

# Лекция 4. Коллекции: словари и множества

Словари, их назначение и операции над ними

Теория множеств

Множества в python, их назначение и операции над ними

Создание вложенных структур данных



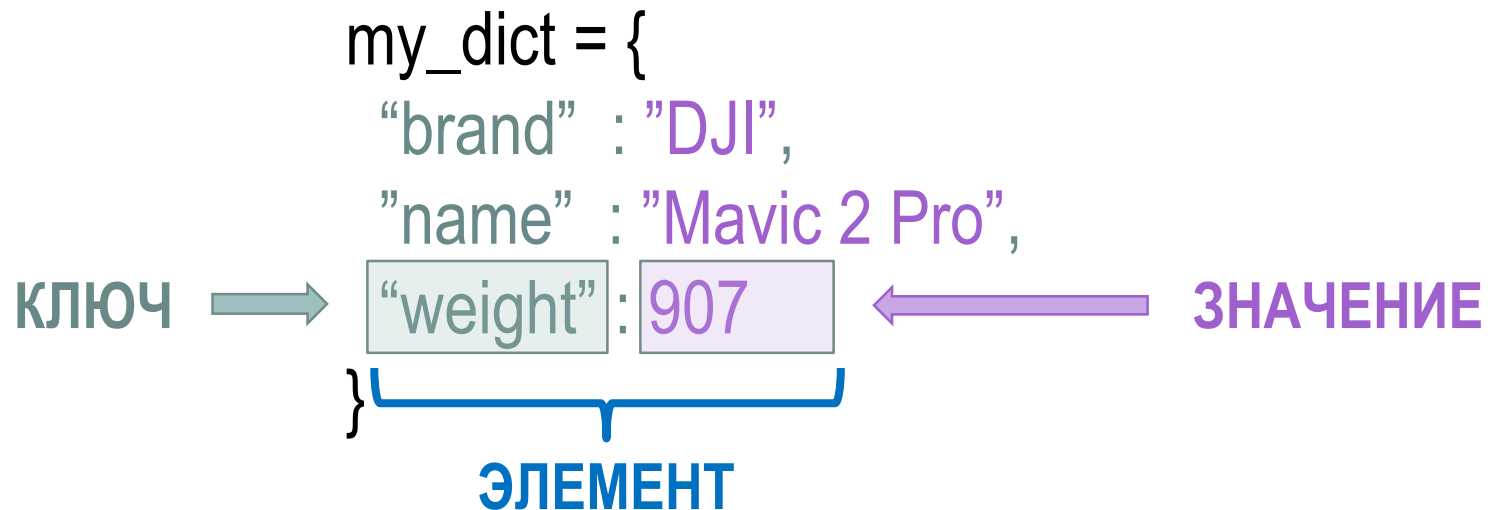


# Словари

Словари и их назначение. Правила для словарей. Операции над словарями. Поиск в словаре: по ключам и по значениям

# Словари

Словари (dictionaries) — неупорядоченная коллекция произвольных объектов с доступом по ключу



# Правила для словарей

- ❑ Элемент словаря состоит из пары **ключ : значение**. Двоеточие между ними обязательно
- ❑ Элементы разделяются запятыми
- ❑ **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



# Создание словаря

## Создание пустого словаря

```
my_dict = dict() → {}
```

```
my_dict = {} → {}
```

## Создание непустого словаря

вручную:

```
my_dict = {"brand": "DJI", "name": "Mavic 2 Pro",  
"weight": 907}
```

из другого типа данных\*:

```
my_dict = dict(("brand", "DJI"), ("name", "Mavic 2  
Pro"), ("weight", 907))
```

\*вряд ли пригодится



# Ключи словаря

Ключи словаря – неизменяемые типы данных. Если нужен ключ из нескольких элементов – берите кортеж

Пример: как оформить данные о человеке?

