Программирование на языке Python

§ 66. Символьные строки

Символьные строки

Начальное значение:

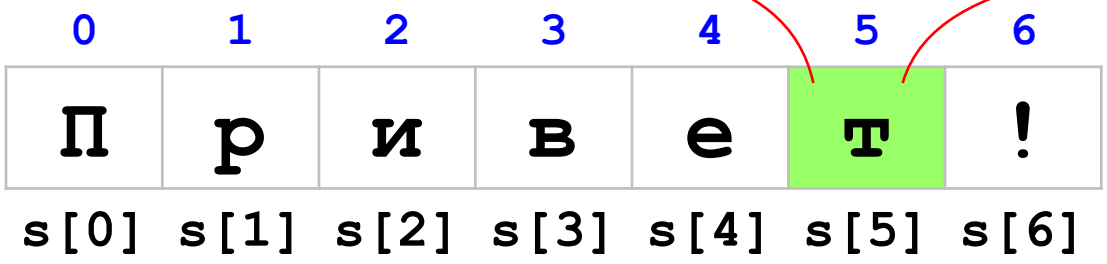
```
s = "Привет!"
```

Вывод на экран:

```
print ( s )
```

```
print ( s[5] )
```

```
print ( s[-2] )
```



s[len(s)-2]

Длина строки:

```
n = len ( s )
```

Строка – это последовательность символов!

Символьные строки

Ввод с клавиатуры:

```
s = input ( "Введите имя: " )
```

Изменение строки:

```
s[4] = "a"
```

Строка – это неизменяемый объект!

... но можно составить новую строку:

```
s1 = s + "a"
```

Символьные строки

Задача: заменить в строке все буквы "а" на буквы "б".

Строка – это неизменяемый объект!

```
s = input( "Введите строку:" )
s1 = ""    # строка-результат
for c in s:
    if c == "а":
        c = "б"
    s1 = s1 + c
print( s1 )
```

перебрать все символы в строке

добавить символ к строке-результату

Задачи

«А»: Ввести с клавиатуры символьную строку и заменить в ней все буквы «а» на «б» и все буквы «б» на «а» (заглавные на заглавные, строчные на строчные).

Пример:

```
Введите строку:
ааббААББссСС
Результат:
ббааББААссСС
```

Задачи

«B»: Ввести с клавиатуры символьную строку и определить, сколько в ней слов. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

Пример:

```
Введите строку:
  Вася  пошел      гулять
Найдено слов: 3
```

Сравнение строк

```
print( "Введите 2 строки:" )
s1 = input()
s2 = input()

if s1 == s2:
  print( s1, "=", s2 )
elif s1 < s2:
  print( s1, "<", s2 )
else:
  print( s1, ">", s2 )
```

Сравнение строк

	А	Б	...	Ё	...	Ю	Я
CP-1251	192	193	...	168	...	222	223
UNCODE	1040	1041	...	1025	...	1070	1071

	а	б	...	ё	...	ю	я
CP-1251	224	225	...	184	...	254	255
UNCODE	1072	1073	...	1105	...	1102	1103

5STEAM < STEAM < Steam < steam

steam < ПАР < Пар < пАр < пар < парк

Операции со строками

Объединение *(конкатенация)* :

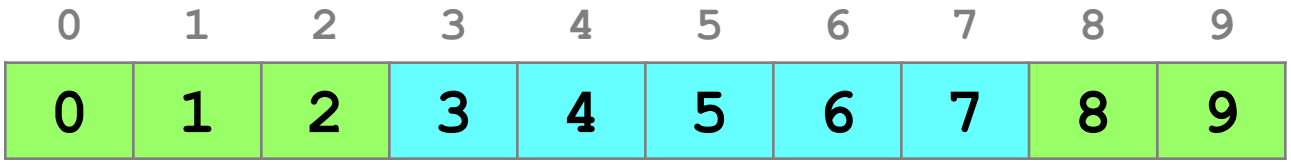
```
s1 = "Привет"
s2 = "Вася"
s  = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

Срезы:

```
s = "0123456789"
s1 = s[3:8]              # "34567"
```

этот символ не входит!



Операции со строками

Срезы:

```
s = "0123456789"
s1 = s[:8]                  # "01234567"
```

от начала строки

```
s = "0123456789"
s1 = s[3:]                  # "3456789"
```

до конца строки

```
s1 = s[::-1]                # "9876543210"
```

реверс строки

Операции со строками

Срезы с отрицательными индексами:

```
s = "0123456789"
s1 = s[:-2]              # "01234567"
```

N-2

```
s = "0123456789"
s1 = s[-6:-2]            # "4567"
```

N-6 N-2

Удаление и вставка символов

Строка – это неизменяемый объект!

Удаление:

```
s = "0123456789"
s1 = s[:3] + s[9:]   # "0129"
```

"012" "9"

Вставка:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
```

"012ABC3456789"

Стандартные функции

Верхний/нижний регистр:

```
s = "aAbBcC"
s1 = s.upper()   # "AABBCC"
s2 = s.lower()   # "aabbcc"
```

Проверка на цифры:

```
s = "abc"
print( s.isdigit() )   # False
s1 = "123"
print( s1.isdigit() )   # True
```

… и много других.

Поиск в строках

```
s = "Здесь был Вася."
n = s.find ( "с" )    # n = 3
if n >= 0:
   print ( "Номер символа", n )
else:
   print ( "Символ не найден." )
```

Находит первое слева вхождение подстроки!

Поиск с конца строки:

```
s = "Здесь был Вася."
n = s.rfind ( "с" )     # n = 12
```

Пример обработки строк

Задача: Ввести имя, отчество и фамилию. Преобразовать их к формату «фамилия-инициалы».

Пример:

```
Введите имя, отчество и фамилию:
Василий Алибабаевич Хрюндиков
Результат:
Хрюндиков В.А.
```

Алгоритм:

- найти первый пробел и выделить имя
- удалить имя с пробелом из основной строки
- найти первый пробел и выделить отчество
- удалить отчество с пробелом из основной строки
- «сцепить» фамилию, первые буквы имени и фамилии, точки, пробелы…

`Алибабаевич Хрюндиков`

`Хрюндиков`

`Хрюндиков В.А.`

Пример обработки строк

```
print ( "Введите имя, отчество и фамилию:" )
s = input()
n = s.find ( " " )
name = s[:n]     # вырезать имя
s = s[n+1:]
n = s.find ( " " )
name2 = s[:n]      # вырезать отчество
s = s[n+1:]        # осталась фамилия
s = s + " " + name[0] + "." + name2[0] + "."
print ( s )
```

Пример обработки строк

Решение в стиле Python:

```
print ( "Введите имя, отчество и фамилию:" )
s = input()
fio = s.split()
s = fio[2] + " " + fio[0][0] + "." + fio[1][0] + "."
print ( s )
```

Василий	Алибабаевич	Хрюндиков
fio[0]	fio[1]	fio[2]

Задачи

«А»: Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр Семёнович
П.С. Иванов
```

Задачи

«B»: Ввести адрес файла и «разобрать» его на части, разделенные знаком "/". Каждую часть вывести в отдельной строке.

Пример:

```
Введите адрес файла:
C:/Фото/2013/Поход/vasya.jpg
C:
Фото
2013
Поход
vasya.jpg
```

Преобразования «строка» – «число»

Из строки в число:

```
s = "123"
N = int ( s )          # N = 123
s = "123.456"
X = float ( s )        # X = 123.456
```

Из числа в строку:

```
N = 123
s = str ( N )                # s = "123"
s = "{:5d}".format(N)        # s = "  123"

X = 123.456
s = str ( X )                # s = "123.456"
s = "{:7.2f}".format(X)      # s = " 123.46"
s = "{:10.2e}".format(X)     # s = "  1.23e+02"
```

Задачи

«A»: Напишите программу, которая вычисляет сумму трех чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45
Ответ: 60
```

«B»: Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются только знаки «+» или «–»). Выражение вводится как символьная строка, все числа целые.

