Шпаргалка: словари и коллекции

Кратко

Словари: основное

Что	Как	О чём помнить

Создать словарь	Схема:
	`<переменная> = {`
	`  <ключ>:<значение>,`
	`  <ключ>: <значение>`
	`}`
	
	Пример:
	`database = {`
	`  'host': '192.168.0.10',`
	`  'port': 5432,`
	`  'user': 'Shrek',`
	`  'password': 'My$wamp235'`
	`}`
		Словарь — это набор парных значений. Каждый элемент словаря состоит из двух частей: **ключ** и **значение**. Между ними стоит двоеточие.
		Все элементы словаря объединяют фигурными скобками.
		Ключи — уникальны, значения — нет. В словаре не может быть двух одинаковых ключей.

Получить значение по ключу	Схема:
	`<название словаря>[<ключ>]`
	Пример:
	`print(shopping_list['мука'])`	

	Метод `get()`
	Схема:
	`print(<имя словаря>.get(<ключ>, <текст ошибки>))`
	Пример:
	`print(backpack.get('Среднее отделение', 'Такого кармана нет'))`
		Лучше использовать метод, а не схему выше. Он сообщит, если не найдёт нужный ключ. Вторым аргументом можно передать текст сообщения об ошибке. Метод выведет его, если не найдёт ключ.

Изменить значение по ключу	**Схема:**
	`<название словаря>[<ключ>] = <значение>`
	Пример:
	`shopping_list['мука'] = 'agony'`
		Команда `название_словаря['ключ'] = 'значение'` работает по-разному. Если ключ существует, она поменяет его значение. Если такого ключа нет, она добавит его с указанным значением.

Добавить один элемент	**Схема:**
	`<название словаря>[<ключ>] = <значение>`
	Пример:
	`shopping_list['клубника'] = 'strawberries'`
		Ключ обязательно должен быть новым. Иначе команда не добавит элемент, а перезапишет значение ключа.
		Если добавить в словарь несколько элементов с одинаковыми ключами, но разными значениями — в словаре появится только один элемент с этим ключом. Ему присвоится последнее из добавленных значений.

Добавить несколько элементов	Метод `update()`, объединение словарей
	Схема:
	`<имя словаря>.update(<имя второго словаря>)`
	Пример:
	`shopping_list.update(additional_products)`
		Второй словарь не поменяется. Изменения произойдут только со словарём, к которому добавили значения.

Извлечь все ключи	Метод `keys()`
	Схема:
	`print(<имя словаря>.keys())`
	Пример:
	`print(shopping_list.keys())`
		Метод выведет все ключи словаря как одну коллекцию. Вот так: `dict_keys(['молоко', 'сахар', 'мука', 'яйца', 'разрыхлитель', 'ваниль'])`. Тут `dict_keys()` — это коллекция, её можно преобразовать в обычный список.

Извлечь все значения	Метод `values()` **Схема:** `print(<имя словаря>.values())` **Пример:** `print(shopping_list.values())`	Тоже соберёт значения в одной коллекции: `dict_values(['milk', 'sugar', 'flour', 'eggs', 'leaven', 'vanilla']`. Коллекция будет называться `dict_values()`.

Сделать из коллекции список	Функция `list()`
	Схема:
	`<новая переменная> = list(<имя словаря>.keys())`
	Пример:
	`shopping_list_ru = list(shopping_list.keys())`
		Так можно сделать с коллекциями, которые получились в результате методов `keys()` и `values()`.

Преобразовать словарь в список	Функция `list()`
	Схема:
	`list(<имя словаря>))`
	Пример:
	`list(shopping_list))`
		Список, который получится в результате, будет включать только **ключи** словаря — без значений

Перебрать словарь в цикле `for`

Что	Как	О чём помнить

Перебрать в цикле и ключи, и значения	Метод `items()`
	Схема:
	`for key, value in <имя словаря>.items():`
	`  <тело цикла>`
	Пример:
	`for key, value in backpack.items():`
	`  print('В ' + key + ' лежит ' + value)`
		Метод извлекает из каждого элемента словаря ключ и значение и передаёт их в переменные.
		
