Лекция. Работа со строками. Методы строк.

Строки

Индексация. Срезы. Методы.

Что такое строка

- считывается со стандартного ввода функцией `input()`
- состоит из последовательности символов

Действия со строками

Конкатенация (сложение)

```
s1 = 'hello'
s2 = 'world'
print(s1 + s2)
'helloworld'
```

Действия со строками

Дублирование строки

```
s1 = 'hello'
print(s1 * 3)
'hellohellohello'
```

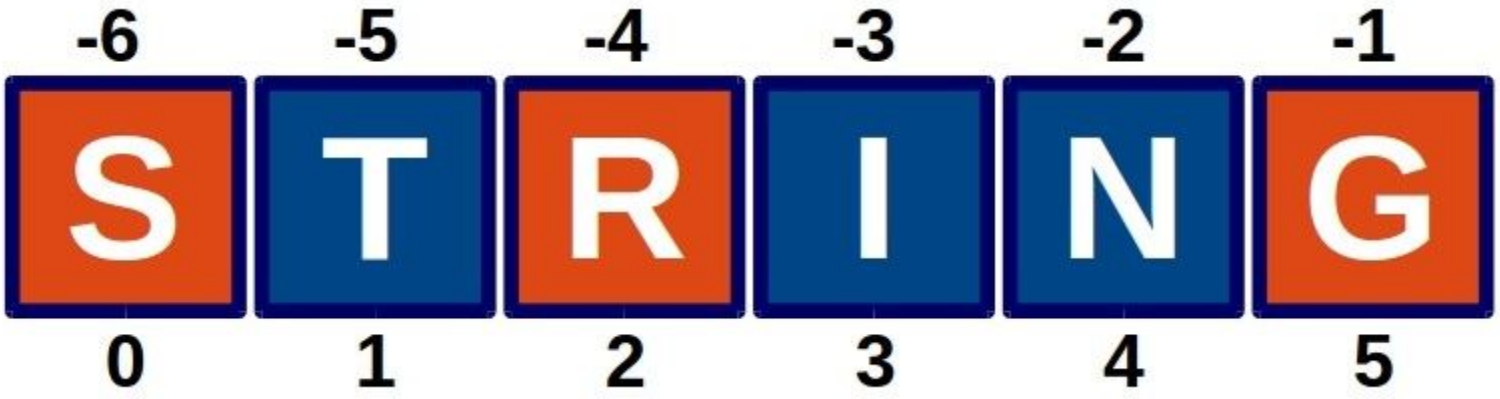
Действия со строками

Длина строки - функция len()

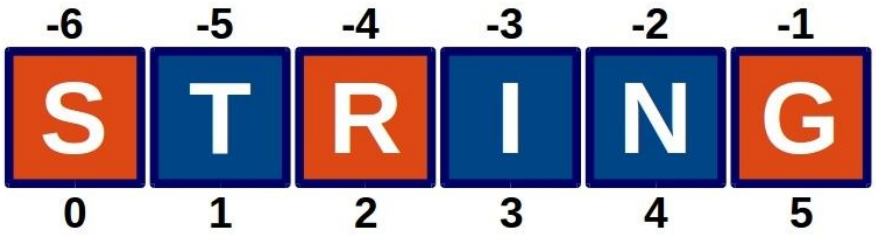
```
s1 = 'hello'
print(len(s1))
```

```
5
```