Пример:

```
Введите выражение:
12-3+45
Ответ: 54
```

Строки в процедурах и функциях

Задача: построить функцию, которая возвращает первое слово в предложении.

```
def firstWord( s ):
  p = s.find( ' ' )
  return s[:p]
    return s
  else:
    return s[:p]
```

? Что плохо?

Однажды весною, в час…

Однажды весною, в час…

Однажды

```
word = firstWord( s )
print( word )
```

Однажды

Замена подстроки

Задача: построить функцию, которая заменяет в строке **s** все вхождения слова-образца **wOld** на слово-замену **wNew**.

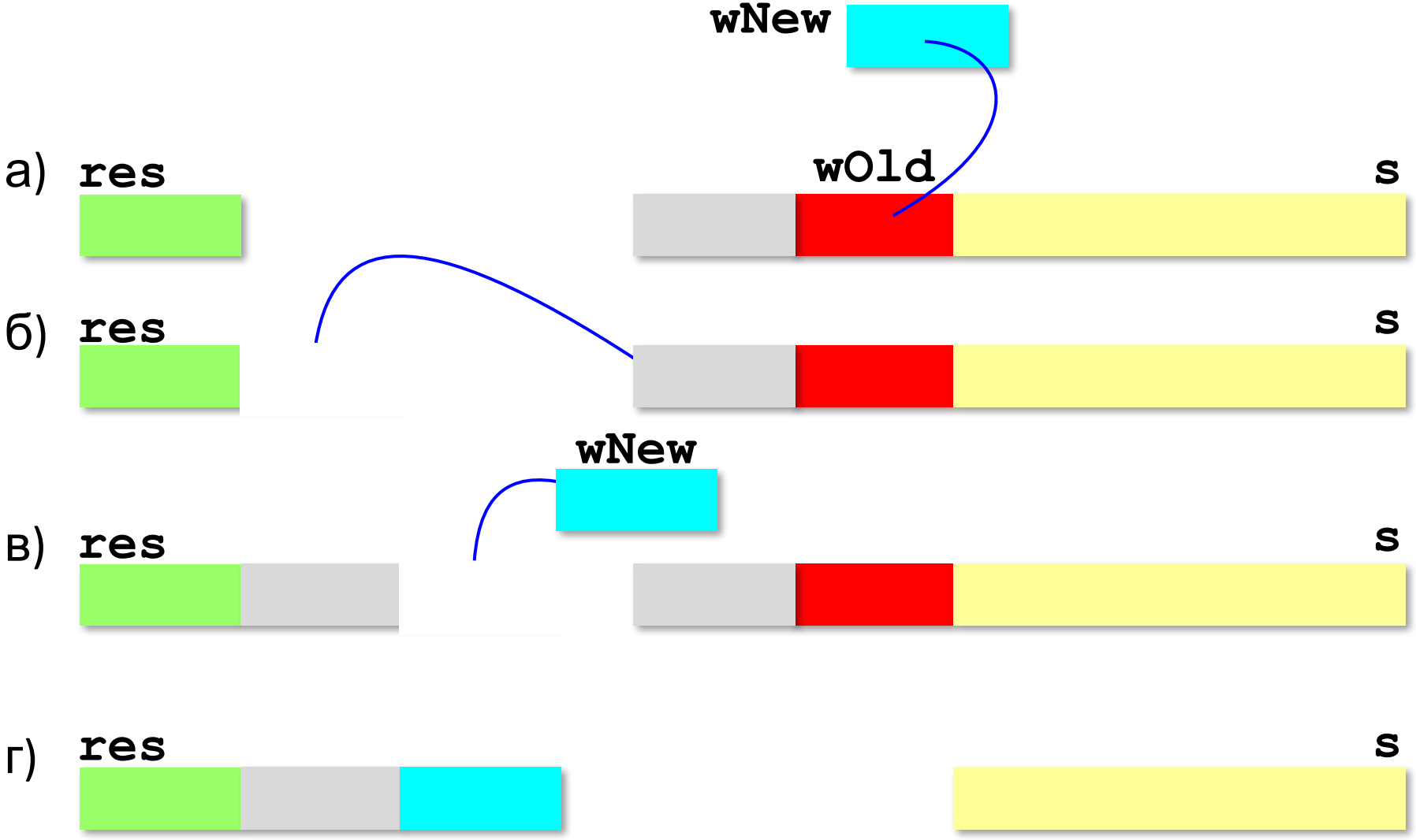
```
пока слово wOld есть в строке s
  удалить слово wOld из строки
  вставить на это место слово wNew
```

