Словари. Продолжение

План урока

1. Целые числа как ключи в словарях
2. Метод `.get()`
3. Методы представления `.keys()` и `.values()`
4. Метод `.items()`
5. Допустимые типы ключей в словарях
6. Словари и функции с переменным количеством аргументов

Аннотация

Это продолжение предыдущего урока о словарях. Рассматривается их более продвинутый, но всё-таки базовый функционал: числа и кортежи в качестве ключей, методы-представления `.keys()` и `.values()`, вложенные словари.

Кроме того, рассказывается о функциях с переменными количеством аргументом, которые реализуются с помощью специального словаря **kwargs.

Ключами в словарях могут быть не только строки

Решим задачу. Дан длинный список целых чисел `numbers`. Мы знаем, что некоторые числа встречаются в этом списке несколько раз. Нужно узнать, сколько именно раз встречается каждое из чисел.

```
numbers = [1, 10, 1, 6, 4, 10, 4, 2, 2, 1, 10, 1]
counts = {}
for number in numbers:
    if number not in counts:
        counts[number] = 1
    else:
        counts[number] += 1
```

Просто так сделать `counts[number] += 1` нельзя: если ключа `number` в словаре нет, возникнет ошибка `KeyError`.

В результате работы этой программы все элементы из списка `numbers` окажутся ключами словаря `counts`. Значением `counts[x]` будет количество раз, которое число `x` встретилось в списке `numbers`. Как это работает?

Цикл `for` перебирает все элементы списка `numbers` и для каждого проверяет, присутствует ли он уже в качестве ключа в `counts`. Если нет — значит, число встретилось нам впервые, и мы инициализируем значение `counts[numbers] = 1`. Иначе увеличим `counts[number]` на единицу, поскольку число `number` встретилось нам повторно.

Почему для этой задачи не стоит использовать список, хотя ключи — обычные целые числа? Потому что используя словарь, мы можем решить эту задачу и для вещественных чисел, и для очень больших целых чисел, и вообще для любых объектов, которые можно сравнивать.

Метод .get()

Взять значение в словаре можно не только с помощью квадратных скобок, но и с помощью метода `get`:

```
article = actors.get('Джонни Депп')
```

Преимущество метода в том, что кроме ключа он может принимать и второй аргумент — значение, которое вернётся, если заданного ключа нет:

```
article = actors.get('Джонни Депп', 'Статья о Джонни Деппа не найдена')
```

Воспользуемся этим приёмом для улучшения нашей программы в задаче о повторяющихся числах:

```
numbers = [1, 10, 1, 6, 4, 10, 4, 2, 2, 1, 10, 1]
counts = {}
for number in numbers:
    counts[number] = counts.get(number, 0) + 1
```

Поймите, почему это работает верно.

Методы представления .keys() и .values()

Все ключи словаря можно перебрать циклом `for`:

```
for actor_name in actors:
    print(actor_name, actors[actor_name])
```

Другой способ сделать то же самое — вызвать метод `.keys()`:

```
for actor_name in actors.keys():
    print(actor_name, actors[actor_name])
```

С помощью метода `.keys()` можно получить список всех ключей словаря:

```
actors_names = list(actors.keys())
```

Есть и парный метод `.values()`, возвращающий все значения словаря:

```
all_articles = list(actors.values())
```

Он позволяет, например, проверить, если ли какое-нибудь значение `value` среди значений словаря:

```
value in d.values()
```

Если вы хотите перебрать элементы словаря d так, чтобы в переменной `key` оказывался ключ, а в `value` — соответствующее ему значение, то это можно сделать с помощью метода `.items()` и цикла `for`.

```
for key, val in d.items():
```

Например:

```
for actor_name, article in actors.items():
    print(actor_name, article)
```

Допустимые типы ключей

Мы уже выяснили, что ключами в словарях могут быть строки и целые числа. Кроме этого, ключами могут быть вещественные числа и кортежи.

Ключами в словаре не могут быть другие словари. В принципе, в одном словаре могут быть ключи разных типов, однако обычно принято использовать однотипные ключи.

Вообще, есть строгий способ определить, может ли объект быть ключом в словаре. Для этого объект должен быть **неизменяемым**. Неизменяемые объекты не могут поменять значение в себе во время выполнения программы. Неизменяемыми в Python являются числа, строки и кортежи. Именно их обычно и используют в качестве ключей словарей.

Вот как может выглядеть список с ключами-кортежами. В качестве ключа используются координаты, а в качестве значения — название города.

```
cities = {(55.75, 37.5): 'Москва', (59.8, 30.3): 'Санкт-Петербург', (54.32, 48.39): 'Ул
ьяновск'}
print(cities[(55.75, 37.5)])
cities[(53.2, 50.15)] = 'Самара'
```

Возможно, нам захочется развернуть этот словарь, то есть построить такой словарь, в котором ключами будут города, а значениями — их координаты.

```
coordinates = {}
for coordinate, city in cities.items():
    coordinates[city] = coordinate
```

Некоторые значения потеряются при разворачивании словаря, если в исходном словаре были повторяющиеся значения. Это объясняется тем, что значения в словаре могут повторяться, а вот ключи обязаны быть уникальными.

Значениями в словаре, в отличие от ключей, могут быть объекты любого типа: числа, строки, кортежи, списки и даже другие словари. Вот, например, как можно сохранить список фильмов для каждого из актёров:

```
films = {
  'Джонни Депп': ['Эдвард Руки-Ножницы', 'Одинокий рейнджер', 'Чарли и шоколадная фабри
ка', ..],
  'Эмма Уотсон': ['Гарри Поттер и философский камень', 'Красавица и Чудовище', ..],
  # ...
}

# Вывести список фильмов, в которых снималась Эмма Уотсон
print(films['Эмма Уотсон'])

# Проверить, снимался ли Джонни Депп в фильме «Чарли и шоколадная фабрика»
if 'Чарли и шоколадная фабрика' in films['Джонни Депп']:
    print('Снимался!')
```

Где словари встречаются в Python

Вы можете сами создавать словари, однако иногда они будут встречаться вам в Python уже готовыми. Как уже обсуждалось в одном из прошлых уроков, именованные аргументы для функции попадают в специальный словарь, который обычно называется **kwargs. Например:

```
def function(*args, **kwargs):
    print('Args = ', args)
    print('KWargs = ', kwargs)
```

Если мы вызовем эту функцию с параметрами

```
function(10, 20, one=1, two='two222')
```

то в списке `args` окажутся числа `[10, 20]`, а в `kwargs` окажется словарь:

```
{'one': 1, 'two': 'two222'}
```