Алексей Васильев

Язык программирования Python

Киев 2019

Лекция 5. Работа с текстом

- Текстовый литерал
- Операции с текстом
- Методы для работы с текстом
- Примеры использования текста

Текстовый литерал

Текстовый литерал представляет собой последовательность символов, заключенная в одинарные или двойные кавычки

Для включения в литерал одинарных и двойных кавычек можем использовать инструкции \' и \"

Обратная косая черта \ используется для разбивки литерала на несколько строк в окне редактора кодов

Двойная косая черта \\ используется для вставки косой черты в литерал

Управляющие символы:

- инструкция \n - переход к новой строке
- инструкции \t - табуляция

Три пары 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

Текстовые литералы

```
A="Язык 'Python' отличается от языка \"Java\"."
print(A)
B='Язык "Java" отличается от языка \'C++\'.'
print(B)
C="Серый\tЖелтый\tКрасный\nСиний\
\tБелый\tЗеленый"
print(C)
print("\\ Омар Хайям \\")
D="""Зачем копить добро в пустыне бытия?
  Кто вечно жил средь 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."""
print(D)
```

Программа (Strings.py)

```
Язык 'Python' отличается от языка "Java".
Язык "Java" отличается от языка 'C++'.
Серый   Желтый  Красный
Синий   Белый   Зеленый
\ Омар Хайям \
Зачем копить добро в пустыне бытия?
  Кто вечно жил средь 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.
```

Префиксы

Префикс — это специальный символ (или символы), который указывается непосредственно перед текстовым литералом

Префикс r или R: литерал (сырой, необработанный) обрабатывается в режиме, в котором косая черта \ интерпретируется как обычный символ

Префиксы f или F используют для создания форматированных литералов. Форматированные литералы могут содержать *поля замены*: специальные инструкции (в фигурных скобках), определяющие, какие значения и в каком формате должны быть добавлены в текстовый литерал в соответствующем месте.

Внутри фигурных скобок указывается *название переменной*, значение которой вставляется в соответствующем месте в текстовом литерале. Также там могут содержаться *дополнительные коды*, определяющие формат (вид и способ) представления значения в литерале

Префиксы

```
# Текстовый литерал без префикса:
A="\"Java\"\n\"Python\""
print(A)
print("Символов:",len(A))
# Текстовый литерал с префиксом:
B=r"\"Java\"\n\"Python\""
print(B)
print("Символов:",len(B))
# Переменная с текстовым значением:
name="Python"
# Текстовый литерал с префиксом:
C=f"Язык {name} - простой и понятный"
print(C)
C=f"Язык {name!r} - простой и понятный"
print(C)
# Переменная с числовым значением:
num=12.34567
# Текстовый литерал с префиксом:
txt=f"Число: {num:9.3f}"
print(txt)
txt=f"Число: {num:09.3f}"
print(txt)
# Новое числовое значение переменной:
num=42
# Формат для отображения целого числа:
txt=f"Число: {num:*>9d}"
print(txt)
```

Программа (UsingPrefix.py)

```
# Формат для отображения
# шестнадцатеричного числа:
txt=f"Число: {num:#09x}"
print(txt)
txt=f"Число: {num:9x}"
print(txt)
txt=f"Число: {num:*<9x}"
print(txt)
# Формат для отображения восьмеричного числа:
txt=f"Число: {num:*^#09o}"
print(txt)
# Формат для отображения двоичного числа:
txt=f"Число: {num:#9b}"
print(txt)
```

```
"Java"
"Python"
Символов: 15
\"Java\"\n\"Python\"
Символов: 20
Язык Python - простой и понятный
Язык 'Python' - простой и понятный
Число:    12.346
Число: 00012.346
Число: *******42
Число: 0x000002a
Число:        2a
Число: 2a*******
Число: **0o52***
Число:  0b101010
```

Инструкции

Инструкция !r означает, что перед вставкой значения используется специальное преобразование: вызывается функция repr(), с помощью которой вычисляется текстовое представление и оно вставляется в литерал. В результате значение "обрастает" кавычками

Символ f в поле замены означает, что отображается число в формате с плавающей точкой.
Инструкция 9.3f: под переменную выделяется поле шириной в 9 позиций, после запятой отображается 3 цифры
Инструкция 09.3f: все "лишние" позиции, выделенные под число, будут заполнены нулями

