

Коллекции данных: словари

Елена Никитина
Руководитель проектов ООО «Аналитические программные решения»



Елена Никитина

О спикере:

- руководитель проектов в ИТ-компании, стаж в отрасли — 19 лет
- 5 лет в разработке летающей робототехники и беспилотных систем
- преподаёт и руководит проектной деятельностью студентов и школьников по программированию и робототехнике в МАИ, «Сириусе» и на Национальной технологической олимпиаде



Вспоминаем прошрое занятие

Мы хотим объединить два набора данных:
("яблоко", "апельсин", "гранат") и ("груша",
"гранат", "грейпфрут").

Вопросы:

- 1 Что получится, если они будут списками?
- 2 Что получится, если они будут множествами?



Вспоминаем прошрое занятие

Мы хотим объединить два набора данных:
("яблоко", "апельсин", "гранат") и ("груша",
"гранат", "грейпфрут").

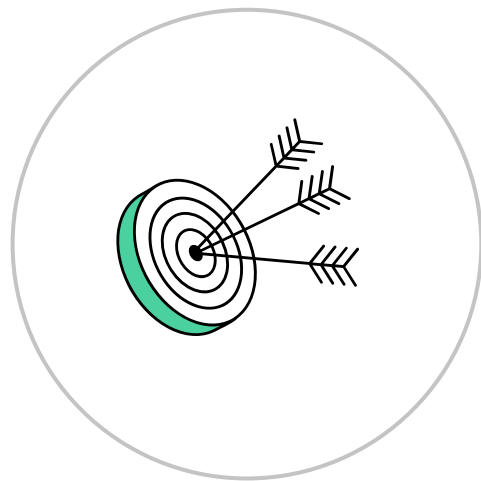
Ответ:

- 1 ["яблоко", "апельсин", "гранат", "груша",
"гранат", "грейпфрут"]
- 2 {"яблоко", "апельсин", "гранат", "груша",
"грейпфрут"}



Цели занятия

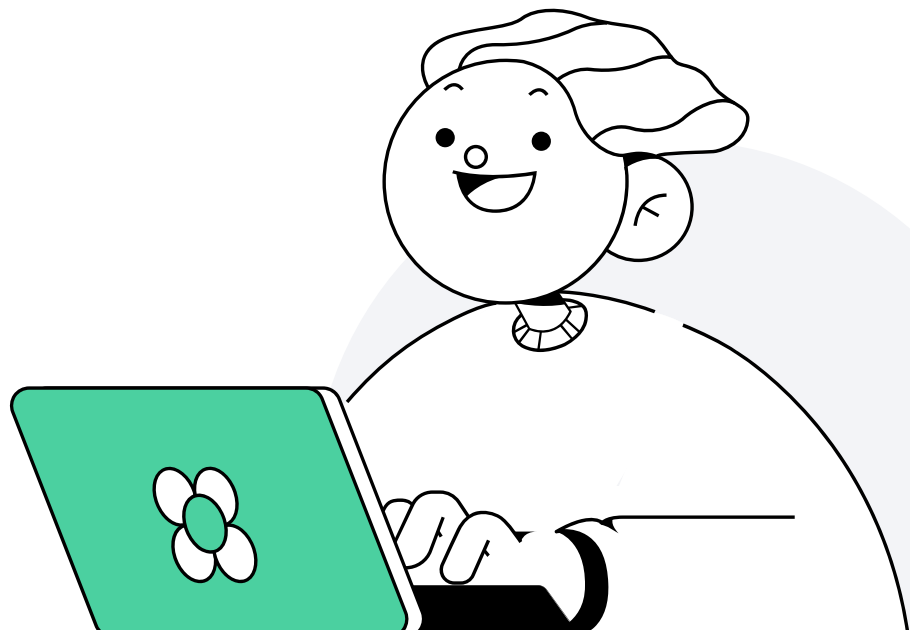
- Познакомимся со словарями и операциями над ними
- Научимся применять словарь как инструмент для поиска данных
- Научимся проектировать вложенные структуры данных



