IT CAREER HUB

Урок 13.
Кортежи

Кортеж (tuple)

Кортеж — это упорядоченная неизменяемая (immutable) коллекция элементов.

Это значит, что после создания кортежа его содержимое нельзя изменить (добавлять, удалять или заменять элементы). Кортежи используются там, где требуется защитить данные от изменений и гарантировать их постоянство.

Основные характеристики кортежа:

1. **Неизменяемость:** После создания кортежа его элементы нельзя изменить.
2. **Упорядоченность:** Элементы в кортеже сохраняют порядок, в котором они были добавлены.
3. **Поддержка дубликатов:** Список может содержать повторяющиеся элементы.
4. **Поддержка различных типов данных:** Кортеж может содержать элементы разных типов.
5. **Индексация:** К кортежам можно обращаться по индексу, как и к спискам.

Создание кортежа с несколькими элементами

Кортеж создаётся с использованием круглых скобок `()`, а элементы внутри разделяются запятыми.

```python
my_tuple = (1, 2, 3, "apple", True)

print(my_tuple)
```

Создание пустого кортежа

Пустой кортеж создаётся с помощью круглых скобок.

```python
empty_tuple = ()

print(empty_tuple)  # Вывод: ()
```

Создание кортежа с одним элементом

Для создания кортежа с одним элементом нужно добавить запятую после элемента, иначе Python воспримет это как математическое выражение, а не как кортеж.

```python
single_element_tuple = (5,)

print(single_element_tuple)
```

Доступ к элементам кортежа

К элементам кортежа можно обращаться с помощью индексов, так же как и к элементам списка. Индексация начинается с 0.

```python
my_tuple = (10, 20, 30, 40)

print(my_tuple[1])

print(my_tuple[-1])
```

Изменяемость кортежей

Кортежи в Python неизменяемы, поэтому после их создания элементы нельзя добавлять, изменять или удалять.

```python
my_tuple = (10, 20, 30)



# Попытка изменить элемент вызовет ошибку

# my_tuple[1] = 40  # TypeError: 'tuple' object does not support item
assignment
```

IT CAREER HUB

Операции с кортежами

С кортежами можно проводить те же операции, что и со списками.

```python
tuple1 = (1, 2)

tuple2 = (3, 4)

print(tuple1 + tuple2)  # Конкатенация кортежей



my_tuple = (1, 2)

print(my_tuple * 3)  # Повторение кортежей



my_tuple = (10, 20, 30)

print(20 in my_tuple)  # Проверка на наличие элемента



my_tuple = (1, 2, 3, 4, 5)

print(len(my_tuple))  # Длина кортежа



# Сравнение кортежей происходит аналогично спискам, то есть поэлементно

tuple1 = (1, 2, 3)

tuple2 = (1, 2, 4)

print(tuple1 == tuple2)  # Сравнение кортежей
```

Преобразование кортежей

Можно преобразовать строку или другую коллекцию в кортеж с помощью функции `tuple()`.

```python
my_list = [1, 2, 3]

my_tuple = tuple(my_list)

print(my_tuple)

print(type(my_tuple))
```

А также можно преобразовывать кортеж в другую коллекцию, например в список.

```python
my_tuple = (1, 2, 3)

my_list = list(my_tuple)

print(my_list)

print(type(my_list))
```

IT CAREER HUB

Задания для закрепления 1

1. **Какой результат будет выведен при выполнении следующего кода?**

```python
my_tuple = (1, 2, 3, 4, 5)

print(my_tuple[2])
```

a. 2
b. 3
c. (3)
d. (3,)

Посмотреть ответ

2. **Как создать кортеж с одним элементом? Выберите все правильные варианты.**

a. `(5)`
b. `(5,)`
c. `tuple([5])`
d. `tuple(5)`

Посмотреть ответ

Цикл for с кортежами

Цикл `for` работает с кортежами так же, как и со списками.

 **Пример**

```python
my_tuple = (10, 20, 30, 40)



for item in my_tuple:

    print(item)
```

Упаковка и распаковка коллекций

 **Упаковка и распаковка — это операции, которые применяются для того, чтобы собрать несколько значений в одну коллекцию (упаковка) или разобрать коллекцию на отдельные переменные (распаковка).**

 **Упаковка — это процесс, при котором несколько значений собираются в одну коллекцию.**

Когда мы присваиваем несколько значений одной переменной, Python автоматически упаковывает их в кортеж.

