# Лекция: Множества (set) в Python

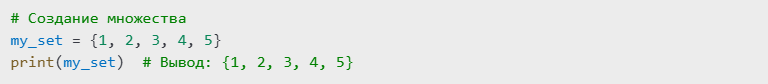
## 1. Введение в множества

Множество (set) — это неупорядоченная коллекция уникальных элементов. В Python множества реализованы как изменяемые объекты, поддерживающие различные операции, включая объединение, пересечение и разность.

## 2. Создание множеств

Создать множество можно несколькими способами:

### 2.1. Использование фигурных скобок {}



Если добавить повторяющиеся элементы, они будут автоматически удалены:



### 2.2. Использование функции set()

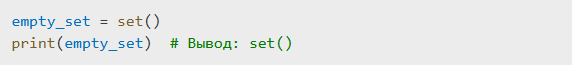


### 2.3. использование (range)



### 2.4. Пустое множество

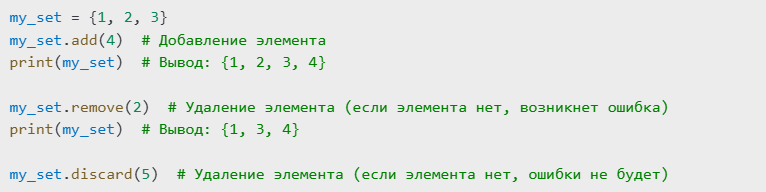
Пустое множество создается **только** через set():



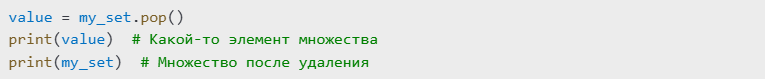
Использование {} создаст пустой **словарь**, а не множество.

## 3. Основные операции с множествами

### 3.1. Добавление и удаление элементов



Метод pop() удаляет и возвращает **случайный** элемент множества:

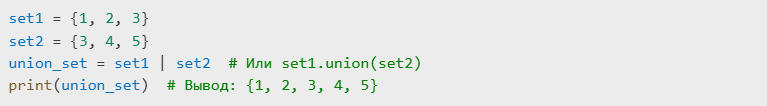


Метод clear() очищает множество:

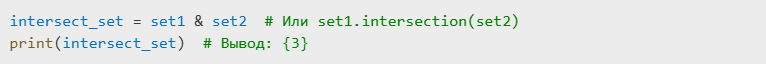


### 3.2. Операции над множествами

#### Объединение (| или union())



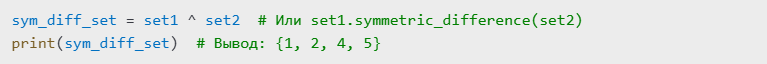
#### Пересечение (& или intersection())



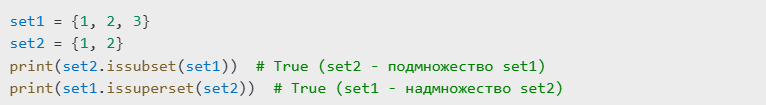
#### Разность (- или difference())



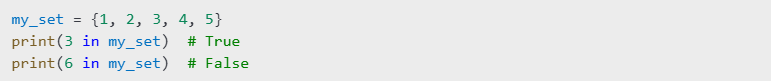
#### Симметрическая разность (^ или symmetric\_difference())



### 3.3. Проверка на подмножество и надмножество

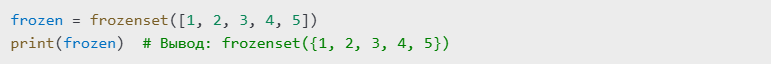


### 3.4. Проверка наличия элемента в множестве



## 4. Замороженные множества (frozenset)

Множества в Python изменяемы, но если требуется неизменяемый вариант, можно использовать frozenset():



Попытка изменения приведет к ошибке:



## 5. Итог

* Множество (set) — это **неупорядоченная** коллекция уникальных элементов.
* Элементы множества **не повторяются**.
* Поддерживаются операции объединения, пересечения, разности и другие.
* frozenset позволяет создать неизменяемое множество.
* Использование множеств позволяет эффективно работать с уникальными значениями и выполнять математические операции с наборами данных.