План занятия

- 1 Словари
- 2 Операции над словарями
- 3 Итоги
- 4 Домашнее задание

* Нажмите на раздел для перехода



Словари

Что такое словарь и как он устроен, чем словари отличаются от других коллекций данных, как создавать словари



1

Словари (dictionaries)

неупорядоченные коллекции
произвольных объектов
с доступом по ключу

Как устроен словарь

В отличие от других типов данных элемент в словаре Python состоит из **двух частей**:

- ключа (**key**), по которому мы ищем
- значения (**value**)

Они разделяются двоеточием: **key:value**

ключ значение ключ значение

{ **key1:value1**, **key2:value2** }

Элемент

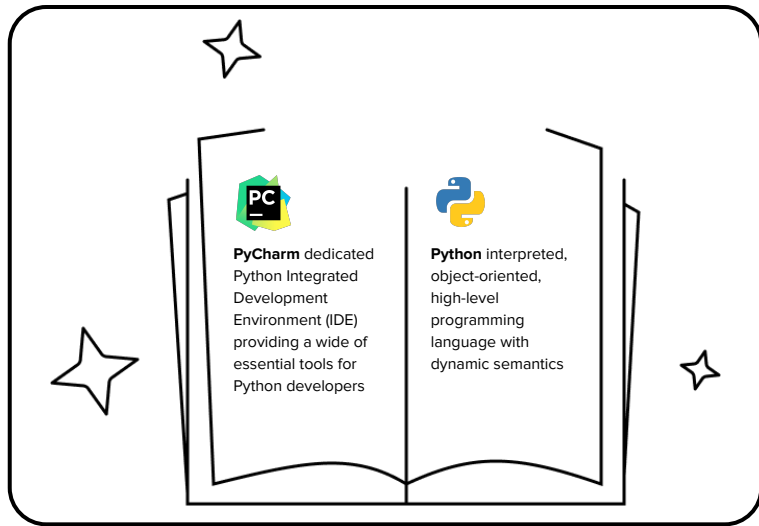
Элемент

Что такое словарь

Словарь в Python очень похож на настоящий толковый словарь.

Слово — это **key**, по которому мы ищем определение слова. Определение слова — это **value**: { **key:value** }

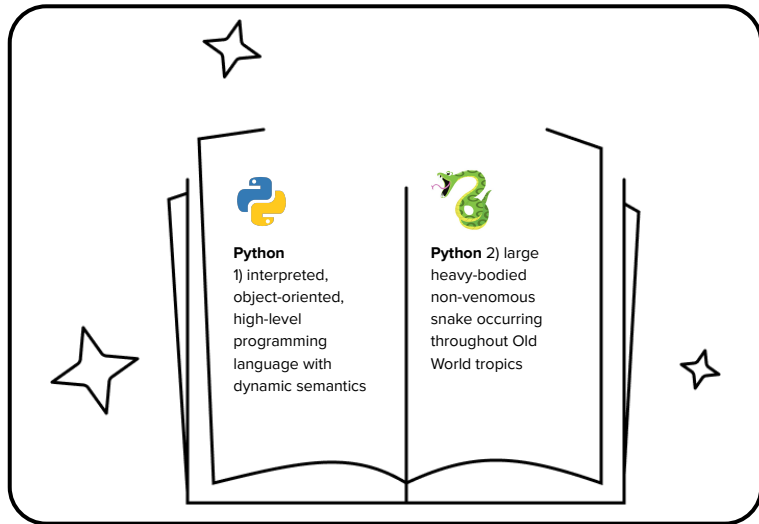
```
{  
    key           value  
    'Pycharm' : 'dedicated Python  
                Integrated Development  
                Environment...',  
    key           value  
    'Python' : 'interpreted,  
               object-oriented, high-level  
               programming language...'  
}
```