Что плохо?

```
wOld: "12"
wNew: "A12B"
```

зацикливание

Замена всех экземпляров подстроки



Замена всех экземпляров подстроки

```
s = "12.12.12"
s = replaceAll ( s, "12", "A12B" )
print( s )
```

рабочая строка s	результат res
"12.12.12"	""

Замена всех экземпляров подстроки

```
def replaceAll ( s, wOld, wNew ):
    lenOld = len(wOld)
    res = ""
    while len(s) > 0:
        p = s.find ( wOld )
        if p < 0:
            res = res + s
            return
        if p > 0: res = res + s[:p]
        res = res + wNew
        if p+lenOld >= len(s):
            s = ""
        else:
            s = s[p+lenOld:]
    return res
```

искать образец

если не нашли

взять начало перед образцом

добавить слово-замену

строка кончилась

взять «хвост»

Замена всех экземпляров подстроки

Встроенная функция:

```
s = "12.12.12"
s = s.replace( "12", "A12B" )
print ( s )
```

Задачи

«А»: Напишите функцию, которая отсекает всю часть строки после первого слова.

Пример:

```
Введите строку: Однажды в студёную зимнюю пору...
Первое слово: Однажды
```

Задачи

«B»: Напишите функцию, которая заменяет расширение файла на заданное новое расширение.

Пример:

```
Введите имя файла: qq
Введите новое расширение: tmp
Результат: qq.tmp
```

Пример:

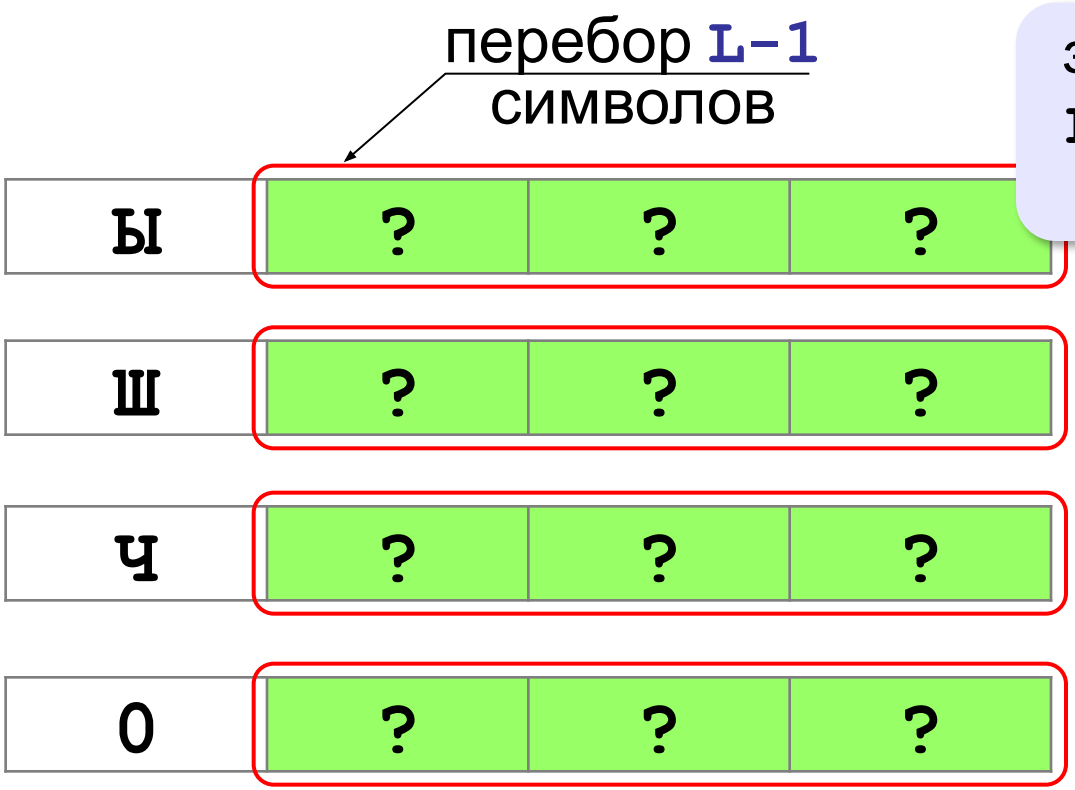
```
Введите имя файла: qq.exe
Введите новое расширение: tmp
Результат: qq.tmp
```