# Лекция: Кортежи (кортеж) на Python

## 1. Введение в кортежи

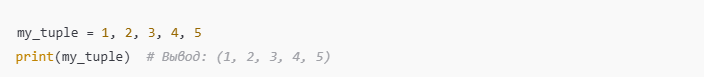
Кортеж ( tuple ) — это **неизменяемая** упорядоченная коллекция элементов. В отличие от списков, кортежи нельзя изменять после создания, что в некоторых случаях делает их более производительными и безопасными.

### Когда использовать кортежи?

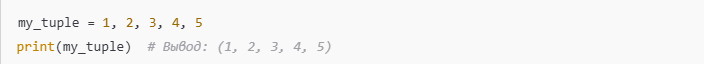
* Когда данные не должны изменяться после создания.
* Когда важна производительность (кортежи работают быстрее списков).
* В качестве ключей в словарях (если кортеж содержит только изменяемые объекты).

## 2. Создание кортежей

### 2.1. Использование круглых скобок( )

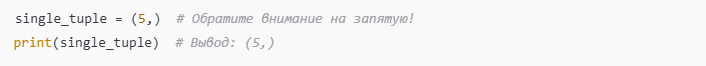


### 2.2. Создание кортежа без скобок (распаковка)



### 2.3. Кортеж из одного элемента

Для создания кортежа с одним элементом **нужно добавить запятую** :



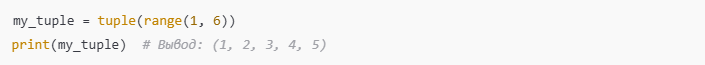
Без запятой Python воспримет ()как просто число:



### 2.4. Использование tuple()



### 2.5. Кортеж с range()



## 3. Доступ к элементам

### 3.1. Influxation

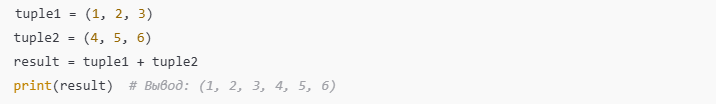


### 3.2.Срезы



## 4. Основные операции с кортежами

### 4.1. Конкатенация ( + )



### 4.2.Повторение ( \* )



### 4.3. Проверка присутствия элемента



## 5. Распаковка кортежей



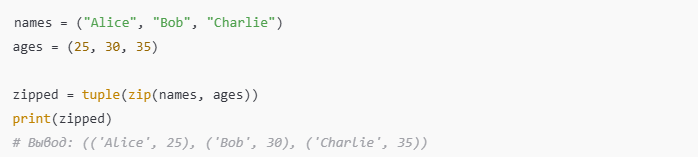
Можно использовать \* для сбора оставшихся элементов:



## 6. Кортежи и функции zip()

Функция zip() позволяет объединять несколько кортежей в непрерывном режиме.

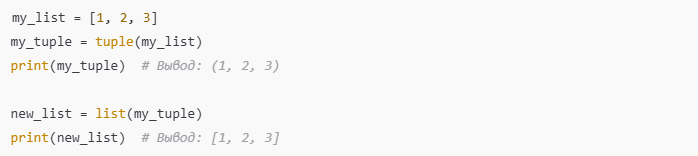
### 6.1. Создание паритетных оценок



**6.2. Распаковка zip()**



## 7. Преобразование списка в суд и обратно



## 8. Итоги

* **Кортеж** — это неизменяемая, упорядоченная коллекция.
* Используется, когда важна постоянство данных.
* Поддерживает индексацию, срезы, объединение ( +) и дублирование ( \*).
* Можно распаковывать элементы кортежа в переменные.
* оказание помощи по zip()последовательности в кортежах.