Что такое словарь

Если наше значение состоит из нескольких определений, тогда значение нужно сделать списком (**list**): {key:[val1, val2]}

```
{  
  key  
  'Python' : value  
  [  
    val1 'interpreted, object-oriented,  
          high-level programming language...',  
    val2 'large heavy-bodied non-venomous  
          snake'  
  ]  
}
```



Тренировочные данные

В качестве примера мы будем использовать простой словарь **countries_dict**, где ключи — названия стран, а значения — столицы этих стран:

```
countries_dict = {  
    "Россия": "Москва",  
    "Белоруссия": "Минск",  
    "Германия": "Берлин"  
}
```

Создание словаря

Создать пустой словарь

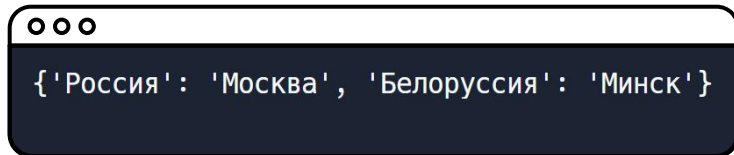
```
countries_dict = dict()  
countries_dict = {}
```



A terminal window with a dark background and a light header bar containing three dots. The terminal displays the code `{}` on a single line.

Создать словарь вручную

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}
```

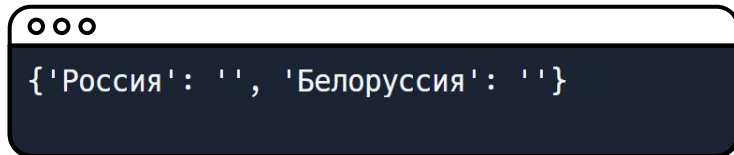


A terminal window with a dark background and a light header bar containing three dots. The terminal displays the code `{'Россия': 'Москва', 'Белоруссия': 'Минск'}` on a single line.

Создать словарь из списка ключей

В этом случае все значения в словаре будут одинаковые (пустая строка вместо столицы):

```
countries_dict = dict.fromkeys(["Россия",  
                                "Белоруссия"], "")
```



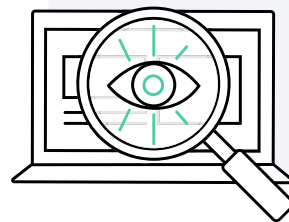
A terminal window with a dark background and a light header bar containing three dots. The terminal displays the code `{'Россия': '', 'Белоруссия': ''}` on a single line.

Правила для словарей

- Словарь обычно создаётся функцией **dict()** или через {}
- Элемент словаря состоит из пары **ключ:значение**. Двоеточие между ними обязательно
- Словарь записывается в фигурных скобках {}
- Элементы (пары **ключ:значение**) в словаре записываются через запятую
- Порядок элементов в словаре не соблюдается. Получить значение в словаре по индексу (как в списке) нельзя
- Доступ к значению словаря — только **по ключу**
- Ключи в словаре должны быть **уникальными**, не могут повторяться
- Ключами могут быть только **неизменяемые** типы данных: числа, строки, кортежи
- Значениями могут быть любые типы данных, включая вложенные списки и словари

Демонстрация

Создание словарей



[Ссылка на код](#)

Операции над словарями

Методы словарей, работа с элементами словаря, итерация по словарю, поиск в словаре



2

Виды операций над словарями

Общие операции:

```
len(), copy(), clear()
```

Работа с ключами и значениями словаря:

```
keys(), values(), items()
```

Добавление и удаление элементов, объединение словарей:

```
del, pop(), [], get(), setdefault(), update()
```

Общие операции

len()

Подсчитывает количество элементов в словаре

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
len(countries_dict)
```

ooo

2

copy()

Создаёт копию данных словаря(так же, как для списков и множеств)

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
copy_dict = countries_dict  
.copy()  
  
id(countries_dict)  
  
id(copy_dict)
```

ooo

140490464333184
140490463491072

clear()

Очищает содержимое множества

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
countries_dict.clear()
```

ooo

{}

Получение элементов

dict[key]

Возвращает значение по ключу. Если такого ключа нет, код сгенерирует исключение `KeyError`

