Лабораторная работа №3

Работа со строками в Python

Цель работы: получение практических навыков работы со строковыми объектами в Python.

Теоретическая часть

Строки в языке питон являются неизменяемыми объектами. Это значит, что в момент запуска скрипта ячейка памяти, на которую ссылается строка, не меняется. Из этого следует, что символы этого типа данных не могут меняться или переприсваиваться.

Они поддерживают различные операторы, могут сравниваться, включают большое количество встроенных методов. Любые действия над строками не модифицируют их, а создают новый объект.

Для того, чтобы приведенные в коде символы интерпретировались как строка, их нужно обернуть в кавычки. Имеется 4 способа это сделать:

– одинарные кавычки

– двойные кавычки

– тройные одинарные кавычки (многострочный текст с сохранением форматирования)

– тройные двойные кавычки (многострочный текст с сохранением форматирования)

Операция сложения или конкатенирования складывает две или более строк в одну. В качестве аргументов могут выступать только строки.

```
  In[1]: 'Hello ' + 'Py'
Out[1]: 'Hello Py'
  In[2]: 'Hello ' + 4
Out[2]: TypeError: can only concatenate str (not "int") to str
```

Операция умножения, применяемая к строкам, подразумевает ее дублирование определенное количество раз (в соответствии со значением числа, которое передано в качестве параметра. Если передать отрицательное число или ноль, то вернется пустая строка).

```
  In[3]: 'Hello ' * 4
Out[3]: 'Hello Hello Hello Hello '
  In[4]: 'Hello ' * -4
Out[4]: ''
```

Всё в Pythom можно сделать строкой (так как все объекты имеют свойство **__str__**, первично унаследованное от прародителя всех классов **object**).

Поэтому любое число, список, объект или функцию можно представить в виде строки.

```
  In[5]: str(1.11)
Out[5]: '1.11'
  In[6]: str(2 + 3j) # комплексное число
Out[6]: '(2+3j)'
  In[7]: str([2, 4, 8])
Out[7]: '[2, 4, 8]'
  In[8]: str({1: 100, 2: 200})
Out[8]: '{1: 100, 2: 200}'
  In[9]: str(sorted)
Out[9]: '<built-in function sorted>'
  In[10]: str(None)
Out[10]: 'None'
```

Не всегда приведение объекта к строке дает нам полезную информацию, но такая возможность имеется.

Метод **encode()** отображает строку в заданной кодировке. По умолчанию используется **utf-8**. Можно использовать и другие варианты кодировки, но при невозможности закодировать строку возникнет ошибка.

```
  In[11]: 'cat'.encode(encoding='ascii')
Out[11]: b'cat'
  In[12]: 'кот'.encode(encoding='utf-8')
Out[12]: b'\xd0\xba\xd0\xbe\xd1\x82'
  In[13]: 'кот'.encode(encoding='cp1251')
Out[13]: b'\xea\xee\xf2'
  In[14]: 'кот'.encode(encoding='utf32')
Out[14]: b'\xff\xfe\x00\x00:\x04\x00\x00>\x04\x00\x00B\x04\x00\x00'
  In[15]: 'кот'.encode(encoding='ascii')
Out[15]: UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-2:
ordinal not in range(128)
```

Срезы строк возвращают новый объект на основании переданных параметров. В общем виде синтаксис следующий:

```
СТРОКА[начало:конец:шаг]
```

Параметры можно опускать, поддерживается и отрицательная индексация (с конца). При любых значениях индексов (даже за рамками длины

строки) возвратится объект (в том числе пустой), и никогда не возникнет ошибка.

Шаг означает пропуски символов в строке: по умолчанию 1 – не пропускать символы, а, например, 3 – брать в срез только каждый третий элемент строки.

Отрицательный шаг берет символы в обратном порядке (от конца текста к началу).

```
  In[16]: 'Это строка'[1]
Out[16]: 'т'
  In[17]: 'Это строка'[::2]
Out[17]: 'Эосрк'
  In[18]: 'Это строка'[::-1]
Out[18]: 'акортс отЭ'
  In[19]: 'Это строка'[1:4]
Out[19]: 'то '
  In[20]: 'Это строка'[:34:3]
Out[20]: 'Э ра'
  In[21]: 'Это строка'[-1:-4:-1]
Out[21]: 'ако'
  In[22]: 'Это строка'[:]
Out[22]: 'Это строка'
```

Согласно **PEP-257** строка документации (**docstring**) - это одно- или многострочный строковый литерал, разделенный тройными одинарными кавычками в начале модуля, функции, класса, метода и описывающий, что делает этот объект.

Важное замечание: **docstring** может быть представлен только в тройных одинарных кавычках, независимо от того, расположен ли он в одной строке или на нескольких.

Почти все объекты (функции, классы, модули или методы) имеют описание, которое можно посмотреть. Для этого используется свойство **__doc__**.

```
  In[23]: import math
  In[24]: math.__doc__
Out[24]: 'This module provides access to the mathematical functions\ndefined by the C
standard.'
  In[25]: math.sqrt.__doc__
Out[25]: 'Return the square root of x.'
```

Стоит отметить, что функция **help()** возвращает более полную информацию, нежели свойство **__doc__**.

Задание на лабораторную работу:

Задание 1

Вариант 1: Напишите функцию search_substr(subst, st), которая принимает 2 строки и определяет, имеется ли подстрока subst в строке st.
В случае нахождения подстроки, возвращается фраза «Есть контакт!», а иначе «Мимо!».

Должно быть найдено совпадение независимо от регистра обеих строк.

