Словари

Date: 07/05/2024

Tochilkina Ksenya Stanislavovna

Словари

Что такое словари?

Словари - это изменяемые неупорядоченные структуры данных, которые хранят в себе пары “ключ-значение”, предназначенные для объединение взаимосвязанной информации. Например ключ имя со значением возраст выглядят так - “имя”:”Виктор”.

В других языках программирования так же есть словари, однако они называются иначе - ассоциативный массив, который так же хранит пары “ключ-значение”.

Для того, чтобы создать пустой словарь будем использовать следующую запись:

```
my_dict_1 = {}
print(type(my_dict_1))
```

Заполненный словарь может выглядеть так:

```python
my_dict_1 = {
    "user": "Iron Man",
    "nickname": "playboy",
    "team": ["SpiderMan", "Capitan America"]
}
```

И так, каждый ключ связан со своим значением и отделяется от него двоеточием(:). Ключи (user, nickname, team) в словарях уникальны и НЕИЗМЕННЫ - то есть нельзя в виде ключа передавать список, так как список это изменяемый тип данных.Чаще всего в виде ключей передаются строки или числа.

А вот значения, напротив, могут быть представлены в виде любого типа данных: числа, строки, списки, множества, другие словари и т.д.

Индексация в словарях

Индексация в словарях происходит с использованием ключей. Чтобы получить значение из словаря, вы указываете ключ в квадратных скобках после имени словаря. Например, если у нас есть словарь my_dict с ключами и значениями, мы можем получить значение, связанное с определенным ключом, используя следующий синтаксис: **my_dict[ключ].**

Вот пример использования индексации в словарях:

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}

value = my_dict['apple']  # value = 4
```

Добавление новых пар в словарь

Чтобы добавить новые пары в словарь в языке программирования Python, вы можете использовать несколько подходов:

- Используйте квадратные скобки и оператор присваивания: словарь[ключ] = значение. Например:

```
my_dict = {'apple': 4, 'orange': 2}
my_dict['kiwi'] = 5
```

- Используйте метод update(), чтобы объединить словарь с другим словарем или с последовательностью пар ключ-значение. Например:

```
my_dict = {'apple': 4, 'orange': 2}
my_dict.update({'kiwi': 5})
```

- Используйте метод setdefault(), чтобы добавить новую пару только в том случае, если ключ отсутствует в словаре. Например:

```
my_dict = {'apple': 4, 'orange': 2}
my_dict.setdefault('kiwi', 5)
```

Перебор словарей

Для перебора значений в словаре в языке программирования Python вы можете использовать различные методы и конструкции циклов. Вот несколько способов:

- Используйте цикл for в сочетании с методом values(), чтобы перебрать все значения в словаре. Например:

```
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
for value in my_dict.values():
    print(value)
```

- Используйте цикл for в сочетании с методом items(), чтобы перебрать все пары ключ-значение в словаре. Например:

```
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
for key, value in my_dict.items():
    print(key, value)
```

- Используйте цикл for в сочетании с методом keys(), чтобы перебрать все ключи в словаре, а затем получить значения по ключам. Например:

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
for key in my_dict.keys():
    value = my_dict[key]
    print(value)
```

В каждом из этих примеров мы используем цикл for, чтобы перебрать элементы словаря. Методы values(), items() и keys() позволяют получить значения, пары ключ-значение и ключи словаря соответственно.

Основные методы при работе со словарями

Основные методы для работы со словарями в языке программирования Python включают:

- Метод keys():
  Возвращает список всех ключей в словаре.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
keys = my_dict.keys()
print(keys)  # ['apple', 'orange', 'kiwi']
```

- Метод values():
  Возвращает список всех значений в словаре.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
values = my_dict.values()
print(values)  # [4, 2, 5]
```

- Метод items():
  Возвращает список всех пар ключ-значение в словаре.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
items = my_dict.items()
print(items)  # [('apple', 4), ('orange', 2), ('kiwi', 5)]
```

- Метод get(key, default):
  Возвращает значение, связанное с указанным ключом. Если ключ не существует, возвращает значение по умолчанию.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
value = my_dict.get('apple', 0)
print(value)  # 4

value = my_dict.get('banana', 0)
print(value)  # 0
```

- Метод update(other_dict):
  Объединяет два словаря, добавляя ключи и значения из другого словаря.

```python
my_dict = {'apple': 4, 'orange': 2}
other_dict = {'kiwi': 5, 'banana': 3}
my_dict.update(other_dict)
print(my_dict)  # {'apple': 4, 'orange': 2, 'kiwi': 5, 'banana': 3}
```

- Метод pop(key):
  Удаляет и возвращает значение, связанное с указанным ключом.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
value = my_dict.pop('apple')
print(value)  # 4
print(my_dict)  # {'orange': 2, 'kiwi': 5}
```

- Метод clear():
  Удаляет все элементы из словаря.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
my_dict.clear()
print(my_dict)  # {}
```

- Метод copy():
  Создает и возвращает копию словаря.

```python
my_dict = {'apple': 4, 'orange': 2, 'kiwi': 5}
new_dict = my_dict.copy()
print(new_dict)  # {'apple': 4, 'orange': 2, 'kiwi': 5}
```

- Метод setdefault(key, default):
  Возвращает значение, связанное с указанным ключом. Если ключ не существует, добавляет ключ с указанным значением.

```
my_dict = {'apple': 4, 'orange': 2}
value = my_dict.setdefault('kiwi', 5)
print(value)  # 5
print(my_dict)  # {'apple': 4, 'orange': 2, 'kiwi': 5}
```

- Метод fromkeys(seq, value):
  Создает новый словарь с ключами из указанной последовательности и значениями, установленными в указанное значение.

```
keys = ['apple', 'orange', 'kiwi']
value = 0
my_dict = dict.fromkeys(keys, value)
print(my_dict)  # {'apple': 0, 'orange': 0, 'kiwi': 0}
```