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
city = countries_dict["Россия"]  
city = countries_dict["Турция"]
```

ooo

"Москва"

`KeyError`

get(key)

Безопасно возвращает значение по ключу. Если ключа нет, возвращает `None`

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
city = countries_dict.get("Россия")  
city = countries_dict.get("Турция")
```

ooo

"Москва"

`None`

Добавление и изменение элементов

`dict[key] = value`

Создаёт в словаре пару **ключ:значение**.
Если такой ключ уже есть, его значение перезаписывается

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
countries_dict["Турция"] = "Анкара"
```

ooo

```
{"Россия": "Москва", "Белоруссия": "Минск",  
 "Турция": "Анкара"}
```

`setdefault(key, value)`

Создаёт ключ и значение, если такого ключа ещё нет в словаре. В противном случае ничего не делает. Если не указать value, присвоит None

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
countries_dict.setdefault("Турция",  
                           "Анкара")  
  
countries_dict.setdefault("Россия", "")
```

ooo

```
{"Россия": "Москва", "Белоруссия": "Минск",  
 "Турция": "Анкара"}
```

Удаление элементов

`del dict[key]`

Удаляет элемент из словаря по ключу. Если такого ключа нет, код сгенерирует исключение `KeyError`

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
del countries_dict["Россия"]  
  
del countries_dict["Турция"]
```

ooo

```
{"Белоруссия": "Минск"}
```

`KeyError`

`pop(key)`

Удаляет ключ и возвращает значение. Если такого ключа нет, код сгенерирует исключение `KeyError`

```
countries_dict = {"Россия": "Москва",  
                  "Белоруссия": "Минск"}  
  
value = countries_dict.pop("Россия")  
  
value = countries_dict.pop("Турция")
```

ooo

```
"Москва", {"Белоруссия": "Минск"}
```

`KeyError`

Работа с ключами и значениями

keys()

Получает все ключи словаря в виде dict_keys. Можно преобразовать в список

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
keys =  
countries_dict.keys()  
  
keys_list = list(keys)
```

ooo

```
["Россия", "Белоруссия"]
```

values()

Получает все значения словаря в виде dict_values. Можно преобразовать в список

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
values =  
countries_dict.values()  
  
values_list = list(values)
```

ooo

```
["Москва", "Минск"]
```

items()

Получает все пары ключ:значение словаря в виде dict_items. Можно преобразовать в список кортежей

```
countries_dict =  
{"Россия": "Москва",  
 "Белоруссия": "Минск"}  
  
items =  
countries_dict.items()  
  
items_list = list(items)
```

ooo

```
[("Россия", "Москва"),  
 ("Белоруссия", "Минск")]
```

Объединение словарей

`dict1.update(dict2)`

Добавляет второй словарь в первый, существующие ключи перезаписываются

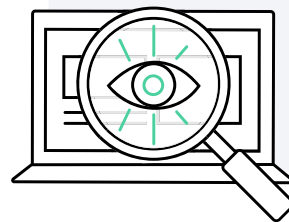
```
countries_dict = {"Россия": "Москва", "Белоруссия": "Минск"}  
countries2_dict = {"Россия": "Москва", "Турция": "Анкара"}  
countries_dict.update(countries2_dict)
```

ooo

```
{"Россия": "Москва", "Белоруссия": "Минск", "Турция": "Анкара"}
```

Демонстрация

Напишем примеры кода для операций над словарями



[Ссылка на код](#)

Как создавать словарь, читать и записывать данные

Создаём словарь, где ключи — страны, значения — их столицы:

```
countries_dict = {  
    "Россия": "Москва",  
    "Белоруссия": "Минск",  
    "Германия": "Берлин"  
}
```

Читаем из словаря значение по ключу (выведет “Москва”):

```
print(countries_dict["Россия"])
```

Записываем в словарь значение по ключу:

```
countries_dict["Турция"] = "Анкара"
```

Итерация по словарю

```
countries_dict = {  
    "Россия": "Москва",  
    "Белоруссия": "Минск",  
    "Германия": "Берлин"  
}
```

Стандартный цикл (идёт только по ключам):