 **Пример упаковки**

```python
# Упаковка нескольких значений в кортеж

packed_tuple = 1, 2, 3
```

```
print(packed_tuple)

print(type(packed_tuple))
```

Распаковка (Unpacking)

Распаковка — это процесс, при котором значения из коллекции присваиваются отдельным переменным.

Если коллекция (список или кортеж) содержит столько же элементов, сколько переменных, Python позволяет легко распаковать её.

Пример распаковки

```python
# Кортеж с тремя элементами

packed_tuple = (1, 2, 3)



# Распаковка значений кортежа в переменные

a, b, c = packed_tuple

print(a, b, c)
```

Распаковка с помощью оператора *

Когда необходимо вывести элементы коллекции по отдельности, из можно распаковать с помощью оператора * и передать в `print()`.

```python
numbers = [1, 2, 3, 4, 5]

# Вывод коллекции

print(numbers)

# Вывод элементов по отдельности

print(*numbers)
```

Упаковка с помощью оператора *

Иногда можно упаковать остаток элементов после распаковки в одну переменную с помощью оператора *. Это удобно, когда количество переменных для распаковки меньше, чем количество элементов в коллекции.

```python
numbers = [1, 2, 3, 4, 5]



# Распаковка первого элемента в a, последнего в b, остальные элементы в other

a, *other, b = numbers



print(a)

print(b)

print(other)
```

Распаковка в цикле

Распаковка особенно полезна при работе с циклами, когда нужно перебирать элементы коллекций с несколькими значениями.

Пример

```python
pairs = [(1, 'apple'), (2, 'banana'), (3, 'cherry')]



for number, fruit in pairs:  # Распаковка элементов кортежа в переменные

    print(f"Число: {number}, Фрукт: {fruit}")
```

Ошибки при распаковке

Количество переменных должно соответствовать количеству элементов в коллекции, за исключением случаев, когда используется оператор *. В противном случае возникнет ошибка.

Пример ошибки

```python
data = (1, 2, 3)



# Попытка распаковать больше переменных, чем элементов

a, b, c, d = data  # Ошибка, т.к. слишком мало элементов

a, b = data  # Ошибка, т.к. слишком много элементов
```

IT CAREER HUB

Функция enumerate

Функция enumerate() — это встроенная функция Python, которая добавляет счётчик к итерируемому объекту, таким как список, кортеж или строка, и возвращает результат в виде объекта enumerate.

Это удобно, когда необходимо получить как индекс, так и значение элемента во время итерации по коллекции.

Синтаксис:

```python
enumerate(iterable, start=0)
```

- `iterable` — это коллекция, которую нужно перебирать (например, список или кортеж).
- `start` (необязательно) — значение, с которого начинается счётчик. По умолчанию — 0.

Отображения содержимого enumerate

Чтобы увидеть содержимое объекта `enumerate`, необходимо преобразовать его в список, кортеж или другую структуру данных, которая может быть напечатана, т.к. содержимое объекта enumerate не отображается напрямую при выводе.

Пример

```python
fruits = ["apple", "banana", "cherry"]
enumerated_fruits = enumerate(fruits)

# Чтобы увидеть содержимое, преобразуем объект enumerate в список
print(list(enumerated_fruits))
```

Использование enumerate()

Функция `enumerate()` часто используется, когда требуется одновременно итерировать по элементам последовательности (например, список, строка или кортеж) и отслеживать их индексы.

Пример: вывод индекса и элемента

```python
fruits = ["apple", "banana", "cherry"]

for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
```

Пример 1: редактирование списка по индексу

```python
numbers = [10, 20, 30, 40]

for index, value in enumerate(numbers):
   numbers[index] = value * 10

print(numbers)
```

Пример 2: доступ к соседним элементам

Функция `enumerate()` также может быть полезна, когда нужно получить доступ к текущему элементу и его соседним элементам в последовательности. Это часто применяется при анализе последовательных данных, где требуется сравнить соседние элементы.