		Переменные обычно называют `key` (для ключей) и `value` (для значений).

Перебрать в цикле только ключи	Метод `keys()`
	Схема:
	`for <переменная> in <имя словаря>.keys():`
	`  <тело цикла>`
	Пример:
	`for pocket in backpack.keys():`
	`  print('В моём рюкзаке есть ' + pocket)`
		Кроме метода `keys()`, есть ещё один способ. Если в условии цикла после названия словаря не указывать метод, цикл будет перебирать ключи словаря: `for <переменная> in <словарь>`.

Перебрать только значения	Метод `values()`
	Схема:
	`for <переменная> in <имя словаря>.values():`
	`  <тело цикла>`
	Пример:
	`for item in backpack.values():`
	`  print('Я взял с собой ' + item)`	

Проверить, есть ли элемент в словаре

Что	Как	О чём помнить

Искать среди ключей	Конструкция `if-in`
	Схема:
	`if <ключ> in <словарь>:`
	`  print(<текст>)`
	`else:`
	`  print(<текст>)`
	Пример:
	`if 'разрыхлитель' in shopping_list:`
	`  print('В словаре: нашлось!')`
	`else:`
	`  print('В словаре: не нашлось :(')`
		Работает только для ключей. Значение не найдёт, даже если оно будет в словаре.

Искать и среди ключей, и среди значений	Конструкция `if-in`, оператор `or`, методы `keys()` и `values()`
	Схема:
	`if <элемент> in <словарь>.keys() or <элемент> in <словарь>.values:`
	`  print(<текст>)`
	`else:`
	`  print(<текст>)`
	Пример:
	`if 'leaven' in shopping_list.keys() or 'leaven' in shopping_list.values():`
	`  print('В словаре: нашлось!')`
	`else:`
	`  print('В словаре: не нашлось :(')`	

Проверить, что элемента нет	Конструкция `if-in`, оператор `not`
	Схема:
	`if <ключ> not in <словарь>:`
	`  print(<текст>)`
	Пример:
	`if 'клубника' not in shopping_list:`
	`  print('Не хватает клубники для украшения панкейков!')`
		Тоже обычно работает только для ключей. Чтобы распространить поиск на значения, нужно использовать методы `keys()` и `values()` + оператор `or`.

Подробно и с примерами

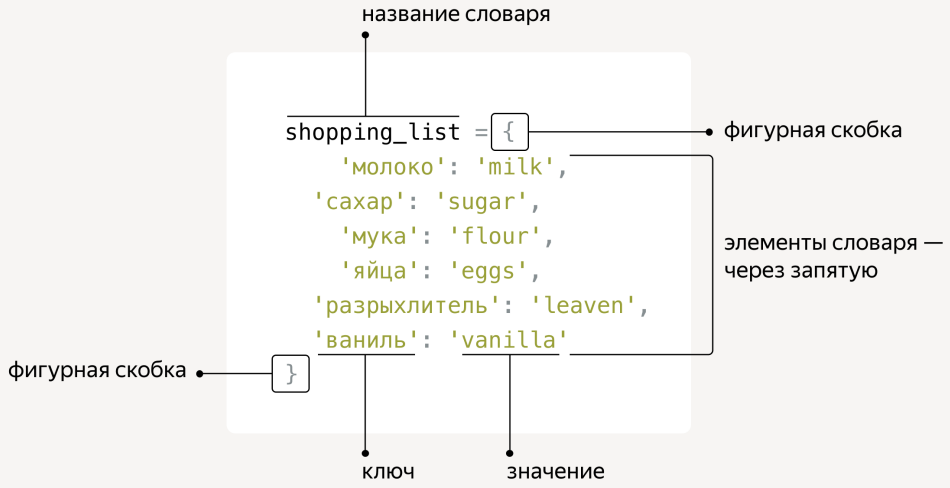
Что такое словарь

Словарь — это набор парных значений. Каждый элемент словаря состоит из двух частей: первая — **ключ**, вторая — **значение**. Их разделяют двоеточием. Например, есть элемент `'молоко': 'milk'`. В этом элементе ключ — `'молоко'`, а значение — `'milk'`.