Теперь ты знаешь всё, что нужно для работы с кортежами!

## ****1. Введение в словари****

Словарь ( словарь , dict) — это **изменяемая** коллекция, которая хранит пару ключей: значение . В отличие от списков и кортежей, словари используют **хеш-таблицы** , что делает результаты поиска по ключу очень быстро.

### ****Когда использовать словари?****

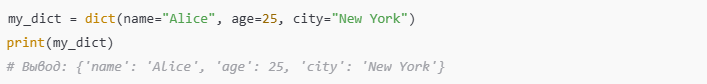
* Когда необходимо сохранить **данные в видеопаре** ( ключ – значение ).
* Когда необходим **быстрый доступ** к данным по ключу (ищите быстрее, чем в списках).
* Когда порядок элементов **не важен** (начиная с Python 3.7, порядок сохраняется).

## ****2. Создание словаря****

### ****2.1. Использование фигурных скобок****{}



### ****2.2. Использование функции**** dict()



### ****2.3. Создание словаря из списка кортежей****



**2.4. Пустой словарь**

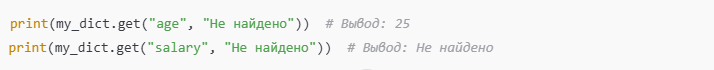


## ****3. Доступ к элементам****

### ****3.1. Доступ по ключу****

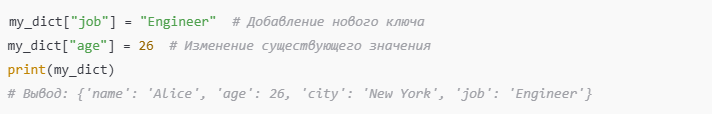


Если переключения нет, нет ошибки KeyError. Во избежание ошибок используйте .get():



## ****4. Основные операции со словарями****

### ****4.1. Добавление и изменение изменений****



### ****4.2. Удаление элементов****



Метод pop() удаления элемента и возвращает его значение:



Метод clear() полностью очищает словарь:



## ****5. Перебор словаря****

### ****5.1. Перебор ключей****



**или**



**5.2. Перебор оценок**



### ****5.3. Перебор пар (ключ, значение)****



## ****6. Проверка электронного переключателя.****



## ****7. Объединение словарей****

### ****7.1.Метод**** update()



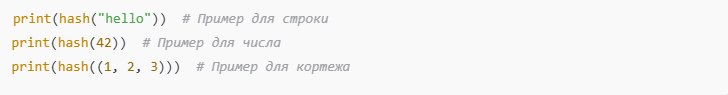
### ****7.2. Новый оператор**** | ****(Python 3.9+)****



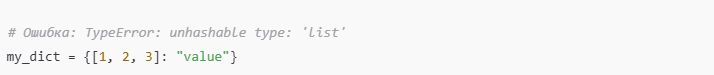
## ****Функция**** hash() ****и хешируемые ключи****

Словари в Python используют **хеш-таблицы** , поэтому ключи должны быть **хешируемыми** (то есть неизменяемыми).

### ****Проверка хешируемости с**** hash()



**Списки, расширения и другие изменяемые объекты нельзя использовать в качестве ключей** . Например, код ниже вызывает ошибку:



Но можно использовать кортеж:



## ****9. Использование**** zip() ****для создания словаря****

Функция zip() позволяет объединить два списка в словаре:



## ****10. Итог****

* **Словарь (** dict**)** — это изменяемая коллекция пар ключ: значение .
* Доступ к элементам **быстрее** , чем в списках.
* Ключи должны быть **хешируемыми** (неизменяемыми).
* В результате hash() теперь можно использовать объект в качестве ключа.
* Функция zip() помогает создать словарь из двух списков.

Теперь ты знаешь, как работать со словарями на Python!