Спецсимволы, экранирование символов, raw-строки

Теперь, когда познакомились со строками и их методами, пришло время узнать, какие специальные символы могут содержать строки в Питоне. С одним из них мы уже сталкивались – это символ перевода строки: ‘\n’

Я напомню, например, когда задается многострочная строка:

```
text = """hello
python"""
```

то в ней автоматически добавляет этот символ перевода между строками:

```
'hello\npython'
```

Причем, это один символ, хотя он и выглядит как два символа: обратный слеш и n. Мы в этом легко можем убедиться, если воспользоваться функцией: `len(text)`

Получим значения 12 = 5 + 6 + 1 – как раз число символов в двух строках плюс один символ перевода строки.

Мало того, мы можем его явно прописывать в любой строке, формируя многострочный текст, например, так: `t = "panda needs\npython"`

и, выводя эту строку с помощью функции: `print(t)`

увидим две строки.

Вообще, в строках языка Python можно прописывать следующие спецсимволы:

Обозначение	Описание
\n	Перевод строки
\\	Символ обратного слеша
\’	Символ апострофа
\"	Символ двойной кавычки
\a	Звуковой сигнал
\b	Эмуляция клавиши BackSpace
\f	Перевод формата
\r	Возврат каретки
\t	Горизонтальная табуляция (размером в 4 пробела)
\v	Вертикальная табуляция

Все их запоминать совсем не обязательно, на практике используются, в основном: \n, \\, \', \", \t

Значительно реже другие варианты. И, обратите внимание, перед каждым спецсимволом записан символ обратного слеша. Это, своего рода, маркер начала спецсимвола. И если после слеша идет одно из обозначений таблицы, то оно будет восприниматься как некая управляющая последовательность.

Давайте я все это продемонстрирую на примерах. Добавим в строку символ табуляции:
```
t = "\tpanda needs\npython"
```

Теперь функция print() интерпретирует его, как особый горизонтальный отступ:
```
print(t)
```

Если же мы уберем букву t:
```
t = "\panda needs\npython"
```

то при печати увидим просто обратный слеш. В действительности, здесь сработала последняя строчка таблицы: когда не подходит ни одна последовательность, то просто печатается обратный слеш.

Но здесь нужно быть осторожным. Предположим, что мы слово needs хотим заключить в обратные слеши:
```
t = "panda \needs\ python"
```

Однако, при печати:
```
print(t)
```

первый слеш пропадет, так как он будет восприниматься началом спецпоследовательности символа переноса строки. Поэтому, для добавления символа обратного слеша в строку, следует записывать два обратных слеша подряд:
```
t = "panda \\needs\\ python"
```

Тогда в строке они будут автоматически заменены на один символ слеша и при выводе мы это и видим. Это называется экранированием, когда мы символы с двойным назначением записываем, добавляя перед ними обратный слеш. В данном случае получаем двойной слеш.

Часто такие символы следует прописывать при определении путей к файлам. Как мы знаем, в ОС Windows маршруты имеют вид:

D:\Python\Projects\stepik\tex1.py

Здесь фигурируют обратные слеши для разделения каталогов. Чтобы правильно описать такой путь, слеши следует экранировать:

```
path = "D:\\Python\\Projects\\ste
```

```
s = "Марка вина "Ягодка""
```

Внутри этой строки имеются кавычки. Но эти же самые кавычки определяют начало и конец строки в Python. Поэтому такая запись приведет к синтаксической ошибке. Чтобы все работало корректно, нужно выполнить экранирование кавычек: `s = "Марка вина \"Ягодка\""`

Или, в данном случае, можно было бы использовать одинарные кавычки для определения строки, а внутри нее записать двойные:

```
s = 'Марка вина "Ягодка"'
```

В завершение этого занятия отмечу, что в Python можно задавать, так называемые, сырые (row) строки. Это строки, в которых игнорируются спецпоследовательности и все символы воспринимаются буквально так, как записаны. Например, если взять строку с путем к файлу:

```
path = "D:\\Python\\Projects\\ste
```

то сейчас, при отображении, мы видим по одному слешу:

```
print(path)
```

Но, если определить эту же строку, как сырую, добавив букву r перед ней: `path = r"D:\\Python\\Projects\\st`

то при печати увидим по два слеша, именно так, как прописали. Поэтому, в таких строках можно убрать спецопределения и записывать строку буквально так, как она должна выглядеть:

```
path = r"D:\Python\Projects\stepi
```