Инструкция *>9d: отображается целое число (символ d), под которое выделяется 9 позиций. Символ > означает, что выравнивание значения выполняется по правому краю. Символ * означает, что "лишние" позиции будут заполнены символом "звездочка" *

Выравнивание содержимого по левому краю: используют символ <. Выравнивание по центру: символ ^. Символ = задает режим: если число отображается со знаком, то между знаком и цифрами в представлении числа выполняется разрыв (так чтобы число занимало всю выделенную под него область), и этот разрыв заполняется соответствующим символом (указанным перед символом, определяющим способ выравнивания)

Инструкции - 2

Символ # определяет специальный режим для отображения числовых значений. Для каждого числового типа (целые числа, числа с плавающей точкой, комплексные числа) этот режим имеет свои особенности. Например, для целых чисел в двоичном, восьмеричном и шестнадцатеричном представлении данный режим означает, что перед собственно значением числа будет отображаться соответственно префикс 0b, 0o и 0x

Инструкция #09x: отображение значения в шестнадцатеричном виде с выделением 9 позиций под число и заполнением "лишних" позиций нулями. Из-за символа # перед шестнадцатеричным кодом отображается префикс 0x

Инструкция *<9x: значение отображается в шестнадцатеричном формате, под число выделяется 9 позиций, выравнивание выполняется по левому краю, "лишние" символы заполняются "звездочкой" *

Инструкция *^#09o: число отображается в восьмеричном представлении, под число выделяется 9 позиций, в представлении числа отображается префикс 0o (символ # в инструкции), выравнивание выполняется по центру (символ ^ в инструкции), "лишние" позиции заполняются "звездочкой" *

Инструкция #9b: числовое значение отображается в двоичном представлении (символ b в инструкции), под значение выделяется поле шириной в 9 символов, а в представлении используется префикс 0b (символ # в инструкции)

Инструкции - 3

Параметры инструкции:

[1] Символ, которым заполняются "лишние" позиции.

[2] Символ (**<**, **>**, **^** или **=**), определяющий способ выравнивания.

[3] Режим отображения знака для числовых значений: если указан "плюс" **+**, то знак будет отображаться и для положительных, и для отрицательных чисел. Если указан "минус" **-**, то знак будет отображаться только для отрицательных чисел.

[4] Можно указать пробел: в этом случае для положительных чисел перед числом делается отступ, а отрицательные числа отображаются со знаком.

[5] Символ **#**: специальный режим отображения значений.

[6] Если для числовых значений указан **0**, то "лишние" позиции заполняются нулями.

[7] Числовое значение: ширина поля для отображения значения.

[8] Символ (запятая **,** символ подчеркивания **_**) для выделения тысячных разрядов.

[9] Через точку указывается точность отображения числового значения.

[10] Символ, который задает тип отображаемого значения: **s** (тексту), **b** (двоичное представление числа), **c** (символ — целочисленное значение интерпретируется как код символа), **d** или **n** (целое число), **o** (восьмеричное представление для числа), **x** или **X** (шестнадцатеричное представление для числа), **e** или **E** (экспоненциальное представление для действительного числа), **f** или **F** (число в формате с плавающей точкой), **g** или **G** (общий формат — способ отображения числового значения зависит от фактического значения числа), **%** (проценты).

Метод format()

Программа (UsingFormat.py)

```
A="Число {}, текст {} и снова число {}"
txt=A.format(123,"Python",321)
print(txt)
txt="Число {0} - это {0:b} или {0:x}".format(42)
print(txt)
txt="Код: {0:05d}, символ: {0:*^5c}".format(65)
print(txt)
txt="Число: {:_>+20.3E}".format(123.468)
print(txt)
B="{0:_{2}{1}s}"
num=6
for k in range(1,num+1):
    print(B.format("*",k,">"),end="")
    print(" "*(2*(num-k)),end="")
    print(B.format("*",k,"<"))
```

```
Число 123, текст Python и снова число 321
Число 42 - это 101010 или 2a
Код: 00065, символ: **A**
Число: __________+1.235E+02
*           *
_*         *_
__*       *__
___*     *___
____*   *____
_____** _____
```

Операции с текстом

Операции с текстом (A и B - текст):

