Raw strings in Python

Posted: 2021-09-06 / Tags: Python, String

In Python, strings prefixed with `r` or `R`, such as `r'...'` and `r"..."`, are called raw strings and treat backslashes `\` as literal characters. Raw strings are useful when handling strings that use a lot of backslashes, such as Windows paths and regular expression patterns.

This article describes the following contents.

Escape sequences

In Python, characters that cannot be represented in a normal string (such as tabs, line feeds. etc.) are described using an escape sequence with a backslash `\` (such as `\t` or `\n`), similar to the C language.

- 2. Lexical analysis - String and Bytes literals — Python 3.9.7 documentation

```
s = 'a\tb\nA\tB'
print(s)
# a b
# A B
```

source: raw_string_escape.py

Raw strings treat backslashes as literal characters

Strings prefixed with `r` or `R`, such as `r'...'` and `r"..."`, are called raw strings and treat backslashes `\` as literal characters. In raw strings, escape sequences are not treated specially.

```
rs = r'a\tb\nA\tB'
print(rs)
# a\tb\nA\tB
```

source: raw_string_escape.py

There is no special type for raw strings; it is just a string, which is equivalent to a regular string with backslashes represented by `\\`.

```
print(type(rs))
# <class 'str'>

print(rs == 'a\\tb\\nA\\tB')
# True
```

source: raw_string_escape.py

In a normal string, an escape sequence is considered to be one character, but in a raw string, backslashes are also counted as characters.

- Get the length of a string (number of characters) in Python

```
print(len(s))
# 7

print(list(s))
# ['a', '\t', 'b', '\n', 'A', '\t', 'B']

print(len(rs))
# 10

print(list(rs))
# ['a', '\\', 't', 'b', '\\', 'n', 'A', '\\', 't', 'B']
```

source: raw_string_escape.py

Windows paths

Using the raw string is useful when representing a Windows path as a string.

Windows paths are separated by backslashes `\`, so if you use a normal string, you have to escape each one like `\\`, but you can write it as is with a raw string.

```
path = 'C:\\Windows\\system32\\cmd.exe'
rpath = r'C:\Windows\system32\cmd.exe'
print(path == rpath)
# True
```

source: raw_string_escape.py

Note that a string ending with an odd number of backslashes raises an error, as described below. In this case, you need to write it in a normal string or write only the trailing backslash as a normal string and concatenate it.

```
path2 = 'C:\\Windows\\system32\\'
```

```
# rpath2 = r'C:\Windows\system32\'
# SyntaxError: EOL while scanning string literal
rpath2 = r'C:\Windows\system32' + '\\'
print(path2 == rpath2)
# True
```

source: raw_string_escape.py

Convert normal strings to raw strings with `repr()`

Use the built-in function `repr()` to convert normal strings into raw strings.

- Built-in Functions - repr() — Python 3.9.7 documentation

```
s_r = repr(s)
print(s_r)
# 'a\tb\nA\tB'
```

source: raw_string_escape.py

The string returned by `repr()` has `'` at the beginning and the end.

```
print(list(s_r))
# ["'", 'a', '\\', 't', 'b', '\\', 'n', 'A', '\\', 't', 'B', "'"]
```

source: raw_string_escape.py

Using slices, you can get the string equivalent to the raw string.

```
s_r2 = repr(s)[1:-1]
print(s_r2)
# a\tb\nA\tB

print(s_r2 == rs)
# True

print(r'\t' == repr('\t')[1:-1])
# True
```

source: raw_string_escape.py

Raw strings cannot end with an odd number of backslashes

Since backslashes escape the trailing `'` or `"`, an error will occur if there are an odd number of backslashes `\` at the end of the string.

- Design and History FAQ - Why can't raw strings (r-strings) end with a backslash? — Python 3.9.7 documentation

```
# print(r'\')
# SyntaxError: EOL while scanning string literal

print(r'\\')
# \\

# print(r'\\\')
# SyntaxError: EOL while scanning string literal
```

source: raw_string_escape.py

Sponsored Link

Share

Tweet Share

Related Categories

- Python
- String

Related Articles

- Create a string in Python (single, double, triple quotes, str())
- Check if a string is numeric, alphabetic, alphanumeric, or ASCII
- Concatenate strings in Python (+ operator, join, etc.)
- Remove a part of a string in Python
- Sort a list of numeric strings in Python
- Convert binary, octal, decimal, and hexadecimal in Python
- Convert a string to a number (int, float) in Python
- Write a long string on multiple lines in Python
- Sort a list, string, tuple in Python (sort, sorted)
- Convert a list of strings and a list of numbers to each other in Python

- Convert Unicode code point and character to each other (chr, ord)