Программирование (Python)

§ 20. Символьные строки

Что такое символьная строка?

Символьная строка – это последовательность символов.

- строка – единый объект
- длина строки может меняться во время работы программы

Символьные строки

Присваивание:

```
s = "Вася пошёл гулять"
```

Ввод с клавиатуры:

```
s = input()
```

Вывод на экран:

```
print(s)
```

Длина строки:

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

length – длина

Сравнение строк

```
print("Введите пароль: ")
s = input()
if s == "sEzAm":
  print("Слушаюсь и повинуюсь!")
else:
  print("Пароль неправильный")
```

Какой правильный пароль?

Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

Сравнение строк

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

Что выведет?

паровоз < пароход

первые отличающиеся буквы

в < х!

Сравниваем с начала: паровоз
пароход

«в»: код 1074

«х»: код 1093

Обращение к символу по номеру

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

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

s[len(s)-2]

0	1	2	3	4	5	6
П	р	и	в	е	т	!
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]

Символы нумеруются с нуля!

составить «кот»

```
s = "информатика"
kot = s[-2]+s[3]+s[-4]
```

Посимвольная обработка строк

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

Строка неизменна!

Задача. Ввести строку и заменить в ней все буквы «э» на буквы «е».

строим новую строку!

```
sNew = ""
for i in range(len(s)):
    if s[i] == "э":
        sNew += "е"
    else:
        sNew += s[i]
```

для каждого символа строки

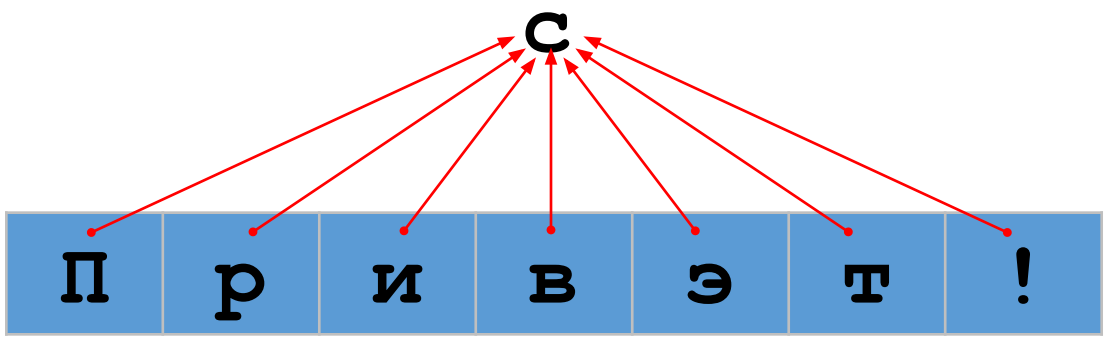
0	1	2	3	4	5	6
П	р	и	в	э	т	!

`len(s)-1`

Цикл перебора символов

```
sNew = ""
for c in s:
    if c == "э":
        sNew += "е"
    else:
        sNew += c
```

перебрать все символы строки



Операции со строками

Объединение *(конкатенация)* :

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

"Привет, Вася!"

Умножение:

```
s = "АУ"
s5 = s*5
```

s5 = s + s + s + s + s

АУАУАУАУАУ

? Что получим?

Срезы строк (выделение части строки)

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

с какого символа

до какого (**не включая 8**)

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

от начала строки

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

до конца строки

Срезы строк

Срезы с отрицательными индексами:

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

len(s)-2

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

len(s)-6

len(s)-2

Операции со строками

Удаление:

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

```
"012"   "9"     "0129"
```

Вставка:

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

```
"012"       "3456789"
   "012ABC3456789"
```

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
```

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"
```