Вариант 2: Напишите функцию search_substr(subst, st), которая принимает 2 строки и определяет, имеется ли подстрока subst в строке st.
В случае нахождения подстроки, возвращается фраза «Есть контакт!», а иначе «Мимо!».

Должно быть найдено совпадение обеих строк с учетом регистра.

Вариант 3: Напишите функцию search_substr(subst, sm), которая принимает строку и символ и определяет количество указанных символов в данной строке.

Должно быть найдено количество указанных символов с учетом регистра. Строка и символ вводятся с клавиатуры.

Вариант 4: Напишите функцию search_substr(subst, sm), которая принимает строку и символ и определяет количество указанных символов в данной строке.

Должно быть найдено количество указанных символов без учета регистра. Строка и символ вводятся с клавиатуры.

Задание 2

Вариант 1: Требуется определить индексы первого и последнего вхождения буквы в строке.

Для этого нужно написать функцию first_last(letter, st), включающую 2 параметра: letter – искомый символ, st – целевая строка.

В случае отсутствия буквы в строке, нужно вернуть кортеж (**None**, **None**), если же она есть, то кортеж будет состоять из первого и последнего индекса этого символа.

Вариант 2: Требуется определить индексы первого и последнего вхождения буквы в строке для трех букв, введенных с клавиатуры. Регистр букв не учитывать. Строка также вводится с клавиатуры. Результат вывести в как кортежи (буква, индекс первого вхождения, индекс последнего вхождения)

Вариант 3: Требуется определить индексы первого и последнего вхождения буквы в строке для трех букв, введенных с клавиатуры. Строка также вводится с клавиатуры. Учитывать регистр букв. Результат вывести в как кортежи (буква, индекс первого вхождения, индекс последнего вхождения). Для этого нужно написать функцию first_last(letter, st1, st2, st3).

Вариант 4: Требуется определить индексы первого и предпоследнего вхождения буквы в строке для двух букв, введенных с клавиатуры. Строка также вводится с клавиатуры. Регистр букв учитывать.

Задание 3

Вариант 1: На основании предоставленного отрывка текста определить 3 наиболее часто встречаемых символа в нем. Пробелы нужно игнорировать (не учитывать при подсчете). Для выведения результатов вычислений требуется написать функцию top3(st).

Итог работы функции представить в виде строки: «символ – количество раз, символ – количество раз…». Текст любой.

Вариант 2: На основании предоставленного отрывка текста определить 2 наименее часто встречаемых символа в нем. Пробелы нужно игнорировать (не учитывать при подсчете). Для выведения результатов вычислений требуется написать функцию top2(st). Итог работы функции представить в виде строки: «символ – количество раз, символ – количество раз…». Текст любой.

Вариант 3: На основании предоставленного отрывка текста определить 2 наименее часто встречаемых слова в нем. Словом считать сочетание символов в количестве не менее 3х, отделенных пробелами или знаками препинания. Текст любой.

Вариант 4: На основании предоставленного отрывка текста определить 2 наиболее часто встречаемых слова в нем. Словом считать сочетание символов в количестве не менее 3х, отделенных пробелами или знаками препинания. Текст любой.

Задание 4:

Вариант 1: Николай решил вспомнить старые времена.

В свое время было модно писать сообщения с чередующимися заглавной и малой буквами.

Он захотел изобрести функцию, которая будет делать с любой предоставленной строкой аналогичное.

Ваша задача: повторить труд студента newst(st) с учетом того, что пробелы и знаки препинания не должны портить чередование регистра символов (они в этом процессе не учитываются, но возвращаются в итоговой строке).

Вариант 2:

Серебряный век русской литературы будут воспевать вечно. Возьмите в качестве входных данных любое поэтическое произведение. Переставьте в нем слова справа на лево, поменяв регистры букв в начале строки и в конце. Знаки препинания сохранить.

Вариант 3:

Ариадна Андреевна любит читать и сохранять статистику по прочитанным произведениям. Возьмите любой текст. Для него посчитайте:

- количество гласных букв
- количество согласных букв
- количество предложений
- количество предложений, с восклицательным и вопросительным знаком на конце (такие предложения должны быть в тексте).
- количество предложений с числом слов менее 50

Вариант 4:

У Петра Васильевича есть собрание сочинений. Он любит сравнивать произведения между собой. Возьмите два текста и выявите:

- в каком из них больше знаков препинания и сколько каких
- в каком из них больше слов и сколько их
- в каком из них больше предложений и сколько

Задание 5:

Вариант 1: Дмитрий считает, что, когда текст пишут в скобках (как вот тут, например), его читать не нужно.

Вот и надумал он существенно укоротить время чтения, написав функцию, которая будет удалять все, что расположено внутри скобок.

Помогите ленивому Диме разработать функцию shortener(st), которая будет удалять все, что внутри скобок и сами эти скобки, возвращая очищенный текст (скобки могут быть вложенными).

Вариант 2: Насте нравится читать, чтобы ускорить время чтения, она придумала словарь, каждому слову, встречающемуся в тексте, впервые был присвоен код, соответствующий его порядковому номеру (включая предлоги). После формирования словаря Настя закодировала исходный текст так, чтоб никто-никто без словаря не мог его прочитать.

Реализуйте задумку Насти в виде функции. На выходе представьте исходный текст, словарь, закодированный текст.

Вариант 3: Закодируйте текст, меняя каждую 3й символ его числовым кодом. Для хранения кодов символов используйте словарь. На выходе представьте исходный текст, словарь, закодированный текст.

Вариант 4: Просчитайте число вхождений каждого символа в текст и запишите в словарь в пары «символ – количество вхождений». Закодируйте текст, меняя каждую нечетный символ его числовым кодом. На выходе представьте исходный текст, словарь, закодированный текст.