UNIT 12

딕셔너리 사용하기

12 딕셔너리 사용하기

딕셔너리 사용하기

- 파이썬에서는 연관된 값을 묶어서 저장하는 용도로 딕셔너리라는 자료형을 제공함
- 게임 캐릭터의 능력치를 딕셔너리에 저장해보자

```
lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

12 딕셔너리 사용하기

딕셔너리 사용하기

- 딕셔너리만 봐도 lux라는 캐릭터의 체력(health)은 490, 마나(mana)는 334, 사거리(melee)는 550, 방어력(armor)은 18.72라는 것을 쉽게 알 수 있음
- 딕셔너리는 값마다 이름을 붙여서 저장하는 방식임
- 사전(dictionary)에서 단어를 찾듯이 값을 가져올 수 있다고 하여 딕셔너리라고 부름

12.1 딕셔너리 만들기

딕셔너리 만들기

- 딕셔너리는 { }(중괄호) 안에 키: 값 형식으로 저장하며 각 키와 값은 ,(콤마)로 구분해줌

```
딕셔너리 = {키1: 값1, 키2: 값2}
```

- 키와 값이 4개씩 들어있는 딕셔너리를 만들어보자

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

- 딕셔너리는 키를 먼저 지정하고 :(콜론)을 붙여서 값을 표현함
- 키에는 값을 하나만 지정할 수 있으며 이런 특성을 따서 키-값 쌍(key-value pair)이라 부름(키-값은 1:1 대응).

12.1 딕셔너리 만들기

키 이름이 중복되면?

- 딕셔너리를 만들 때 키 이름이 중복되면 어떻게 될까? (파이썬 3.6 기준)

```
>>> lux = {'health': 490, 'health': 800, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux['health']    # 키가 중복되면 가장 뒤에 있는 값만 사용함
800
>>> lux    # 중복되는 키는 저장되지 않음
{'health': 800, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

- 딕셔너리 lux를 만들 때 'health': 490이 있고 그 뒤에 'health': 800을 넣었음
- 키 'health'가 중복됨
- lux['health']를 출력해보면 800이 나옴
- 딕셔너리에 키와 값을 저장할 때 키가 중복되면 가장 뒤에 있는 값만 사용
- 중복되는 키는 저장되지 않음

12.1 딕셔너리 만들기

딕셔너리 키의 자료형

- 딕셔너리의 키는 문자열뿐만 아니라 정수, 실수, 불도 사용할 수 있으며 자료형을 섞어서 사용해도 됨
- 값에는 리스트, 딕셔너리 등을 포함하여 모든 자료형을 사용할 수 있음

```
>>> x = {100: 'hundred', False: 0, 3.5: [3.5, 3.5]}
>>> x
{100: 'hundred', False: 0, 3.5: [3.5, 3.5]}
```

- 키에는 리스트와 딕셔너리를 사용할 수 없음

```
>>> x = {[10, 20]: 100}
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    x = {[10, 20]: 100}
TypeError: unhashable type: 'list'
>>> x = {{'a': 10}: 100}
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    x = {{'a': 10}: 100}
TypeError: unhashable type: 'dict'
```

12.1 딕셔너리 만들기

빈 딕셔너리 만들기

- 딕셔너리 = {}
- 딕셔너리 = dict()

```
>>> x = {}
>>> x
{}
>>> y = dict()
>>> y
{}
```

12.1 딕셔너리 만들기

dict로 딕셔너리 만들기

- dict는 다음과 같이 키와 값을 연결하거나, 리스트, 튜플, 딕셔너리로 딕셔너리를 만들 때 사용함
  - 딕셔너리 = dict(키1=값1, 키2=값2)
  - 딕셔너리 = dict(zip([키1, 키2], [값1, 값2]))
  - 딕셔너리 = dict([(키1, 값1), (키2, 값2)])
  - 딕셔너리 = dict({키1: 값1, 키2: 값2})
- dict에서 키=값 형식으로 딕셔너리를 만들 수 있음
- 키에 ' '(작은따옴표)나 " "(큰따옴표)를 사용하지 않아야 함
- 키는 딕셔너리를 만들고 나면 문자열로 바뀜

```
>>> lux1 = dict(health=490, mana=334, melee=550, armor=18.72)    # 키=값 형식으로 딕셔너리를 만듦
>>> lux1
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

12.1 딕셔너리 만들기

dict로 딕셔너리 만들기

- 두 번째 방법은 dict에서 zip 함수를 이용하는 방법
- 키가 들어있는 리스트와 값이 들어있는 리스트를 차례대로 zip에 넣은 뒤 다시 dict에 넣어주면 됨

```
>>> lux2 = dict(zip(['health', 'mana', 'melee', 'armor'], [490, 334, 550, 18.72]))    # zip 함수로
>>> lux2                                                                              # 키 리스트와 값 리스트를 묶음
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