ФИО: Иванов Павел Андреевич

Паспорт: 1111 101202

Дата рождения: 01.01.1980

Адрес проживания: 11402, г. Москва, ул. Кетчерская, д.16



# Ключи словаря (продолжение)

Простой (и совсем плохой) вариант:

```
my_dict = {"Иванов":  
           ["Павел", "Андреевич", "1111 101202", "01.01.1980", "11402, г.  
Москва, ул. Кетчерская, д.16"]}
```

Хороший вариант:

```
my_dict = {"last_name": "Иванов", "name": "Павел", "surname": "Андреевич",  
           "passport": "1111 101202", "date_of_birth": "01.01.1980", "address": "11402, г.  
Москва, ул. Кетчерская, д.16"}
```



# Ключи словаря (продолжение)

Хороший вариант промежуточных данных для анализа:

```
my_dict = {("Иванов", "Павел", "Андреевич"):  
            {"passport": "1111 101202", "date_of_birth": "01.01.1980",  
             "address": "11402, г. Москва, ул. Кетчерская, д.16"}}
```

И еще один:

```
my_dict = {("Иванов", "Павел", "Андреевич"):  
            ["1111 101202", "01.01.1980", "11402, г. Москва, ул.  
            Кетчерская, д.16"]}
```





# Получение данных из словаря

```
my_dict = {"brand": "DJI", "name": "Mavic 2 Pro", "weight": 907}
```

Получить **значение** по ключу:

```
my_dict["brand"] → "DJI"
```

Получить **все ключи**:

```
my_dict.keys() → ["brand", "name", "weight"]
```

Получить **все значения**:

```
my_dict.values() → ["DJI", "Mavic 2 Pro", 907]
```



# Операции со словарями

|   |   |  |
|---|---|--|
| <b>dict.fromkeys(keys)</b><br><b>dict.fromkeys(keys, value)</b> | <b>Создает словарь из списка ключей</b><br>если задано value, то оно<br>присваивается всем ключам. Если не<br>задано, то присваивается None | <code>my_dict.fromkeys(["key1", "key2"]) →</code><br><code>{"key1": None, "key2": None}</code><br><code>my_dict.fromkeys(["key1", "key2"], 99) →</code><br><code>{"key1": 99, "key2": 99}</code> |
| <b>dict[key]</b>  | <b>Возвращает значение по ключу</b><br>если такого ключа нет, код упадет с<br>ошибкой   | <code>my_dict = {"key1": 1, "key2": 2}</code><br><code>my_dict["key1"] → 1</code><br><code>my_dict["key3"] → ошибка KeyError</code>  |
| <b>dict.get(key)</b>  | <b>Безопасно возвращает значение<br/>по ключу</b><br>если ключа нет, возвращает None  | <code>my_dict = {"key1": 1, "key2": 2}</code><br><code>my_dict.get("key1") → 1</code><br><code>my_dict.get("key3") → None</code>   |



# Операции со словарями (продолжение)

|                                    |  |   |
|------------------------------------|--|---|
| <b>dict.setdefault(key, value)</b> | <b>Создает ключ и значение, если такого ключа еще нет в словаре</b><br>в противном случае ничего не делает. Если не указать value, присвоит None | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.setdefault("key3", 99) →</code><br><code>{"key1":1,"key2":2,"key3":99}</code> |
| <b>dict.keys()</b>                 | <b>Возвращает все ключи словаря</b>  | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.keys() → dict_keys(["key1", "key2"])</code>                                   |
| <b>dict.values()</b>               | <b>Возвращает все значения словаря</b>   | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.values() → dict_values([1,2])</code>  |
| <b>dict.items()</b>                | <b>Возвращает пары ключ-значение</b>   | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.items() →</code><br><code>dict_items([("key1", 1), ("key2", 2)])</code>       |



