# Урок 14.01. Форматирование строк: format и f-string.

Мы уже довольно хорошо знаем списки и много работали с их методами. Теперь понимаем, что они довольно сильно облегчают жизнь программистам.

Как мы уже знаем, строки это тоже отдельный вид данных и как оказалось довольно не обычный. И так как программист сталкивается с ними постоянно, то логично предположить, что уже кто-то придумал за нас отдельные методы для работы со строками. И сегодня мы с вами посмотрим какие методы могут сделать нашу работу со строками легкой и не принужденной.

Давайте рассмотрим задачу «Путь к файлу», которую мы уже встречали давным-давно в одном из наших заданий. Как мы знаем, у каждого файла на компьютере есть путь, например, 'C:/user/docs/folder/new\_file.txt'. И нам надо написать программу, которая на место user и new\_file подставит данные, которые введёт пользователь.

Давайте рассмотрим решение данной задачи, которое мы уже делали:

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

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

path = 'C:/' + user\_name + '/docs/folder/' + file\_name + '.txt'

print('Путь к файлу:', path)

И тут всё просто. Использовали операцию конкатенации и вывели строку. Но для продвинутого программиста эта запись будет выглядеть немного странной. И странность для программиста со стажем заключается в том, что при каждой конкатенации строк в памяти python выделяется место, а выделение места, копирование строк и подобные операции замедляют программу и ведут к нерациональному расходу памяти. И здесь мы создаем строку 4 раза, а если у нас все параметры получаются из вне, например, имя диска, имя папки, расширения, то и строка будет создаваться много-много раз. И если папок будет больше чем две и так наши проблемы начнут увеличиваться как снежный ком. И как вы уже догадались программисты и эту проблему решили. И давайте о нем узнаем. Для начала уберем старое решение, и вспомним, что начальная строка выглядели вот так: 'C:/user/docs/folder/new\_file.txt', но в этой строке нам нужно подставить свои значения переменных. А вот тут нам поможет такой метод строк, который называется format. И применяется он следующим образом:

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

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

path = 'C:/{user}/docs/folder/{new\_file}.txt'.format(user=user\_name, new\_file=file\_name)

print('Путь к файлу:', path)

Давайте посмотрим, что у нам тут написано. В конце строки мы дописали через точку ключевое слово format, далее внутри указали параметры user и new\_file, и присвоили им соответственно значения user\_name и file\_name. А в самой строке слова user и new\_file взяли в фигурные скобки. Тогда получается, что в функцию формат мы передаем имя переменной, которую мы выделили в строке и присваиваем ей то значение, которое хотим подставить. Всё довольно просто. И такой код выглядит уже лучше.

На этом фишки функции format не закончились, можно написать ещё вот таким образом:

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

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

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

 user=user\_name,

 new\_file=file\_name

)

path\_2 = 'C:/{0}/docs/folder/{1}.txt'.format(

 user\_name,

 file\_name

)

print('Путь к файлу:', path)

Здесь имена переменных стоят по порядку, и в строке они перечисляются по порядку, начиная с 0.

Давайте рассмотрим ещё один способ написания строк со вставкой переменных (f-strings):

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

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

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

 user=user\_name,

 new\_file=file\_name

)

path\_2 = 'C:/{0}/docs/folder/{1}.txt'.format(

 user\_name,

 file\_name

)

path\_3 = f'C:/{user\_name}/docs/folder/{file\_name}.txt'

print('Путь к файлу:', path)

Этот метод является наиболее лаконичным и наглядным, а еще этот метод работает быстрее. Однако, если есть такой хороший f-strings, то зачем нам format. Всё так. Но в ряде случаев format является более удобной формой, и как правило это относится к строкам, которые генерируется вне самой программы.