```
for country in countries_dict:  
    print(country, countries_dict[country])
```

Цикл с items() (идёт по ключам и значениям одновременно):

```
for country, city in countries_dict.items():  
    print(country, city)
```

Получить ключи и значения

Получить ключи:

```
keys = countries_dict.keys()
print(list(keys))
```

Получить значения:

```
values = countries_dict.values()
print(list(values))
```

Получить значения:

ooo

```
['Россия', 'Белоруссия', 'Германия', 'Франция', 'Турция']
['Москва', 'Минск', 'Берлин', 'Париж', 'Анкара']
```

«Опасные» операции

Пытаться прочесть или удалить ключ, которого не существует:

```
countries_dict = {  
    "Россия": "Москва",  
    "Белоруссия": "Минск"  
}  
  
print(countries_dict["Турция"])  
del countries_dict["Турция"]
```

ooo

KeyError: 'Турция'

ooo

KeyError: 'Турция'

Безопасное чтение или удаление:

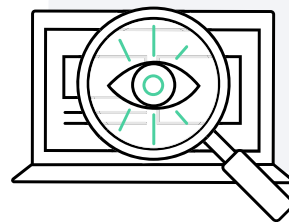
```
print(countries_dict.get("Турция"))  
if "Турция" in countries_dict.keys():  
    del countries_dict["Турция"]
```

ooo

None

Демонстрация

Рассмотрим задачу поиска по словарю, выясним, как задача поиска влияет на структуру словаря, поймём, почему иногда одного словаря может быть недостаточно



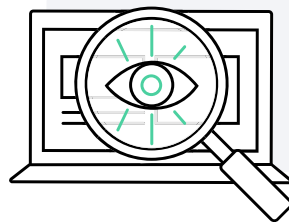
[Ссылка на код](#)

Демонстрация

Разберём кейс «Сравним курсы Нетологии по программированию»
и рассмотрим принципы проектирования вложенных структур данных



[Ссылка на код](#)



Правила проектирования структур данных

- Если нужна типовая структура с описанием данных, создавайте словарь (dict)
- Если нужно много раз повторить типовую структуру, поместите её в список (list)
- Если значение словаря состоит из нескольких элементов, создавайте значение-список (list)
- Если вам требуется поиск по различным критериям, создавайте несколько связанных словарей (dict)

Итоги

Сегодня мы:

- 1 Познакомились со словарями, научились их создавать и изменять
- 2 Научились искать данные с использованием словарей и исследовали случай, когда требуется связывать несколько словарей между собой
- 3 Потренировались проектировать вложенные структуры данных

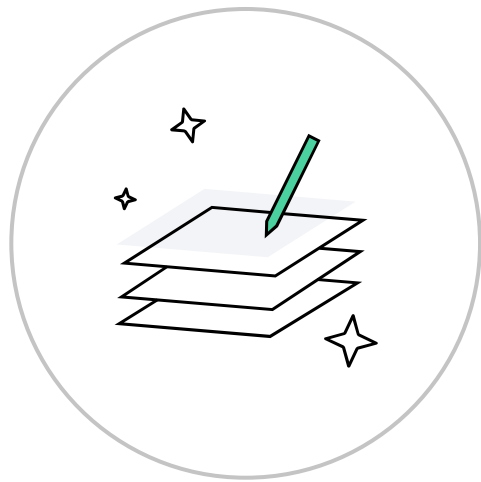


Домашнее задание

Давайте посмотрим ваше домашнее задание.

- 1 Вопросы по домашней работе задавайте в чате группы
- 2 Задачи можно сдавать по частям
- 3 Зачёт по домашней работе ставят после того, как приняты все задачи

А чтобы улучшить практические навыки программирования, выполните **задания в тренажёре** в личном кабинете



Задавайте вопросы и пишите отзыв о лекции

Елена Никитина
Руководитель проектов ООО «Аналитические программные решения»