Все элементы словаря объединяют фигурными скобками.

Так выглядит словарь целиком:

```
shopping_list = {
    'молоко': 'milk', # Первый элемент словаря (в каждом элементе две части!)
    'сахар': 'sugar',  # Второй элемент словаря
    'мука': 'flour',  # Третий элемент
    'яйца': 'eggs',  # Четвёртый элемент
    'разрыхлитель': 'leaven',  # Пятый элемент
    'ваниль': 'vanilla'
}
```



Ключи — уникальны, значения — нет

Ключами словаря могут быть:

- числа,
- строки,
- булевы значения `True` и `False`.

При этом в словаре **не может быть двух одинаковых ключей**.

🔑 **Ключ** — это уникальный адрес, по которому можно найти значение, поэтому он **не может** повторяться.

🔒 **Значением** может быть что угодно: числа, строки, списки и даже другие словари. Значения не обязательно уникальны, они могут повторяться.

Пример. Вот словарь с ключами и значениями разных видов — числами, строками, булевыми значениями, списками, словарями:

```
dump = {
    1: 'единица',               # Ключ — число, значение — строка.
    'земляника': 'ягода',       # И ключ, и значение — строки.
    'помидор': 'ягода',         # Значение 'ягода' — не уникально. Так можно.
    False: 0,                   # Ключ — булево значение, значение — число.
    'лук': ['овощ', 'оружие'],  # Ключ — строка, значение — список.
                                # Ключ — строка, а значение — словарь. Так тоже можно!
    'англо-русский словарь': {'молоко': 'milk',
                              'сахар': 'sugar',
                              'мука': 'flour'
                               },
}

print(dump)
```

Что такое коллекция

Списки и словари относят к одному виду конструкций — **коллекциям**. Как в реальности бывают коллекции монет или картин, так и в Python можно хранить вместе разные значения.

📎 Коллекция — это набор значений, которые хранятся вместе и к которым можно применять специальные функции.

Как получить значение по ключу

Чтобы получить значение из словаря в Python, нужно указать его ключ. Это ещё называют **получить доступ по ключу**.

Доступ по ключу похож на доступ по индексу в списках: нужно написать название словаря, а в квадратных скобках — нужный ключ.

Пример. Получить значение по ключу `'мука'` можно так:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

print(shopping_list['мука'])
# Будет напечатано: flour
```

Как изменить значение по ключу

В любом элементе словаря можно заменить существующее значение на новое. Для этого нужно обратиться к элементу по ключу и присвоить ему новое значение.

Пример. Нужно заменить слово «мука́» на «муќа». В исходном словаре ключу `'мука'` соответствует значение `'flour'` — можно заменить его на `'agony'`.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

# Элементу с ключом 'мука' присвоили новое значение
shopping_list['мука'] = 'agony'

print(shopping_list['мука'])
# Будет напечатано: agony
```

Добавить один элемент

Ты уже знаешь, как изменить значение по ключу: `<название словаря>['ключ'] = 'значение'`. Чтобы добавить элемент, нужно написать такую же конструкцию, только с новым ключом и значением.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