Допустим, у нас есть список чисел, и мы хотим найти все такие элементы, которые больше следующего элемента в последовательности.

```python
numbers = [10, 20, 15, 30, 25, 35]


for index, value in enumerate(numbers[:-1]):   # Проходим по списку, кроме последнего элемента
    if value > numbers[index + 1]:
        print(f"{value} больше, чем {numbers[index + 1]}")
```

Изменение стартового значения enumerate()

Вы можете изменить значение, с которого начинается счётчик, с помощью параметра start.

```python
languages = ("Python", "Java", "C++")


for index, language in enumerate(languages, start=1):
    print(f"{index}: {language}")
```

Использование enumerate() со строкой:

Функция `enumerate()` может быть использована для перебора символов строки с индексами.

```python
text = "Python"


for index, letter in enumerate(text):
```

```
    print(f"{index}: {letter}")
```

IT CAREER HUB

Задания для закрепления 2

1. **Что произойдет при попытке распаковать кортеж с тремя элементами в две переменные?**

```python
my_tuple = (1, 2, 3)

a, b = my_tuple
```

a. a = 1, b = 2
b. a = (1, 2), b = 3
c. a = 1, b = (2, 3)
d. Ошибка

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
fruits = ["apple", "banana", "cherry"]


for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}", end=' ')
```

a. `0: apple 1: banana 2: cherry`
b. `1: apple 2: banana 3: cherry`
c. `Ошибка`
d. `0: apple 1: banana 2: cherry на разных строках`

Посмотреть ответ

IT CAREER HUB

Методы кортежа

Кортежи — это неизменяемые коллекции, поэтому они поддерживают ограниченное количество методов.

Кортежи предоставляют два метода: `count()` и `index()`, которые позволяют работать с данными внутри кортежа.

Метод count()

Метод `count()` используется для подсчёта, сколько раз определённый элемент встречается в кортеже.

Синтаксис:

```python
tuple.count(value)
```

- `value` — элемент, количество вхождений которого нужно подсчитать.

Пример использования count()

```python
my_tuple = (1, 2, 3, 2, 4, 2)

count_of_twos = my_tuple.count(2)

print(count_of_twos)
```

Метод index()

Метод index() возвращает индекс первого вхождения указанного элемента в кортеже. Если элемент не найден, будет вызвана ошибка ValueError.

Синтаксис:

```JavaScript
tuple.index(value, start=0, end=None)
```

- `value` — элемент, индекс которого нужно найти.
- `start` (необязательно) — начальная позиция для поиска.
- `end` (необязательно) — конечная позиция для поиска.

Пример использования index()

```Python
my_tuple = (10, 20, 30, 20, 40)

index_of_first_twenty = my_tuple.index(20)

print(index_of_first_twenty)
```

Пример с указанием диапазона поиска

```Python
my_tuple = (10, 20, 30, 20, 40)

index_of_twenty_in_range = my_tuple.index(20, 2)  # Начинаем поиск с индекса 2

print(index_of_twenty_in_range)
```

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Кортеж с одним элементом	Ответ: b, c
Задания на закрепление 2	Вернуться к заданиям
1. Распаковка кортежа	Ответ: d
2. Результат выполнения кода	Ответ: a

IT CAREER HUB

Практическая работа

1. Кортеж уникальных

Напишите программу, которая обрабатывает кортеж чисел. Программа должна вернуть кортеж, в котором будут только уникальные элементы.

Не используйте неизученные коллекции.

Данные:

```python
numbers = (1, 2, 3, 2, 1, 4, 5, 3, 6)
```

Пример вывода:

```python
Уникальные элементы: (1, 2, 3, 4, 5, 6)
```

Решение:

```python
numbers = (1, 2, 3, 2, 1, 4, 5, 3, 6)
unique_numbers = tuple()
for num in numbers:
    if num not in unique_numbers:
        unique_numbers += (num,)
print("Уникальные элементы:", unique_numbers)
```

IT CAREER HUB

2. Кортеж выше среднего

Напишите программу, которая обрабатывает кортеж чисел. Программа должна вернуть кортеж, состоящий из элементов, которые больше среднего значения всех элементов исходного кортежа.

Данные:

```python
numbers = (10, 15, 20, 25, 30)
```

Пример вывода:

```python
Элементы больше среднего: (25, 30)
```

Решение:

```python
numbers = (10, 15, 20, 25, 30)
average = sum(numbers) / len(numbers)
greater_than_avg = tuple()


for num in numbers:
    if num > average:
        greater_than_avg += (num,)


print("Элементы больше среднего:", greater_than_avg)
```