[1] Конкатенация: A+B

[2] Конкатенация литералов через пробел:
"Изучаем " "Python" (результат "Изучаем Python")

[3] Умножение текст на число: 3*"A" (результат "AAA")

[4] Функция ord(): определение кода символа

[5] Функция chr(): определение символа по коду

[6] Текст можно использовать в операторе цикла for

[7] Для текста можно выполнять срез

Операции с текстом

```
# Исходный текст:
txt="Hello Python"
print(txt)
# Текст в обратном порядке:
A=txt[::-1]
print(A)
# Первое слово в тексте:
B=txt[:5]
print(B)
# Последнее слово в тексте:
C=txt[6:]
print(C)
# Переменная с текстовым значением:
new_txt=""
# Переменная с целочисленным значением:
delta=ord("a")-ord("A")
# Перебор символов в тексте:
for s in txt:
    # Если буква в диапазоне от "a" до "z":
    if(ord(s)>=ord("a") and ord(s)<=ord("z")):
        s=chr(ord(s)-delta)
    # Добавление символа к тексту:
    new_txt+=s
# Текст из больших букв:
print(new_txt)
```

Программа (UsingText.py)

```
Hello Python
nohtyP olleH
Hello
Python
HELLO PYTHON
```

Задание - 1

Напишите программу, в которой на основе введенного текста создается новый текст, в котором соседние символы меняются местами: первый со вторым, третий с четвертым, и так далее

Задание - 2

Напишите программу для шифрования текста: каждый символ заменяется на следующую букву в алфавите (а последняя буква алфавита - на первую)

Методы

upper() - текст, в котором все буквы большие

lower() - текст, из маленьких букв

swapcase() - большие буквы заменены на маленькие, а маленькие — на большие

title() - каждое слово начинается с большой буквы

capitalize() - первое слово начинается с большой буквы

isalnum() - результат **True** если текст непустой и состоит из букв и/или чисел

isalpha() - результат **True** если текст непустой и состоит из букв

isascii() - значение **True** если текст пустой или состоит из символов кодовой таблицы

isdecimal() - результат **True**, если текст непустой и состоит из десятичных цифр

isdigit() - результат **True**, если текст непустой и состоит из цифр

isidentifier() - результат **True**, если текст содержит название зарегистрированного в языке идентификатора или ключевого слова

islower() - значение **True**, если текст непустой и состоит из маленьких букв

isnumeric() - результат **True**, если текст непустой и состоит из числовых символов

isprintable() - значение **True**, если текст пустой или состоит из печатных (тех, которые отображаются в области вывода) символов

isspace() - значение **True**, если текст непустой и состоит из пробелов

istitle() - результат **True**, если текст непустой и каждое слово в тексте начинается с большой буквы

isupper() - значение **True**, если текст непустой и состоит из больших букв

Методы - 2

Поиск символа: методы find(), rfind(), index() и rindex(). Аргумент - искомая подстрока.

Методы find() и index(): результат - индекс первого вхождения подстроки в текст. Если подстроки в тексте нет: метод find() возвращает значение -1, метод index() генерирует исключение класса ValueError.

Методы rfind() и rindex() отличаются от методов find() и index(): выполняется поиск не первого, а последнего хождения подстроки в текст.

count() - количество вхождений подстроки или символа в тексте

endswith() - заканчивается ли текст подстрокой, переданной аргументом

startswith() - начинается ли текст подстрокой, переданной аргументом

expandtabs() - текст, в котором инструкции табуляции заменены на соответствующее количество пробелов

strip(), lstrip() и rstrip(): позволяют удалять начальные и/или конечные символы в тексте

replace() - замена в тексте одной подстроки на другую подстроку

Методы - 3

join() - создает строку на основе списка

partition() - разбивает текст на три части. Аргумент - разделитель для разбивки текста на блоки. Точка разбивки - место, где разделитель встречается первый раз. Результат - кортеж: текст до разделителя, разделитель, текст после разделителя

rpartition() - аналогичен методу **partition()**, но разбивка - в месте, где разделитель последний раз встречается в тексте

split() - результат - список слов в тексте

rsplit() - аналогичен методу **split()**, но разбивка на слова выполняется справа налево

splitlines() - список подстрок, на которые разбивается исходный текст. Разделитель - инструкция перехода к новой строке

center() - строку, выравненная по центру поля, выделенного для отображения текста. Ширина поля (в символах) - аргумент метода

ljust() и **rjust()** - аналогичны методу **center()**, но методом **ljust()** выравнивание выполняется по левому краю, а методом **rjust()** текст выравнивается по правому краю