Индекс строк

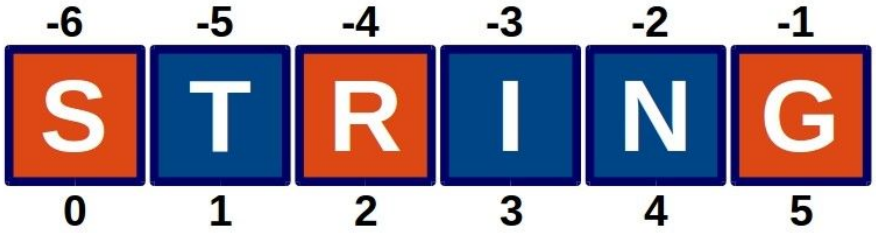


Индекс строк



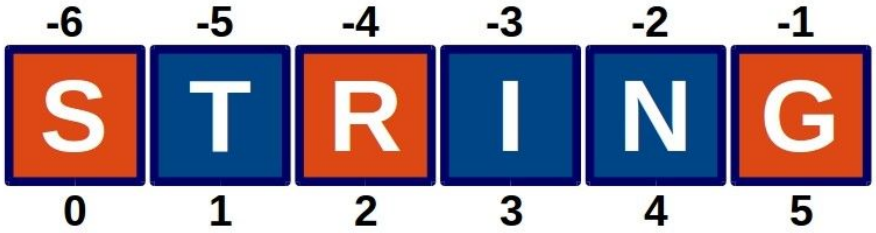

```
s1 = 'string'

Print(s1[0])

s
```

Индекс строк



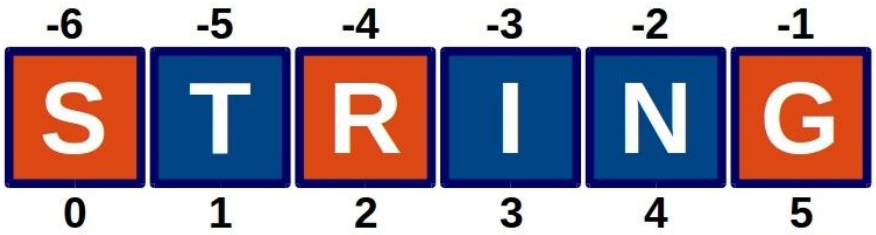

```
s1 = 'string'
s1[0] - 's'
s1[4]
```

Индекс строк



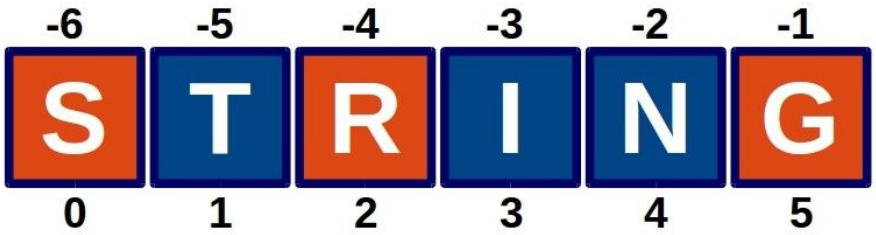

```
s1 = 'string'
s1[0] - 's'
s1[4]
```

Индекс строк



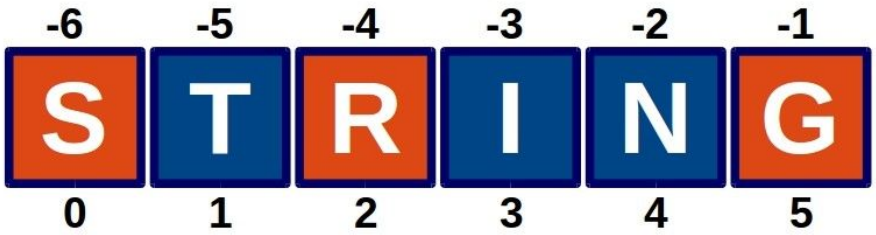

```
s1 = 'string'

s1[0] - 's'

s1[4] - 'n'

s1[-1]
```

Индекс строк



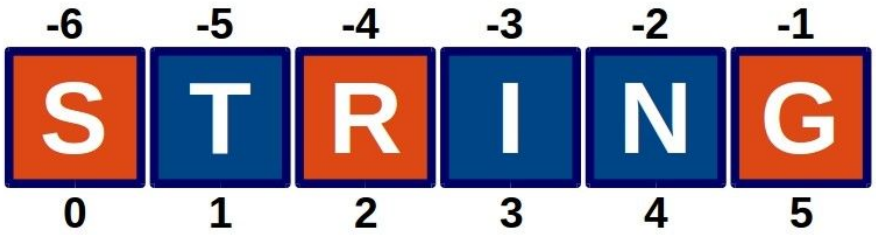

```
s1 = 'string'          s1[-4]

s1[0] - 's'

s1[4] - 'n'

s1[-1] - 'g'
```

Индекс строк



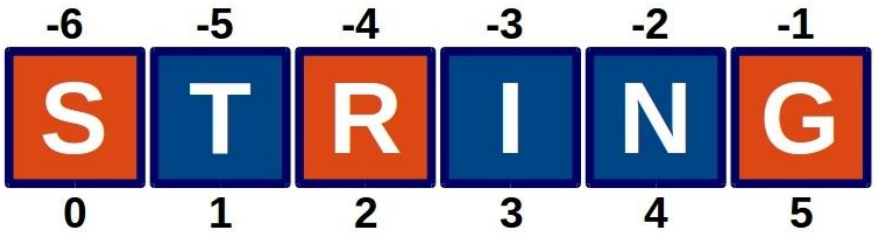

```
s1 = 'string'
s1[0] - 's'
s1[4] - 'n'
s1[-1] - 'g'
s1[-4] - 'r'
```

