**45 Методы обработки строк**

**Методы**

Метод — это функция, применяемая к объекту, в данном случае — к строке. Метод вызывается в виде:

**Имя\_объекта.Имя\_метода(параметры)**

Например, **S.find("e")** — это применение к строке S метода find с одним параметром "e".

Рассмотрим несколько полезных методов для обработки строк.

**1,2** Методы **upper** и **lower** изменяют регистр всех символов строки, соответственно, на верхний и нижний.

Попробуем:

**str = "это пример 1 строки....wow!!!"**

**print ("str.upper : ",str.upper())**

**str = "ЭТО ПРИМЕР 2 СТРОКИ....WOW!!!"**

**print ("str.lower : ",str.lower())**

**3** **(рис 2-1)** Логический метод **isdigit** проверяет, только ли цифры содержит строка. Он возвращает значение **«истина», если строка состоит только из цифр**, в противном случае будет возвращено значение «ложь». Этот метод можно использовать для того, чтобы определить, является ли строка целым положительным числом. Для отрицательных чисел и для чисел с плавающей точкой этот метод будет возвращать значение «ложь».

**4** **(рис 2-2)**Часто бывает полезен метод **find**. Он возвращает **индекс элемента**, с которого начинается первое вхождение заданной подстроки в строку, или же -1, если заданной подстроки в строке не обнаружено. Для того, чтобы найти последнее вхождение подстроки в строку, можно использовать метод **rfind**. Попробуем:

|  |  |
| --- | --- |
| **find** | **rfind** |
| **S = 'Hello'**  **print(S.find('e')) # вернёт 1**  **print(S.find('ll')) # вернёт 2**  **print(S.find('L')) # вернёт -1** | **S = 'Hello'**  **print(S.find('l')) # вернёт 2**  **print(S.rfind('l')) # вернёт 3** |

**(рис 2-3)** Данные любого из основных типов можно преобразовать в символьную строку, для этого используется функция **str**. Для того, чтобы преобразовать строку в число, можно воспользоваться одной из уже известных нам функций преобразования – **int** или **float**. Однако, если использовать эту функцию для строки, содержимое которой не является числом, то исполнение программы будет завершено ошибкой несоответствия типов. Поэтому прежде, чем преобразовывать строку в число или другой тип данных, нужно проверить ее содержимое.

**5,6** **(рис 2-4)**Методы **split** и **join**, первый из них делит исходную строку на подстроки по заданной строке-разделителю. Результат этого метода – список подстрок исходной строки. Второй метод, напротив, соединяет список строк в одну строку, вставляя между ними заданную строку-разделитель.

**7** Также часто бывает полезен метод **s.replace(а,b)**, который **заменяет** в исходной строке (s) все вхождения одной подстроки (a) на другую подстроку(b).

Если методу replace задать еще один параметр: **s.replace(а,b,x)**, то заменены будут не все вхождения, а только не больше, чем первые x из них. Попробуем:

|  |  |
| --- | --- |
| **replace(а,b)** | **replace(а,b,x)** |
| **print('Hello'.replace('l', 'L')) # вернёт HeLLo** | **print('abrakadabra'.replace('a', 'A', 2))**  **# вернёт 'AbrAkadabra'** |

**8** Метод **count.** Подсчитывает количество вхождений одной строки в другую строку. Форма вызова **s.count(t)** возвращает число вхождений строки **t** внутри строки **s**. При этом подсчитываются только непересекающиеся вхождения, например:

|  |  |
| --- | --- |
| **print('abracadabra'.count('a')) # вернёт 5** | **print(('a' \* 10).count('aa')) # вернёт 5** |

**Сравнение строк**

**(рис 2-5)** Важным нюансом работы со строками является то, что их можно сравнивать между собой. Например, если сравнить строки «Вася» и «Петя», нам ясно, что по алфавиту «Петя» идет позже, значит эта строка в нашем примере больше. Также можно сравнить строки «самосвал» и «самолет». Первый символ, который в них не совпадает, имеет индекс «четыре». Так как буква «С» идет по алфавиту после буквы «Л», то строка «самосвал» будет больше.

Как же компьютер сравнивает символьные строки? **(рис 2-6)** Для сравнения компьютер последовательно сравнивает коды соответствующих символов в кодовой таблице. Важно при этом понимать, что в кодовой таблице раньше любых букв последовательно следуют цифры. Русские буквы следуют последовательно после английских. При этом заглавные буквы, как русские, так и английские, следуют перед строчными. Буква «е», как заглавная, так и строчная, следует вне алфавитов. А символ пробела следует раньше цифр. Из того, что строки можно сравнивать между собой, следует то, что их можно сортировать.

**Мы узнали:** **(рис 2-7)**

· **Символьная строка** – это набор нумерованных символов, которые хранятся в оперативной памяти компьютера под одним именем и располагаются последовательно друг за другом.

· Для обработки строк в языке Python определен ряд методов и операций.

· Символьные строки **можно сравнивать и сортировать**.

