

Практическая работа 5 - Словари (dict) и работа с ними

Для хранения пар (ключ-значение) в Питон используются словари. В ходе урока мы научимся создавать, а также использовать словари. Дополнительно мы изучим функции по работе со словарями в Python.

Словари отличаются способом их создания, а также форматом индекса. Если в обычном списке в качестве индексов выступают числа (0, 1, 2...), то здесь на их замену приходят "ключи". Так, мы можем создать некий ключ (Студент Федя, например) и добавить к нему целый ряд характеристик (список) в качестве одного элемента. Теперь найти нужного нам студента будет намного проще, так как мы просто будем оперировать ключами, а не числами.

Во многих других языках программирования такие словари зачастую называются ассоциативными массивами, поэтому не редко можно услышать и такое название.

Чтобы создать словарь вы можете воспользоваться следующей конструкцией:

```
words = {'short': 'Jo', 'long': 'John'}
```

Методы словарей

На фото ниже представлены различные методы, которые можно использовать при работе со словарями:

dict.clear() - очищает словарь.

dict.copy() - возвращает копию словаря.

classmethod **dict.fromkeys(seq[, value])** - создает словарь с ключами из seq и значением value (по умолчанию None).

dict.get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).

dict.items() - возвращает пары (ключ, значение).

dict.keys() - возвращает ключи в словаре.

dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).

dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

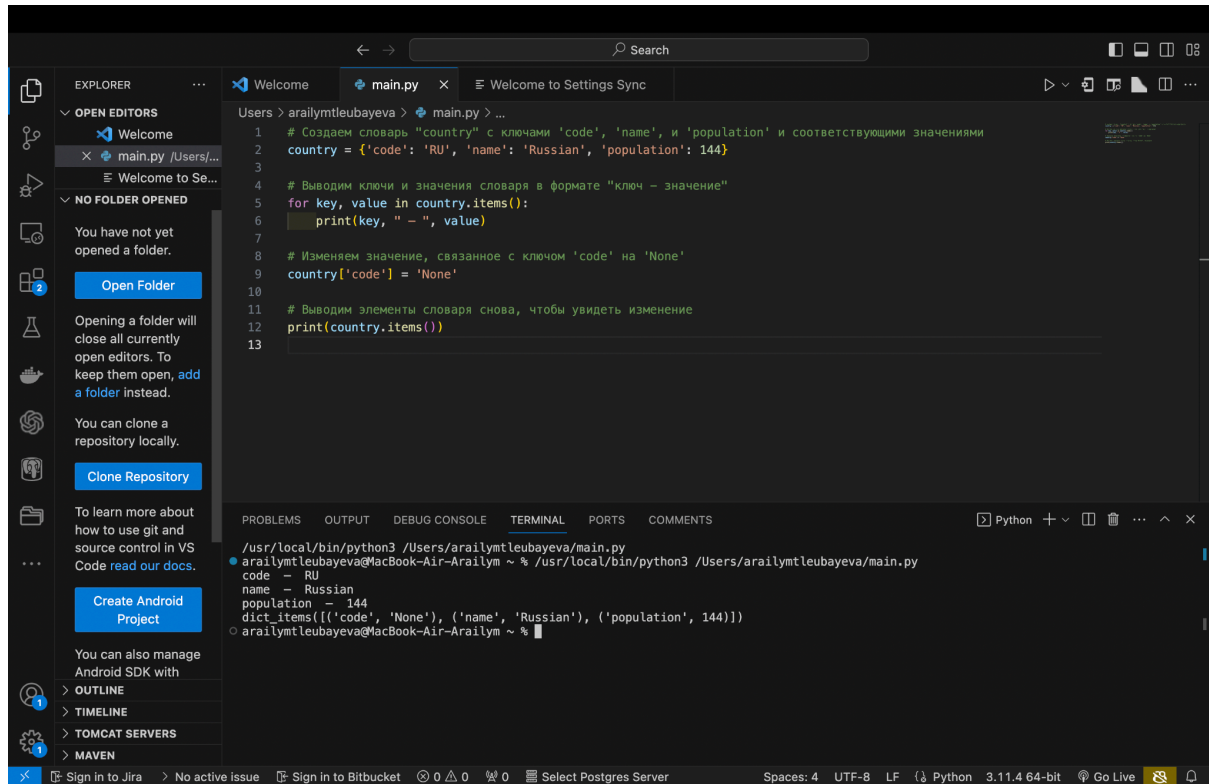
dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None).

dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

dict.values() - возвращает значения в словаре.

