КОД БУДУЩЕГО
обучение талантливых школьников 8 -11 классов современным языкам программирования

Минцифры России

НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ ЦИФРОВАЯ ЭКОНОМИКА

20.35 УНИВЕРСИТЕТ

Программирование на языке Python. Базовый уровень

Модуль 2. Строки и списки

Строковый тип данных (занятие 1)

академия ПРОСВЕЩЕНИЕ

Строки в Python – упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

В Python строковый тип данных имеет название str (сокр. от string – струна, ряд).

П

Строка	Описание
S=’’	Пустая строка
S=’A’	Строка, состоящая из одного символа
S=’Hello’	Строка, состоящая из нескольких символов, заключенных в апострофы
S=”good”	Строка, состоящая из нескольких символов, заключенных в кавычки

Запомнить. После того, как строка будет создана, ее нельзя изменить, то есть все операции над строками в результате создают новую строку. Например, нельзя изменить отдельный символ строки: обращение s[3]=’x’ вызовет ошибку.

Строки можно создать несколькими способами:

1. С помощью одинарных и двойных кавычек.

```
first_string = 'Я текст в одинарных кавычках'
second_string = "Я текст в двойных кавычках"
```

2. С помощью тройных кавычек.

```
my_string = '''Это очень длинная
строка, ей нужно
много места'''
```

3. С помощью метода str().

```
my_num = 12345
my_str = str(my_num)
```

Операции над строками

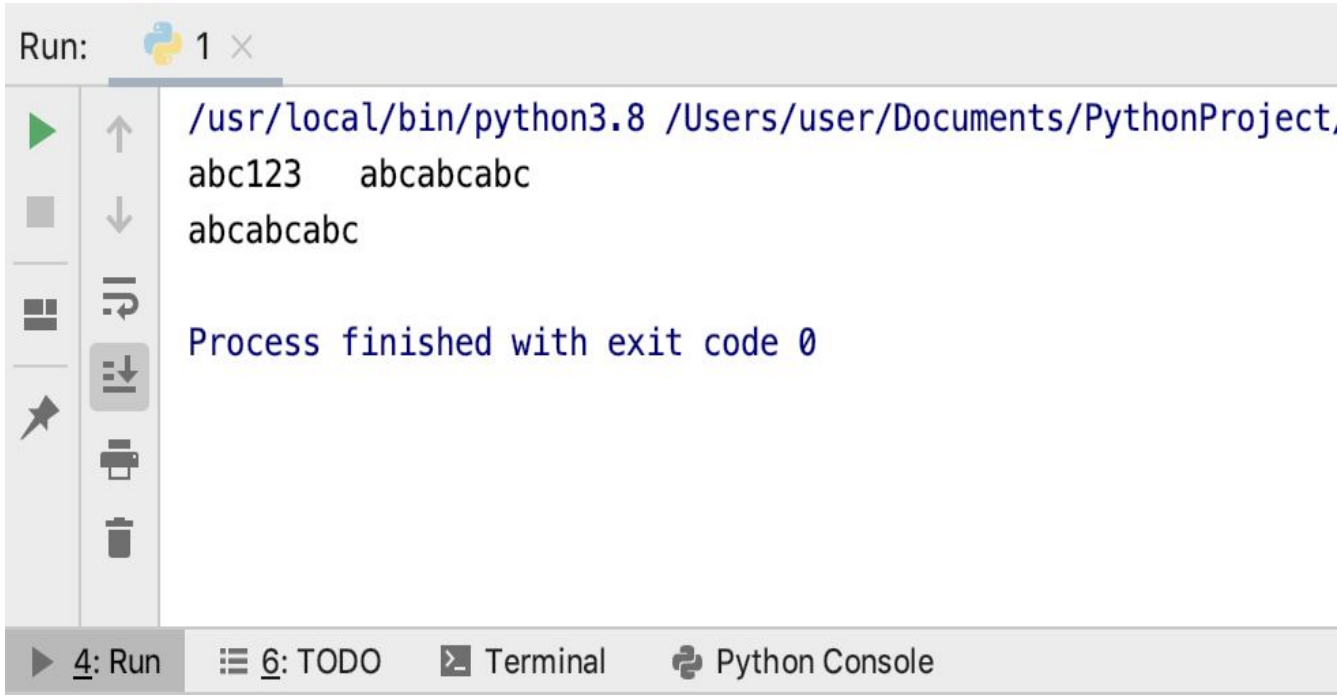
Операция	Описание	Пример
+	Сложение (конкатенация) строк. В результате применения возвращается строка, равная «склейке» указанных строк	a= 'py' b = 'th' b = 'on' s=a+b+c
*	Умножение строк. Оператор создает несколько копий строки, формат операции s*n или n*s, где s – это строка, а n – натуральное число	s=’ab’ sn=s*4

Операции над строками

Операция	Описание	Пример
in	Оператор принадлежности, который возвращает True, если подстрока входит в строку, и False, если нет	If ‘z’ in s: print(5)
>, <, >=, <=, ==, !=.	Сравнение строк	S1=”ab S2=”xy” If S1>S2: print(“Ok”)

Пример.

```
s1="abc"
s2="123"
s3=s1+s2
s4=s1*3
print(s3,' ',s4)
if s3>s4:
    print(s3)
else:
    print(s4)
```



Длиной строки называется количество символов, из которых она состоит. Чтобы посчитать длину строки используем встроенную функцию len() (от слова length – длина).

Пример.

```
s1 = 'abcdef'
length1 = len(s1)            # считаем длину строки из переменной s1
length2 = len('Python rocks!')  # считаем длину строкового литерала
print(length1)
print(length2)
```

Результат:
6
13

При подсчете длины строки считаются все символы, включая

Индексация строк

Для обращения к определенному символу строки используют индекс – порядковый номер элемента. Python поддерживает два типа индексации – положительную, при которой отсчет элементов начинается с 0 и с начала строки, и отрицательную, при которой отсчет начинается с -1 и с конца

Положительные индексы	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Пример строки	P	r	o	g	l	i	b
Отрицательные индексы	-7	-6	-5	-4	-3	-2	-1

Обратите внимание: если длина строки s равна len(s), то при положительной нумерации слева направо, последний элемент имеет индекс равный len(s) - 1, а при отрицательной индексации справа налево, первый элемент имеет индекс равный -len(s).

Пример.

```
# Спрашиваем имя пользователя
first = input("Введите имя: ")
# Вычисляем длину строки
num_chars = len(first)
# Отображаем результат
print("В вашем имени", num_chars, "символов")
```

Очень часто нужно просканировать всю строку целиком, обрабатывая каждый ее символ. Для этого удобно использовать цикл for.

Пример.

Напишем программу, которая выводит каждый символ строки на отдельной строке:

```
s = 'abcdef'
for i in range(len(s)):
    print(s[i])
```

```
s = 'abcdef'
for c in s:
    print(c)
```

Результат:

a
b
c
d
e
f