Словари

Вы уже знаете, что...

Множество – это коллекция и типа такой мешок. В него можно:

– положить что-то, чтобы оно там лежало

– убрать что-то, чтобы оно там не лежало

– посмотреть, лежит ли там что-то определённое

Множество в Python имеет тип `set`

```
vowels = {'a', 'e', 'i', 'o', 'u', 'y'}     # так можно создать множество элементов
empty_set = set()                           # так можно создать пустое множество

print(vowels)                       # {'e', 'u', 'i', 'a', 'o', 'y'}
print(empty_set)                    # set()
print(len(vowels), len(empty_set))  # 6 0
```

Словарь – это тоже коллекция

Словарь – это типа такое множество ключей и каждому ключу соответствует значение.

В него можно:

- положить значение по ключу
- посмотреть значение по ключу
- убрать ключ (и его значение)
- посмотреть, лежит ли там определённый ключ

Словарь в Python имеет тип `dict`

```
empty_dict = {}     # так можно создать пустой словарь
rooms = {301: 'информатика', 317: 'история', 322: 'старостат'}

print(empty_dict)                   # {}
print(rooms)                        # {301: 'информатика', 317: 'история', 322: 'старостат'}
print(rooms[301])                   # информатика
print(len(empty_dict), len(rooms))  # 0 3
```

А главное – зачем?

Можно было бы перечислить номера кабинетов и их названия в списке кортежей.
Тогда, чтобы найти кабинет 322, код выглядел бы так:

```
rooms = [(301, 'информатика'), (317, 'история'), (322, 'старостат')]

for elem in rooms:
    number = elem[0]
    room = elem[1]
    if number == 322:
        print(room)     # старостат
        break
else:
    print('Не найден')
```

Вместо этого мы создаём словарь, в котором ключи – номера кабинетов, значения - кабинеты.
И обращаемся к словарю по ключу:

```
rooms = {301: 'информатика', 317: 'история', 322: 'старостат'}
print(rooms[322])                    # старостат
```

in, for

- С помощью оператора in можно проверить, содержится ли ключ в словаре.
- По ключам словаря можно пройтись циклом for.

```
rooms = {301: 'информатика', 317: 'история', 322: 'старостат'}
number = 404
if (number in rooms):
    print(f'Room {number} found: {rooms[number]}')
else:
    print(f'Room {number} not found')   # Room 404 not found


for key in rooms:
    print(key, rooms[key], end=', ')    # 301 информатика, 317 история, 322 старостат,
```

Некоторые операции со словарём

Операция	Описание	Пример	Вывод
len(d)	**Возвращает** количество ключей в словаре	d = {"a": 1, "b": 2, "c": 3} print(len(d))	3
d.pop(key, default)	**Удаляет** ключ и **возвращает** значение (если оно есть, иначе возвращает default)	d = {"a": 1, "b": 2, "c": 3} x = d.pop("a") print(x)	1
del d[key]	**Удалить** ключ из словаря. Если ключа нет, то вызывается исключение KeyError	d = {"a": 1, "b": 2, "c": 3} del d["b"] print(d)	{"a": 1, "c": 3}
dict.clear()	**Очистить** словарь	d = {"a": 1, "b": 2, "c": 3} d.clear() print(d)	{}
dict.get(key, default)	**Возвращает** значение по ключу key. Если ключа нет, то возвращает значение default	d = {"a": 1, "b": 2, "c": 3} print(d.get("e", "Ключа нет в словаре"))	Ключа нет в словаре
d[key]	**Возвращает** значение по ключу key. Если ключа нет, вызывается исключение KeyError	d = {"a": 1, "b": 2, "c": 3} print(d[“a”])	1
dict.keys()	**Возвращает** итерируемый объект, состоящий из ключей словаря	d = {"a": 1, "b": 2, "c": 3} for key in d.keys(): print(key)	a b c
dict.values()	**Возвращает** итерируемый объект, состоящий из значений словаря	d = {"a": 1, "b": 2, "c": 3} for value in d.values(): print(value)	1 2 3
dict.items()	**Возвращает** итерируемый объект, состоящий из кортежей (ключ, значение) словаря	d = {"a": 1, "b": 2, "c": 3} for key, value in d.items(): print(key, value)	a 1 b 2 c 3

Что бывает, когда обращаешься по несуществующему ключу

Бывает KeyError

```
rooms = {301: 'информатика', 317: 'история', 322: 'старостат'}
print(rooms[404])

# print(rooms[404])
#       ~~~~~^^^^^
# KeyError: 404
```

Ключи уникальны, значения - нет

```
rooms = {301: 'информатика', 317: 'история', 322: 'старостат'}
rooms[303] = 'информатика'
print('\n'.join(map(str, rooms.items())))

# (301, 'информатика')
# (317, 'история')
# (322, 'старостат')
# (303, 'информатика’)

inversed_rooms = {'информатика': 301, 'история': 317, 'старостат': 322}
inversed_rooms['информатика'] = 303
print('\n'.join(map(str, inversed_rooms.items())))

# ('информатика', 303)
# ('история', 317)
# ('старостат', 322)
```

Если очень нужно

сложить несколько значений под один ключ, то можно сложить их в коллекцию

```
def add(dictionary, key, value):
    if key in dictionary:
        dictionary[key].append(value)
    else:
        dictionary[key] = [value]

inversed_rooms = {'информатика': [301], 'история': [317], 'старостат': [322]}
add(inversed_rooms, 'информатика', 303)

print('\n'.join(map(str, inversed_rooms.items())))
# ('информатика', [301, 303])
# ('история', [317])
# ('старостат', [322])

print(inversed_rooms['информатика'][1])     # 303
```