Строки

ЛЕКЦИЯ 4

Содержание

- Строки
- Индексирование
- Извлечение среза
- Конкатенация и повторение
- Неизменяемость
- Специальные методы строк

Введение

Последовательности - это упорядоченные хранилища(коллекции) объектов.

- Последовательности поддерживают порядок размещения элементов, которые они содержат, слева направо.
- Элементы последовательности сохраняются и извлекаются исходя из их позиции.

Строки - способ записи текстовой информации или произвольных последовательностей байтов

Строки являются последовательностями одно символьных строк (проще говоря – символов)

Индексирование

- Каждый элемент строки имеет **индекс**

Индекс - значение смещения от начала последовательности

- Первый элемент последовательности является 0-ым (смещение на 0 от начало последовательности)
- Последний элемент последовательности носит индекс -1

Доступ к элементам строки

Определить длину строки можно с помощью функции len()

```
s = 'my_string'

print(len(s))
```

```
9

Process finished with exit code 0
```

- Доступ к элементам строки, так же как и к элементам большинства последовательностей осуществляется с помощью квадратных скобок «[]»

```
s = 'my_string'

print(s[0])          # печать первого символа из строки
print(s[len(s)-1])   # печать последнего элемента в строке
print(s[-1])         # упрощенный способ печати последнего элемента
```

```
m
g
g

Process finished with exit code 0
```

Извлечение среза

Извлечение среза – способ выделения целого сегмента последовательности за одну операцию

```
s = 'my_string'

print(s[1:4])
```

```
y_s

Process finished with exit code 0
```

Операция **X[I:J]** означает: извлечь из **X** все элементы начиная со смещения **I** и до смещения **J**, не включая его

Извлечение среза: Примеры

```
s = 'my_string'

print(s[1:])    # Все, кроме первого элемента (1:len(s))
print(s[0:8])   # Все, кроме последнего элемента
print(s[:8])    # То же, что и s[0:8]
print(s[:-1])   # То же, что и s[0:8]
print(s[:])     # Вся строка, как копия
```

```
y_string
my_strin
my_strin
my_strin
my_string

Process finished with exit code 0
```

Конкатенация и повторение

Конкатенация – это объединение двух строк в одну. Для представления этой операции используется знак «+»

```
s1 = 'my_string1'
s2 = 'my_string2'

print(s1+s2)
```

```
my_string1my_string2

Process finished with exit code 0
```

Повторение – это создание новой строки за счет многократного повторения другой строки. Для представления этой операции используется знак «*»

```
s1 = 'my_string1'

print(s1*3)
```

```
my_string1my_string1my_string1

Process finished with exit code 0
```

Неизменяемость

- Все строки и числа в языке Python относятся к неизменяемым объектам. Это означает, что под действие различных операций оригинальная строка/число не меняется, а создается новый объект с необходимыми параметрами

Специальные методы при работе со строками

Метод **find()** выполняет поиск подстроки в строке. Данный метод возвращает значение смещения (индекс) переданной ему подстроки или -1 если поиск не увенчался успехом.

```
s1 = 'my_string1'
s2 = 'string'

print(s1.find(s2))
```

```
3

Process finished with exit code 0
```

Метод **replace()** производит поиск подстроки с заменой

```
s1 = 'my_string1'
s2 = 'string'

print(s1.replace(s2, 'xyz'))
```

```
my_xyz1

Process finished with exit code 0
```

Специальные методы при работе со строками

Метод **split()** разбивает строки по разделителю, указываемому в скобках и создает ***список*** строк

```
s = 'my_string1'

print(s.split('_'))
```

```
['my', 'string1']

Process finished with exit code 0
```

Методы **upper**() и **lower**() переводят символы из строки в верхний и нижний регистры соответственно

```
s = 'my_string1'
s = s.upper()
print(s)
s = s.lower()
print(s)
```

```
MY_STRING1
my_string1

Process finished with exit code 0
```

Получение справки об используемом методе

Для того чтобы быстро получить справку о том или ином методе следует воспользоваться функцией **help**() в качестве аргумента которой следует передать объект с интересующим методом **БЕЗ** указания аргументов этого метода

```
s = 'my_string'

help(s.replace)
```

```
Help on built-in function replace:

replace(...) method of builtins.str instance
    S.replace(old, new[, count]) -> str

    Return a copy of S with all occurrences of substring
    old replaced by new.  If the optional argument count is
    given, only the first count occurrences are replaced.


Process finished with exit code 0
```