# Урок 14.03. Методы строк startswith, endswith, upper, lower. Подстановка.

Мы продолжаем изучать методы строк, которые позволят нам проще, быстрее и удобнее работать со строками. Давайте вспомним нашу программу по формированию пути к файлу.

Нам нужно было написать программу формирующую путь к файлу «C:/user/docs/folder/new\_file.txt». Этот путь начинается на диске C и заканчивается расширением файла «.txt». И пусть пользователь вводит полное имя файла, т.е. пусть вводит еще и расширение файла. Но мы точно знаем, что файл должен быть с расширением «.txt». Может показаться, что мы нагружаем пользователя вводом расширения, однако чаще всего пользователю дается возможность выбрать файл, а не писать его, мы пока этот этап пропустим и предположим, что нам поступает полное имя файла, просто в нашей программе попросим пользователя его вводить. Но самое важное здесь проверить, что имя введенного файла оканчивается на «.txt». Возьмем нашу недавнюю программу и для начала уберем из имени файла расширение. И с помощью срезов можно проверить строку, а именно нам нужно проверить последние 4 символа имени файла.

user\_name = input('Введите имя пользователя: ')

file\_name = input('Введите имя файла: ')

path = 'C:/{user}/docs/folder/{new\_file}'.format(

user=user\_name,

new\_file=file\_name

)

if path[-4:] == '.txt':

print('Путь к файлу:', path)

else:

print('Ошибка. Неверное расширение файла.')

Запустим программу и убедимся, что она работает верно. Однако на практике такой подход не очень то и хорошо. Во-первых, это выражение не очень наглядно, а, во-вторых, при извлечении среза создается новый объект, что приводит к увеличению времени работы такой конструкции и занимаемой памяти, это особенно становится очевидным при работе с большим объемом данных, а, в-третьих, это не правильно с точки зрения стиля на python. Гораздо правильнее, здесь использовать один из новых методов «endswith»:

user\_name = input('Введите имя пользователя: ')

file\_name = input('Введите имя файла: ')

path = 'C:/{user}/docs/folder/{new\_file}'.format(

user=user\_name,

new\_file=file\_name

)

if path.endswith('.txt'):

print('Путь к файлу:', path)

else:

print('Ошибка. Неверное расширение файла.')

И такой код уже является и читаемым и быстрым и по памяти оптимальным. Запустим и проверим программу. Всё работает.

И как вы уже догадались, что если можно проверить чем строка заканчивается, то можно проверить и с чего строка начинается, для этого используется метод «startswith». Проверим это путем проверки имени диска:

user\_name = input('Введите имя пользователя: ')

file\_name = input('Введите имя файла: ')

path = 'C:/{user}/docs/folder/{new\_file}'.format(

user=user\_name,

new\_file=file\_name

)

if not path.startswith('C:/'):

print('Ошибка. Неверное имя диска.')

elif not path.endswith('.txt'):

print('Ошибка. Неверное расширение файла')

else:

print('Путь к файлу:', path)

Запустим и убедимся, что программа работает.

В этой программе есть небольшая загвоздка: что если расширение или имя диска будет введено большими буквами, то программа перестанет работать. Давайте это исправлять. Для изменения регистра нам поможет такие функции как lower и up, программа будет выглядеть вот так:

user\_name = input('Введите имя пользователя: ')

file\_name = input('Введите имя файла: ')

path = 'C:/{user}/docs/folder/{new\_file}'.format(

user=user\_name,

new\_file=file\_name

)

if not path.upper().startswith('C:/'):

print('Ошибка. Неверное имя диска.')

elif not path.lower().endswith('.txt'):

print('Ошибка. Неверное расширение файла')

else:

print('Путь к файлу:', path)

Запустим и получим работающую программу, которая учитывает регистр.

Конечно, методов у списков очень много и все мы не будем рассматривать, однако вам нужно самостоятельно их изучить, что понадобиться в заданиях.

Подстановка.

В python также можно использовать способ форматирования строк чтобы вывести числа определенным образом.

Рассматрим задачу «Детали».

У нас есть 5 000 000 деталей, стоимость детали 23. 8589578 руб, и наценка на продажу детали составляет 4,5678. И нам нужно вывести продажи цену детали и суммарную стоимость всех деталей. И для этого в фигурных скобках используется так называемый «placeholder» и вот так он выглядит:

count\_det = 500000000

price = 23.8589578

increase = 0.045678

print('На складе {:,d} деталей.'.format(count\_det))

print('Каждая деталь стоит {:.2f} руб.'.format(price))

print('Цена увеличилась {:.1%}.'.format(increase))

Остальные placrholder’ы вы можете посмотреть сами.