `values()`, `items()`, `get()`, `clear()`, `in()` 등이 있다.

```
dictB = {'A' : 1, 'B': 'Baby', 2:3, 4: 'four'}
```

`keys()`

`dictB.keys()` #딕셔너리의 키를 리스트로 가져온다.

`values()`

`dictB.values()` #딕셔너리의 값을 리스트로 가져온다.

`items()`

`dictB.items()` #딕셔너리의 키-값 쌍을 튜플로 묶어 가져온다.

`get()`

`dictB.get(키)` #딕셔너리의 '키'에 맞는 값을 가져온다.

`in()`

`키 in dictB` #딕셔너리 안에 '키'가 있는지를 확인한다.

`clear()`

`dictB.clear()` #딕셔너리의 모든 요소를 삭제한다.

Summary



- 리스트는 컬렉션 타입의 데이터로, 여러 개의 데이터를 하나로 정의하여 관리하는 자료형이다.
- 리스트를 정의할 때는 다양한 자료형을 섞어 정의하되, 요소는 쉼표(,) 로 구분하고, 대괄호[] 로 감싸 표현한다.
- 어떤 리스트에 요소를 추가할 때는 append, insert 함수를 사용한다.
- 어떤 리스트에 요소를 삭제할 때는 remove, pop, del 을 사용한다.
- 리스트의 요소에 접근할 때는 인덱싱이나 슬라이싱을 사용한다.
 - 인덱싱은 요소의 위치를 가지고 리스트의 값에 접근하는 방식이다.
 - 슬라이싱은 리스트의 일부를 잘라 가져오는 방식이다.
- 딕셔너리는 컬렉션 타입의 데이터로, 키-값의 쌍 여러 개가 모여 정의하는 자료형이다.
- 딕셔너리를 정의할 때는 다양한 자료형을 섞어 정의하되, 요소는 쉼표(,)로 구분하고, 중괄호 { } 로 감싸 표현한다.
- 딕셔너리에 요소를 추가할 때는 딕셔너리[키] = 값 으로 키와 값을 모두 사용하여 추가한다.
- 딕셔너리에는 keys(), values(), items() 와 같이 각각 키, 값, 키와 값의 쌍(튜플) 을 인쇄하는 함수가 있다.