Срезы строк (slice)



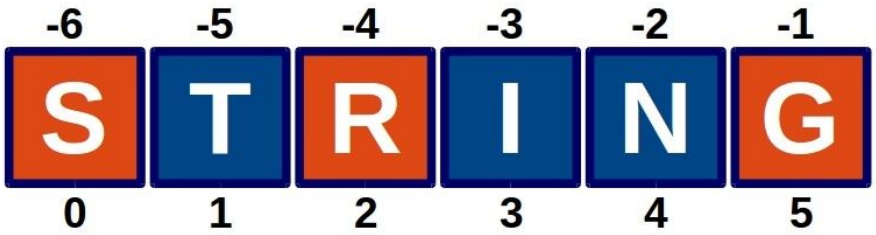

```
s1 = 'string'
s1[0:2]
```

Срезы строк (slice)



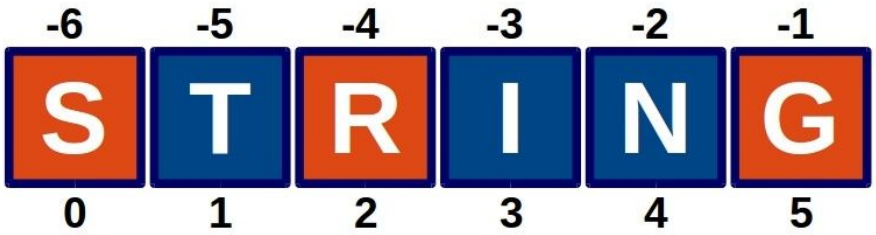

```
s1 = 'string'

s1[0:2] - 'st'

s1[2:5]
```

Срезы строк (slice)



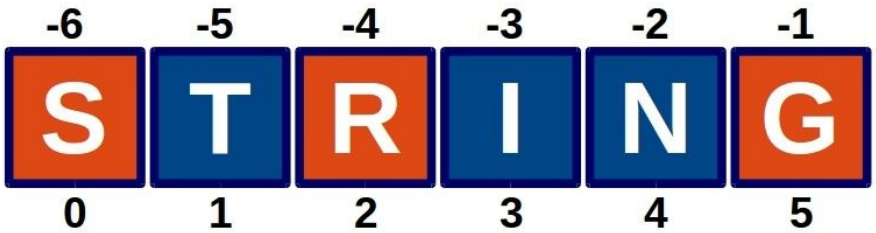

```
s1 = 'string'

s1[0:2] - 'st'

s1[2:5] - 'rin'

s1[-3:-1]
```

Срезы строк (slice)




```
s1 = 'string'          s1[:3]

s1[0:2] - 'st'

s1[2:5] - 'rin'

s1[-3:-1] - 'in'
```

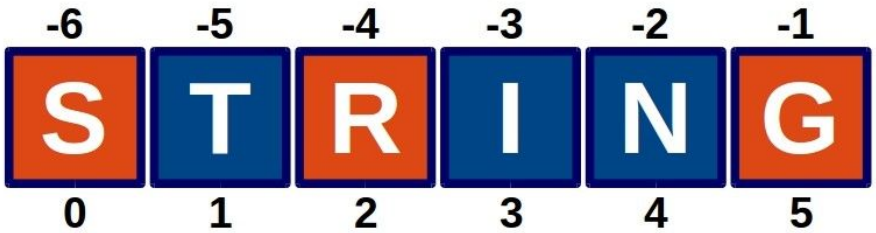
Срезы строк (slice)




```
s1 = 'string'
s1[0:2] - 'st'
s1[2:5] - 'rin'
s1[-3:-1] - 'in'
```