## Операции со словарями (продолжение)

|                           |  |   |
|---------------------------|--|---|
| <b>dict.pop(key)</b>      | Удаляет ключ и возвращает значение   | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.pop("key1") → 1, {"key2":2}</code>                                  |
| <b>del dict[key]</b>      | Удаляет элемент из словаря по ключу  | <code>my_dict = {"key1":1,"key2":2}</code><br><code>del my_dict["key1"] → {"key2":2}</code>                                     |
| <b>dict.update(dict2)</b> | Добавляет второй словарь в первый<br>существующие ключи перезаписываются                       | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.update({"key1":99,"key3":3}) → {"key1":99,"key2":2,"key3":3}</code> |
| <b>dict.copy()</b>        | Возвращает копию словаря<br>создает точную копию словаря, никак не связанную с первым словарем | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict2 = my_dict.copy() → my_dict2 = {"key1":1,"key2":2}</code>           |
| <b>dict.clear()</b>       | Очищает словарь  | <code>my_dict = {"key1":1,"key2":2}</code><br><code>my_dict.clear() → {}</code>   |





# Множества

Теория множеств. Множества в python, их назначение и операции над ними

# Множество

**Множество** — это неупорядоченная совокупность произвольных уникальных элементов. Множества позволяют очень быстро (за секунды или доли секунды) сравнивать между собой огромные наборы элементов

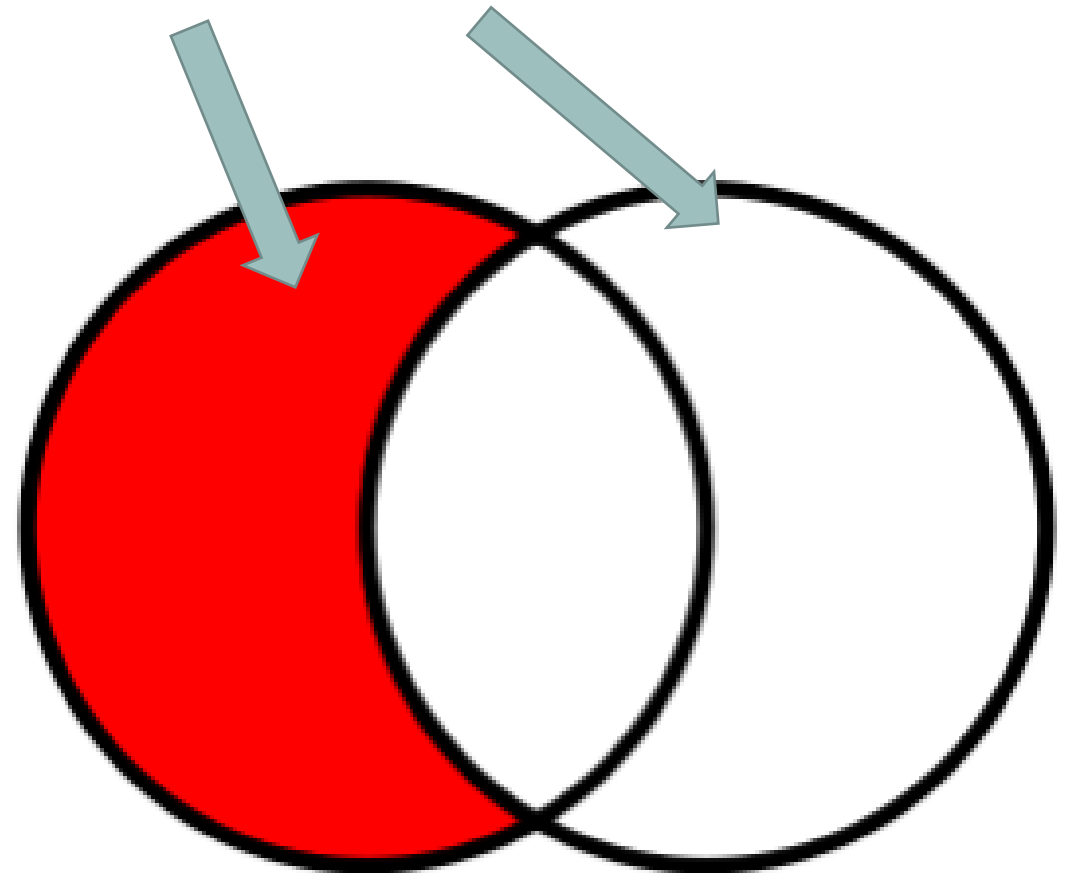
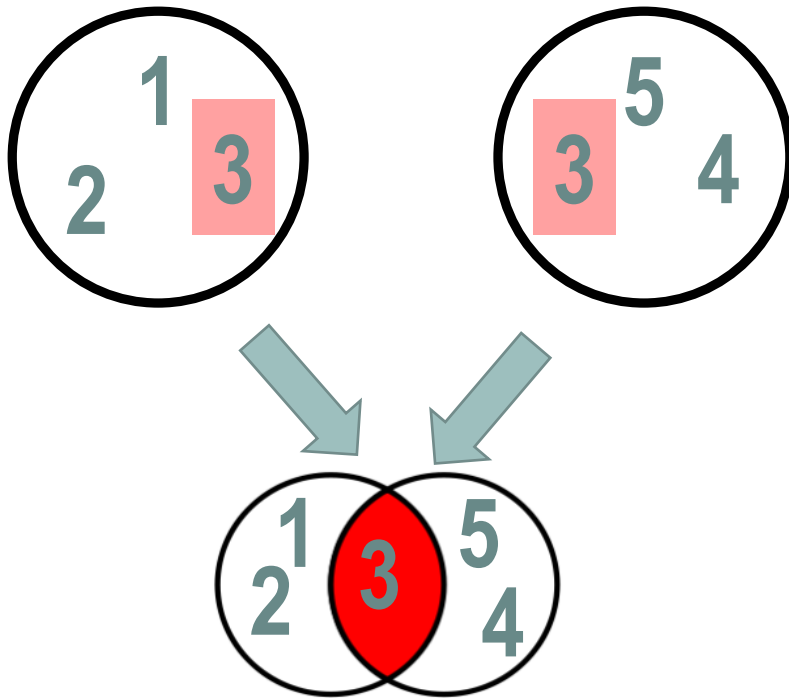
Правила для множеств:

- ❑ Элементы разделяются запятыми
- ❑ **Порядок** элементов в множестве **не соблюдается**. Получить значение по индексу (как в списке) — нельзя...
- ❑ ... но множество можно преобразовать в список и тогда - можно
- ❑ Элементы в множестве должны быть **уникальными** (не могут повторяться)
- ❑ Элементы множества реализуют **теорию множеств**

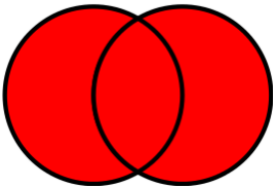
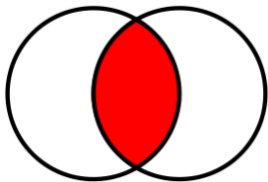
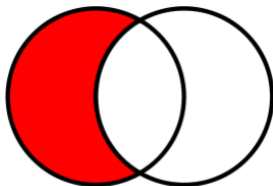
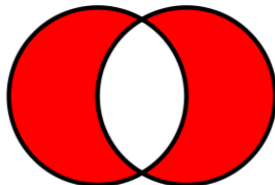


# Теория множеств

КРУГИ ЭЙЛЕРА (ДИАГРАММЫ ВЕННА)



# Теория множеств (продолжение)

|   |   |  |
|---|---|--|
| Объединение(union)<br>Логическое “ИЛИ” (or)       |    | $\text{set\_a} = \{1, 2, 3\}$<br>$\text{set\_b} = \{3, 4, 5\}$<br>$\text{set\_c} = \text{set\_a} \mid \text{set\_b} \rightarrow \{1, 2, 3, 4, 5\}$ |
| Пересечение(intersection)<br>Логическое “И” (and) |    | $\text{set\_c} = \text{set\_a} \& \text{set\_b} \rightarrow \{3\}$   |
| Разность (difference)                             |   | $\text{set\_c} = \text{set\_a} - \text{set\_b} \rightarrow \{1, 2\}$<br>$\text{set\_c} = \text{set\_b} - \text{set\_a} \rightarrow \{4, 5\}$       |
| Симметрическая разность<br>(symmetric_difference) |  | $\text{set\_c} = \text{set\_a} \wedge \text{set\_b} \rightarrow \{1, 2, 4, 5\}$  |





# Визуализация операций над множествами

Перед вами классическая иллюстрация **невозможности** получить все и сразу, выраженная через **круги Эйлера** 😊

Например, **быструю** и **качественную** работу может сделать только профессионал – но профессионал стоит **дорого**.