# Создаём новый элемент словаря через доступ по ключу
shopping_list['клубника'] = 'strawberries'
```

Команда `название_словаря['ключ'] = 'значение'` работает по-разному в зависимости от того, существует ли такой ключ.

📌 Если ключ уже существует в словаре, эта команда заменит его значение. Если такого ключа ещё нет — добавится новый элемент.

Если добавить в словарь несколько элементов с одинаковыми ключами, но разными значениями — в словаре появится только один элемент с этим ключом. Ему присвоится последнее из добавленных значений.
Пример:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

# Создаём несколько новых элементов словаря с одинаковым ключом
shopping_list['лук'] = 'onion'  # Будет создан элемент с ключом 'лук'.
shopping_list['лук'] = 'bow'  # Значение под ключом 'лук' будет заменено.

print(shopping_list)
```

```
# Будет напечатано: {'молоко': 'milk', 'сахар': 'sugar', 'мука': 'flour', 'яйца': 'eggs',
'разрыхлитель': 'leaven', 'ваниль': 'vanilla', 'лук': 'bow'}
```

Добавить сразу несколько элементов

Как и у списков, у словарей есть много встроенных методов. Один из них — `update()`. Он позволяет объединить два словаря: добавить в один словарь элементы другого.

Пример. Есть новый словарь `additional_products`. В нём хранятся названия продуктов, которые нужно включить в `shopping_list`.

```
additional_products = {'клубника': 'strawberries', 'малина': 'raspberries'}
```

Можно добавить словарь `additional_products` к словарю `shopping_list`:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

# новый словарь
additional_products = {'клубника': 'strawberries', 'малина': 'raspberries'}

# Добавим в словарь shopping_list элементы словаря additional_products
shopping_list.update(additional_products)

# Посмотрим, что теперь хранится в словаре shopping_list
print(shopping_list)

# Заодно выясним, что произошло со словарём additional_products
print(additional_products)
```

В словаре `shopping_list` появились новые элементы, а словарь `additional_products` остался таким, как был.

Метод `get()`

Как ты знаешь, чтобы получить значение словаря по ключу, нужно написать название словаря, а в квадратных скобках — нужный ключ. Вот так:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}
print(backpack['Большое отделение'])
print(backpack['Укреплённое отделение'])
```

Значения часто получают таким способом. Но у этого подхода есть минусы.

Например, если при поиске `словарь[ключ]` ты обратишься к несуществующему ключу, программа выдаст ошибку и выполнение кода прервётся:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}
print(backpack['Большое отделение'])
print(backpack['Среднее отделение'])
print(backpack['Укреплённое отделение'])

# При обращении к ключу 'Среднее отделение' программа выдаст ошибку KeyError
```

Поэтому лучше использовать метод `get()`. Как и `update()`, это уже встроенный метод для работы со словарями.

Преимущества `get()`

Плюс метода `get()` в том, что он предупредит, если ты попробуешь обратиться к несуществующему ключу.

Пример с методом `get()`:

```
backpack = {
    'Большое отделение': 'Ноутбук',
    'Боковое отделение': 'Ключи',
    'Укреплённое отделение': 'Очки',
```

```
    'Малое отделение': 'Зарядное устройство'
}

print(backpack.get('Среднее отделение'))
# Будет напечатано: None
```

По умолчанию метод `get()` возвращает `None`, если искомого ключа нет. При этом ошибка не появится и программа продолжит выполняться.

Ещё пример:

```
backpack = {
    'Большое отделение': 'Ноутбук',
    'Боковое отделение': 'Ключи',
    'Укреплённое отделение': 'Очки',
    'Малое отделение': 'Зарядное устройство'
}

print(backpack.get('Большое отделение'))
print(backpack.get('Среднее отделение'))
print(backpack.get('Укреплённое отделение'))
# Будет напечатано:
# Ноутбук
# None
# Очки
```

Чтобы метод `get()` выводил какое-нибудь сообщение об ошибке, нужно передать вторым аргументом текст ошибки.

Пример:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}

print(backpack.get('Среднее отделение'))
print(backpack.get('Карман на спине','Такого кармана нет')) # Выведет "Такого кармана нет"
```

Метод `keys()`

Например, тебе нужно напечатать все слова на русском языке из словаря `shopping_list`.

Чтобы получить все ключи, есть метод `keys()` («ключи»).