Исходный код Примеры

Создание словарей



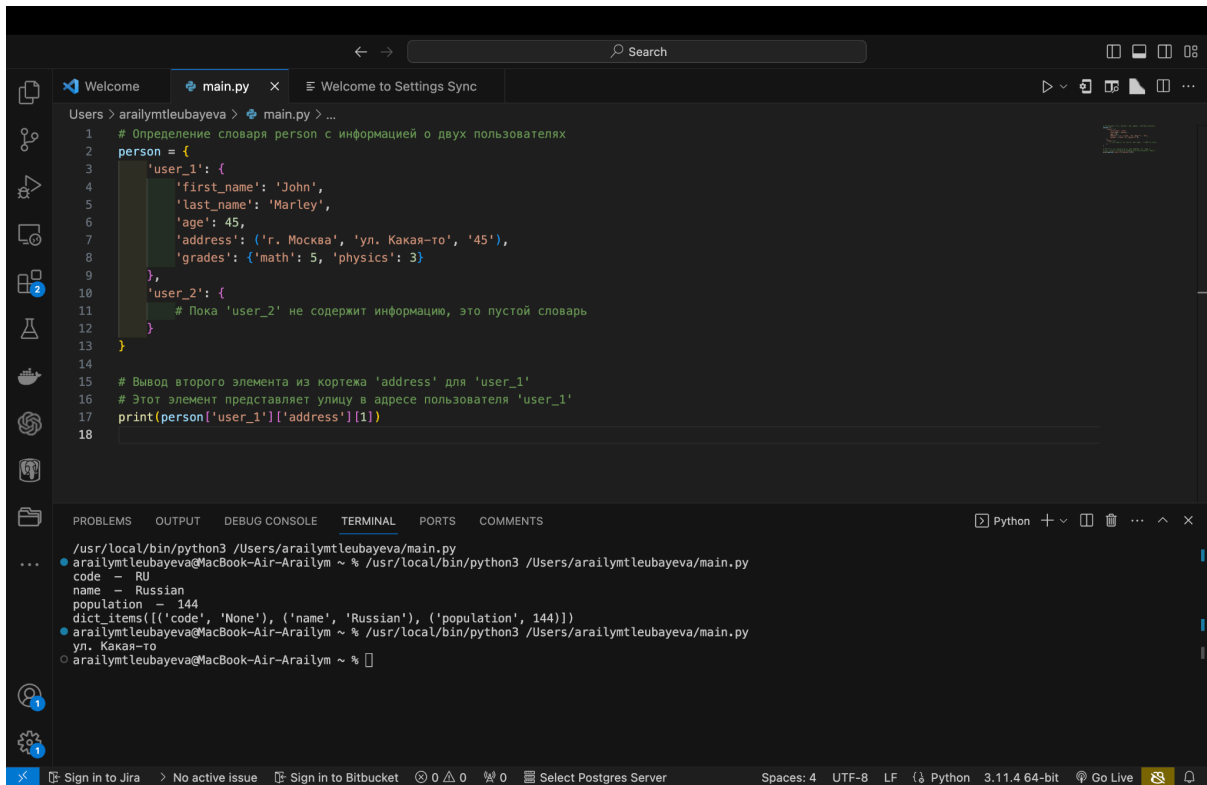
The screenshot shows a Visual Studio Code editor with a Python file named `main.py` open. The code creates a dictionary, prints its items, and then updates a value. Below the code, the terminal shows the execution of the script, displaying the dictionary's contents before and after the update.

```
1 # Создаем словарь "country" с ключами 'code', 'name', и 'population' и соответствующими значениями
2 country = {'code': 'RU', 'name': 'Russian', 'population': 144}
3
4 # Выводим ключи и значения словаря в формате "ключ - значение"
5 for key, value in country.items():
6     print(key, " - ", value)
7
8 # Изменяем значение, связанное с ключом 'code' на 'None'
9 country['code'] = 'None'
10
11 # Выводим элементы словаря снова, чтобы увидеть изменение
12 print(country.items())
13
```

```
/usr/local/bin/python3 /Users/arailymtleubayeva/main.py
arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/main.py
code - RU
name - Russian
population - 144
dict_items([('code', 'None'), ('name', 'Russian'), ('population', 144)])
arailymtleubayeva@MacBook-Air-Arilym ~ %
```

Большой словарь

- Вывод user1 - first_name, last_name, age, address
- Вывод user2 - first_name, age, grades



The screenshot shows a code editor with a Python script named `main.py`. The script defines a dictionary `person` with two keys: `'user_1'` and `'user_2'`. `'user_1'` contains a nested dictionary with fields like `'first_name'`, `'last_name'`, `'age'`, `'address'`, and `'grades'`. `'user_2'` is an empty dictionary. The script prints the address of `'user_1'`. The terminal output shows the execution of the script, displaying the nested dictionary structure for `'user_1'`.

```
1 # Определение словаря person с информацией о двух пользователях
2 person = {
3     'user_1': {
4         'first_name': 'John',
5         'last_name': 'Marley',
6         'age': 45,
7         'address': ('г. Москва', 'ул. Какая-то', '45'),
8         'grades': {'math': 5, 'physics': 3}
9     },
10    'user_2': {
11        # Пока 'user_2' не содержит информации, это пустой словарь
12    }
13 }
14
15 # Вывод второго элемента из кортежа 'address' для 'user_1'
16 # Этот элемент представляет улицу в адресе пользователя 'user_1'
17 print(person['user_1']['address'][1])
18
```

Terminal output:

```
/usr/local/bin/python3 /Users/arailymtleubayeva/main.py
code - RU
name - Russian
population - 144
dict_items([('code', 'None'), ('name', 'Russian'), ('population', 144)])
arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/main.py
ул. Какая-то
arailymtleubayeva@MacBook-Air-Arilym ~ %
```

Самостоятельная работа

Работа со словарями

Создайте словарь, описывающий учеников в классе. Словарь должен содержать следующие ключи:

- имя
- возраст
- средняя оценка.

Выведите данные на экран через цикл `for`. В цикле помимо значений также выведите ключи.

2) Кортежи вместо ключей

Создайте словарь, в котором вместо ключей будут кортежи.

Выведите первый элемент такого словаря.

Стоит заметить, что подобное нельзя сделать со списками. Только кортежи могут выступать в качестве ключей для словаря.

3) Генерация словаря

При помощи цикла `for` сгенерируйте словарь из квадратов чисел от 1 до 7.

4) Способы создания словаря

Создайте словарь несколькими способами:

через фигурные скобки;
при помощи функции dict;
при помощи метода fromkeys.

5) Встроенные функции

Есть следующий словарь:

```
d = {"Один" : "Питон", "Два" : "C++", "Три" : "Java", "Четыре" : "C#"}
```

Выполните операции:

- сделайте копию словаря
- удалите оригинал
- из копии удалите ключ «Три» вместе с его значением
- добавьте новый элемент в конец словаря: "Новое" -> "Kotlin"