Словари

Что такое словарь?

Словарь в Python - это структура данных, которая представляет собой коллекцию пар ключ-значение, где каждый ключ связан с определенным значением. Основные свойства словарей:

- *Уникальность ключей:* Каждый ключ в словаре должен быть уникальным. Если попытаться добавить уже существующий ключ, значение этого ключа будет обновлено.
- *Неупорядоченность:* Элементы в словаре не имеют определенного порядка. Порядок следования элементов может меняться при выполнении различных операций, таких как добавление, удаление или изменение элементов.
- *Изменяемость:* Словари являются изменяемыми структурами данных, то есть их можно изменять после создания, добавлять и удалять элементы.
- *Использование различных типов ключей и значений*: Ключи словаря могут быть любого неизменяемого типа данных, такого как строки, числа или кортежи. Значения могут быть любого типа данных, включая другие словари или даже функции.

Создание словарей в Python можно выполнить несколькими способами:

Литеральный способ: Мы можем создать словарь, перечислив его элементы в фигурных скобках {}, где каждый элемент представляет собой пару ключ-значение, разделенную двоеточием :. Элементы разделяются запятыми. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
```

Использование функции dict(): Мы также можем создать словарь с помощью встроенной функции dict(), передавая в нее последовательность (список кортежей) пар ключ-значение. Например:

```
my_dict = dict([('apple', 3), ('banana', 2), ('orange', 5)])
```

Использование генератора словарей: Мы можем создать словарь с помощью генератора словарей, который позволяет создавать словари более компактным и элегантным способом. Например:

```
my_dict = {x: x**2 for x in range(5)}
```

Операции с добавлением и удалением элементов в словаре включают следующие действия:

Добавление элементов:

Чтобы добавить новую пару ключ-значение в словарь, вы можете назначить значение для нового ключа, как если бы это уже существовало. Если ключ уже существует, его значение будет перезаписано новым значением. Например:

```
my_dict = {'apple': 3, 'banana': 2}
my_dict['orange'] = 5  # Добавление новой пары ключ-значение
```

Удаление элементов:

Для удаления элемента из словаря можно использовать оператор del, передав ключ элемента, который вы хотите удалить. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
del my_dict['banana']  # Удаление элемента с ключом 'banana'
```

Метод pop() также может быть использован для удаления элемента по его ключу и возврата соответствующего значения. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
value = my_dict.pop('banana')  # Удаление элемента
```

Метод popitem() удаляет и возвращает последнюю вставленную пару ключ-значение из словаря. Так как словари в Python являются неупорядоченными коллекциями, порядок вставки не гарантирован. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
key, value = my_dict.popitem()  # Удаление и получ
```

Метод clear() удаляет все элементы из словаря. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
my_dict.clear()  # Удаление всех элементов из словаря
```

Операции доступа к значениям по ключу и проверки наличия ключа в словаре включают следующие действия:

Доступ к значениям по ключу:

Для доступа к значению по ключу используется синтаксис индексации, в котором указывается ключ в квадратных скобках после имени словаря. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
apple_count = my_dict['apple']  # Получение значения
```

Если ключ отсутствует в словаре, будет вызвано исключение KeyError. Для избежания исключения можно использовать метод get(), который позволяет указать значение по умолчанию в случае отсутствия ключа. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
pear_count = my_dict.get('pear', 0)  # Получение значения
```

Проверка наличия ключа в словаре:

Для проверки наличия ключа в словаре можно использовать оператор in. Он возвращает логическое значение True, если ключ присутствует в словаре, и False, если ключ отсутствует. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
is_apple_present = 'apple' in my_dict  # Проверка наличия ключа
```

Также можно использовать методы keys() и values() для получения всех ключей и всех значений словаря соответственно, а затем использовать оператор in для проверки наличия конкретного ключа или значения. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
all_keys = my_dict.keys()  # Получение всех ключей словаря
is_apple_present = 'apple' in all_keys  # Проверка наличия ключа 'apple'
```

Метод items() возвращает представление пар ключ-значение словаря. Он позволяет проверить наличие конкретной пары ключ-значение. Например:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
all_items = my_dict.items()  # Получение представления пар
is_apple_present = ('apple', 3) in all_items  # Проверка
```

Использование словарей в циклах - это мощный инструмент для обработки данных и выполнения различных операций над элементами словаря. Вот основные способы работы со словарями в циклах:

Итерация по ключам:

В Python можно перебирать ключи словаря, используя цикл for. Это делается с помощью метода keys() или просто перебором самого словаря:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
for key in my_dict.keys():
    print(key)
```

Итерация по значениям:

Аналогично, можно перебирать значения словаря, используя метод values():

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
for value in my_dict.values():
    print(value)
```

Итерация по парам ключ-значение:

Часто требуется получить как ключ, так и значение при переборе словаря. Для этого используется метод items(), который возвращает представление пар ключ-значение:

```
my_dict = {'apple': 3, 'banana': 2, 'orange': 5}
for key, value in my_dict.items():
    print(key, value)
```

Практические задания

1. Напишите функцию для обновления словаря новыми значениями из другого словаря. Если ключи уже существуют, значения должны быть обновлены.
2. Напишите функцию для удаления элементов из словаря по заданному списку ключей.
3. Напишите функцию, которая создает словарь из двух списков: один список для ключей, другой для значений. Если списки разной длины, использовать минимальную длину.
4. Напишите функцию, которая принимает словарь и список ключей, а затем проверяет, содержит ли словарь все эти ключи.
5. Напишите функцию, которая принимает список элементов и возвращает словарь, где ключами являются уникальные элементы списка, а значениями - количество повторений каждого элемента.
6. Напишите функцию, которая принимает словарь и значение, а затем возвращает список ключей, соответствующих этому значению.
7. Напишите функцию, которая принимает словарь и ключ, а затем возвращает значение, соответствующее этому ключу. Если ключ отсутствует в словаре, функция должна вернуть значение по умолчанию.
8. Напишите функцию для обращения словаря, то есть создания нового словаря, где ключи и значения поменяны местами.