Пример. Код ниже выведет на экран все ключи словаря:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

print(shopping_list.keys())
# Будет напечатано: dict_keys(['молоко', 'сахар', 'мука', 'яйца', 'разрыхлитель', 'ваниль'])
```

Метод `values()`

Значения словаря тоже можно извлечь: для этого есть метод `values()`.
Пример. Код ниже выведет на экран все значения словаря.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

print(shopping_list.values())
# dict_values(['milk', 'sugar', 'flour', 'eggs', 'leaven', 'vanilla'])
```

Метод `values()` возвращает коллекцию типа `dict_values()`, а метод `keys()` — коллекцию типа `dict_keys()`.

`dict_values()` и `dict_keys()` — это коллекции, похожие на списки. Их можно преобразовать в списки и работать с ними. Например, если нужно извлечь значения из словаря и составить из них упорядоченный список.

Функция `list()`

Чтобы сделать из коллекции список, понадобится функция `list()`:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}
# Собираем ключи словаря в коллекцию
# и преобразуем эту коллекцию в список
shopping_list_ru = list(shopping_list.keys())

# Собираем значения словаря в коллекцию
# и преобразуем эту коллекцию в список
shopping_list_en = list(shopping_list.values())

# Печатаем списки
print(shopping_list_ru)
# Будет напечатано: ['молоко', 'сахар', 'мука', 'яйца', 'разрыхлитель', 'ваниль']

print(shopping_list_en)
# Будет напечатано: ['milk', 'sugar', 'flour', 'eggs', 'leaven', 'vanilla']
```

Преобразование словаря в список

Словарь целиком тоже можно преобразовать в список — той же функцией `list()`.

Но здесь есть особенность: список, который получится в результате, будет включать только **ключи** словаря — без значений.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

print(list(shopping_list))
# Будет напечатано: ['молоко', 'сахар', 'мука', 'яйца', 'разрыхлитель', 'ваниль']
```

Перебор словаря в цикле

Ты знаешь, как цикл работает со списками: каждый элемент списка по очереди передаётся в переменную, а значение переменной обрабатывается в теле цикла. И так до тех пор, пока цикл не переберёт все элементы списка.

Словари, как и списки, — это коллекции. Поэтому для них всё работает похоже. Отличие в том, что элементы словаря состоят из двух частей. Поэтому в цикле нужно указать, что перебрать:

- и ключи, и значения,
- только ключи,
- только значения.

Перебрать в цикле и ключи, и значения

Для словарей есть метод `items()`: он извлекает из каждого элемента словаря ключ и значение и передаёт их в переменные.

Переменные можно назвать как угодно, но обычно их называют `key` (англ. «ключ») и `value` (англ. «значение»). Так удобнее читать код.

Пример:

```
backpack = {
    'Большое отделение': 'Ноутбук',
    'Боковое отделение': 'Ключи',
    'Укреплённое отделение': 'Очки',
    'Малое отделение': 'Зарядное устройство'
}

for key, value in backpack.items():
    print('В ' + key + ' лежит ' + value)
# Будет напечатано:
# В Большое отделение лежит Ноутбук
# В Боковое отделение лежит Ключи
# В Укреплённое отделение лежит Очки
# В Малое отделение лежит Зарядное устройство
```

Перебрать отдельно ключи и значения

При обработке словаря в цикле необязательно извлекать и ключ, и значение каждого элемента. Можно перебрать только ключи или только значения.

Перебор ключей словаря в цикле ещё называют «итерировать по ключам»; перебирать значения словаря — «итерировать по значениям».

Для перебора ключей и значений понадобятся методы `keys()` и `values()`.

Если в цикле применён метод `keys()` или `values()`, то нужно объявлять только одну переменную цикла.

