

11. Работа с кортежами

- Кортежи
- Функции и методы кортежей

#ShoXCode

Кортежи очень похожи на списки, но имеют одно важное отличие – они *неизменяемые*. В остальном, они также могут состоять из данных разных типов и использовать индексы, которые определяют конкретный порядок элементов.

Индекс начинается с 0, как и в случае списков, а отрицательный индекс – с -1. Этот индекс указывает на последний элемент кортежа.

Они полезны в тех случаях, когда необходимо передать данные, не позволяя изменять их. Эти данные могут быть использованы, но в оригинальной структуре изменения не отобразятся

Кортежи

Эта структура данных используется для хранения последовательности **упорядоченных** и **неизменяемых элементов**.

Кортеж можно записать как набор значений, разделенных запятыми , заключенных в круглые скобки (). Круглые скобки необязательны, но их использование является хорошей практикой.

```
var = ('t', 'o', 'm', 'a', 't', 'o')
print(f'{var}\nТип переменной var - {type(var)}')
```

Рисунок.1 – Создание кортежа

```
('t', 'o', 'm', 'a', 't', 'o')
Тип переменной var - <class 'tuple'>
Process finished with exit code 0
```

Рисунок.1.1 – Результат Рис.1

Создание кортежа с одним элементом немного отличается. Нам нужно поставить запятую после элемента, чтобы объявить кортеж

```
var_1 = ("Python")
# Создание кортежа с одним элементом
var_2 = ("Python", )
print(f"Тип переменной var_1 - {type(var_1)}\nТип переменной var_2 - {type(var_2)}")
```

Рисунок.1 – Создание кортежа с одним элементом

```
Тип переменной var_1 - <class 'str'>
Тип переменной var_2 - <class 'tuple'>
Process finished with exit code 0
```

Рисунок.2.1 – Результат Рис.2

Кортежи могут включать однородные и разнородные значения

```
mixed_type = ("Лондон", "Москва", 21, False, 6.5)

for item in mixed_type:
    print(f"{item} - {type(item)}")
```

Рисунок.3 – Создание кортежа из разных типов данных

```
Лондон - <class 'str'>
Москва - <class 'str'>
21 - <class 'int'>
False - <class 'bool'>
6.5 - <class 'float'>

Process finished with exit code 0
```

Рисунок.3.1 – Результат Рис.3

После создания кортежа вы не можете вносить изменения в него.

```
>>> vegetables = ("помидор", "огурец", "картофель")
>>> vegetables[1] = "морковка"
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Рисунок.4 – Кортеж – неизменяемый

Индексирование и нарезка в кортеже

Индексация и нарезка в кортеже аналогичны списку. Индексация в кортеже начинается с 0 и продолжается до `len(tuple) - 1`

Доступ к элементам кортежа можно получить с помощью оператора `[]`. Python также позволяет использовать оператор *двоеточия* Рис.6 для доступа к нескольким элементам в кортеже

```
nums = (10, 20, 30, 40, 50, 60, 70)
print(nums[0])
print(nums[1])
print(nums[2])
# это даст IndexError
print(nums[8])
```

Рисунок.5 – Индексирование и нарезка в кортеже

```
print(nums[8])
IndexError: tuple index out of range
10
20
30

Process finished with exit code 1
```

Рисунок.5.1 – Результат Рис.5

```
nums = (10, 20, 30, 40, 50, 60, 70)
# от 1-го элемента до конца
print(nums[1:])
# от 0-го элемента до 3-го включительно
print(nums[:4])
# от 1-го элемента до 4-го включительно
print(nums[1:5])
# от 0-го до 6-го элемента с шагом 2
print(nums[0:6:2])
```

Рисунок.6 – Индексирование и нарезка в кортеже

```
(20, 30, 40, 50, 60, 70)
(10, 20, 30, 40)
(20, 30, 40, 50)
(10, 30, 50)
Process finished with exit code 0
```

Рисунок.6.1 – Результат Рис.6

Отрицательная индексация в кортеже

Доступ к элементу кортежа также может осуществляться с помощью отрицательной индексации. Индекс `-1` обозначает самый правый элемент, `-2` – второй последний элемент и так далее.

Элементы слева направо обходятся с использованием отрицательной индексации.

