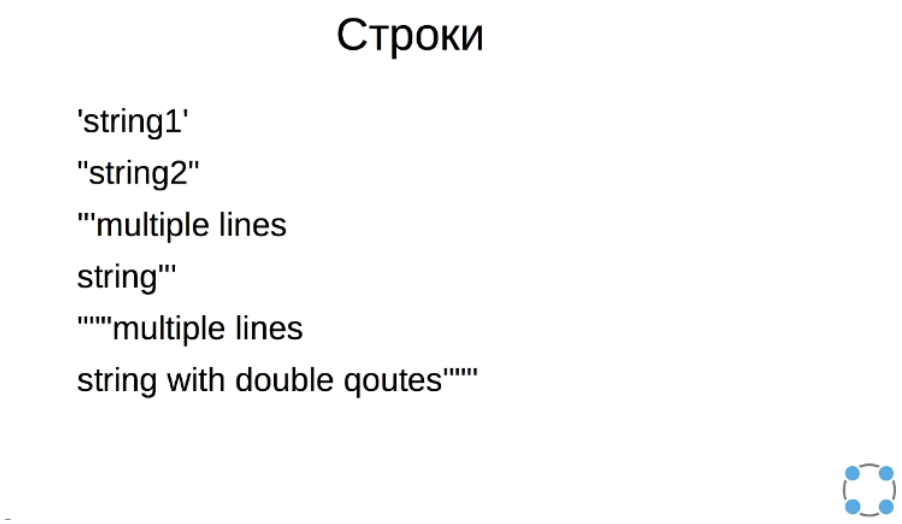
Step 1

Step on Stepik: https://stepik.org/lesson/3362/step/1

В нескольких программах нам с вами уже встречались строки и комментарии. Однако остановимся на на них чуть подробнее. Строки в Python - это просто последовательность символов. Символы могут быть абсолютно произвольными: буквы (большие и маленькие, латинские, русские), цифры, пробелы, переводы строк, и все что угодно. Есть несколько способов записать строку.

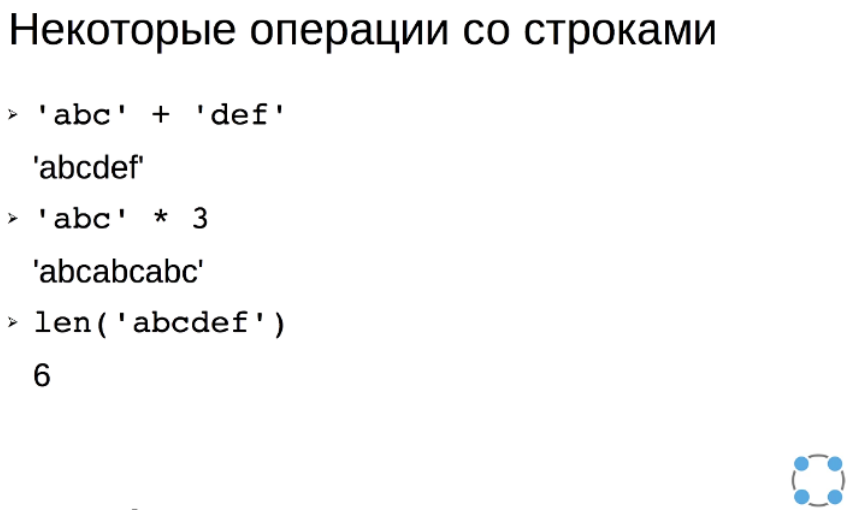


Первый способ - это использовать одинарные кавычки, мы ставим одинарную кавычку, пишем строку и в конце строки ставим еще одну одинарную кавычку, для того чтобы показать, что что строка закончилась. Можно использовать символ двойная кавычка. Между использованием одинарных и двойных кавычек никакой принципиальной разницы нет. Можно использовать и так и так.

