

Текстовые литералы

Текстовые литералы

- Текстовый литерал представляет собой последовательность символов, заключенную в одинарные или двойные кавычки. Это самое общее правило. Дальше начинаются частности.

На заметку

- По умолчанию при отображении текстовых значений в области вывода обычно используются одинарные кавычки (если по контексту команды текст должен отображаться в кавычках), даже если исходный литерал создавался с использованием двойных кавычек.

Текстовые литералы

- Первый вопрос, который возникает, связан тем, как в самом текстовом литерале использовать двойные или одинарные кавычки (речь о ситуации, когда текст, формирующий текстовый литерал, содержит одинарные или двойные кавычки). Самый простой (но не всегда применимый) выход из ситуации состоит в следующем: если в тексте содержатся двойные кавычки, то весь литерал содержит одинарные кавычки.

Текстовые литералы

- Если же текст содержит одинарные кавычки, то весь литерал можно заключить в двойные кавычки. Правда, такой вариант в силу разных причин тоже может быть неприемлемым. На этот случай существует другая стратегия: мы можем использовать обратную косую черту \ и после нее указать одинарные или двойные кавычки (в зависимости от потребностей). То есть мы можем использовать в текстовом литерале инструкции \' и \" для включения в литерал, соответственно, одинарных и двойных кавычек.

Текстовые литералы

- Сама по себе обратная косая черта \ используется для разбивки литерала на несколько строк в окне редактора кодов. Имеется в виду следующее: если текстовое значение слишком «длинное» для того, чтобы поместиться в одну строку, или в силу каких-то причин его нужно вводить в нескольких строках в окне редактора кодов, то в конце строки, в том месте, где выполняется перенос, ставится символ \. При отображении такого литерала в области вывода в соответствующем месте переноса строки не будет, а символ \ при этом не отображается. Обратная косая черта в литерале служит индикатором того, что перенос текстового значения выполнен осознанно.

На заметку

- Просто так разбить значение текстового литерала на несколько строк не получится — возникнет ошибка. Обратная косая черта ставится в конце строки. После нее не должно быть никаких символов. Если после обратной косой черты какой-то символ все же поставить, то такая комбинация (обратная косая черта и символ после нее) будет интерпретироваться как управляющая инструкция.

Текстовые литералы

- Также обратная косая черта используется в управляющих инструкциях. Управляющая инструкция состоит из обратной косой черты и символа. Такие инструкции обозначают определенное действие или операцию, которые должны выполняться в процесс «отображения» инструкции — например, когда соответствующее текстовое значение передано аргументом функции `print()`. Скажем, инструкция `\n` в текстовом литерале означает, что в процессе отображения этого текста в области вывода с том месте, где размещена инструкция, выполняется переход к новой строке. Другим примером управляющей инструкции является инструкция табуляции `\t`. Выполнение этой инструкции сводится к тому, что курсор перемещается к ближайшей справа позиции табуляции

Подробности

- Область вывода условно разбивается на последовательность позиций, в которых отображаются символы. Позиции табуляции — это равноудаленные позиции в этой последовательности. Обычно позициями табуляции являются 1-я, 9-я, 25-я и так далее (с интервалом, равным 8) позиции. В нашем случае интервал между позициями табуляции определяется настройками среды разработки.

На заметку

- Кроме инструкции перехода к новой строке `\n` и инструкции табуляции `\t`, есть и другие управляющие инструкции. Мы на них останавливаться не будем, поскольку не предполагается их использование. Кроме того, корректность применения этих управляющих инструкций достаточно сильно зависит от среды разработки и версии интерпретатора. Вообще, в некоторых версиях Python если интерпретатор «не узнает» управляющую инструкцию, то она обрабатывается как обычная последовательность символов. Однако следует понимать, что все равно такая ситуация является ошибочной, и нет гарантии, что в будущих версиях Python такой «либерализм» будет сохранен.

Текстовые литералы

- Если нам нужно в самом литерале использовать обратную косую черту (не как часть управляющей инструкции, а как самостоятельный символ), то используют инструкцию `\\`.

Текстовые литералы

- Достаточно удобный способ задавать текстовые литералы базируется на использовании трех пар двойных или одинарных кавычек. Такой текстовый литерал можно вводить (в окне редактора) в нескольких строках (без использования обратной черты в качестве переноса строки), и отображаться он будет так, как введен в окне редактора. Небольшой пример, в котором разными способами создаются текстовые литералы, представлен в листинге на скриншоте чуть ниже:

Текстовые литералы

```
66     a = "Язык 'Python' отличается от языка \"Java\"."
67     print(a)
68     b = 'язык "Java" отличается от языка \'C++\''
69     print(b)
70     c = "Серый\tЖёлтый\tКрасный\nСиний\" \" \" \
71         "\tБелый\tЗелёный"
72     print(c)
73     print("\\0мар Хайям\\")
74     d = """Зачем копить добро в пустыне бытия?
75         Кто вечно жил среди нас? Таких не видал я.
76         Ведь жизнь нам в долг дана, и то – на срок недолгий,
77         А то, что в долг дано, не собственность твоя."""
78     print(d)
79
```

Текстовые литералы

```
Run: main x
C:\Users\Admin\PycharmProjects\EGE_projct\venv\Scripts\python.exe C:/Users/Adm
Язык 'Python' отличается от языка "Java".
язык "Java" отличается от языка 'C++'.
Серый Жёлтый Красный
Синий" Белый Зелёный
\Омар Хайям\
Зачем копить добро в пустыне бытия?
    Кто вечно жил среди нас? Таких не видал я.
        Ведь жизнь нам в долг дана, и то – на срок недолгий,
            А то, что в долг дано, не собственность твоя.

Process finished with exit code 0
```

Текстовые литералы

- В этой программе создается несколько текстовых значений, и затем эти значения отображаются в области вывода. В частности, мы встречаем примеры использования одинарных и двойных кавычек в текстовых значениях, которые записываются в переменные A и B. Значение, которое записывается в переменную C, содержит несколько инструкций табуляции \t, инструкцию перехода к новой строке \n и инструкцию переноса литерала в следующую строку \.