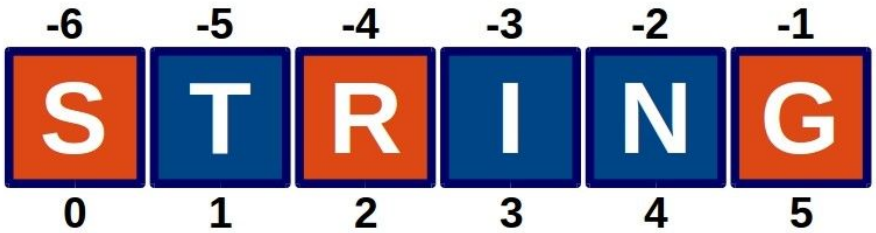
```
s1[:3] - 'str'
s1[3:]
```

Срезы строк (slice)



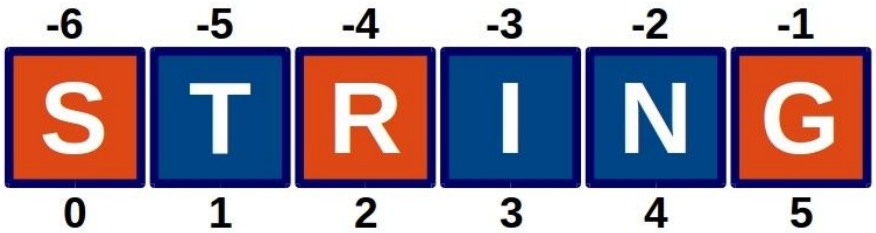

```
s1 = 'string'
s1[0:2] - 'st'
s1[2:5] - 'rin'
s1[-3:-1] - 'in'
s1[:3] - 'str'
s1[3:] - 'ing'
s1[:]
```

Срезы строк (slice)



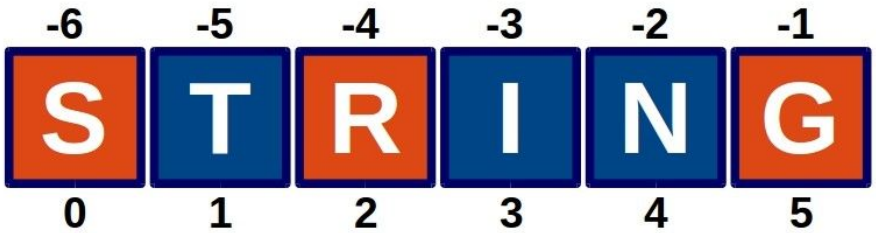

```
s1 = 'string'
s1[0:2] - 'st'
s1[2:5] - 'rin'
s1[-3:-1] - 'in'
s1[:3] - 'str'
s1[3:] - 'ing'
s1[:] - 'string'
s1[::2]
```

Срезы строк (slice)



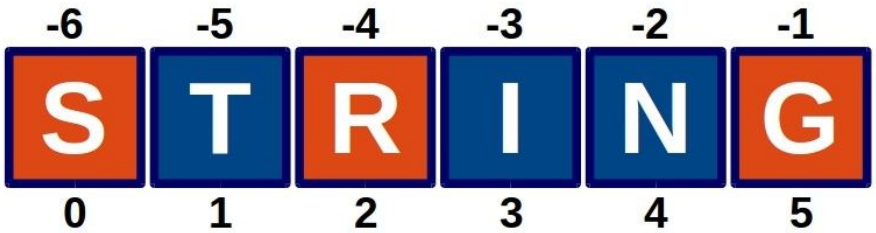

```
s1 = 'string'          s1[:3] - 'str'          s1[::3]
s1[0:2] - 'st'         s1[3:] - 'ing'
s1[2:5] - 'rin'        s1[:] - 'string'
s1[-3:-1] - 'in'       s1[::2] - 'srn'
```

Срезы строк (slice)




```
s1 = 'string'
s1[0:2] - 'st'
s1[2:5] - 'rin'
s1[-3:-1] - 'in'
s1[:3] - 'str'
s1[3:] - 'ing'
s1[:] - 'string'
s1[::2] - 'srn'
s1[::3] - 'si'
s1[::-1]
```

Срезы строк (slice)




```
s1 = 'string'       s1[:3] - 'str'       s1[::3] - 'si'
s1[0:2] - 'st'      s1[3:] - 'ing'       s1[::-1] - 'gnirts'
s1[2:5] - 'rin'     s1[:] - 'string'
s1[-3:-1] - 'in'    s1[::2] - 'srn'
```