Если нам захочется создать строку, которая будет содержать символы перевода строк, то можно использовать тройные кавычки (три одинарные кавычки), далее идет ваша длинная строка и в конце мы закрываем ее тройными кавычками. Можно также использовать в качестве тройных кавычек три двойных кавычки.

```
In [18]: a = 'string a'
         b = "string b"

         c = '''mutiple lines
         string'''

         d = """mutiple lines
         string with double qoutes"""

         print(a, b, c, d, sep = "\n\n")

         string a

         string b

         mutiple lines
         string

         mutiple lines
         string with double qoutes

In [ ]:
```

К срокам можно применять некоторые стандартные операции. Строки можно складывать, при этом результатом будет конкатенация этих строк. Конкатенация означает - соединение строк, к концу первой строки присоединяется вторая строк, без всяких дополнительных символов. Строку можно умножать на целое число.



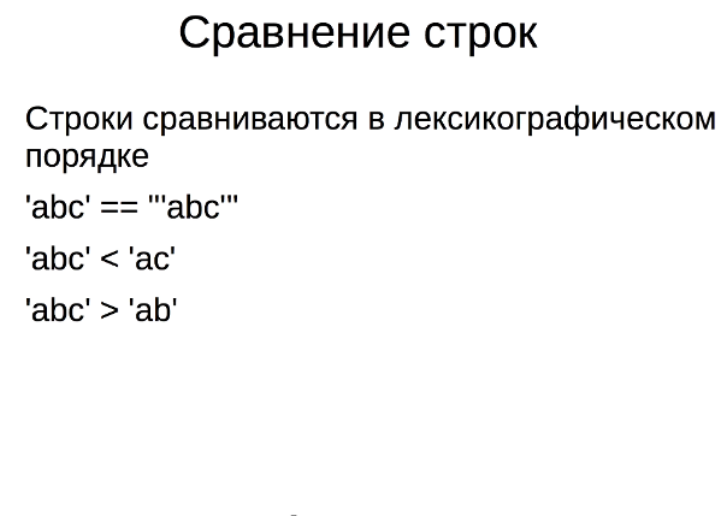
Работает это следующим образом, если мы умножаем строку, к примеру, на 3, то в результате получаем нашу строку записаную три раза. В общем случае умножая строку на **к**, мы получаем в результате строку, которая будет в **к** раза длинней и **к** раз содержать исходную строку.

Есть полезная функция **len**, на вход которой можно передать строку, которая возвращает количество символов в этой строке. Для

примера: **len('abcdef')** вернет 6.

```
In [19]: 'abc' + 'def'
Out[19]: 'abcdef'

In [20]: 'abc' * 3
Out[20]: 'abcabcabc'

In [21]: len('abcdef')
Out[21]: 6

In [ ]:
```

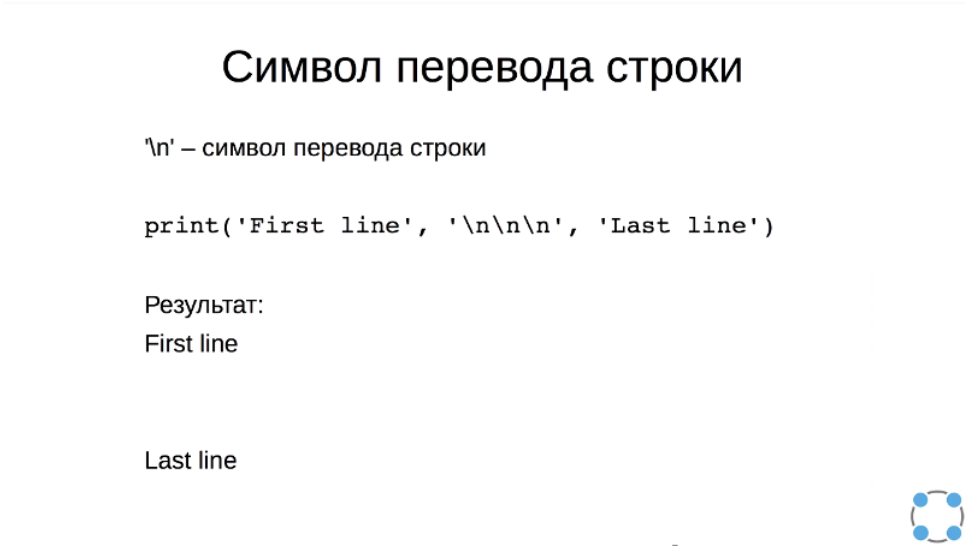
Строки можно сравнивать и проверять на равенство. Две строки равны, если все их символы по порядку равны.



