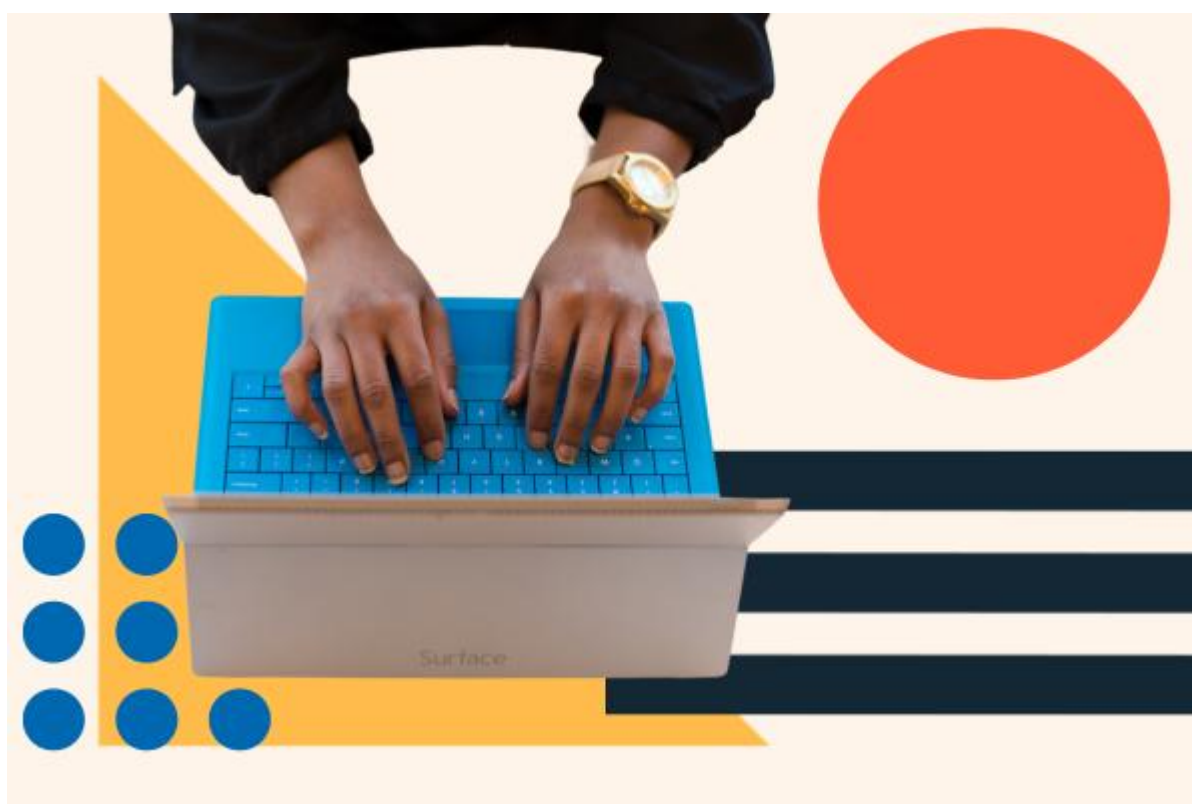


# Как использовать индекс в Python [+Примеры]

09.06.2023

Освоение индексации в Python необходимо всем, кто хочет эффективно манипулировать элементами в последовательности. Из этого руководства вы узнаете об основах индексирования в Python и о том, как применять его в строках, списках и кортежах. Давайте погрузимся.



## Что такое индекс в Python?

Вы можете использовать квадратные скобки для доступа к определенному элементу в последовательности. Просто введите значение его индекса в скобках. Например, если вы хотите получить доступ к шестому элементу в списке, вы напишете: `my_list = ["яблоко", "банан", "вишня", "дуриан", "баклажан", "инжир"]; my_list[5]`. В Python также есть удобная функция `index()`, которая делает этот процесс еще проще. Она принимает

два аргумента – элемент, который вы хотите найти, и последовательность, в которой он находится, – и возвращает значение его индекса. Например, если вы хотите найти индекс “fig” в приведенном выше списке, вы напишете: `my_list.index(“fig”)`.

Python вернет 5 в качестве значения индекса для этого элемента. Важно отметить, что Python допускает отрицательную индексацию, то есть вы можете использовать отрицательные числа для подсчета элементов с конца последовательности. Это может быть полезно в некоторых ситуациях. Например, если вам нужен доступ только к последнему элементу списка, вы можете написать: `my_list[-1]`; Python укажет на элемент со значением индекса -1 (в данном случае “fig”).