Специалист **высокого класса**, пообещавший вам сделать работу **со скидкой**, будет делать ее в последнюю очередь – а значит, **долго**.

**Быстро** и **дешево** можно сделать только в спешке и экономя на всем подряд – то есть не очень хорошо или вовсе **криво**.



# Создание множества

## Создание пустого множества

```
my_set = set() → {}
```

## Создание непустого множества

вручную:

```
my_set = {"Mavic 2 Pro", "Phantom  
4", "Mini 2"}
```

## Создание непустого множества

из списка:

```
my_set = set(["Mavic 2 Pro", "Phantom 4", "Mini 2"])  
→ {"Mavic 2 Pro", "Phantom 4", "Mini 2"}
```

из словаря:

```
my_set = set({'brand': 'DJI', 'name': 'Mavic 2 Pro',  
'weight': 907}) → {"brand", "name", "weight"}
```

из строки:

```
my_set({"Мама"}) → {"М", "а", "м"}
```



# Операции с множествами

|                   |  |   |
|-------------------|--|---|
| <b>len(set)</b>   | Подсчитывает количество элементов в множестве                              | <code>len({1, 2}) → 2</code>  |
| <b>X in set</b>   | Проверяет, есть ли элемент X в множестве                                   | <code>1 in {1, 2} → True</code><br><code>1 in {"1", "2"} → False</code>                                       |
| <b>set.copy()</b> | Создает копию множества чтобы не испортить исходное множество              | <code>set1 = {1, 2}</code><br><code>set2 = set1.copy() → set2 = {1, 2}, но set1 != set2</code>                |
| <b>set.add(X)</b> | Добавляет элемент X в множество<br>если этого элемента в множестве еще нет | <code>my_set = {1, 2}</code><br><code>my_set.add(1) → {1, 2}</code><br><code>my_set.add(3) → {1, 2, 3}</code> |



# Операции с множествами (продолжение)

|                       |  |  |
|-----------------------|--|--|
| <b>set.remove(X)</b>  | <b>Удаляет элемент X из множества</b><br>если этого элемента в множестве нет, код упадет с ошибкой <code>KeyError</code> | <code>my_set = {1, 2}</code><br><code>my_set.remove(1) → {2}</code><br><code>my_set.remove(3) → ошибка KeyError</code> |
| <b>set.discard(X)</b> | <b>Безопасно удаляет элемент X из множества</b><br>если элемента нет, ничего не происходит                               | <code>my_set = {1, 2}</code><br><code>my_set.discard(1) → {2}</code><br><code>my_set.discard(3) → {2}</code>           |
| <b>set.clear()</b>    | <b>Очищает содержимое множества</b>  | <code>my_set = {1, 2}</code><br><code>my_set.clear() → {}</code>   |



# Операции с множествами (сравнение)

|                              |  |  |
|------------------------------|--|--|
| <b>set1 == set2</b>          | <b>Сравнение двух множеств</b><br>если множества полностью совпадают по составу элементов, возвращает True, иначе - False  | set1 = {1, 2}<br>set1 == {1, 2, 3} → False<br>set1 == {1, 2} → True                |
| <b>set1.issubset(set2)</b>   | <b>Является ли set1 подмножеством set2?</b><br>set1 либо входит в set2, либо они идентичны (то есть выполняется ==)        | set1 = {1, 2}<br>set1.issubset({1, 2, 3}) → True<br>set1.issubset({1, 2}) → True   |
| <b>set1.issuperset(set2)</b> | <b>Является ли set2 подмножеством set1?</b><br>set1 либо включает в себя set2, либо они идентичны (то есть выполняется ==) | set1 = {1, 2}<br>set1.issuperset({1, 2, 3}) → False<br>set1.issuperset({1}) → True |