Пример:

```
Введите имя файла: qq.work.xml
Введите новое расширение: tmp
Результат: qq.work.tmp
```

Рекурсивный перебор

Задача. В алфавите языка племени «тумба-юмба» четыре буквы: «**Ы**», «**Ш**», «**Ч**» и «**О**». Нужно вывести на экран все слова, состоящие из **L** букв, которые можно построить из букв этого алфавита.



задача для слов длины **L** сведена к задаче для слов длины **L-1**!

Рекурсивный перебор

```
перебор L символов
   w[0]="Ы"
    # перебор последних L-1 символов
   w[0]="Ш"
    # перебор последних L-1 символов
   w[0]="Ч"
    # перебор последних L-1 символов
   w[0]="О"
    # перебор последних L-1 символов
```

Рекурсивный перебор

алфавит

слово

нужная длина слова

```
def TumbaWords ( A, w, L ):
  if len(w) == L:
    print ( w )
    return
  for c in A:
    TumbaWords ( A, w+c, L )
```

слово полной длины

по всем символам алфавита

```
# основная программа
TumbaWords ( "ЫШЧО", "", 3 )
```

Рекурсивный перебор + счётчик

```
count = 0

def TumbaWords ( A, w, L ):
  global count
  if len(w) == L:
    print ( w )
    count += 1
    return
  for c in A:
    TumbaWords ( A, w+c, L )

TumbaWords ( "ЫШЧО", "", 3 )
print( count )
```

будем менять глобальную переменную

увеличение счётчика

Рекурсивный перебор + условие

```
count = 0
```

```
def TumbaWords ( A, w, L ):
  global count
  if len(w) == L:
    if not "ОО" in w:
      print ( w )
      count += 1
    return
  for c in A:
    TumbaWords ( A, w+c, L )
```

условие отбора

```
TumbaWords ( "ЫШЧО", "", 3 )
print( count )
```

Рекурсивный перебор + условие (функция)

```
def valid ( s ):
  if not "OO" in w:
    return True
  else:
    return False
```

```
return not "OO" in w
```

```
def TumbaWords ( A, w, L ):
  global count
  if len(w) == L:
    if valid(w):
      print ( w )
      count += 1
    return
  for c in A:
    TumbaWords ( A, w+c, L )
```

условие отбора

Задачи

«A»: В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых вторая буква «Ы». Подсчитайте количество таких слов.

«B»: В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, стоящие рядом. Подсчитайте количество таких слов.
Программа не должна строить другие слова, не соответствующие условию.

Сортировка строк

```
aS = []     # пустой список строк
print ( "Введите строки для сортировки:" )
while True:
   s1 = input()
   if s1 == "": break
   aS.append ( s1 ) # добавить в список
aS.sort()           # сортировка
print ( aS )
```

Задачи

«А»: Вводится 5 строк, в которых сначала записан порядковый номер строки с точкой, а затем – слово. Вывести слова в алфавитном порядке.

Пример:

```
Введите 5 строк:
1. тепловоз
2. арбуз
3. бурундук
4. кефир
5. урядник
Список слов в алфавитном порядке:
арбуз, бурундук, кефир, тепловоз, урядник
```

Задачи

«B»: Вводится несколько строк (не более 20), в которых сначала записан порядковый номер строки с точкой, а затем – слово. Ввод заканчивается пустой строкой. Вывести введённые слова в алфавитном порядке.

Пример:

```
Введите слова:
1. тепловоз
2. арбуз

Список слов в алфавитном порядке:
арбуз, тепловоз
```