**Задание 1-0 Напишем функцию проверки того, является ли содержимое символьной строки числом (по описанию алгоритма).**

1. Назовем нашу функцию **IsNumber**. (**def IsNumber (s):**) Она будет принимать на вход строку, которую нужно проверить. В функцию строка передается в виде своей копии.
2. Если строка пуста, то она не является числом, в этом случае функция вернет значение «ложь».
3. Если строка состоит из одного символа, то для того, чтобы строка являлась числом, этот символ должен быть цифрой. Если это условие выполняется, то функция должна вернуть значение «истина», в противном же случае – «ложь».
4. Если в строке два символа, то для того, чтобы строка являлась числом, это должны быть, соответственно, знак минус и цифра, или же две цифры. Если это условие выполняется, то функция вернет значение «истина», в противном случае – «ложь».
5. Если в строке больше двух символов, то для того, чтобы она являлась числом, ее первый символ должен быть либо знаком минус, либо цифрой, последний же символ должен быть цифрой. Если это условие не выполняется, то функция вернет значение «ложь». Каждый же символ, начиная с позиции с индексом 1 и заканчивая предпоследним, должен быть либо цифрой, либо точкой. Для проверки этого условия запишем цикл с параметром **i**, изменяющимся в диапазоне от 1 до индекса предпоследнего символа строки **s**. В цикле запишем ветвление с условием, что символ строки **s[i]** не является цифрой или точкой. Если это условие выполняется, то функция вернет значение «ложь».
6. Далее проверим, не идет ли в начале строки точка после минуса. Если это условие выполняется, функция вернет значение «ложь».
7. Осталось лишь проверить количество точек в строке **s**. Если их больше одной, то строка не является числом. Заведем для этого переменную-счетчик **k** и присвоим ей значение 0, так как запятые мы еще не считали. Далее напишем цикл с символьным параметром **c**, в котором будем перебирать символы строки **s**. Если на каком-то шаге цикла параметр **c** стал равен точке, то мы увеличим **k** на 1. После цикла в переменной **k** будет храниться количество точек в строке **s**. И мы вернем истинность высказывания о том, что **k < 2**.

Напишите функцию, проверяющую, является ли исходная строка целым числом или нет.

**Задание 1-1 Напишем модуль по формированию новой строки (по описанию алгоритма).**

В строке через запятую перечислены 4 слова. Сформировать из этих слов новую строку, в которой они будут перечислены через пробел и отсортированы по убыванию без учета регистра.

1. Вначале с помощью инструкции **print** выведем на экран запрос на ввод исходной строки. Далее с помощью функции **input** считаем исходную строку в переменную **s**.
2. Так как сортировать слова нужно без учета регистра, приведем строку **s** к нижнему регистру, используя метод **lower**.
3. Теперь сформируем 4 строковые переменные (s1,s2,s3,s4), состоящие из слов строки **s**. Для этого к строке **s** применим метод **split**, в котором в качестве разделителя укажем подстроку, состоящую из запятой.
4. Отсортируем переменные по убыванию и запишем их в этом порядке в другие переменные (с1=max1, с2=max2, c3=max3, c4=max4), воспользуемся операцией сравнения.
5. Сформируем из них результирующую строку, которую обозначим **r**. Для этого используем метод **join**, в котором в качестве строки-разделителя укажем символ «пробел».
6. С помощью инструкции **print** с соответствующим поясняющим сообщением выведем на экран результирующую строку.

**собака,кот,почтальон,гусь собака почтальон кот гусь**

**Задание 2**

Что будет выведено на экран при исполнении инструкций?

**s = 'АРАРАТ'**

**s.replace ('АРА', 'ФА')**

**print (s[0:4])**

**Задание 3**

Что будет выведено на экран при исполнении инструкций?

**s, t = 'Владикавказ', 'Владивосток'**

**if t > s:**

**print (s)**

**else:**

**print (t)**

**Задание 4**

Сопоставьте методы символьных строк, с результатом, который они вернут при **s = 'Самолёт'**?

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **'САМОЛЁТ'** |  | **s.split ('о')** |
| **2** | **-1** |  | **s.upper ()** |
| **3** | **4** |  | **s.rfind ('Лёт')** |
| **4** | **'Сам', 'лёт'** |  | **s.isdigit ()** |
| **5** | **False** |  | **s.find ('лёт')** |

**Задание 5**

В какой из инструкций допущена ошибка, если s = 'мама мыла раму'?

* **t = s[0:4]**
* **t = s[3]**
* **s[3] = ' '**
* **s = s[0:4]**

**Задание 6**

Что будет выведено на экран при исполнении инструкций?

**s = '-5'**

**print (s.isdigit ())**

**Задание 7**

Что будет выведено на экран при исполнении инструкций?

**a = 'flimsy'**

**b = 'miserable'**

**c = b[0:1] + a[2:]**

**print(c)**

**Задание 8**

Дана строка **s** и целые числа **a, b, c, d.** Вывести строку, полученную из **s**, начиная с символа с индексом **a** до **b**, и с **c** до **d** включительно (между “кусками” - пробел).

**Sample Input: abcdefghijklmnopqrstuvwxyz 2 5 8 15**

**Sample Output: cdef ijklmnop**

**Задание 9**

Есть строка **s = "abcdefghifdgffdgfgsddasdsfgjk"**.

Запишите строки (без кавычек), полученные в результате следующих операций:

**s[3:7] s[:9] s[3:] s[::-1]**

**s[-4:] s[:-9] s[-1:-11:-2]**

**Задание 10** **«Число десятков» (PT)**

Дано натуральное число. Найдите число десятков в его десятичной записи.