## Когда использовать функцию индекса в Python

Функция `index()` в Python помогает определить местоположение элемента в определенной последовательности, включая списки, кортежи и строки. Она определяет начальное вхождение этого элемента, возвращая связанное с ним числовое значение индекса. Если по какой-то причине он не может найти указанный элемент, то этот замечательный инструмент выдает сообщение `ValueError`, чтобы предупредить пользователя.

Давайте посмотрим на формат функции `index()`:

`SEQUENCE.INDEX(ELEMENT, START, END)`

- `sequence`: Определяет порядок поиска элемента.
- `element`: Определите элемент, который нужно найти в последовательности.
- (необязательно): Определите начальную точку поиска.
- `end` (необязательно): Определяет начальную точку поиска.

Когда вам нужно найти элементы в строке или списке Python,

функция `index()` может стать бесценным инструментом. Вот несколько сценариев, в которых эта функция может оказаться полезной.

## When To Use Index in Python

- 1 Searching for an Element in a Sequence
- 2 Checking if an Element Exists in a Sequence
- 3 Finding the First Occurrence of an Element
- 4 Searching for an Element Within a Subset of a Sequence

## Поиск элемента в последовательности

Если у вас есть серия элементов и вы хотите определить индекс одного конкретного элемента, вы можете использовать удобную функцию `index()`. Вот пример:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY', 'BANANA']  
INDEX = FRUITS.INDEX('CHERRY')
```

```
PRINT(INDEX) # ВЫВОД: 2
```

Допустим, у вас есть список фруктов, и вы хотите найти индекс 'cherry'. Все, что вам нужно, это указать значение 'cherry' в качестве аргумента при использовании функции `index()`, и она вернет его позицию в списке, которая, как оказалось, равна 2.

# Проверка наличия элемента в последовательности

Когда у вас есть набор значений, в котором необходимо найти определенный элемент, оператор `in` и функция `index()` обеспечивают эффективный способ проверки его существования. Позвольте мне проиллюстрировать это на примере:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY', 'BANANA']
IF 'CHERRY' IN FRUITS:
PRINT('CHERRY EXISTS')
ELSE:
PRINT('CHERRY DOES NOT EXIST')
```

```
IF FRUITS.COUNT('CHERRY') > 0:
PRINT('CHERRY EXISTS')
ELSE:
PRINT('CHERRY DOES NOT EXIST')
```

```
IF FRUITS.INDEX('CHERRY') >= 0:
PRINT('CHERRY EXISTS')
ELSE:
PRINT('CHERRY DOES NOT EXIST')
```

Чтобы определить, существует ли `'cherry'` в списке фруктов, в этом примере используются три метода:

- Использование оператора `in`.
- Подсчет с помощью функции `count()`.
- Отслеживание его индексного номера с помощью функции `index()`.

Несомненно, все эти усилия будут вознаграждены положительным ответом – ведь элемент под названием “вишня” действительно существует.

## Поиск первого появления элемента

Когда у вас есть список значений и вам нужно найти первое присутствие элемента, функция `index()` – это то, что вам нужно.

Давайте посмотрим на это в действии:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY', 'BANANA']  
INDEX = FRUITS.INDEX('BANANA')
```

```
PRINT(INDEX) # ВЫВОД: 1
```

В данном случае у нас есть список фруктов, и нам нужно найти индекс первого “банана”. Для этого мы просто передаем ‘banana’ в качестве аргумента через функцию `index()`. Она возвращает 1, указывая на то, что банан находится на первой позиции в нашем списке фруктов.

## Поиск элемента в подмножестве последовательности

Если у вас есть последовательность значений и вам нужно найти элемент в подмножестве этой последовательности, вы можете использовать функцию `index()` с параметрами `start` и `end`.

Вот пример:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY', 'BANANA']
```

```
INDEX =
```

## Как создать индекс в Python

В Python вы можете создать индекс двумя уникальными способами: используя функцию `enumerate()` или словарь. Первый способ позволяет назначить индексы каждому элементу итерируемого объекта, например, кортежа или списка. Во втором случае ключи служат индексами, а соответствующие значения представляют элементы в наборе. Вот несколько способов создания индекса в

Python.