# Операции с множествами (объединение)

|  |  |   |
|--|--|---|
| <b>set1.union(set2, ...)</b><br>или<br><b>set1   set2   ..</b> | <b>Объединение нескольких множеств</b><br>объединяет set1 и любое количество множеств в скобках. Также можно использовать операцию  . В результате создается новое множество | set1 = {1, 2}<br>set1.union({4, 5}, {3}) → {1,2,3,4,5}<br>set1   {4, 5}   {3} → {1,2,3,4,5}                           |
| <b>set1.update(set2, ...)</b><br>или <b>set1  = set2   ..</b>  | <b>Объединение множеств в set1</b><br>записывает объединение сразу в set1  | set1 = {1, 2}<br>set1.update({4, 5}, {3}) →<br>set1 = {1,2,3,4,5}<br>или<br>set1  = {4, 5}   {3} → set1 = {1,2,3,4,5} |



# Операции с множествами (пересечение)

|  |   |   |
|--|---|---|
| <b>set1.intersection(set2, ...)</b><br>или<br><b>set1 &amp; set2 &amp; ..</b>      | <b>Пересечение нескольких множеств</b><br>делает пересечение set1 и любого количества множеств в скобках. Также можно использовать операцию <b>&amp;</b> . В результате создается новое множество, содержащее элементы, которые совпали во всех без исключения множествах. Если таких совпадений нет, создается пустое множество {} | set1 = {1, 2}<br>set1.intersection({1, 5}, {3}) → {}<br>set1.intersection({1, 5}, {1, 3}) → {1}<br>set1 & {1, 5} & {1, 3} → {1} |
| <b>set1.intersection_update(set2, ...)</b> или<br><b>set1 &amp;= set2 &amp; ..</b> | <b>Пересечение множеств в set1</b><br>записывает пересечение сразу в set1   | set1 = {1, 2}<br>set1.intersection_update({1, 5}, {1, 3})<br>→ set1 = {1}<br>или<br>set1 &= {1, 5} & {1, 3} → set1 = {1}        |



# Операции с множествами (разность)

|  |   |  |
|--|---|--|
| <b>set1.difference(set2, ..)</b><br>или <b>set1 – set2 - ...</b>                     | <b>Разность множеств</b><br>вычитает из set1 все элементы,<br>присутствующие в множествах set2, ..<br>Создает новое множество | set1 = {1, 2, 3, 4, 5}<br>set1.difference({1, 5}, {3}) → {2, 4}<br>{6, 7} – set1 → {6, 7}                                |
| <b>set1.difference_update</b><br><b>(set2, ...)</b> или<br><b>set1 -= set2 - ...</b> | <b>Разность множеств в set1</b><br>записывает разность сразу в set1   | set1 = {1, 2, 3, 4, 5}<br>set1.difference_update({1, 5}, {3}) →<br>set1 = {2, 4}<br>или<br>set1 -= {1, 5} - {3} → {2, 4} |





# Операции с множествами (симметрическая разность)

|  |  |  |
|--|--|--|
| <b>set1.symmetric_difference</b><br>(set2, ..) или<br><b>set1 ^ set2 ^ ...</b>               | <b>Симметрическая разность множеств</b><br>оставляет в set1 и остальных множествах<br>set2, .. только те элементы, которые<br>уникальны. Создает новое множество | set1 = {1, 2}<br>set1.symmetric_difference({3, 2}) →<br>{1, 3}   |
| <b>set1.<br/>symmetric_difference_update</b><br>(set2, ...) или<br><b>set1 ^= set2 ^ ...</b> | <b>Симметрическая разность множеств в<br/>set1</b><br>записывает симметрическую разность<br>сразу в set1   | set1 = {1, 2}<br>set1.symmetric_difference_update<br>({3, 2}) → set1 = {1, 3}<br>или<br>set1 ^= {3, 2} → set1 = {1, 3} |



# Статичное множество

Статичное множество (`frozenset`) — это такое же множество, в котором нельзя изменять элементы. Все остальные операции над ними работают как в обычных множествах.

Статичные множества — это то же самое, что кортежи для списков