- 키와 값을 리스트가 아닌 튜플에 저장해서 zip에 넣어도 됨
- 세 번째 방법은 리스트 안에 (키, 값) 형식의 튜플을 나열하는 방법

```
>>> lux3 = dict([('health', 490), ('mana', 334), ('melee', 550), ('armor', 18.72)])
>>> lux3                                                  # (키, 값) 형식의 튜플로 딕셔너리를 만듦
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

12.1 딕셔너리 만들기

dict로 딕셔너리 만들기

- 네 번째 방법은 dict 안에서 중괄호로 딕셔너리를 생성하는 방법

```
>>> lux4 = dict({'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72})     # dict 안에서
>>> lux4                                                                        # 중괄호로 딕셔너리를 만듦
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

- 딕셔너리는 키를 통해서 값의 의미를 파악하기 쉬움
- 딕셔너리는 예제의 게임 캐릭터 능력치처럼 특정 주제에 대해 연관된 값들을 모아둘 때 주로 사용

12.2 딕셔너리의 키에 접근하고 값 할당하기

딕셔너리의 키에 접근하고 값 할당하기

- 딕셔너리의 키에 접근할 때는 딕셔너리 뒤에 [ ](대괄호)를 사용하며 [ ] 안에 키를 지정해주면 됨

• 딕셔너리[키]

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux['health']
490
>>> lux['armor']
18.72
```

12.2 딕셔너리의 키에 접근하고 값 할당하기

딕셔너리의 키에 값 할당하기

- 딕셔너리는 [ ]로 키에 접근한 뒤 값을 할당함

- 딕셔너리[키] = 값

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux['health'] = 2037    # 키 'health'의 값을 2037로 변경
>>> lux['mana'] = 1184      # 키 'mana'의 값을 1184로 변경
>>> lux
{'health': 2037, 'mana': 1184, 'melee': 550, 'armor': 18.72}
```

- 딕셔너리에서 키의 값을 출력할 때와 마찬가지로 [ ]에 키를 지정한 뒤 값을 할당하면 됨
- 딕셔너리에 없는 키에 값을 할당하면 해당 키가 추가되고 값이 할당됨

```
>>> lux['mana_regen'] = 3.28    # 키 'mana_regen'을 추가하고 값 3.28 할당
>>> lux
{'health': 2037, 'mana': 1184, 'melee': 550, 'armor': 18.72, 'mana_regen': 3.28}
```

12.2 딕셔너리의 키에 접근하고 값 할당하기

- 딕셔너리에 없는 키에서 값을 가져오려고 하면 에러가 발생함

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux['attack_speed']    # lux에는 'attack_speed' 키가 없음
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    lux['attack_speed']
KeyError: 'attack_speed'
```

12.2 딕셔너리의 키에 접근하고 값 할당하기

딕셔너리에 키가 있는지 확인하기

- 딕셔너리에서 키가 있는지 확인하고 싶다면 in 연산자를 사용하면 됨
  - **키 in 딕셔너리**

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> 'health' in lux
True
>>> 'attack_speed' in lux
False
```

- 딕셔너리에 특정 키가 있으면 True, 없으면 False가 나옴
- 딕셔너리 lux에 키 'health'가 있으므로 True, 'attack_speed'가 없으므로 False가 나옴
- 반대로 in 앞에 not을 붙이면 특정 키가 없는지 확인함
  - **키 not in 딕셔너리**

```
>>> 'attack_speed' not in lux
True
>>> 'health' not in lux
False
```

- not in은 특정 키가 없으면 True, 있으면 False가 나옴

12.2 딕셔너리의 키에 접근하고 값 할당하기

딕셔너리의 키 개수 구하기

- 딕셔너리를 사용하다 보면 딕셔너리의 키 개수(길이)를 구할 필요가 있음
- 딕셔너리의 키와 값을 직접 타이핑할 때는 키의 개수를 알기가 쉬움
- 실무에서는 함수 등을 사용해서 딕셔너리를 생성하거나 키를 추가하기 때문에 키의 개수가 눈에 보이지 않음
- 키의 개수는 len 함수를 사용하여 구함(키와 값은 1:1 관계이므로 키의 개수는 곧 값의 개수임)

- len(딕셔너리)

```
>>> lux = {'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> len(lux)
4
>>> len({'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72})
4
```

12.2 딕셔너리의 키에 접근하고 값 할당하기

딕셔너리의 키 개수 구하기

- len(lux)와 같이 len에 딕셔너리 변수를 넣어서 키의 개수를 구해도 되고, len에 딕셔너리를 그대로 넣어도 됨
- 딕셔너리를 생성할 때는 { }(중괄호)를 사용하고, 키와 값을 1:1 관계로 저장한다는 점이 중요함
- 딕셔너리는 특정 주제에 대해 연관된 값을 저장할 때 사용한다는 점도 꼭 기억해두자
- 이 부분이 리스트, 튜플과 딕셔너리의 차이점