## Использование функции `enumerate()`

Функция `enumerate()` принимает объект `iterable` и возвращает итератор, который выдает кортежи, содержащие индекс и значение каждого элемента в объекте `iterable`. Вы можете преобразовать этот итератор в список, чтобы создать индекс для объекта `iterable`.

Вот пример:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY'].  
FRUIT_INDEX = LIST(ENUMERATE(FRUITS))
```

```
PRINT(FRUIT_INDEX)
```

**Выходные данные:**

```
[(0, 'APPLE'), (1, 'BANANA'), (2, 'CHERRY')].
```

Чтобы построить `fruit_index`, мы можем использовать понимание словаря для перебора списка фруктов и создания отображения пар индекс-значение. Функция `enumerate()` позволяет нам получить соответствующий индекс и элемент из каждого элемента списка фруктов; этот результат затем создается как переменная `fruit_index`.

## Использование директории

Для создания индекса можно использовать словарь, сделав ключами индекса значения из списка. Например:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY'].  
FRUIT_INDEX = {I: FRUIT FOR I, FRUIT IN ENUMERATE(FRUITS)}
```

```
PRINT(FRUIT_INDEX)
```

**Выходные данные:**

```
{0: 'APPLE', 1: 'BANANA', 2: 'CHERRY'}
```

Чтобы построить `fruit_index`, мы можем использовать понимание словаря для перебора списка фруктов и создания отображения пар индекс-значение. Функция `enumerate()` позволяет нам получить соответствующий индекс и элемент из каждого элемента списка фруктов; этот вывод затем создается как наша переменная `fruit_index`.

## Индекс Python в действии

### Пример 1: `index()` с начальным и конечным параметрами

Допустим, мы хотим найти индекс `'banana'` в определенном разделе нашего списка. Мы можем сделать это, добавив начальный и конечный параметры следующим образом:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY', 'BANANA'].  
INDEX = FRUITS.INDEX('BANANA', 0, 2)
```

```
PRINT(INDEX) # ВЫХОД: 1
```

Здесь функция `index()` возвращает 1, поскольку "банан" занимает первую позицию в нашем списке. Параметры поиска ограничивают диапазон обнаружения диапазоном между 0 (включено) и 2 (исключено).

### Пример 2: `enumerate()` и распаковка кортежей

Вот немного более сложный пример, демонстрирующий, как сочетать функции `index()` и `enumerate()`, а также распаковку кортежей.

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY']  
FOR I, FRUIT IN ENUMERATE(FRUITS):  
    INDEX = FRUITS.INDEX(FRUIT, I+1)  
    IF INDEX > 0:  
        PRINT('THE ELEMENT {0} IS ALSO AT POSITION {1}'.FORMAT(FRUIT,
```

```
INDEX))
```

**Выходные данные:** Элемент банан также находится в позиции 3

Используя технику распаковки кортежа для присвоения переменных “i” и “fruit”, мы перемещаемся по нашему списку фруктов. Затем мы используем функцию `index()` с начальным значением, которое опускает текущий элемент (в данном примере `i+1`). При этом мы обнаруживаем, что “банан” встречается в нашем списке дважды и находится под индексом 3.

## Пример 3: Использование словаря для создания индекса

Наконец, допустим, мы хотим использовать словарь для создания индексов для нашего списка фруктов. Вот как мы можем это сделать, используя предыдущий пример в качестве руководства:

```
FRUITS = ['APPLE', 'BANANA', 'CHERRY'].
FRUIT_INDEX = {I: FRUIT FOR I, FRUIT IN ENUMERATE(FRUITS)}

PRINT(FRUIT_INDEX)
```

**Выходные данные:** {0: 'apple', 1: 'banana', 2: 'cherry'}

Используя понимание словаря, мы можем создать индекс для нашего списка фруктов. Итерация по списку и использование функции `enumerate()` для создания последовательности пар индекс-значение дает нам основу для построения словаря – таким образом, мы получаем легкий доступ ко всем элементам данной коллекции.

## Начало работы с индексированием в Python

Индексирование – невероятно полезный инструмент в Python, и понимание того, как эффективно его использовать, может стать отличным подспорьем. С помощью функций `index()`, `enumerate()` и



словаря comprehension вы можете эффективно создавать индексы для любого списка элементов. С мощью Python на кончиках ваших пальцев творческие возможности не ограничены.