Занятие 19. Основные операции со строковыми типами данных

План занятия

1. СТРОКОВЫЙ ТИП ДАННЫХ И ВВОД-ВЫВОД СТРОК

В строковых данных можно хранить фамилии и имена, адреса, названия товаров и т.д. Чтобы задать в программе строку необходимо заключить последовательность символов в двойные кавычки:

```
a="У лукоморья дуб зеленый"
b=""          # Это пустая строка
```

Считать строковую переменную с клавиатуры можно при помощи стандартной функции input, при этом пользователь также должен заключить вводимую строку в кавычки. Пример:

```
a=input()
```

2. КОНКАТЕНАЦИЯ СТРОК

Главная операция со строками — их объединение, когда одна строка записывается после другой строки. Эта операция называется конкатенацией и для нее используется оператор +:

```
a="abc"
b="xyz"
c=a+b
print c       # Будет напечатано abcxyz
```

Также при помощи оператора * можно многократно повторять одну и ту же строку. "Умножать" строку можно только на натуральное число:

```
print "="*20  # Будет напечатано 20 знаков "=" подряд
```

3. ВЫБОР ПОДСТРОКИ

Со строками можно работать во многом как и со списками, рассматривая строку, как список отдельных символов. Функция len для строки возвращает ее длину. Все символы строки пронумерованы начиная с

0, а также в обратном порядке (с конца) начиная с -1. Например, если s="abc", то len(s) возвращает значение 3, s[0] и s[-3] — это символ "a", s[1] и s[-2] — это "b", s[2] и s[-1]— это символ "c". Обращение к несуществующему элементу строки приведет к ошибке IndexError: string index out of range.

Символ строки: a b c

Его индекс: 0 1 2

или такой: -3 -2 -1

Помимо обращения к отдельным символам строки, из строки можно выбрать подстроку в виде s[x:y], где s — идентификатор строки, x — номер первого символа подстроки, y — номер первого символа, который не будет включен в подстроку. Пример:

```
s="abcdefgh"
print s[2:6]   # Будет напечатано cdef
```

В этом примере из строки s выбирается подстрока s[2:6], состоящая из символов с индексами от 2 до 5.

При выборе подстроки можно не указать номер начального символа, тогда считается, что он равен 0 (то есть подстрока начинается с начала строки) или последний символ (тогда он считается равным длине строки, то есть подстрока заканчивается концом исходной строки). Также можно использовать отрицательные числа для задания номеров символов. Таким образом, извлечь из строки s первые три символа можно так: s[:3], а последние три символа: s[-3:]. Указание недопустимых для данной строки индексов при выборе подстроки допускается и не приводит к ошибке.

В отличии от списков, изменять отдельные элементы элементы (то есть символы в строке) нельзя. Вместо этого можно пользоваться конкатенацией строк. Например, если хочется в строке s заменить первый символ на букву "a", то это можно сделать при помощи конкатенации строки "a" и всей строки s за исключением первого символа: s="a"+s[1:].

4. ПРИМЕР

Рассмотрим пример программы, которая считывает строку с клавиатуры и находит в ней первое слово (то есть все символы до первого пробела.

```
s=input()              # 1
for i in range(len(s)):      # 2
  if s[i]==" ":              # 3
    print s[:i]              # 4
    break                    # 5
```

В первой строке программы в переменную s считывается строка с клавиатуры. Затем организовывается цикл, в котором переменная s меняется от 0 до len(s)-1, то есть переменная i принимает подряд все номера символов в строке. В третьей строке программы проверяется, является ли i-й символ строки пробелом (то есть совпадает ли он со строкой из одного пробела). Если проверяемое условие истинно, то был найден первый слева строки пробел, на экран печатается первые i символов строки, то есть все символы с начала строки до найденного пробела, после чего выполнение цикла завершается инструкцией break.

5. СРАВНЕНИЕ СТРОК

Строки можно сравнивать между собой на равенство или неравенство при помощи операторов сравнения == и !=. Также можно сравнивать строки при помощи операторов <, <=, >, >=, при этом строки сравниваются в лексикографическом (то есть алфавитном) порядке. Например, верны будут следующие неравенства: "a"<"aa"<"ab"<"abc"<"c"<"z". При этом сравнение двух отдельных символов осуществляется сравнением номеров, которыми они кодируются в двоичном представлении, в частности, все заглавные

латинские буквы будут "меньше" строчных латинских букв, а порядок сравнения русских букв (в системе Linux) будет отличаться от алфавитного.

Приведем пример программы, которая по данной строке выбрасывает из нее все заглавные буквы латинского алфавита:

```
s=input()                    # 1
t=""                         # 2
for i in range(len(s)):      # 3
   if s[i]>="A" and s[i]<="Z":   # 4
      continue                  # 5
   t=t+s[i]                     # 6
print t                         # 7
```

В первой строке программы мы считывает строку s, затем заводим пустую строку t, в которую будем по очереди копировать все символы строки s при условии, что они не являются заглавными буквами. Далее в строке 3 организовывается цикл, в котором переменная i принимает всевозможные значения индексов элементов строки s, то есть от 0 до len(s)-1. Внутри цикла мы проверяем, является ли символ s[i] заглавной буквой, сравнив его со строками "A" и "Z" (строка 4 программы). В случае истинности проверяемого условия пропускаем этот символ (строка 5 программы), иначе выполняется строка 6, в которой в конец строки t дописывается символ s[i]. Завершается программа выводом строки t на экран.

6. УПРАЖНЕНИЯ

Упражнение 1. Из введённой строки удалить пробелы, а запятые заменить точками.

Упражнение 2. Посчитать количество содержащих букву "о" слов в строке .