Примеры

Программа (Text_01.py)

```
txt="Язык PYTHON проще, чем язык JAVA!"
print(txt)
print(txt.upper())
print(txt.lower())
print(txt.swapcase())
print(txt.title())
print(txt.capitalize())
```

```
Язык PYTHON проще, чем язык JAVA!
ЯЗЫК PYTHON ПРОЩЕ, ЧЕМ ЯЗЫК JAVA!
язык python проще, чем язык java!
яЗЫК python ПРОЩЕ, ЧЕМ ЯЗЫК java!
Язык Python Проще, Чем Язык Java!
Язык python проще, чем язык java!
```

Программа (Text_02.py)

```
txt=input("Введите текст: ")
symb=input("Какую букву найти? ")
num=txt.count(symb)
if num==0:
    print("Такой буквы в тексте нет!")
else:
    print(f"В тексте {num} букв(ы) '{symb}'")
```

```
Введите текст: Программировать нужно правильно
Какую букву найти? р
В тексте 4 букв(ы) 'р'
```

```
Введите текст: Программировать нужно правильно
Какую букву найти? ы
Такой буквы в тексте нет!
```

Примеры - 2

Программа (Text_03.py)

```
txt=input("Введите текст: ")
symb=input("Какую букву найти? ")
num=txt.find(symb)
L=[]
while num!=-1:
    L.append(num)
    num=txt.find(symb,num+1)
if len(L)==0:
    print("Такой буквы в тексте нет!")
else:
    print(f"Позиции буквы '{symb}' в тексте: {L}")
```

```
Введите текст: Программировать нужно правильно
Какую букву найти? р
Позиции буквы 'р' в тексте: [1, 4, 9, 23]
```

```
Введите текст: Программировать нужно правильно
Какую букву найти? ы
Такой буквы в тексте нет!
```

Примеры - 3

Программа (Text_04.py)

```
txt="Мы изучаем язык Python"
print(txt)
A=txt.replace(" ","_*_")
print(A)
B=txt.replace(" ","\n")
print(B)
C=txt.replace(" "," не ",1).replace("Python","Java")
print(C)
D=txt.replace("язык ","")
print(D)
```

```
Мы изучаем язык Python
Мы_*_изучаем_*_язык_*_Python
Мы
изучаем
язык
Python
Мы не изучаем язык Java
Мы изучаем Python
```

Примеры - 4

Программа (Text_05.py)

```
A=["Alpha","Bravo","Charlie"]
print("Список:",A)
B=", ".join(A)
print("Текст:",B)
C=B.split(", ")
print("Снова список:",C)
txt="""Прошли года
И в свете лет
Есть мудрость
А вот счастья нет"""
print(txt)
D=txt.splitlines()
print(D)
```

```
Список: ['Alpha', 'Bravo', 'Charlie']
Текст: Alpha, Bravo, Charlie
Снова список: ['Alpha', 'Bravo', 'Charlie']
Прошли года
И в свете лет
Есть мудрость
А вот счастья нет
['Прошли  года',  'И  в  свете  лет',  'Есть  мудрость',  'А  вот счастья нет']
```

Примеры - 5

Программа (Text_06.py)

```
txt="PYTHON"
num=20
A=txt.ljust(num,"_")
B=txt.center(num)
C=txt.rjust(num,"*")
print("|",A,"|")
print("|",B,"|")
print("|",C,"|")
```

```
| PYTHON______________ |
|         PYTHON       |
| **************PYTHON |
```

Домашнее задание

[1] Напишите программу, в которой пользователь вводит два текстовых значения, и на их основе создается новый текст. В этот новый текст поочередно включаются буквы из текстов, введенных пользователем. Когда один из текстов заканчивается, в качестве символов из этого текста используется "звездочка" *

[2] Напишите программу, в которой на основе текста, введенного пользователем, создается новый текст. По сравнению с исходным, в нем слова расположены в обратном порядке. Под словами подразумевать блоки текста, разделенные пробелами.