Текстовые литералы

- В команде `print("\\ Омар Хайям \\")` мы использовали текстовый литерал, содержащий двойную обратную косую черту `\\`. С помощью такой комбинации мы отображаем в области вывода одинарную обратную косую черту. Наконец, литерал, присваиваемый в качестве значения переменной `D`, заключается в тройные «двойные кавычки». Такой литерал отображается так, как он введён в окне редактора, с учётом всех переносов и отступов.

Текстовые литералы

- Текстовые литералы можно создавать с использованием специальных префиксов. Префикс — это специальный символ (или символы), который указывается непосредственно перед текстовым литералом. Например, если используется префикс `r` или `R`, то такой литерал обрабатывается в режиме, в котором обратная косая черта `\` интерпретируется как обычный символ (созданный с помощью префикса `r` или `R` литерал называется необработанным или сырым). Скажем, выражение `"\n"` представляет собой литерал, который состоит из инструкции перехода к новой строке `\n`.

Текстовые литералы

- Эта инструкция интерпретируется как один символ. А вот выражение `r"\n"` представляет собой литерал, который состоит из двух символов: это символы `\` и `n`.

Текстовые литералы

- С использованием префикса `f` или `F` создаются форматированные литералы. Главное преимущество форматированных литералов состоит в том, что они могут содержать поля замены. Это специальные инструкции, определяющие, какие значения и в каком формате должны быть добавлены в текстовый литерал в соответствующем месте. Поле замены представляет собой инструкцию, заключенную в фигурные скобки.

Текстовые литералы

- Внутри этих скобок указывается название переменной, значение которой вставляется в соответствующем месте в текстовом литерале. Также в них могут содержаться дополнительные коды, определяющие формат (вид и способ) представления значения в литерале. Детали лучше рассмотреть на конкретном примере. В листинге 5.2 представлена программа, в которой есть примеры использования литералов с префиксами.

Текстовые литералы

```
79
80 p = "\"Java\"\\n\"Python\""
81 print(p)
82 print("Символов:", len(p))
83 z = r "\"Java\"\\n\"Python\""
84 print(z)
85 print("Символов:", len(z))
86 name = "Python"
87 k = f"Язык {name} - простой и понятный"
88 print(k)
89 k = f"Язык {name!r} - простой и понятный"
90 print(k)
91
```

Текстовые литералы

- Сначала в программе переменной A в качестве значения присваивается текстовый литерал `"\Java\\"\n\"Python\""`. Он содержит управляющие инструкции, в том числе и инструкцию переноса строки `\n`. При отображении значения переменной A получаем вполне ожидаемый результат. Переменной B в качестве значения присваивается литерал `r"\Java\\"\n\"Python\""` с префиксом `r` (именно префиксом `r` данный литерал отличается от предыдущего). Но при отображении значения переменной B получаем совершенно иной результат. На этот раз все управляющие инструкции интерпретируются как обычная последовательность символов. Более того, результаты выражений `len(A)` и `len(B)`, определяющие длину соответствующих текстовых значений, различны. Причина та же — в случае с переменной B управляющие инструкции обрабатываются как обычные последовательности символов.

Текстовые литералы

- Еще один пример литерала с префиксом (на этот раз использован префикс `f`) дается командой `C = f"Язык {name} — простой и понятный"`. При этом предварительно переменной `name` присвоено текстовое значение `"Python"`. Текстовое значение переменной формируется так: присваиваемый в качестве значения литерал в месте размещения инструкции `{name}` вставляется значение переменной `name`.

Подробности

- Вместо инструкции `{name}` вставляется то значение переменной `name`, которое она имеет на момент присваивания значения переменной. Если впоследствии значение переменной `name` изменится, то на значение переменной это уже никак не повлияет.

Текстовые литералы

- Это самый простой способ инкапсулировать (вставить) в текстовый литерал значение переменной. В общем случае, как отмечалось, поле замены может содержать инструкции (или коды) форматирования. Например, если вместо инструкции {name} использовать {name !r}, то результат будет несколько иным — в текстовом литерале значение переменной name будет заключено в кавычки. Инструкция !r после имени переменной name означает, что перед вставкой значения переменной используется специальное преобразование. В данном случае вызывается функция repr(), с помощью которой для переменной name вычисляется текстовое представление, и уже оно вставляется в литерал. В результате таких манипуляций значение «обрастает» кавычками.

Подробности

- Практически для каждого объекта предусмотрено или может быть предусмотрено текстовое представление, применимое для отображения в области вывода. Получить такое представление можно с помощью функции `repr()`. Кроме инструкции `!r`, после имени переменной в поле замены можно использовать инструкцию `!s` (в этом случае для выполнения преобразования вызывается функция `str()`) и `!a` (для выполнения преобразования вызывается функция `ascii()`). Во всех этих случаях получаем текстовое представление для значения переменной — разница лишь в алгоритмах получения этого преобразования и способе обработки некоторых символов.

Подробности

- 219 страница учебника