Основные методы строк

isalpha(): возвращает True, если строка состоит только из алфавитных символов

islower(): возвращает True, если строка состоит только из символов в нижнем регистре

isupper(): возвращает True, если все символы строки в верхнем регистре

isdigit(): возвращает True, если все символы строки - цифры

isnumeric(): возвращает True, если строка представляет собой число

startswith(sub_str): возвращает True, если строка начинается с подстроки sub_str

endswith(str): возвращает True, если строка заканчивается на подстроку str

Основные методы строк

lower(): переводит строку в нижний регистр
upper(): переводит строку в вехний регистр
title(): начальные символы всех слов в строке переводятся в верхний регистр
capitalize(): переводит в верхний регистр первую букву только самого первого слова строки
lstrip(): удаляет начальные пробелы из строки
rstrip(): удаляет конечные пробелы из строки
strip(): удаляет начальные и конечные пробелы из строки
ljust(width): если длина строки меньше параметра width, то справа от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по левому краю

Основные методы строк

rjust(width): если длина строки меньше параметра width, то слева от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по правому краю

center(width): если длина строки меньше параметра width, то слева и справа от строки равномерно добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по центру

find(str[, start [, end]): возвращает индекс подстроки в строке. Если подстрока не найдена, возвращается число -1

replace(old, new[, num]): заменяет в строке одну подстроку на другую

split([delimeter[, num]]): разбивает строку на подстроки в зависимости от разделителя

join(strs): объединяет строки в одну строку, вставляя между ними определенный разделитель

Основные методы строк

С помощью **метода isnumeric()** можно проверить, введено ли в действительности число, и если так, то выполнить операцию преобразования:

```
string = input("Введите число: ")
if string.isnumeric():
    number = int(string)
    print(number)
```

Основные методы строк

Проверка, начинается или оканчивается строка на определенную подстроку:

```
file_name = "hello.py"

starts_with_hello = file_name.startswith("hello")
# True
ends_with_exe = file_name.endswith("exe")
# False
```

Основные методы строк

Удаление пробелов в начале и в конце строки:

```
string = "   hello  world!   "
string = string.strip()
print(string)          # hello  world!
```

Основные методы строк

Дополнение строки пробелами и выравнивание:

```
print("iPhone 7:", "52000".rjust(10))
print("Huawei P10:", "36000".rjust(10))
```

Результат:

```
iPhone 7:      52000
Huawei P10:      36000
```

Поиск в строке

Для поиска подстроки в строке применяется **метод find()**, который возвращает индекс первого вхождения подстроки в строку и имеет три формы:

find(str): поиск подстроки str ведется с начала строки до ее конца
find(str, start): параметр start задает начальный индекс, с которого будет производиться поиск
find(str, start, end): параметр end задает конечный индекс, до которого будет идти поиск

Поиск в строке

```
welcome = "Hello world! Goodbye world!"
index = welcome.find("wor")
print(index)       # 6

# поиск с 10-го индекса
index = welcome.find("wor",10)
print(index)       # 21

# поиск с 10 по 15 индекс
index = welcome.find("wor",10,15)
print(index)       # -1
```

Замена в строке

Для замены в строке одной подстроки на другую применяется **метод replace()**:

replace(old, new): заменяет подстроку old на new

replace(old, new, num): параметр num указывает, сколько вхождений подстроки old надо заменить на new

Замена в строке

```
phone = "+1-234-567-89-10"

# замена дефисов на пробел
edited_phone = phone.replace("-", " ")
print(edited_phone)     # +1 234 567 89 10

# удаление дефисов
edited_phone = phone.replace("-", "")
print(edited_phone)     # +12345678910

# замена только первого дефиса
edited_phone = phone.replace("-", "", 1)
print(edited_phone)     # +1234-567-89-10
```

Разделение на подстроки

Метод split() разбивает строку на список подстрок в зависимости от разделителя. В качестве разделителя может выступать любой символ или последовательность символов. Данный метод имеет следующие формы:

split(): в качестве разделителя используется пробел
split(delimeter): в качестве разделителя используется delimeter
split(delimeter, num): параметр num указывает, сколько вхождений delimeter используется для разделения. Оставшаяся часть строки добавляется в список без разделения на подстроки

Разделение на подстроки

```
text = "Это был огромный, в два обхвата дуб, с обломанными ветвями и с обломанной корой"
# разделение по пробелам
splitted_text = text.split()
print(splitted_text)
print(splitted_text[6])     # дуб,
# разбиение по запятым
splitted_text = text.split(",")
print(splitted_text)
print(splitted_text[1])     # в два обхвата дуб
# разбиение по первым пяти пробелам
splitted_text = text.split(" ", 5)
print(splitted_text)
print(splitted_text[5])     # обхвата дуб, с обломанными ветвями и с обломанной корой
```

