Python: операции с последовательностями, срезы, диапазоны | Я.Шпора

Общие операции для последовательностей

Определить количество элементов последовательности

Для этой задачи используется функция `len()` ; аргументом передаётся последовательность, возвращается число — количество элементов.

```
any_string = 'Сколько же элементов в этой строке?'
print('Символов в строке any_string:', len(any_string))
# Вывод в терминал: Символов в строке any_string: 35

any_range = range(10, 57, 2)
print('Элементов в диапазоне any_range:', len(any_range))
# Вывод в терминал: Элементов в диапазоне any_range: 24

any_list = [2, 12, 85, '06', 'это мой номер']
print('Элементов в списке any_list', len(any_list))
# Вывод в терминал: Элементов в списке any_list 5
```

Конкатенация — объединение последовательностей

Выполняется с помощью оператора `+` . Объединять можно только последовательности одного типа. Результат конкатенации — новая последовательность.

```
first_vegetable_list = ['помидор', 'огурец', 'баклажан']

second_vegetable_list= ['морковь', 'лук', 'чеснок']

full_vegetable_list = first_vegetable_list + second_vegetable_list
print(full_vegetable_list)
# Вывод в терминал: ['помидор', 'огурец', 'баклажан', 'морковь', 'лук', 'ч
```

```
hybrid = second_vegetable_list[3] + first_vegetable_list[1]
print(hybrid)
# Вывод в терминал: чеснокогурец
```

Умножение последовательностей

Все последовательности, кроме `range`, можно умножить на число. В результате вернётся новая последовательность, где элементы будут дублироваться заданное количество раз.

```
dig_more = 'копаю грядку! ' * 4
print(dig_more)
# Вывод в терминал: копаю грядку! копаю грядку! копаю грядку! копаю грядку
```

Проверка наличия элемента в последовательности

Оператор `in` позволяет узнать, есть ли заданный элемент в последовательности, и возвращает `True` или `False`.

```
numbers = [1, 2, 3, 4, 5]
result = 3 in numbers  # True, тройка есть в списке чисел
```

Сравнение последовательностей

При сравнении последовательностей поочерёдно сравниваются элементы с одинаковыми индексами, и при первом же различии определяется результат сравнения.

Операторы `<`, `>`, `<=`, `>=` работают только при сравнении последовательностей одного типа и однотипных элементов в этих последовательностях.

```
list_one = [10, 12, 17, 9, 1, 4]
list_two = [10, 12, 17, 3, 1, 4]

print(list_one > list_two)
# Вывод в терминал: True
```

Если в одной последовательности больше элементов, чем в другой, а сравниваемые элементы оказались равны, большей будет признана та последовательность, которая длиннее.

> При сравнении строк сравниваются не сами символы в строках, а их числовые коды.

Поиск наименьшего и наибольшего значения в последовательности

Используются встроенные функции `min()` и `max()`. Искать наименьшее и наиболее значение можно только в последовательностях с элементами одного типа.

```
str_of_numbers = '12345'
print('Максимальное значение в строке str_of_numbers:', max(str_of_numbers
# Максимальное значение в строке str_of_numbers: 5

list_of_str = ['abc', 'Abc']
print('Минимальное значение в списке list_of_str:', min(list_of_str))
# Abc
```

Срезы

Чтобы из последовательности `sequence` получить срез, в который войдут элементы, расположенные один за другим между заданными индексами, применяют синтаксис `sequence[start:end]`, где `start` и `end` — индексы элементов, определяющие диапазон, из которого будет взят срез.

> Если в срезе указан конечный индекс, элемент с этим индексом не включается в срез.

```
vegetable_name = 'Ананас'
print(vegetable_name[1:5])
# Вывод в терминал: нана
```

Для получения среза можно указать только начальный или конечный индекс,

тогда второй границей среза будет конец или начало исходной последовательности.

```
vegetable_name = 'Картофелина'
print(vegetable_name[7:])
# Вывод в терминал: лина

print(vegetable_name[:7])
# Вывод в терминал: Картофе
```

Чтобы получить срез из элементов, расположенных с определённым шагом, применяют синтаксис `sequence[start:end:step]`.

```
vegetables = ['Помидоры', 'Огурцы', 'Баклажаны', 'Перец', 'Капуста']
print(vegetables[0:5:2])
# Вывод в терминал: ['Помидоры', 'Баклажаны', 'Капуста']
```

Диапазон (range)

Диапазон — неизменяемая последовательность чисел. Последовательность `range` создаётся при вызове функции `range()`; границы создаваемого диапазона передаются в аргументах.

```
days = range(0, 30)
```

В переменной `days` будет создан диапазон чисел от 0 до 29 включительно.

Функция `range()` может принимать от одного до трёх аргументов:

1. `range(stop)`: сгенерирует числа от 0 до `stop - 1`.

   `range(100)` сгенерирует числа от 0 до 99 включительно.

2. `range(start, stop)`: сгенерирует числа от `start` до `stop - 1`.

   `range(50, 100)` сгенерирует числа от 50 до 99 включительно. Объявления диапазонов `range(0, 100)` и `range(100)` вернут последовательности с одинаковым набором элементов.

3. `range(start, stop, step)`: сгенерирует числа от `start` до `stop - 1` с шагом `step`.

   `range(50, 100, 9)` сгенерирует числа в пределах от 50 до 99 включительно

с шагом 9:

50, 59, 68, … 95. Последним значением будет 95.

Шпаргалка — это карта,
помогающая мудрому
мореплавателю выбрать путь.

Ягдетто Там, испанский мореплаватель