```
nums = (10, 20, 30, 40, 50)
# последний элемент
print(nums[-1])
# элемент под индексом -4
print(nums[-4])
# от -3-го элемента до -1-го не включительно
print(nums[-3:-1])
# от 0-го элемента до -1-го не включительно
print(nums[0:-1])
# от -2-го элемента до -1-го включительно
print(nums[-2:])
```

Рисунок.7 – Отрицательная индексация в кортеже

```
50
20
(30, 40)
(10, 20, 30, 40)
(40, 50)
Process finished with exit code 0
```

Рисунок.7.1 – Результат Рис.7

В отличие от списков, элементы кортежа не могут быть удалены с помощью ключевого слова `del`, поскольку кортежи неизменяемы. Чтобы удалить весь кортеж, мы можем использовать ключевое слово `del` вместе с именем кортежа

```
>>> nums = (1, 2, 3, 4)
>>> print(nums)
(1, 2, 3, 4)
>>> del nums[0]
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
>>> print(nums)
(1, 2, 3, 4)
>>> del nums
>>> print(nums)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'nums' is not defined
```

Рисунок.8 – Удаление кортежа

Для определения количества элементов кортежа, используйте функцию `len()`

```
vegetables = ("помидор", "огурец", "лук")
print(f"Длина кортежа vegetables = {len(vegetables)}")
```

Рисунок.9 – Длина кортежа

```
Длина кортежа vegetables = 3
Process finished with exit code 0
```

Рисунок.9.1 – Результат Рис.9

Конструктор `tuple()`

Функция `tuple()` используется для конвертации данных в кортеж

```
vegetable = tuple('cucumber')
print(vegetable)
```

Рисунок.10 – Преобразование строки в кортеж

```
('c', 'u', 'c', 'u', 'm', 'b', 'e', 'r')
Process finished with exit code 0
```

Рисунок.10.1 – Результат Рис.10

Также можно превратить список в кортеж

```
nums_1 = [1,2,3,4,5]
nums_2 = tuple(nums_1)
print(type(nums_2))
```

Рисунок.10.2 – Преобразование списка в кортеж

```
<class 'tuple'>
Process finished with exit code 0
```

Рисунок.10.3 – Результат Рис.10.2

Присваивание несколько кортежей

Кортежи можно использовать для присваивания нескольких значений одновременно

```
var_t = ('один', 'два', 'три')
(one, two, three) = var_t
print(one)
print(two)
print(three)
```

Рисунок.11 – Присваивание несколько кортежей

```
один
два
три
Process finished with exit code 0
```

Рисунок.11.1 – Результат Рис.11

`var_t` – это кортеж из трех элементов и `(one, two, three)` – кортеж трех переменных. Присваивание `(one, two, three)` кортежу `var_t` присваивает каждое значение `var_t` каждой переменной: `one`, `two` и `three` по очереди. Это удобно, если нужно присвоить определенному количеству переменных значений в кортеже

Итерация по кортежу

Вы можете перебирать элементы кортежа с помощью цикла **for**

```
vegetables = ("помидор", "огурец", "лук")
for item in vegetables:
    print(f"Это {item}.")
```

Рисунок.12 – Итерация по кортежу

```
Это помидор.
Это огурец.
Это лук.
Process finished with exit code 0
```

Рисунок.12.1 – Результат Рис.12

Основные операции с кортежами

«Основные операции с кортежами» Таблица.1

| Оператор | Описание |
|----------|----------------------------------------------------|
| * | Позволяет повторять элементы кортежа несколько раз |
| + | Конкатенирует кортеж |

| | |
|---------------|---------------------------------------------------------------------------------------------|
| in | Возвращает True , если определенный элемент существует в кортеже, иначе False |
| for in | Цикл for используется для итерации по элементам кортежа |

Функции и методы кортежей

В отличие от списков у кортежей нет методов, таких как `append()`, `remove()`, `extend()`, `insert()` или `pop()` опять-таки из-за их **неизменяемости**. Но есть другие

`tuple.count()` возвращает количество повторений элемента в кортеже

`t.count()` используется для определения количества вхождений определенного элемента в кортеже

```
>>> var_t = (11, 22, 33, 44, 55, 22, 99, 99, 22)
>>> var_t.count(22)
3
```

Рисунок.13 – Метод `t.count()`

`tuple.index()` по указанному значению и возвращает его индекс

`t.index()` используется для получения индекса конкретного элемента в кортеже

```
>>> mixed_type = ("Лондон", 46.23, "Москва", [1, 2, 3], "Париж", 53, False)
>>> mixed_type.index("Москва")
2
```

Рисунок.14 – Метод `t.index()`

Встроенные функции

«Встроенные функции» Таблица.2

| Функция | Описание |
|-----------------------------------|---------------------------------------------------|
| <code>len(tuple)</code> | Вычисляет длину кортежа |
| <code>max(tuple)</code> | Возвращает максимальный элемент кортежа |
| <code>min(tuple)</code> | Возвращает минимальный элемент кортежа |
| <code>tuple(seq)</code> | Преобразует указанную последовательность в кортеж |
| <code>sorted(tuple)</code> | Отсортирует кортеж |

`min()` и `max()`

Функция `max()` возвращает самый большой элемент последовательности, а `min()` – самый маленький