Разделение на подстроки

```
text = "Это был огромный, в два обхвата дуб, с обломанными ветвями и с обломанной корой"
# разделение по пробелам
splitted_text = text.split()
print(splitted_text)
print(splitted_text[6])     # дуб,
# разбиение по запятым
splitted_text = text.split(",")
print(splitted_text)
print(splitted_text[1])     # в два обхвата дуб
# разбиение по первым пяти пробелам
splitted_text = text.split(" ", 5)
print(splitted_text)
print(splitted_text[5])     # обхвата дуб, с обломанными ветвями и с обломанной корой
```

Соединение строк

Ещё одну возможность для соединения строк представляет **метод join()**: он объединяет список строк. Причем текущая строка, у которой вызывается данный метод, используется в качестве разделителя:

```
words = ["Let", "me", "speak", "from", "my", "heart", "in", "English"]

# разделитель - пробел
sentence = " ".join(words)
print(sentence)  # Let me speak from my heart in English
********print(*words)

# разделитель - вертикальная черта
sentence = " | ".join(words)
print(sentence)  # Let | me | speak | from | my | heart | in | English
```

Соединение строк

Ещё одну возможность для соединения строк представляет **метод join()**: он объединяет список строк. Причем текущая строка, у которой вызывается данный метод, используется в качестве разделителя:

```
words = ["Let", "me", "speak", "from", "my", "heart", "in", "English"]

# разделитель - пробел
sentence = " ".join(words)
print(sentence)  # Let me speak from my heart in English

# разделитель - вертикальная черта
sentence = " | ".join(words)
print(sentence)  # Let | me | speak | from | my | heart | in | English
```

Соединение строк

Вместо списка в метод join можно передать простую строку, тогда разделитель будет вставляться между символами этой строки:

```
word = "hello"
joined_word = "|".join(word)
print(joined_word)      # h|e|l|l|o
```

Дана строка.

Сначала выведите третий символ этой строки.

Во второй строке выведите предпоследний символ этой строки.

В третьей строке выведите первые пять символов этой строки.

В четвертой строке выведите всю строку, кроме последних двух символов.

В пятой строке выведите все символы с четными индексами (считая, что индексация начинается с 0, поэтому символы выводятся начиная с первого).

В шестой строке выведите все символы с нечетными индексами, то есть начиная со второго символа строки.

В седьмой строке выведите все символы в обратном порядке.

В восьмой строке выведите все символы строки через один в обратном порядке, начиная с последнего.

В девятой строке выведите длину данной строки.

Входные	Выходные

`Hello`	`l`
	`l`
	`Hello`
	`Hel`
	`Hlo`
	`el`
	`olleH`
	`olH`
	`5`

Задачи

Дана строка, состоящая из слов, разделенных пробелами. Определите, сколько в ней слов. Используйте для решения задачи метод `count`.

Входные данные	Выходные данные
`Hello world`	`2`
`q w e`	`3`

Задачи

Дана строка. Разрежьте ее на две равные части (если длина строки — четная, а если длина строки нечетная, то длина первой части должна быть на один символ больше). Переставьте эти две части местами, результат запишите в новую строку и выведите на экран.

При решении этой задачи не стоит пользоваться инструкцией `if`.

Входные данные	**Выходные данные**
`Hello`	`loHel`
`Qwerty`	`rtyQwe`

Задачи

Дана строка, состоящая ровно из двух слов, разделенных пробелом. Переставьте эти слова местами. Результат запишите в строку и выведите получившуюся строку.

При решении этой задачи не стоит пользоваться циклами и инструкцией `if`.

Входные данные	**Выходные данные**
`Hello, world!`	`world! Hello,`

Задачи

Дана строка. Замените в этой строке все цифры `1` на слово `one`.

Входные данные	**Выходные данные**
`1+1=2`	`one+one=2`

Задачи

Дана строка. Удалите из этой строки все символы `@`.

Входные данные	Выходные данные
`@W@E@E@E@R`	`WEEER`

Задачи

Дана строка. Удалите из нее все символы, чьи индексы делятся на 3.

Входные данные	Выходные данные
`Python`	`yton`
`Hello`	`elo`