Пример с `keys()`:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}

# А в этом цикле извлечём и напечатаем только ключи (keys) словаря
for pocket in backpack.keys():
  print('В моём рюкзаке есть ' + pocket)

# Будет напечатано:
# В моём рюкзаке есть Большое отделение
# В моём рюкзаке есть Боковое отделение
# В моём рюкзаке есть Укреплённое отделение
# В моём рюкзаке есть Малое отделение
```

Пример с `values()`:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}

# Извлечём и напечатаем только значения (values) каждого элемента
for item in backpack.values():
  print('Я взял с собой ' + item)

# Будет напечатано:
# Я взял с собой Ноутбук
# Я взял с собой Ключи
# Я взял с собой Очки
# Я взял с собой Зарядное устройство
```

По умолчанию Python для итерации по словарю использует ключи. Поэтому можно итерироваться по ключам без метода `keys()`.

Если при объявлении цикла после названия словаря не указать метод — цикл будет перебирать ключи словаря.

Пример без `keys()` и `values()`:

```
backpack = {
    'Большое отделение' : 'Ноутбук',
    'Боковое отделение' : 'Ключи',
    'Укреплённое отделение' : 'Очки',
    'Малое отделение' : 'Зарядное устройство'
}


for pocket in backpack:  # Это то же самое, что for pocket in backpack.keys()
  print(pocket + ' - уже занят')
# Будет напечатано:
# Большое отделение - уже занят
# Боковое отделение - уже занят
# Укреплённое отделение - уже занят
# Малое отделение - уже занят
```

Как проверить наличие элементов в словаре

Это можно сделать условной конструкцией `if` с оператором `in` («внутри»).

Конструкция `if-in`

Например, вот так выглядит поиск по слову «разрыхлитель»:

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven'
    'ваниль': 'vanilla'
}
# Есть ли элемент 'разрыхлитель' в словаре shopping_list?
if 'разрыхлитель' in shopping_list:
    print('В словаре: нашлось!')
else:
```

```
    print('В словаре: не нашлось :(')
# Будет напечатано: В словаре: нашлось!
```

Такой элемент нашёлся. Теперь попробуем найти не ключ, а значение — leaven.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven'
    'ваниль': 'vanilla'
}
# Есть ли элемент 'leaven' в словаре shopping_list?
if 'leaven' in shopping_list:
    print('В словаре: нашлось!')
else:
    print('В словаре: не нашлось :(')

# Будет напечатано: В словаре: не нашлось :(
```

Поиск ничего не дал: элемент есть в словаре, но оператор его не нашёл.

Дело в том, что в работе с оператором `in` у словарей есть особенность. Этот оператор проводит поиск только по **ключам**, а в словаре `shopping_list` слово leaven — это **значение** элемента. Поэтому элемент и не нашёлся.

Поиск по значению

Чтобы проверить, есть ли элемент в словаре среди ключей или значений, можно применить методы `values()` и `keys()`.

Понадобится та же конструкция `for-in`, но к ней нужно добавить `or` («или»). Так ты говоришь программе, что нужно искать среди ключей **или** значений.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven'
    'ваниль': 'vanilla'
}
# Есть ли элемент 'leaven' среди ключей ИЛИ значений словаря shopping_list?
```

```
if 'leaven' in shopping_list.keys() or 'leaven' in shopping_list.values():
    print('В словаре: нашлось!')
else:
    print('В словаре: не нашлось :(')
# Будет напечатано: В словаре: нашлось!
```

Проверить, что элемента нет

Если нужно убедиться, что элемента **нет** в коллекции, — поможет логический оператор `not` («не»). Обрати внимание: без методов и оператора `or`, поиск будет идти только по ключам.

```
shopping_list = {
    'молоко': 'milk',
    'сахар': 'sugar',
    'мука': 'flour',
    'яйца': 'eggs',
    'разрыхлитель': 'leaven',
    'ваниль': 'vanilla'
}

if 'клубника' not in shopping_list:
    print('Не хватает клубники для украшения панкейков!')

# Будет напечатано: Не хватает клубники для украшения панкейков!
```