```
>>> var_t = (1, 2, 4, -96, 1254, -43, 1222, -80)
>>> min(var_t) # минимальный элемент кортежа
-96
>>> max(var_t) # максимальный элемент кортежа
1254
```

Рисунок.15 – Функции `max()` и `min()`

Эти функции можно использовать и для кортежей со строками

```
>>> # Строка "Apple" автоматически преобразуется в последовательность символов
>>> fruit = ('Apple')
>>> max(fruit)
'p'
>>> min(fruit)
'A'
>>> 'A' > 'p'
```

Рисунок.15.1 – Функции `max()` и `min()`

`sum()`

С помощью этой функции можно вернуть сумму элементов в кортеже. Работает только с числовыми значениями

```
>>> var_t = (1, 2, 4, -96, 1254, -43, 1222, -80)
>>> sum(var_t)
2264
```

Рисунок.16 – Функция `sum()`

`sorted()`

Чтобы получить кортеж с отсортированными элементами, используйте `sorted()`

```
>>> vegetable = tuple('cucumber')
>>> sorted(vegetable)
['b', 'c', 'c', 'e', 'm', 'r', 'u', 'u']
>>> # эта функция не меняет исходную коллекцию
>>> vegetable
('c', 'u', 'c', 'u', 'm', 'b', 'e', 'r')
```

Рисунок.17 – Функция `sorted()`

Но важно отметить, что **возвращаемый тип - список**, а не **кортеж**. При этом последовательность в оригинальном объекте неизменна, а сам он остается кортежем

```
>>> var_t = (1, 2, 4, -96, 1254, -43, 1222, -80)
>>> sorted(var_t)
[-96, -80, -43, 1, 2, 4, 1222, 1254]
>>> var_t
(1, 2, 4, -96, 1254, -43, 1222, -80)
```

Рисунок.17.1 – Функция `sorted()`

Где используется кортеж (tuple)?

Использование кортежа вместо списка используется в следующем сценарии.

- Использование кортежа вместо списка дает нам четкое представление о том, что данные кортежа являются постоянными и не подлежат изменению.
- Кортеж может имитировать словарь без ключей. Обратите внимание на следующую вложенную структуру, которую можно использовать в качестве словаря.
 - [(101, "John", 22), (102, "Mike", 28), (103, "Dustin", 30)]

Отличие кортежа (tuple) от списка (list)

«Отличие кортежа (tuple) от списка (list)» Таблица.3

| Список (list) | Кортеж (tuple) |
|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис списка обозначается символом [] | Синтаксис кортежа обозначается знаком () |
| Список является изменяемым | Кортеж является неизменяемым |
| Список имеет переменную длину | Кортеж имеет фиксированную длину |
| Список обеспечивает большую функциональность, чем кортеж | Кортеж обеспечивает меньшую функциональность, чем список. |
| Список используется в сценарии, в котором нам нужно хранить простые коллекции без ограничений, где значение элементов может быть изменено | Кортеж используется в случаях, когда нам нужно хранить коллекции только для чтения, т.е. значение элементов не может быть изменено. Его можно использовать в качестве ключа внутри словаря |
| Списки занимают больше памяти, чем кортежи | Кортежи более эффективны с точки зрения использования памяти благодаря своей неизменяемости |

Теперь вы знаете следующее

Некоторые кортежи (*которые содержат только неизменяемые объекты: строки и так далее*) — **неизменяемые**, а другие (*содержащие изменяемые типы, например, списки*) **изменяемые**. Но это очень обсуждаемая тема среди программистов на Python и необходимы кое-какие познания, чтобы полностью понять ее. В целом же кортежи **неизменяемые**.

- Вы не можете добавлять в них новые элементы. У этого типа нет методов `append()` или `extend()`
- Удалять элементы тоже нельзя, также из-за неизменяемости Методов `remove()` и `pop()` нет
- Искать элементы в кортеже можно, потому что этот процесс его не меняет
- Разрешено использовать оператор **in** для проверки наличия элемента в кортеже

Так что, если вы планируете использовать постоянный набор значений для перебора, используйте кортеж вместо списка. Он будет работать *быстрее*. Плюс, это *безопаснее*, ведь такой тип данных защищен от записи

#ShoXCode