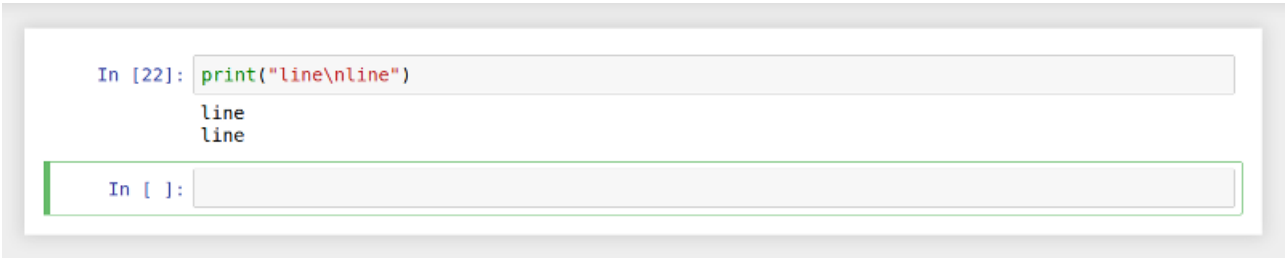
В примере на слайде для строк используются разные кавычки. От того какой тип кавычек используется (одинарные или двойные) результат не меняется и будет True. Строки можно сравнивать на больше - меньше. При этом одна строчка считается меньше, если она лексикографически идет раньше. Лексикографический порядок - это порядок слов в словаре. Как происходит сравнение? Вначале, сравниваются первые символы, если они равны, сравниваются следующие. Если они не равны, то меньшей будет та строка, символ в которой меньше. В случае, если в одной из строк в результате такого стравнения закончились символы, то меньшей будет та строка, в которой символы закончились.

Полезно знать, что есть специальный символ - перевод строк. Он

обозначается '**\n**'



Мы можем его использовать в рамках какой нибудь строки. Например строка '**line\nline**' , будет выведена на двух строках.



Еще один пример приведен на слайде выше. Функции **print** мы передали три параметра. Какой будет результат? Вначале будет выведена первая строка, затем последует тройной перевод строки, и наконец, в последней строке выведется третий переданный аргумент.

Почти всегда в программе полезно писать комментарии.

Комментарии

```
# это комментарий
x = 5 # комментарий к действию

'''
Многострочный комментарий – это просто
строка
'''
```

Есть однострочные комментарии которые начинаются с помощью символа решетка '**#**' за которым может идти любой текст. Комментарием называется такой текст, который никаким образом не влияет на работу программы. Он пишется для того, чтобы людям читающим код, было понятно, что происходит в программе. Например, мы можем написать операцию:

```
In [23]: x = 5  # переменной x присвоено значение 5

In [ ]:
```

и в комментарии указать, что в этой строчке мы заводим переменную x и записываем в нее число 5. Многострочные комментарии Python фактически представляют собой строки, обрамленные с обоих сторон тройными кавычками.

```
In [24]: print('Строка до комментария.')
         '''
             Пример многострочного комментария.
             При выполнении этого кода он будет игнорироваться
             интерпретатором.
         '''
         print('Строка после комментария.')

         Строка до комментария.
         Строка после комментария.

In [ ]:
```

Ссылки:

https://neerc.ifmo.ru/wiki/index.php?title=%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA